Prognose freier Rechenkapazitäten zur besseren Nutzung von Grid-Ressourcen

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Alek Opitz

geboren am 24. Mai 1976 in Halle (Saale)

Gutachter: Prof. Dr.-Ing. habil. Hartmut König

Gutachter: Prof. Dr. Hans-Ulrich Heiß

Gutachter: Prof. Dr. rer. nat. habil. Wolfgang Freudenberg

Tag der mündlichen Prüfung: 30. Juni 2009

Abstract

Vor allem in der Forschung und in den Entwicklungsabteilungen von Unternehmen gibt es eine Vielzahl von Problemen, welche nur mit Programmen zu lösen sind, für deren Ausführung die zur Verfügung stehende Rechenleistung kaum groß genug sein kann. Gleichzeitig ist zu beobachten, dass ein großer Teil der mit der installierten Rechentechnik vorhandenen Rechenkapazität nicht ausgenutzt wird. Dies gilt insbesondere für Einzelrechner, die in Büros, Computer-Pools oder Privathaushalten stehen und sogar während ihrer eigentlichen Nutzung selten ausgelastet sind. Eines der Ziele des Grid-Computings besteht darin, solche nicht ausgelasteten Ressourcen für rechenintensive Anwendungen zur Verfügung zu stellen.

Die eigentliche Motivation für die beabsichtigte bessere Auslastung der Ressourcen liegt dabei nicht primär in der höhreren Auslastung, sondern in einer möglichen Einsparung von Kosten gegenüber der Alternative der Neuanschaffung weiterer Hardware. Ein erster Beitrag der vorliegenden Arbeit liegt in der Analyse und Quantifizierung dieses möglichen Kostenvorteils. Zu diesem Zweck werden die relevanten Kosten betrachtet und schließlich verschiedene Szenarien miteinander verglichen. Die Analyse wird schließlich konkrete Zahlen zu den Kosten in den verschiedenen Szenarien liefern und somit das mögliche Potential zur Kosteneinsparung bei der Nutzung brach liegender Rechenkapazitäten aufzeigen.

Ein wesentliches Problem beim Grid-Computing besteht jedoch (vor allem bei der Nutzung von Einzelrechnern zur Ausführung länger laufender Programme) darin, dass die zur Verfügung stehenden freien Rechenkapazitäten im zeitlichen Verlauf stark schwanken und Berechnungsfortschritte durch plötzliche anderweitige Verwendung bzw. durch Abschalten der Rechner verloren gehen. Um dennoch auch Einzelrechner sinnvoll für die Ausführung länger laufender Jobs nutzen zu können, wären Vorhersagen der in der nächsten Zeit zu erwartenden freien Rechenkapazitäten wünschenswert. Solche Vorhersagen könnten u. a. hilfreich sein für das Scheduling und für die Bestimmung geeigneter Checkpoint-Zeitpunkte. Für die genannten Anwendungszwecke sind dabei Punktvorhersagen (wie z. B. Vorhersagen des Erwartungswertes) nur bedingt hilfreich, weshalb sich die vorliegende Arbeit ausschließlich mit Vorhersagen der Wahrscheinlichkeitsverteilungen beschäftigt. Wie solche Vorhersagen erstellt werden sollen, ist Gegenstand der restlichen Arbeit.

Dabei werden zunächst Möglichkeiten der Bewertung von Prognoseverfahren diskutiert, die Wahrscheinlichkeitsverteilungen vorhersagen. Es werden wesentliche Probleme bisheriger Bewertungsverfahren aufgezeigt und entsprechende Lösungsvorschläge gemacht. Unter Nutzung dieser werden in der Literatur zu findende und auch neue Vorgehensweisen zur Prognoseerstellung empirisch miteinander verglichen. Es wird sich zeigen, dass eine der neu entwickelten Vorgehensweisen im Vergleich zu bisher in der Literatur dargestellten Vorhersageverfahren klare Vorteile bzgl. der Genauigkeit der Prognosen erzielt.

Inhaltsverzeichnis

1. Einleitung

1.1. Nutzung freier Rechenkapazitäten durch Grid-Computing

In der Forschung, in den Entwicklungsabteilungen von Unternehmen, zum Teil aber auch im operativen Geschäft gibt es eine Vielzahl von Problemen, für deren Lösung die zur Verfügung stehende Rechenleistung kaum groß genug sein kann. Gleichzeitig ist zu beobachten, dass moderne Rechentechnik nur in geringem Maße ausgelastet ist. Dieser Gegensatz führte in den letzten Jahren zu verstärkten Anstrengungen zur besseren Ausnutzung der bereits vorhandenen Rechenkapazitäten.

Eine der Hauptursachen der bisher geringen Rechner-Auslastung ist dabei, dass Rechner meist nur für einen ganz bestimmten Zweck angeschafft werden und für diesen auch bei Spitzenlast ausreichend sein sollen. Eine geringe durchschnittliche Auslastung der Rechner ist daher eine logische Konsequenz. Ein wesentlicher Ansatzpunkt zur Lösung bzw. Minderung dieses Problems besteht folgerichtig darin, einerseits dieselben Rechenressourcen für unterschiedlichste Einsatzzwecke zu verwenden und andererseits dieselben Anwendungen je nach Verfügbarkeit auf unterschiedlichen Ressourcen auszuführen. Die damit verbundene Virtualisierung der Ressourcen kann nicht nur helfen, Kosten einzusparen, sondern bietet auch erhebliches Potential zur Einsparung von Energie (vgl. [Schä07]). Der zweite Punkt dürfte unter dem Gesichtspunkt der aktuellen Klimadiskussion und der rasch knapper werdenden fossilen Brennstoffe in Zukunft sogar noch an Bedeutung gewinnen.

Für die Virtualisierung von Ressourcen sind zwei grundsätzliche Herangehensweisen denkbar. Zum einen können Ressourcen angeschafft werden, die ausschließlich dazu dienen, anderen Nutzern für deren Zwecke zur Verfügung zu stehen. Beispielsweise besitzt die Firma Sun Rechenzentren, deren Ressourcen ausschließlich dazu dienen, gegen Entgelt zur Ausführung von Programmen fremder Nutzer vermietet zu werden ([Comp06]). Zum anderen gibt es die Möglichkeit, Ressourcen mit einem bestimmten primären Einsatzzweck zu Zeiten der nicht vollständigen Auslastung auch für andere Zwecke zur Verfügung zu stellen. Bei solchen Ressourcen kann es sich bspw. um Rechner in Computerpools oder Büros, um Server oder auch um Privatrechner handeln. Eine Vision für diese beiden, selbstverständlich kombinierbaren Herangehensweisen ist die, künftig analog zum Strom aus der Steckdose bei Vorhandensein einer entsprechenden Netzwerkanbindung je nach Bedarf auf (nahezu unbegrenzte) Rechenleistung zugreifen zu können. Die Vision einer solchen Infrastruktur wird in Anlehnung an das Stromnetz (engl.: *power grid*) Rechen-Grid[1] genannt. Die Nutzung eines solchen Grids wird als Grid-Computing, ein entsprechendes Anwendungsprogramm als Grid-Anwendung bezeichnet. Ein konkreter Auftrag zur Ausführung einer Grid-Anwendung

[1] Eine grau gestrichelte Unterstreichung weist bei der jeweils ersten Verwendung eines Begriffes innerhalb eines Kapitels darauf hin, dass im Glossar eine kurze Erklärung zu diesem Begriff nachgeschlagen werden kann.

mit bestimmten Parametern wird Grid-Job genannt. Die Idee des Grid-Computings ist in Abbildung 1-1 zusammengefasst: Eine rechenintensive (Grid-)Anwendung wird auf einer Infrastruktur aus Rechnern ausgeführt, die nicht primär für diese Anwendung zur Verfügung stehen, sondern von vielen verschiedenen Anwendungen genutzt werden können und u. U. auch über verschiedene Organisationen verteilt sind.

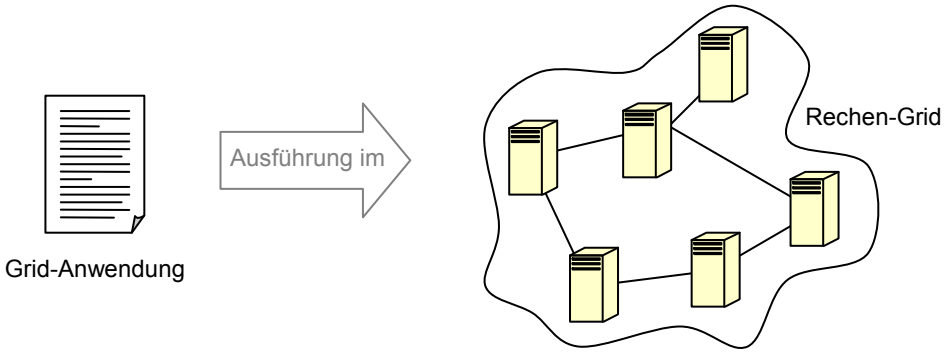

Abbildung 1-1: Idee des Grid-Computings: Ausführung von Anwendungen in einer virtualisierten Infrastruktur

Natürlich gibt es zahlreiche Unterschiede zwischen einem Rechen-Grid und einem Stromnetz. Es lassen sich aber auch einige wichtige Gemeinsamkeiten erkennen. Insbesondere geht es in beiden Fällen darum, dass ein bestimmter, zeitlich schwankender Bedarf (an Rechenleistung bzw. Strom) zuverlässig und kostengünstig zu decken ist. Dafür kommt jeweils ein bunter Mix von Ressourcen zum Einsatz. Im Falle des Stromnetzes handelt es sich bspw. um Wind-, Wasser-, Kohle- und Kernkraftwerke. Jede der verschiedenen Ressourcen hat bestimmte Vor- und Nachteile, aufgrund derer es überhaupt erst zu diesem Mix kommt (vgl. [RWE91], [Kalt95], [Gies98]). Beispielsweise gilt die Stromerzeugung durch Windkraftwerke als ökologisch besonders vorteilhaft und ist geeignet, die knapper werdenden Vorräte an fossilen Brennstoffen zu schonen. Auf der anderen Seite unterliegt die Leistungsfähigkeit von Windkraftwerken bestimmten Schwankungen, durch die Windkraftwerke niemals allein die Stromerzeugung sicherstellen könnten. Diesbezüglich vorteilhafter ist ein Kernkraftwerk, das (von Störfällen und Wartungsarbeiten abgesehen) sehr gleichmäßig Energie liefern kann. Da jedoch der Stromverbrauch nicht konstant ist, wird auch eine variable Stromerzeugung benötigt, für die Kernkraftwerke weniger geeignet sind. Diesbezüglich sind Wasserkraftwerke in der Form von Speicherkraftwerken günstiger, können jedoch nur in begrenztem Umfang zur Verfügung stehen. Die Thematik der Stromversorgung soll hier nicht weiter vertieft werden. Es sollte jedoch klar geworden sein, dass der Mix von verschiedenen Stromquellen in erster Linie die Ursache darin hat, dass es momentan für die Stromerzeugung keine eindeutig beste Form gibt, die auch in ausreichender Menge zur Verfügung steht.

An dieser Stelle wird die Analogie zum Rechen-Grid deutlich. Auch für dieses existiert keine eindeutig beste Form der Bereitstellung von Rechenkapazitäten. So gibt es Anwendungen, für deren Ausführung spezielle Parallelrechner benötigt werden, die

heutzutage meist in Form von Clustern bereitgestellt werden. Andere Anwendungen benötigen sehr zuverlässig zur Verfügung stehende Rechner, wie sie vor allem in entsprechend ausgestatteten Rechenzentren zu finden sind. Schließlich gibt es Anwendungsprogramme, die zwar sehr rechenintensiv sind, für deren Abarbeitung es jedoch keine feste Frist gibt und die prinzipiell auch auf vergleichsweise billigen Einzelrechnern[1] (statt bspw. auf Clustern) laufen können. Eine Nutzung von Einzelrechnern ist insbesondere deshalb interessant, da sie (vor allem in der Funktion als Arbeitsplatzrechner) eine oft besonders geringe Auslastung aufweisen ([Bers02]), in Summe also über große freie Rechenkapazitäten verfügen.

Im Rahmen der vorliegenden Arbeit soll es vor allem um den zuletzt genannten Typ von Anwendungsprogrammen gehen, die zwar so schnell wie möglich, aber i. Allg. auch so kostengünstig wie möglich abgearbeitet werden sollen. Gerade bei solchen Anwendungen erscheint die oben skizzierte Variante des Grid-Computings denkbar, zur Einsparung von Kosten rechenintensive Anwendungen auf Einzelrechnern auszuführen, deren primärer Anwendungszweck ein anderer als die Ausführung rechenintensiver Programme ist. Die Vielfalt der Anwendungen, die von einem solchen Szenario profitieren könnte, wird im nächsten Abschnitt verdeutlicht.

1.2. Anwendungsbeispiele

Die skizzierte Variante der Nutzung von Einzelrechnern zur Ausführung rechenintensiver Programme ist nicht für alle Anwendungen geeignet. Es stellt sich daher die Frage, was für Anwendungen von einer Nutzung solcher Rechner profitieren können. Da in der Literatur diesbezüglich oftmals nur Anwendungen genannt werden, deren Nutzen zumindest zweifelhaft ist (z. B. die Suche nach außerirdischen Signalen, die Suche nach Mersenne-Primzahlen oder Versuche des Brechens von Verschlüsselungen ([Hips], [BBC01], [Ther05])), könnte der Eindruck entstehen, dass es nur wenige sinnvolle Anwendungen gibt. Um die tatsächlich vorhandene große Vielfalt möglicher Anwendungen zu demonstrieren, wird im Folgenden eine Reihe praxisrelevanter, zum Teil sehr unterschiedlicher Anwendungen vorgestellt.

1.2.1. Optimierung des Crash-Verhaltens von Fahrzeugen

Zur Optimierung des Crash-Verhaltens von Fahrzeugen werden in immer stärkerem Maße Simulationen eingesetzt ([Blum02]). Gegenüber der Verwendung von Prototypen ergeben sich zum einen i. d. R. bessere Konstruktionen, zum anderen auch deutliche Zeitgewinne und Kostenvorteile in der Entwicklung.

---

[1] Unter Einzelrechnern seien hier solche Rechner verstanden, die von ihrer Leistungsfähigkeit und ihrer grundlegenden Architektur her in etwa einem modernen PC entsprechen. Explizit ausgeschlossen sind Cluster und andere Massiv-parallele Rechner. Unter diese Definition des Einzelrechners fallen somit insbesondere Arbeitsplatz- und Privatrechner, Workstations in Computer-Pools und auch viele Server.

Im Prozess der Optimierung sind dabei aus verschiedenen Gründen eine ganze Reihe von Simulationsläufen nötig. Deren Laufzeiten variieren von wenigen Minuten bis zu einigen Tagen, je nachdem, ob einzelne Komponenten oder ganze Fahrzeuge zu optimieren sind. Gründe für die Vielzahl benötigter Simulationsläufe zur Lösung eines einzelnen Optimierungsproblems liegen zum einen in der Exploration des Lösungsraums, also dem Testen unterschiedlicher Konstruktionen des Fahrzeugs. Neben dem Test verschiedener Konstruktionen ist darüber hinaus zu prüfen, ob sich für eine einzelne Konstruktion ein stabiles Crash-Verhalten ergibt. So können kleine numerische Fehler, aber auch Fertigungstoleranzen zu erheblichen Veränderungen des Verhaltens führen, was i. Allg. nicht wünschenswert ist. Um die Stabilität zu prüfen, sind daher für eine Konstruktion möglichst viele Simulationen mit geringen Abweichungen der Parameter durchzuführen.

Die erforderliche große Anzahl aufwendiger Simulationsläufe zeigt zum einen, dass enorme Rechenkapazitäten für den Prozess der Optimierung benötigt werden und im Grunde genommen nie genug davon zur Verfügung stehen können. Zum anderen sind viele der Simulationsläufe absolut parallel durchführbar. Insofern ist die Optimierung des Crash-Verhaltens von Fahrzeugen eine ideale Anwendung für die Verwendung eines Rechen-Grids, in dem freie Rechenkapazitäten von Rechnern genutzt werden, um die Simulationen durchzuführen. Dies gilt um so mehr, als viele der sinnvoll einsetzbaren Verfahren recht robust gegenüber dem Fehlschlagen eines kleinen Teils der Simulationen sind.

1.2.2. Arzneimittelforschung: Virtual Screening

Das nächste, in [Wasz01] beschriebene Beispiel betrifft die Suche nach neuen Medikamenten. Diese kann man in zwei Phasen einteilen. In der ersten Phase werden Verbindungen identifiziert, die zumindest ansatzweise die gewünschte Wirkung (z. B. die Bindung an bestimmte Rezeptoren) zeigen. In der zweiten Phase werden dann diese Verbindungen, welche auch als Leitsubstanzen bezeichnet werden, optimiert (und ausgiebig getestet), damit aus ihnen nach Möglichkeit tatsächlich Medikamente entstehen.

Eine Variante zur Suche von Leitsubstanzen ist das Virtual Screening. Bei diesem Verfahren ist der Ausgangspunkt die Kenntnis der räumlichen Struktur eines Ziel-Rezeptors. Für diesen Ziel-Rezeptor werden mittels entsprechender Computer-Berechnungen chemische Verbindungen gesucht, die an diesem Ziel-Rezeptor andocken können. Erforderlich dafür sind entsprechende Datenbestände mit den räumlichen Strukturen der Verbindungen, die gegen den Rezeptor getestet werden sollen. Je nach Genauigkeit der verwendeten Simulationsprogramme ergibt das Virtual Screening die gesuchten Leitsubstanzen oder aber eine Menge von Kandidaten für solche Leitsubstanzen. Im zweiten Fall wird die durchsuchte Bibliothek also lediglich geeignet eingeschränkt (z. B. auf ein Prozent der Verbindungen), so dass dann mittels kombinatorischer Chemie und High-Throughput-Screening nur noch auf dieser deutlich kleineren Menge der Verbindungen gesucht werden muss, um geeignete Leitsubstanzen zu finden.

Die beim Virtual Screening erforderliche Berechnung des Dockings ist sehr rechenintensiv und gut parallelisierbar, weshalb eine Anwendung des Grid-Computings denkbar ist. So wird bspw. bei Novartis unternehmensintern, aber Standort-übergreifend das Grid-Computing eingesetzt, um ein solches Virtual Screening durchzuführen. Genutzt werden alle Desktop-PCs größerer Niederlassungen, die in LANs mit einer Bandbreite von wenigstens 100 MBit/s angeschlossen sind. ([NeuZ03])

1.2.3. Entwicklung von Animationsfilmen

Eine weitere Anwendungsmöglichkeit des Grid-Computings ist das so genannte Rendering von Animationsfilmen. Wie in [Pixa04] beschrieben ist, werden bspw. in Pixars Animationsstudios Filme so entwickelt, dass zunächst die Story entwickelt wird, danach Skizzen erstellt werden und daraus 3D-Modelle der Figuren entworfen werden. Diese 3D-Modelle sind zusammengesetzte Gebilde, deren einzelne Teile durch Gelenke („avars“) miteinander verbunden sind, so dass die Modelle sich auch bewegen können. Mit den 3D-Modellen der Figuren und Gegenstände werden dann die Modelle der einzelnen Szenen entwickelt. In diesen Modellen wird festgelegt, wo sich die einzelnen Figuren und Gegenstände befinden und in welcher Position. Die Bewegungsabläufe der Figuren müssen dabei heutzutage nicht mehr rein manuell entwickelt werden; stattdessen kann der Computer die Bilder zwischen vorgegebenen Anfangs- und Endbildern entwickeln. Schließlich ist es noch erforderlich, dass die Oberflächen der einzelnen Modelle und die Lichtquellen festgelegt werden. Wenn all dies getan ist, werden die einzelnen Bilder gerendert. Dabei wird ausgehend vom entwickelten Modell das zweidimensionale Bild berechnet, das schließlich im Film erscheinen soll.

Gemäß [Pixa04] benötigt ein einziges Bild etwa sechs Stunden Rechenzeit, manche Bilder brauchen sogar 90 Stunden. Diese Zeitaufwände stimmen auch in etwa mit den Angaben aus [HP04] überein. Berücksichtigt man, dass ein Film in einer einzigen Sekunde (etwa) 24 Bilder benötigt und längst nicht jedes gerenderte Bild auch wirklich im Film verwendet wird, dann wird schnell klar, dass die Produktion eines einzigen Animationsfilms in Spielfilmlänge ein sehr aufwändiges Unterfangen ist. In [HP04] wurde bspw. für den Film „Shrek 2“ angegeben, dass über eine Million Bilder gerendert wurden, für die eine Rechenzeit von etwa zehn Millionen Stunden benötigt wurde.

Wie in verschiedenen Quellen angemerkt (z. B. [Gray03], [Smit03]), ist das Rendern von Bildern nicht nur rechenintensiv, sondern auch sehr gut parallelisierbar und dementsprechend prinzipiell für das Grid-Computing geeignet. Die Idee ist dabei, dass das Rendern der einzelnen Bilder auf unterschiedlichen Rechnern erfolgen kann, grundsätzlich aber auch die Berechnung eines einzelnen Bildes auf verschiedene Rechner verteilt werden kann.

1.2.4. Genetische Algorithmen

Eine Vielzahl weiterer möglicher Anwendungen des Grid-Computings ergibt sich bei der Nutzung genetischer Algorithmen. Auf diese wird in [Marc04] ausführlich einge-

gangen; insbesondere werden zahlreiche Anwendungsbeispiele aufgeführt, die auch für das Grid-Computing in Frage kommen.

Grundlagen genetischer Algorithmen

Ganz allgemein werden genetische Algorithmen im Rahmen von Aufgaben der Optimierung eingesetzt. Der grundsätzliche Ansatz folgt Prinzipien der Evolution, wie sie in der Natur zu beobachten ist. So wird mit einer (möglichst vielfältigen) Menge von Lösungskandidaten (Individuen) begonnen, aus denen sukzessive bessere Individuen abgeleitet werden sollen. Dieses Ableiten ist an genetische Mechanismen angelehnt. Einer dieser Mechanismen ist die Selektion. Diese wird dadurch simuliert, dass es eine Bewertungsfunktion für die Individuen gibt, mit der die Eignung für die gegebene Problemstellung eingeschätzt wird. Entsprechend der Bewertung ergeben sich dann für die einzelnen Individuen unterschiedliche Überlebenswahrscheinlichkeiten; je besser die repräsentierte Lösung ist, um so größer ist ihre Wahrscheinlichkeit, nicht verworfen zu werden.

Aus den übrig bleibenden Individuen werden dann neue erzeugt. Dazu werden bspw. Mutationen simuliert, indem zufällig bestimmte Eigenschaften (Merkmale) verändert werden. Die Hoffnung ist dabei die, auf bessere Lösungen zu stoßen. Neben diesen Mutationen gibt es ferner die Variante der Kombination verschiedener Individuen. Bei dieser Kombination entsteht ein neues Individuum als Mischung bisheriger Individuen (häufig aus genau zwei verschiedenen). Für die einzelnen Merkmale des neuen Individuums wird von den entsprechenden Merkmalswerten der zu kombinierenden Individuen einer zufällig ausgewählt. Eine Veranschaulichung für eine mögliche Kombination aus zwei verschiedenen Individuen findet sich in Abbildung 1-2.

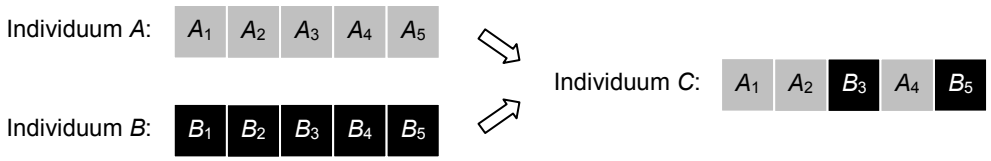

Abbildung 1-2: Beispiel einer möglichen Kombination von zwei Individuen *A* und *B* mit jeweils 5 Merkmalen zu einem neuen Individuum *C*

Die Anwendung der beschriebenen Maßnahmen zur Nachbildung der Selektion, der Mutation und der Kombination wird wiederholt ausgeführt; mögliche Abbruchkriterien sind das Erreichen einer bestimmten Zahl von Iterationen oder ausbleibende weitere Verbesserungen der bisher gefundenen Lösungen.

Als besonders günstig wird bei solchen genetischen Algorithmen angesehen, dass sie sehr universell anwendbar sind, dass sie parallelisierbar sind und dass sie auch gut geeignet sind, um lokale Optima zu überwinden und somit auch völlig neuartige, von bisher bekannten Lösungen stark abweichende Varianten erzeugen können. Insbesondere die gute Parallelisierbarkeit macht diese Algorithmen dabei auch für das Grid-Computing interessant.

Konkrete Beispiele der Anwendung genetischer Algorithmen

Eines der in [Marc04] angeführten Anwendungsbeispiele beschreibt die Optimierung eines Diesel-Motors ([Sche00]). Optimiert wurde dabei zunächst nur der Betrieb eines solchen Motors, also bspw. die Menge an zugeführter Luft, die Menge an zugeführtem Kraftstoff sowie das entsprechende zeitliche Verhalten. Zur Anwendung kamen genetische Algorithmen, mit denen sowohl die Effizienz des Motors als auch der Schadstoffausstoß deutlich verbessert werden konnte. Die Optimierung benötigte unter Nutzung von 32 Prozessoren einen Zeitraum von insgesamt etwas mehr als zwei Wochen. Zukünftige Pläne bestehen darin, auch den Aufbau (insbesondere die Form) des Motors durch die Nutzung solcher genetischen Algorithmen zu optimieren.

Ein weiteres Beispiel für die Anwendung genetischer Algorithmen ist in [Robi03] beschrieben und betrifft die in der Halbleitertechnik zum Einsatz kommende Elektronenstrahllithographie. Bei dieser geht es darum, mittels genau dosierter Elektronenstrahlen feinste dreidimensionale Strukturen in Materialien zu erzeugen. Problematisch ist dabei, dass ein auftreffender Elektronenstrahl sich nicht nur auf den anvisierten Punkt auswirkt, sondern durch Streuungen auch auf Nachbarbereiche. Dieser Effekt lässt sich zwar mittels Simulationen am Computer berechnen, macht jedoch das Finden geeigneter Dosierungen der Strahlen für die einzelnen Punkte der zu bearbeitenden Fläche außerordentlich schwierig. Faktisch bleibt nur das Experimentieren mit verschiedenen Dosierungsparametern, die jedoch in großer Zahl vorhanden sind. Aus diesem Grunde bieten sich genetische Algorithmen zur Lösung dieses Problems an und können [Robi03] zufolge ausgesprochen gute Ergebnisse liefern. Für die dort untersuchte zu erzeugende Struktur wurde mit 100.000 Iterationen gearbeitet, für die insgesamt auf einer 500-MHz-CPU mit einer Bearbeitungsdauer von 20 bis 60 Stunden zu rechnen ist.

Auch bei der Konstruktion von Antennen können genetische Algorithmen hilfreich sein, um wünschenswerte Eigenschaften zu erreichen ([Veld98]). Der große Vorteil bei der Anwendung solcher Algorithmen im Vergleich zur manuellen Konstruktion besteht darin, dass automatisch auch unkonventielle Designs betrachtet werden, auf die ein menschlicher Konstrukteur gar nicht kommen würde. In dem genannten Beispiel konnten dabei deutliche Fortschritte gegenüber bisher angewendeten Designs erzielt werden.

1.2.5. Weitere Anwendungsbeispiele

Neben den genannten Beispielen gibt es eine Vielzahl weiterer Anwendungsmöglichkeiten für die hier diskutierte Nutzung freier Rechenkapazitäten von Einzelrechnern. So wird in [Nimm99] über die Verwendung normaler Workstations für Simulationen bei Intel berichtet. Typische Aufgaben sind dabei Performance- oder Logik-Überprüfungen von Chip-Designs. Die einzelnen Jobs sind meist voneinander unabhängig, so dass sich die Ausführung auf Einzelrechnern anbietet, wie es im genannten Fall auch getan wird. Der Bereich typischer Rechenzeiten der Jobs reicht dabei von einigen Minuten bis zu mehreren Tagen, in Ausnahmefällen sogar bis zu mehreren Wochen.

In [Wies98] ist mit evolutionären Algorithmen[1] gearbeitet worden, um die mehrschichtige Vergütung von optischen Linsen zu optimieren. Bei diesen Vergütungen geht es insbesondere darum, dass ohne solche Beschichtungen das Licht einer bestimmten Frequenz typischerweise anteilig durchgelassen, absorbiert und reflektiert wird. Für eine bestimmte Frequenz ist es in vielen Fällen jedoch gewünscht, diese ganz oder gar nicht durchzulassen. Um diesem Ziel näher zu kommen, werden die angesprochenen Vergütungen eingesetzt, bspw. bei Zoom-Objektiven oder auch bei Brillengläsern. Die Wirksamkeit dieser Vergütungen wird jedoch von den Fertigungstoleranzen beeinflusst. Bei der Parametrisierung der Beschichtungen ist daher darauf zu achten, dass das gewünschte Ziel trotz der gegebenen Toleranzen mit hoher Wahrscheinlichkeit erreicht wird.

Abschließend sei auf die im Rahmen der vorliegenden Arbeit durchgeführten empirischen Untersuchungen verwiesen. In diesen waren zahlreiche Simulationsdurchläufe erforderlich, die von wenigen Minuten bis zu einigen Stunden Rechenzeit benötigten. Da jeder einzelne Durchlauf völlig unabhängig von den anderen Durchläufen war, stellten vorhandene Einzelrechner eine ideale Ausführungsumgebung dar. Dies gilt um so mehr, als dass der Speicherbedarf sehr gering und daher unproblematisch war.

1.3. Ziel der Arbeit: Prognose freier Rechenkapazitäten

Wie ausführlich in den beiden vorangegangenen Abschnitten dargelegt wurde, kann es durchaus sinnvoll sein, freie Rechenkapazitäten von ohnehin vorhandenen Einzelrechnern auszunutzen, um rechenintensive (Grid-)Anwendungen auszuführen. Damit jedoch die zusätzliche Nutzung von Einzelrechnern zur Ausführung rechenintensiver Grid-Anwendungen in der Praxis akzeptiert wird, dürfen die primären Nutzer nicht (spürbar) in ihrer Nutzung der Rechner gestört werden. Folglich dürfen die Grid-Anwendungen nicht mit den primären Anwendungen um die Ressourcen konkurrieren, sondern lediglich die übrig bleibenden Kapazitäten nutzen. Es ist also ausgeschlossen, dass für die auszuführenden Grid-Jobs Ressourcen reserviert werden können.

Die Verfügbarkeit freier Rechenkapazitäten wird daher entscheidend durch die lokalen Nutzer beeinflusst. Sie können bspw. jederzeit die Rechner herunterfahren oder rechen- bzw. speicherintensive Programme starten, die dann absichtlich gegenüber den Grid-Anwendungen priorisiert sind. Aus Sicht des Grids bzw. der Grid-Anwendungen sind somit die freien Rechenkapazitäten solcher Einzelrechner als unzuverlässig anzusehen. Dies wirkt sich insbesondere bei der Ausführung von Programmen negativ aus, die über einen längeren Zeitraum (bspw. mehrere Stunden) laufen. In Analogie zum Stromnetz könnte man daher Einzelrechner als die „Windkraftwerke der Rechen-Grids“ ansehen: Einerseits können sie aufgrund ihrer Unzuverlässigkeit den Bedarf an Rechenkapazität

[1] Die in Abschnitt 1.2.4 dargestellten genetischen Algorithmen sind Spezialformen der hier erwähnten evolutionären Algorithmen. Die Unterschiede liegen vor allem in der Repräsentation und in den daraus resultierenden Einschränkungen der zu variierenden Merkmale. Die für die Parallelisierbarkeit vorteilhaften Eigenschaften sind gleich.

nicht alleine decken, andererseits können sie aber eine empfehlenswerte bzw. kostengünstige Variante der Ausführung darstellen.

In den Stromnetzen wird der Unzuverlässigkeit der Windkraftwerke dadurch begegnet, dass bspw. die (im Wesentlichen vom Wetter abhängigen) zu erwartenden Strommengen prognostiziert werden. Ein analoges Vorgehen erscheint auch in einem Rechen-Grid denkbar und ist das Thema der vorliegenden Arbeit: Es geht hier darum zu untersuchen, mit welchen Prognoseverfahren die zu erwartenden freien Rechenkapazitäten von als unzuverlässig einzustufenden Einzelrechnern möglichst genau vorhergesagt werden können. Diskutiert werden sollen dabei Zeithorizonte von bis zu 24 Stunden. Der vielfältige Nutzen solcher Prognosen wird im Folgenden dargelegt.

1.3.1. Mögliche Anwendungen der Prognosen

Prognosen der freien Kapazitäten können für verschiedene Zwecke hilfreich sein. Auf einige dieser wird hier näher eingegangen.

Verbindlichkeit

Beim Grid-Computing geht es i. Allg. auch darum, von fremden Anbietern die Ressourcen nutzen zu können. Wenn ein Nutzer aber für die Ressourcennutzung bezahlt, möchte er vermutlich auch verbindliche Zusicherungen darüber haben, wann der Job fertig sein wird bzw. mit welcher Wahrscheinlichkeit er in der nächsten Zeit t fertiggestellt wird. Dies wäre für den Nutzer auch dahingehend wichtig, um ggf. auf einen anderen Ressourcen-Anbieter auszuweichen.

Gerade bei der Nutzung von Einzelrechnern ist es jedoch für einen Ressourcen-Anbieter alles andere als trivial, Zusicherungen irgendeiner Art zu machen. Aus diesem Grunde wäre es für den Anbieter hilfreich, über möglichst genaue Verfahren für die Vorhersage der zu erwartenden freien Rechenkapazitäten verfügen zu können. Mittels dieser Verfahren könnten zumindest statistische Vorhersagen gemacht werden, dass bestimmte Jobs mit einer gewissen Wahrscheinlichkeit erfolgreich abgearbeitet werden können. Ein Anbieter mit schlechten Vorhersagen könnte dabei aus verschiedenen Gründen gegenüber anderen Anbietern Nachteile erleiden. So würden Kunden vermutlich andere Anbieter bevorzugen, wenn sie den Eindruck haben, dass die von ihrem bisherigen Anbieter gemachten Zusagen zu optimistisch sind. Die Abwanderung der Kunden könnte noch dadurch verstärkt werden, wenn bei Vergleichsstudien bzw. -untersuchungen die Nichteinhaltung der Zusagen festgestellt und publik gemacht wird. Bei speziellen Vereinbarungen mit Kunden, die regelmäßig Rechenleistung in Anspruch nehmen, könnte es schließlich sogar zur Zahlung von Vertragsstrafen kommen, wenn über einen bestimmten Zeitraum die zugesagten Wahrscheinlichkeiten über dem Anteil tatsächlich erfolgreich ausgeführter Jobs liegen. Um derartige negativen Konsequenzen zu vermeiden, sollte ein Anbieter für die eigenen Ressourcen genaue Vorhersagen abgeben können, damit die verbindlichen Zusagen eingehalten werden können.

Scheduling

Ein weiterer wichtiger Punkt ist der des Schedulings oder – genauer gesagt – der der Auswahl eines geeigneten Rechners zur Ausführung eines konkreten Jobs mit bestimmter benötigter Rechenleistung. Zur Optimierung dieser Auswahl wäre es hilfreich zu wissen, welche Rechenleistung die einzelnen Rechner in nächster Zeit zur Verfügung stellen können, ohne zwischendurch ausgeschaltet zu werden.

Wahl der Granularität der Jobs

Manche der komplexeren, also länger laufenden Jobs lassen sich recht gut in Teilaufgaben aufteilen, die an verschiedene Rechner verteilt werden können und dort unabhängig voneinander berechnet werden können (vgl. Abschnitt 1.2). Auf diese Weise lässt sich die Berechnungsdauer pro Prozess verringern, so dass auch mit Ressourcen gearbeitet werden kann, die jeweils nur begrenzte Zeit zur Verfügung stehen. Allerdings gilt zu beachten, dass mit stärkerer Aufteilung der Grid-Jobs i. Allg. auch der Overhead steigt. Beispielsweise muss bei der Verteilung auf mehrere Rechner in aller Regel zumindest ein Teil der Daten mehrfach übertragen werden. Aus diesem Grunde muss eine möglichst optimale Aufteilung gefunden werden, die einerseits nur einen geringen Overhead aufweist, andererseits aber auch mit hoher Wahrscheinlichkeit die erfolgreiche Abarbeitung der gestarteten Teiljobs ermöglicht, ohne dass die Rechner vorher ausgeschaltet werden. Zum Finden einer solchen Aufteilung wäre eine möglichst zuverlässige Abschätzung der in der nächsten Zeit auftretenden freien Rechenkapazitäten hilfreich.

Wahl der Zeitpunkte für Checkpoints

Ein Mittel, um mit unzuverlässig zur Verfügung stehenden Ressourcen auch länger laufende Jobs abarbeiten zu können, sind Checkpoints, die den erreichten Stand der Berechnungen festhalten. Bei einem möglicherweise erfolgten Neustart des Rechners muss dann der Job nicht unbedingt komplett neu berechnet werden. Stattdessen kann auf dem im letzten Checkpoint vermerkten Stand aufgebaut werden.

Problematisch ist die Wahl der Zeitpunkte für die einzelnen Checkpoints. Da jede Erstellung eines Checkpoints auch einen gewissen Aufwand erfordert, ist eine möglichst gute Wahl für die Intervalle zwischen Checkpoints entscheidend für die Effizienz der Ausführung der Jobs. Werden nur wenige Checkpoints erstellt, ist im Falle eines Neustarts meist ein großer Teil der bereits erfolgten Berechnungen zu wiederholen. Bei sehr häufigen Checkpoints hingegen mindert bereits die Erstellung der Checkpoints die Effizienz der Ausführung.

Für eine Entscheidung bzgl. der Zeitpunkte für die Checkpoints wären daher ebenfalls Abschätzungen der noch verbleibenden freien Rechenkapazität wünschenswert.

1.3.2. Abgrenzung der Arbeit

Diese Arbeit beschränkt sich im Weiteren darauf, möglichst optimale Prognosen für die freien Rechenkapazitäten zu finden. Ausdrücklich nicht diskutiert wird, wie derartige

Prognosen in den verschiedenen Einsatzszenarien möglichst optimal ausgenutzt werden können. Insbesondere werden bspw. weder Scheduling- noch Checkpoint-Verfahren näher betrachtet.

1.4. Gliederung der Arbeit

Der Rest der Arbeit gliedert sich wie folgt. Zunächst wird in Kapitel 2 geprüft, inwieweit der hier als Motivation angeführte Kostenvorteil bei der Ausnutzung freier Rechenkapazitäten vorhandener Rechner tatsächlich zutreffend ist. Zu diesem Zweck werden die relevanten Kosten betrachtet und schließlich verschiedene Szenarien miteinander verglichen. Die Analyse wird einerseits deutlich machen, dass die Ausführung von Programmen auch auf existierenden, zeitweilig nicht ausgelasteten Rechnern immer Kosten in relevanter Höhe verursacht. Andererseits wird deutlich werden, welch großes Potential zur Einsparung von Kosten besteht, wenn statt der klassischerweise exklusiven Nutzung von Ressourcen verbleibende Kapazitäten von Rechnern verwendet werden.

Im sich anschließenden Kapitel 3 erfolgt eine genaue Definition der zu prognostizierenden freien Rechenkapazitäten. Es wird darauf eingegangen, warum Vorhersagen von Wahrscheinlichkeitsverteilungen sinnvoller sind als Punktschätzungen. Ausführlich wird auch das Problem der Bewertung solcher prognostizierten Wahrscheinlichkeitsverteilungen diskutiert. Es wird sich zeigen, dass diese Bewertung alles andere als trivial ist und spezielle Maße erfordert, die in dem Kapitel entwickelt werden. Aufbauend auf diesen Maßen wird in Kapitel 4 auf mögliche Prognoseverfahren und deren empirischen Vergleich eingegangen. Dazu wird zunächst das methodische Vorgehen zum Vergleich verschiedener Verfahren vorgestellt, dann werden konkrete Varianten zur Prognoseerstellung diskutiert und diese schließlich in empirischen Untersuchungen miteinander verglichen.

Kapitel 5 setzt schließlich die vorgestellten Verfahren in Beziehung zu existierenden Ansätzen aus der Literatur und macht die entscheidenden Unterschiede deutlich. Abgeschlossen wird die Arbeit in Kapitel 6 mit einer Zusammenfassung und einem Ausblick auf mögliche weiterführende Arbeiten.

2. Kosten des Grid-Computings

Das Argument einer möglichen Kostenersparnis ist für das Grid-Computing, wie es in dieser Arbeit betrachtet wird, die eigentliche Motivation. Bisher finden sich in der Literatur jedoch kaum Arbeiten, welche die möglichen Kostenvorteile fundiert analysieren. Zum Teil wird sogar das Bild gezeichnet, dass die Nutzung freier Ressourcen zum Nulltarif möglich wäre ([Silv05]). Eine solche Sichtweise stellt jedoch eine sehr starke Vereinfachung dar, durch die erhebliche Kosten vernachlässigt werden.

Eine Kenntnis der Kosten ist in vielen Fällen Voraussetzung, den Nutzen des Grid-Computings einschätzen zu können. Beispielsweise wurden Ansätze entwickelt, um Grid-Computing auch auf nicht vertrauenswürdigen Rechnern auszuführen. Dabei werden die Aufgaben auf mehrere Rechner verteilt, damit die Aufgaben mehrfach ausgeführt und die Ergebnisse schließlich miteinander verglichen werden können ([Sarm99], [Skaw02]). Inwieweit mit einer solchen Nutzung der Ressourcen unter realen Bedingungen aber überhaupt noch Kostenvorteile erreicht werden können, wird nicht thematisiert. Ganz allgemein lassen sich in der Literatur bisher nur wenige, rudimentäre Ansätze zur Einschätzung der Kosten des Grid-Computings finden.

Einer dieser Ansätze ist in [Gray03] enthalten. In diesem werden verschiedenen Operationen, die in einem Grid-Job auftreten können, Preise (bzw. Kosten) zugeordnet. Explizit betrachtet wurden bspw. die Übertragung von Daten über das Internet, der Zugriff auf Datenbanken oder auch die Auslastung einer CPU. Für das Jahr 2003 ergaben sich die in Tabelle 2-1 aufgeführten Operationen, die jeweils einem US-Dollar entsprachen.

Senden von 1 GB Daten über ein WAN
Auslastung einer CPU für 8 Stunden
Belegung oder Reservierung von 1 GB Festplattenspeicher
10 Millionen Zugriffe auf eine Datenbank
Übertragung von 10 TB von bzw. zu einer Festplatte

Tabelle 2-1: Mögliche Entsprechungen für einen US-Dollar im Jahr 2003 (nach [Gray03])

Aus den Werten wurde geschlussfolgert, dass Grid-Anwendungen rechenintensiv sein müssen, damit sie sich wirtschaftlich lohnen. Konkrete Werte für den Vergleich der Kosten zwischen einer Grid-Anwendung und einer entsprechenden konventionellen Anwendung sind jedoch nicht angegeben. Ferner wurde auch nicht die konkrete Ausgestaltung des Grids betrachtet, obwohl diese (wie im Weiteren noch zu sehen sein wird) erhebliche Auswirkungen auf die Kosten hat.

Ein anderer Ansatz zur Quantifizierung des Kostenvorteils Kostenvorteil von Grid-Anwendungen gegenüber vergleichbaren herkömmlichen Anwendungen wurde in [Foga04] verfolgt. Dazu wurde eine Tabellenkalkulation erstellt, in die verschiedene Parameter der zu erstellenden (Grid-)Anwendung eingegeben werden müssen, u. a. die

benötigte Hard- und Software sowie der erforderliche Arbeitsaufwand zum Aufsetzen des Systems. Nach der Eintragung all dieser Parameter errechnet die Tabellenkalkulation schließlich die Kostenersparnis der Grid-Variante gegenüber dem herkömmlichen System. Es bleibt auch hier unklar, von welcher Art von Grid die Rede ist. Ferner ist es so, dass die Abschätzungen in [Foga04] auf pauschalen Angaben von Anbietern von Grid-Software bzw. -Systemen basieren. Aus diesem Grunde ist es fraglich, inwieweit die Daten tatsächlich auf realistischen Annahmen beruhen bzw. inwieweit sie von den Marketing-Abteilungen beeinflusst wurden.

Aufgrund der fehlenden Verfügbarkeit weitergehender Untersuchungen zu den Kosten des Grid-Computings, diese jedoch für die Motivation der vorliegenden Arbeit eine wesentliche Rolle spielen, soll in diesem Kapitel versucht werden, die beim Grid-Computing entstehenden Kosten genauer zu quantifizieren. Dabei wird der Fokus auf den Kosten liegen, die einem Anbieter von Grid-Ressourcen entstehen. Die damit implizit angenommene Beziehung zwischen einem Kunden (dem Grid-Nutzer) und einem Lieferanten (dem Ressourcen-Anbieter) ist in der folgenden Abbildung 2-1 dargestellt. Während der Kunde einfach nur seine Programme abgearbeitet bekommen haben möchte, muss der Lieferant die notwendigen Maßnahmen ergreifen, damit diese Jobs auf seinen Ressourcen ordnungsgemäß abgearbeitet werden können. Auch wenn in vielen realen Grid-Projekten die beiden Rollen des Kunden und des Lieferanten nicht organisatorisch getrennt sind, lassen sich die entstehenden Kosten dennoch den entsprechenden Rollen zuordnen.

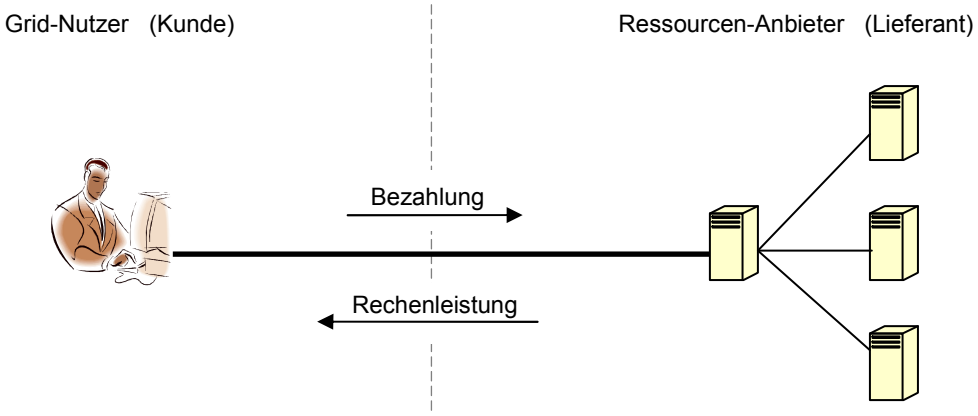

Abbildung 2-1: Grid-Nutzer und Ressourcen-Anbieter

Für die Analyse der Kosten des Ressourcen-Anbieters werden zunächst in Abschnitt 2.1 die relevanten Kostenfaktoren ermittelt und für diese typische Werte angegeben. Darauf aufbauend werden in Abschnitt 2.2 die Kosten für drei reale Grids analysiert. Die ermittelten Kosten werden schließlich zu aktuellen Marktpreisen in Beziehung gesetzt (Abschnitt 2.3). Eine Diskussion der Ergebnisse erfolgt in Abschnitt 2.4.

2.1. Relevante Faktoren für die Kosten eines Anbieters von Grid-Ressourcen

Für einen Anbieter von Grid-Ressourcen sind verschiedene Kostenfaktoren relevant. Insbesondere wurden diesbezüglich die folgenden Faktoren identifiziert:

- Server-Hardware
- Infrastruktur
- Stromverbrauch der Rechentechnik
- Software
- Administration, Service und Support
- Verbindung zum Internet

Diese Faktoren werden in den folgenden Unterabschnitten genauer untersucht.

2.1.1. Kosten für die Server-Hardware

Mit dem Kostenfaktor Server-Hardware sind hier die Kosten für die Hardware der reinen Rechentechnik gemeint. Die Kosten für die umgebende Infrastruktur werden in Abschnitt 2.1.2 diskutiert.

Kosten für Neuanschaffungen

Bei der Berechnung der Kosten für die Server-Hardware sind zunächst die Kosten für die Erstanschaffung zu betrachten. Denkbar sind sowohl der Einsatz von Desktop-PCs als auch der Einsatz von Servern. Ausgangspunkt ist dabei im Folgenden immer der Preis pro CPU. Für diesen wurde eine realistische Preisspanne von 300 bis 6000 Euro ermittelt ([Idea07], [Prei06], [Prei06a], [Prei07]). Um die Preise in jährliche Kosten umzurechnen, ist die wirtschaftliche Lebensdauer einzubeziehen. Zu dieser Lebensdauer gibt es in der Literatur kaum zuverlässige Angaben, meist wird aber von einer Spanne von 3 bis 5 Jahren ausgegangen ([Anse06], [Patt07], [Schä07], [Weer04]). Folglich ist mit jährlichen Kosten zwischen 60 und 2000 Euro pro CPU zu rechnen.

Die offensichtlich sehr breite Spanne für die Kosten der Server-Hardware ist für eine genaue Schätzung der Kosten recht ungünstig. In vielen Fällen sind jedoch diese Kosten gar nicht relevant. Wenn die Ressourcen nämlich für einen anderen Zweck als das Grid-Computing beschafft wurden und dann nur während Zeiten des Ungenutzt-Seins für Grid-Jobs verwendet werden, müssen die Anschaffungskosten nicht in die Kostenabschätzung für das Grid einbezogen werden.

Ersetzen defekter Hardware-Komponenten

Neben der Neuanschaffung von Server-Hardware ist es gelegentlich erforderlich, defekte Komponenten auszutauschen. Die damit verbundenen, in vielen Fällen dominierenden Kosten der menschlichen Arbeit werden dabei hier noch nicht betrachtet,

sondern erst in Abschnitt 2.1.5. Hier geht es erst einmal ausschließlich um die reinen Preise der zu ersetzenden Komponenten.

Um die jährlichen Kosten für das notwendige Ersetzen von Hardware-Komponenten abschätzen zu können, ist die Fehlerhäufigkeit der einzelnen Komponenten zu berücksichtigen. Diese wird typischerweise durch die Mean Time to Failure (MTTF) angegeben. Eine umfangreiche Literaturrecherche ergab die in Tabelle 2-2 zusammengestellten Werte.

Komponente	Kosten einer Einheit (€)	MTTF (h)	Kosten pro Komponente und Jahr (€)
CPU-Kühler	30 – 80	50.000 – 150.000	1,8 – 14
Netzteil	20 – 80	50.000 – 80.000	2,2 – 14
Mainboard	90 – 250	45°C: 100.000 – 200.000	3,6 – 22
		55°C: 80.000 – 170.000	4,5 – 27
CPU	120 – 7200	30°C: 360.000	1,2 – 72
		60°C: 160.000	2,8 – 160
		80°C: 80.000	5,4 – 320
Festplatte	50 – 400	25°C: 600.000	0,5 – 4
		52°C: 300.000	1 – 8

Tabelle 2-2: Kosten durch auszutauschende Hardware-Komponenten
(Quellen: [Ande03], [Boeg03], [Idea06], [Kozi01], [Mabs05], [MSI06], [Port06], [Prei06a], [Radi02], [Schr07])

Die angegebenen Werte für die Kosten der Ersatzteile sind u. U. nicht genau die, welche für die hier vorzunehmende Kostenabschätzung anzusetzen sind. Wenn die Ressourcen nicht exklusiv für das Grid-Computing verwendet werden, dann müssten die genannten Beträge genau genommen anteilig auf die Nutzungsarten verteilt werden. Da jedoch der typische Auslastungsgrad der Ressourcen nur im Bereich zwischen 5 und 20 Prozent liegt ([Andr02], [HP07], [Röhr04], [Schä07]), werden hier der Einfachheit halber die vollen Kosten dem Grid-Computing zugeschrieben.

Zu berücksichtigen bleiben ferner eventuelle Garantieansprüche gegenüber den Herstellern bzw. Händlern. Wenn z. B. eine zweijährige Garantie gewährt wird, dann müssten bei einer Nutzungsdauer von zwei Jahren gar keine Kosten für zu ersetzende Komponenten kalkuliert werden, bei einer Nutzungsdauer von fünf Jahren hingegen nur drei Fünftel der in Tabelle 2-2 angegebenen Werte.

2.1.2. Infrastrukturkosten

Der Betrieb der Grid-Ressourcen erfordert eine technische Infrastruktur, zu der Gebäude, Kühlanlagen, Beleuchtungen, Notstromversorgungen und anderes gehören. Um die Kosten dieser Infrastruktur abzuschätzen, werden hier Kalkulationsempfehlungen aus [Turn06] verwendet, welche auf den Untersuchungen von 16 größeren Projekten der jüngsten Vergangenheit basieren. In der genannten Arbeit sind verschiedene Formeln

aufgeführt, welche die ungefähren Kosten eines Rechenzentrums in Abhängigkeit von der Nennleistung der Stromversorgung für die IT-Hardware und der benötigten Grundfläche angeben. Für unterschiedliche Kategorien (engl.: Tiers) der Rechenzentren gibt es separate Formeln, wobei sich die Kategorien hauptsächlich hinsichtlich der Verfügbarkeit (also Zuverlässigkeit) unterscheiden (vgl. [Turn06a]). Die höchste Zuverlässigkeit wird bei Kategorie 4 erreicht, die geringste bei Kategorie 1.

Typische, für das Grid-Computing gut geeignete Aufgaben sind rechenintensiv und kommen häufig aus dem Bereich der Forschung und Entwicklung. Da es dort vor allem darum geht, überhaupt die benötigte Rechenleistung zu bekommen, jedoch keine so hohen Anforderungen an die Verfügbarkeit der verwendeten Rechentechnik gestellt werden, sind Rechenzentren der Kategorien 3 und 4 normalerweise nicht erforderlich. Dementsprechend werden in [Turn06a] Rechenzentren der Kategorie 2 als angemessen angesehen. Die vorgeschlagene Formel zur Berechnung der gesamten Kosten während der Lebensdauer eines Rechenzentrums der Kategorie 2 lautet:[1]

$$Gesamtkosten = Grundfläche \cdot 2.400 \frac{\$}{\mathrm{m}^2} + (Nennleistung\ der\ Stromversorgung) \cdot 11.000 \frac{\$}{\mathrm{kW}}$$

In diesen Kosten ist die vollständige Infrastruktur enthalten inkl. deren Stromverbrauch, nicht aber die zu installierende Rechentechnik oder deren unmittelbarer Stromverbrauch. (Letzterer wird in Abschnitt 2.1.3 betrachtet.) Da in die Formel auch die Grundfläche des Rechenzentrums einfließt, werden zur Abschätzung der Kosten pro CPU weitere Informationen benötigt. Um einen groben Richtwert zu bekommen, wird auf das in dem bereits erwähnten Papier [Turn06] angeführte realistische Beispiel eines Tier-2-Rechenzentrum zurückgegriffen. Für dieses ergaben sich über die Lebensdauer Gesamtkosten von 15,4 Millionen US-Dollar, wobei die Nennleistung 1.000 kW betrug. Die folgende Abschätzung stützt sich auf dieses Beispiel.[2] Dabei bleibt jedoch noch zu klären, mit wie vielen CPUs zu kalkulieren ist. Da sich die Dimensionierung der Stromversorgung eines Rechenzentrums typischerweise nicht an den real zu messenden Stromverbräuchen der Hardware orientiert, sondern eher an den angegebenen Nennleistungen, sind die im folgenden Abschnitt 2.1.3 angegebenen Werte zum realen Stromverbrauch für die hier erforderliche Umrechnung wenig hilfreich, so dass auf andere Informationen zurückgegriffen werden muss.

In [Knue05] wurde mit 250 W pro CPU gerechnet. Die Angaben für aktuell verkaufte Hardware sind jedoch meist etwas höher, so dass im Folgenden mit 300 W gerechnet

---

[1] Es sei an dieser Stelle darauf hingewiesen, dass Grid-Jobs natürlich auch auf Ressourcen ausgeführt werden können, die in einem Rechenzentrum der Kategorie 3 oder 4 stehen. Diese Variante ist jedoch nur dann sinnvoll, wenn es wirklich nur um eine zusätzliche Auslastung dieser Ressourcen geht, also nicht um speziell für das Grid-Computing reservierte Ressourcen. Dementsprechend wären in einem solchen Fall aber gar keine Kosten für die Infrastruktur zu kalkulieren, da deren Kosten durch das Grid-Computing nicht erhöht würden.

Bei der möglichen Verwendung eines Rechenzentrums der Kategorie 1 würden sich nur geringfügig andere Kosten ergeben als bei einem der Kategorie 2, so dass die angegebene Formel auch für eine Abschätzung dieses Falls geeignet ist.

[2] Das Beispiel dient also dazu, einen realistischen Wert für das Verhältnis zwischen der Grundfläche und der Nennleistung der Stromversorgung zu bekommen.

wird. Damit entsprechen die 1000 kW des angegebenen Beispiels einer Zahl von 3333 CPUs, so dass sich pro CPU Gesamtkosten von 4620 $ für die Infrastruktur über deren Lebenszeit hinweg ergeben. In [Patt07] wird von einer typischen Nutzungsdauer von 15 Jahren ausgegangen. Ähnliche Werte werden in [HP07] (10 bis 15 Jahre) und in [Anse06] (etwa 20 Jahre) angenommen. Im Weiteren soll daher von einer typischen Spanne der Nutzungsdauer von 10 bis 20 Jahren ausgegangen werden. Dann ergeben sich jährliche Kosten pro CPU von 231 bis 462 US-Dollar bzw. umgerechnet 165 bis 330 Euro.[1] Diese Kosten sind nicht einzuberechnen, wenn die Ressourcen auch ohne Grid-Computing diese Infrastruktur benötigen.

2.1.3. Stromverbrauch der Rechentechnik

Zur Abschätzung des Stromverbrauchs für das Grid-Computing ist insbesondere der Unterschied zwischen dem Stromverbrauch im unbelasteten Zustand (Idle-Zustand) und dem bei laufender Grid-Anwendung von Interesse, da bei ohnehin laufenden Systemen nur diese Differenz dem Grid-Computing zuzurechnen ist.

Die hier angegebenen Werte stützen sich im Wesentlichen auf [Hübn04]. In dieser Studie wurden Computersysteme mit unterschiedlichen Prozessoren bzgl. ihres Energieverbrauchs miteinander verglichen. Zum einen sind die Werte für den unbelasteten Zustand aufgeführt, zum anderen für den Fall, dass die Seti@home-Software läuft ([Skaw02]). Die Ergebnisse sind in Abbildung 2-2 dargestellt. Es wird deutlich, dass die untersuchten Systeme auch im unbelasteten Zustand einen erheblichen Energiebedarf aufweisen.

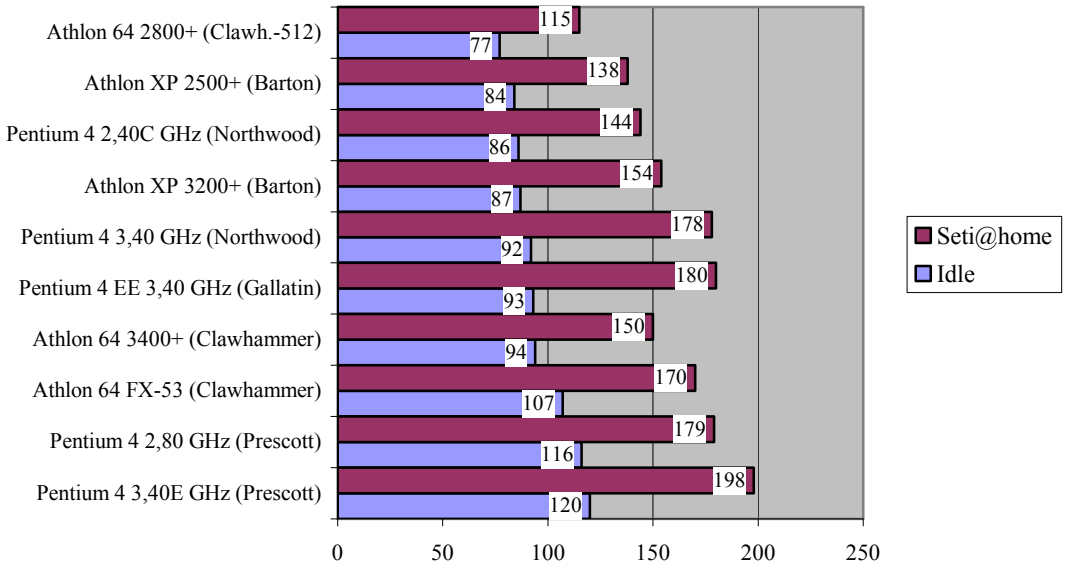

Abbildung 2-2: Stromverbrauch von Computersystemen mit unterschiedlichen Prozessoren in Watt ([Hübn04])

[1] In dieser Arbeit wird mit einem Kurs zwischen Euro und Dollar von 1:1,4 gerechnet.

Für die schließlich vorzunehmende Abschätzung der Energiekosten eines Grids muss geprüft werden, in welchen Zeiträumen die Ressourcen ohnehin angeschaltet sind und in welchen sie exklusiv für das Grid-Computing betrieben werden. Benötigt werden weiterhin die Kosten pro kWh. Bei diesen ist zu unterscheiden zwischen den Kosten für private Nutzer und den Kosten für die Industrie. Während im ersten Fall innerhalb der EU durchschnittlich 0,15 € zu bezahlen sind, fallen für die Industrie nur durchschnittliche Kosten von 0,09 € an ([Euro06]).

2.1.4. Kosten für die Software

Für das Betreiben eines Grids ist eine entsprechende Middleware-Software erforderlich.[1] Dabei kommen eine Reihe von Möglichkeiten in Betracht. Eine Variante ist eine Eigenentwicklung, die jedoch in den wenigsten Fällen sinnvoll sein dürfte. Relevanter sind die beiden Fälle der Nutzung kommerzieller oder freier Software. Im zweiten Fall ergeben sich zunächst keine unmittelbaren Kosten für die Beschaffung. Allerdings ist ein Service-Vertrag nicht zum Nulltarif zu bekommen. Ferner ist die Nutzung von Open-Source-Software häufig schwieriger und damit arbeitsaufwendiger, was insbesondere auch für die bekannteste Open-Source-Grid-Plattform, das Globus Toolkit, gilt ([Math07], [Plas06], [Rica05]). Da sich der zusätzliche Arbeitsaufwand nur außerordentlich schwer beziffern lässt, wird hier nur auf die Kosten bei der Verwendung kommerzieller Middleware eingegangen. Auch dafür soll im Rahmen dieser Arbeit keine allgemeine Betrachtung durchgeführt werden. Stattdessen wird exemplarisch die N1 Grid Engine 6 von Sun betrachtet ([Bulh04], [Sun02]). Diese bietet den Grid-Nutzern den Zugriff auf Rechner, die über verschiedene Organisationen verteilt sein können. Neben dem reinen Zugriff für die Grid-Nutzer werden ferner Funktionen zur Administration bereitgestellt. Mit diesen lässt sich bspw. einstellen, in welchem Maße die involvierten Ressourcen durch die Grid-Jobs belastet werden dürfen, damit die primären Nutzer nicht in ihrer Arbeit gestört werden.

Die N1 Grid Engine benötigt neben den ausführenden Prozessoren mindestens einen Master-Host, der für die Verteilung der Jobs verantwortlich ist. Ein solcher Master-Host kann dabei für bis zu etwa 10.000 Prozessoren verantwortlich sein. Die Lizenz-Kosten für die N1 Grid Engine sind in Tabelle 2-3 aufgelistet. Diese Kosten fallen dabei nur an, wenn die Software mit einem Service-Vertrag erworben wird. Da anderenfalls aber anderweitig Kosten für die nötige Weiterbildung und Arbeitszeit der Mitarbeiter anfallen, wird hier mit den angegebenen Kosten gerechnet.

Nicht ganz klar ist jedoch, auf welchen Zeitraum der Nutzung sich die angegebenen Kosten beziehen. Zum einen könnte es sein, dass der mit dem Erwerb der Software kombinierte Service-Vertrag bspw. auf ein Jahr begrenzt ist. Zum anderen wird Software typischerweise ohnehin nur für eine sehr begrenzte Zeit genutzt. Genaue Angaben

[1] Auf die Kosten für die Anwendungsentwicklung soll hier nicht eingegangen werden, da hier ausschließlich die Kosten des Ressourcen-Anbieters betrachtet werden. Insbesondere wird hier also nicht untersucht, inwieweit sich die Kosten für die Anwendungssoftware im Falle des Grid-Computings gegenüber herkömmlichen Szenarien verändern.

zu typischen Nutzungszeiten einer Software lassen sich jedoch praktisch nicht finden. Lediglich in betriebswirtschaftlicher Literatur finden sich gelegentlich Angaben für typische Nutzungsdauern. Beispielsweise wird in [Löw05] von 3 bis 5 Jahren ausgegangen, die dann auch für die Abschreibung anzusetzen ist. Inwieweit derartige Angaben jedoch mit der Realität übereinstimmen, bleibt eher unklar.

Max. Prozessorzahl	Anzahl der Master-Hosts	Kosten [$]
50	1	10.000
250	1	30.000
2.000	1	80.000
10.000	1	150.000
120.000	unbeschränkt	600.000

Tabelle 2-3: Lizenzkosten für die N1 Grid Engine ([Sun06])

Neben den Lizenzkosten fallen ggf. weitere Kosten für die benötigte Hardware oder deren Aufrüstung an. Als Minimalanforderungen für den Master-Host sind 80 MB freier Hauptspeicher und 100 MB freie Plattenkapazität angegeben. Für die anderen Prozessoren sind entsprechende Werte von 20 MB für den Hauptspeicher bzw. 50 MB für die Festplatte angegeben. In den meisten Fällen dürften diese Kosten jedoch weitgehend vernachlässigbar sein.

2.1.5. Kosten für Administration, Service und Support

Die Kosten für Administration, Service und Support sind im Wesentlichen Personalkosten. Im Unterschied zu den anderen hier betrachteten Kostenfaktoren hängen Personalkosten außerordentlich stark vom Land ab, in dem die entsprechenden Arbeiten durchgeführt werden. Eine allgemeingültige Betrachtung dieser Kosten soll daher im Rahmen dieser Arbeit nicht durchgeführt werden. Stattdessen wird exemplarisch die Situation in Deutschland betrachtet, um zumindest Anhaltspunkte zu haben, welche Punkte bei einer entsprechenden Abschätzung zu berücksichtigen sind.

Eine Übersicht über Gehälter im IT-Bereich in Deutschland wird in [Apfe07] gegeben. Für die hier diskutierten Bereiche Administration sowie Service und Support sind die ermittelten Durchschnittsgehälter in Abbildung 2-3 und Abbildung 2-4 dargestellt. Die durchschnittlichen jährlichen Bruttolöhne im Bereich der Administration lagen 2006 also zwischen 38.500 € und 55.400 €, während sie im Bereich von Service und Support zwischen 24.600 € und 66.200 € lagen.

Die für den Arbeitgeber anfallenden Kosten liegen deutlich über den Bruttolöhnen. Ausgehend von den Bruttolöhnen können die Gesamtkosten für den Arbeitgeber grob abgeschätzt werden, indem die Bruttolöhne mit einem Faktor von 1,8 multipliziert werden ([Sess03], [Pech05]). Aus den hier ermittelten Durchschnittslöhnen ergeben sich daher pro Mitarbeiter Kosten zwischen 70.000 € und 100.000 € für den Bereich der Administration, während beim Service und Support die entsprechenden Kosten zwischen 45.000 € und 120.000 € liegen.

Abbildung 2-3: Durchschnittliche Gehälter in Euro im Jahr 2006 im Bereich der Administration in Deutschland ([Apfe07])

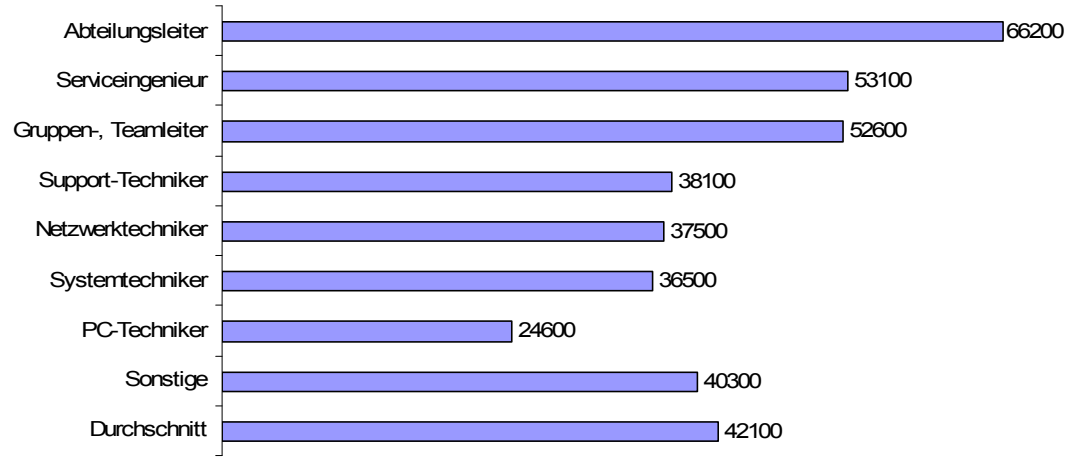

Abbildung 2-4: Durchschnittliche Gehälter in Euro im Jahr 2006 im Bereich von Service und Support in Deutschland ([Apfe07])

Die hier dargelegten Punkte zu den Personalkosten für die Administration sowie den Service und Support stellen nur einen Ansatzpunkt dar, um in konkreten Projekten die Kosten abschätzen zu können. Im Rahmen dieser Arbeit wird dieser Ansatz jedoch nicht weiter fortgeführt. Neben den oben genannten Problemen der fehlenden Allgemeingültigkeit für unterschiedliche Länder liegt dies daran, dass für die in Abschnitt 2.2 diskutierten realen Projekte andere Möglichkeiten der Schätzung dieser Kosten bestehen.

2.1.6. Kosten für die Verbindung zum Internet

Der letzte hier betrachtete Kostenfaktor ist die Verbindung zum Internet.[1] Um die entsprechenden Kosten auch für größere Organisationen abschätzen zu können, werden die Preise von Internet-Verbindungen mit Datenraten oberhalb gewöhnlicher xDSL-Anschlüsse benötigt. Mangels anderer Zahlen wird auf die Angebote des Betreibers des deutschen Forschungsnetzes (DFN) zurückgegriffen. Die Angebote, so wie sie 2007 galten ([DFN07]), sind in Tabelle 2-4 zusammengefasst.

Übertragungsrate [Mbit/s]	Preis pro Jahr [€]
10	17.800
20	25.900
100	62.300
200	103.800
1.000	285.400
2.000	389.200
10.000	570.900

Tabelle 2-4: Preise für Internetverbindungen beim Deutschen Forschungsnetz DFN ([DFN07])

Für eine Internetverbindung mit bestimmter Datenübertragungsrate ist anzumerken, dass die Kosten bzw. Preise stark fallend sind, so dass kein Preis genannt werden kann, der über mehrere Jahre hinweg auch nur in etwa stimmt. Aus diesem Grunde wird im Weiteren nicht mit den Kosten für eine bestimmte Übertragungsrate gearbeitet, sondern mit den Kosten pro CPU und Jahr. Die diesbezüglich anzusetzenden Netzwerkkosten sind deutlich stabiler, da die fallenden Kosten für eine bestimmte Übertragungskapazität zumindest ungefähr dadurch ausgeglichen werden, dass die geschalteten Leitungen von Jahr zu Jahr leistungsfähiger werden, also eine immer größere Übertragungskapazität aufweisen. Um die typischen Kosten für die Internetverbindung pro CPU ermitteln zu können, wurde nach entsprechenden Angaben zu den Netzwerkverbindungen von Rechenzentren gesucht.

Die erste Angabe stammt vom Grid Computing Centre Karlsruhe (GridKa), welches zum LHC-Grid gehört, das Rechenleistung für die Experimente am CERN (Europäische Organisation für Kernforschung) bereitstellt. Im Jahr 2007 beherbergte das GridKa 3220 CPUs ([Grid07]). Diese waren mittels vier 10-Gbit/s-Leitungen und vier kleineren Leitungen mit anderen Rechenzentren verbunden. Wenn hier für die vier kleineren Leitungen eine Übertragungskapazität von jeweils 1 Gbit/s angenommen wird, dann ergeben sich entsprechend der in Tabelle 2-4 aufgeführten Zahlen jährliche Kosten von 3,4 Millionen Euro. Diese Kosten lassen sich umrechnen in 0,12 € pro CPU und Stunde.

[1] Es geht hier um die Verbindung der Ressourcen eines Standorts zu anderen Standorten. Theoretisch müsste dies nicht unbedingt eine Internet-Verbindung sein, allerdings ist eine Internet-Verbindung die typische Variante. Deshalb wird hier auch nur diese Variante betrachtet.

Ein weiteres Rechenzentrum, zu dem entsprechende Informationen gefunden werden konnten, gehört der Hetzner Online AG ([Hetz06]). Bei diesem handelt es sich um ein kommerzielles Rechenzentrum, das bis zu 3500 Server aufnehmen kann. Aufgrund fehlender weiterer Angaben wird hier diese Zahl als die Zahl der CPUs angenommen. Zwar ist damit zu rechnen, dass ein Server mehr als eine CPU beinhalten kann – andererseits sind Rechenzentren nur selten voll bestückt. Bei der hier getroffenen Annahme wird also unterstellt, dass sich die beiden Fehler zumindest in etwa aufheben. Das Rechenzentrum ist mit zwei 10-Gbit/s-Leitungen und sechs 1-Gbit/s-Leitungen mit der Außenwelt verbunden. Entsprechend der Informationen aus Tabelle 2-4 ergeben sich so jährliche Kosten von 2,9 Millionen Euro bzw. 0,09 € pro CPU und Stunde.

An dieser Stelle sei darauf hingewiesen, dass die hier vorgestellte Abschätzung der Kosten für die Internetverbindung so nur gültig ist, wenn die Verbindung exklusiv für das Grid-Computing verwendet wird, ohne das Grid-Computing also gar keine Verbindung zum Internet notwendig wäre. Falls dies in einem konkreten Szenario nicht zutreffend ist, dann sind für das Grid-Computing nur die entstehenden Zusatzkosten zu berechnen. Wenn z. B. ein Rechenzentrum eines großen Unternehmens ohnehin mit dem Internet verbunden sein muss, dann könnte das zusätzlich eingeführte Grid-Computing durchaus die Anforderungen an die Internet-Verbindung erhöhen. Doch selbst ein drastischer Anstieg von 1 Gbit/s auf 10 Gbit/s würde nur etwa 50 Prozent des Preises einer 10-Gbit/s-Leitung kosten, obwohl fast die gleiche zusätzliche Übertragungskapazität hinzukäme (vgl. Tabelle 2-4). Dementsprechend ist bei einer solchen geteilten Nutzung von Internet-Verbindungen mit deutlich geringeren Kosten als den oben angegebenen zu rechnen.

2.2. Kostenschätzung für reale Grid-Projekte

Im vorherigen Abschnitt wurden die verschiedenen Kostenfaktoren analysiert, die beim Grid-Computing eine Rolle spielen. Die Diskussion hat insbesondere aufgezeigt, wie bei der Abschätzung der entsprechenden Kosten vorgegangen werden kann. Es wurden ferner wichtige Informationen zu den konkreten Größen dieser Kostenfaktoren aufgeführt. Allerdings führte die Diskussion noch nicht zu Angaben zu den Gesamtkosten, die sich für einen Ressourcen-Anbieter beim Grid-Computing ergeben. Ein wesentlicher Grund dafür liegt darin, dass es sehr unterschiedliche Grids gibt, für die sich auch recht unterschiedliche Kosten ergeben können. Daher soll in diesem Abschnitt anhand ausgewählter realer Projekte erarbeitet werden, welche Gesamtkosten sich für unterschiedliche Typen von Grids ergeben.

Zum Zwecke einer möglichst guten Anschaulichkeit und Vergleichbarkeit der Werte werden die Gesamtkosten umgerechnet in Kosten pro CPU und Stunde. Mit dieser Angabe ist insbesondere auch ein Vergleich mit Marktpreisen möglich, da diese sich ebenfalls an dieser Größe orientieren. Auf entsprechende kommerzielle Angebote wird in Abschnitt 2.3 eingegangen.

Die im Folgenden präsentierten Abschätzungen realer Projekte weisen mehr oder weniger große Spannen für die möglicherweise entstandenen Kosten pro Stunde und CPU auf. Der Grund liegt vorwiegend darin, dass zu einer ganzen Reihe von Punkten keine genauen projektspezifischen Angaben vorlagen, sondern mit allgemeinen Werten (bzw. Spannen) gerechnet wurde. Die Nutzung solcher Spannen ist für das hier verfolgte Ziel, die beim Grid-Computing entstehenden Kosten abzuschätzen, aber nicht von Nachteil. Schließlich sind weniger die projektspezifischen Werte von Interesse als typische Werte für vergleichbare Grids. Für die Einschätzung dieser typischen Werte sind die ermittelten Spannen hilfreicher als Punktschätzungen für die betrachteten Projekte.

2.2.1. EGEE I

Das Projekt *Enabling Grids for E-SciencE* (EGEE) hat den Zweck, über verschiedene Forschungseinrichtungen in Europa verteilte Rechen-Ressourcen miteinander zu verknüpfen. Das Projekt wurde im Rahmen des 6. Rahmenprogramms der EU gefördert. Das Projekt gliedert sich in zwei Phasen. Die erste (EGEE I) lief von April 2004 bis März 2006, die zweite Phase (EGEE II) erstreckt sich über die darauf folgenden beiden Jahre. Das EGEE-Projekt ist kein Forschungsprojekt, sondern ein Projekt zur Bereitstellung eines Produktionsgrids für die Ausführung rechenintensiver Anwendungen aus der Forschung ([Gagl05]). Zu den Bereichen, die von EGEE profitieren, zählen u. a. die Astro-Physik und die Bio-Medizin.

In diesem Abschnitt geht es zunächst nur um die erste Phase des Projekts, also um EGEE I. Die Kostenabschätzung für die zweite Phase (EGEE II) wird in Abschnitt 2.2.2 vorgestellt.

Kosten für die Server-Hardware

Das EGEE-Projekt dient zwar der Verknüpfung von verteilt vorliegenden Ressourcen, finanziert jedoch nicht die Ressourcen selbst. Dementsprechend sind die Kosten für die Hardware nicht in den in [Cord07] genannten Gesamtkosten enthalten. Da die Hardware stattdessen durch die entsprechenden miteinander verknüpften Forschungseinrichtungen beschafft und bezahlt wurde, sind auch praktisch keine belastbaren Zahlen zu den konkret eingesetzten Ressourcen zu bekommen. Daher bleibt hier nur der Ausweg, die recht allgemeinen Kosten, so wie sie in Abschnitt 2.1.1 abgeschätzt wurden, anzusetzen. Dazu wird zunächst die durchschnittliche Zahl der CPUs benötigt, die im EGEE eingesetzt wurden. Laut [Bird07] entwickelte sich diese Zahl wie in Abbildung 2-5 dargestellt. Damit ergibt sich eine durchschnittliche Anzahl von 11.770 CPUs für das zwei Jahre laufende Projekt. Wird diese Zahl entsprechend der in Abschnitt 2.1.1 genannten Kosten pro CPU in jährliche Kosten umgerechnet, ergibt sich die breite Spanne von 706.000 € bis zu 23.500.000 € für die benötigten Neuanschaffungen.

Zu den so ermittelten Kosten sind die Kosten für das Ersetzen defekter Komponenten hinzuzurechnen. Entsprechend Abschnitt 2.1.1 werden hier zunächst nur die reinen Preise der zu ersetzenden Komponenten berücksichtigt, während die Arbeitskosten erst

mit den weiter unten diskutierten Administrationskosten berücksichtigt werden. Um die Komponentenkosten abzuschätzen, wird der Einfachheit halber angenommen, dass zu jeder CPU ein Mainboard, ein CPU-Kühler, eine Festplatte und ein Netzteil gehört. Die tatsächlichen Relationen werden mit ziemlicher Sicherheit von dieser Annahme abweichen. Dennoch dürfte die hier getroffene Annahme als Schätzgrundlage durchaus geeignet sein. Damit ergeben sich jährliche Kosten zwischen 121.000 € und 2.700.000 €. Wenn nun – wie in der EU zumindest im Privatbereich üblich – eine Gewährleistung von zwei Jahren angenommen wird, dann ist bei einer Nutzungsdauer von drei Jahren nur mit einem Drittel dieser Kosten zu rechnen. Bei einer Nutzungsdauer von fünf Jahren hingegen müssten 60 Prozent dieser Kosten angesetzt werden. In Summe ergibt sich damit eine Spanne für die jährlichen Hardware-Kosten von 780.000 € bis zu 42.000.000 €.

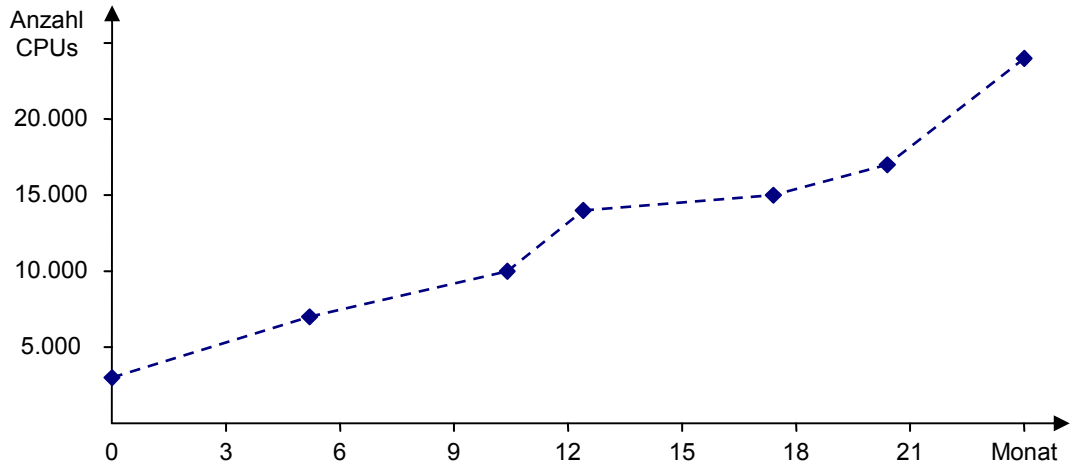

Abbildung 2-5: Rechen-Ressourcen in EGEE I ([Bird07])

Infrastrukturkosten

Da die im EGEE verwendeten Rechen-Ressourcen ausschließlich dem Grid-Computing dienen, sind auch die Kosten für die Infrastruktur zu berücksichtigen. Gemäß Abschnitt 2.1.2 ist dabei mit Kosten zwischen 165 € und 330 € pro CPU und Jahr zu rechnen. Für EGEE I ergibt dies jährliche Infrastrukturkosten zwischen 1,9 und 3,9 Millionen Euro.

Stromverbrauch der Rechentechnik

Die typischen Werte für den Stromverbrauch pro CPU liegen, wie in Abschnitt 2.1.3 diskutiert, bei laufenden Grid-Jobs zwischen 115 W und 198 W. Rechnet man dies auf das Jahr hoch und nimmt dabei eine Vollauslastung der Ressourcen an, ergibt sich ein Gesamtverbrauch pro CPU zwischen etwa 1000 kWh und 1700 kWh. Wird dafür der für die Industrie ermittelte Durchschnittspreis von 0,09 € pro kWh angewendet, ergeben sich Kosten zwischen 90 € und 156 € pro CPU und Jahr. Insgesamt sind damit für EGEE I jährliche Stromkosten zwischen 1,1 und 1,8 Millionen Euro zu veranschlagen.

Softwarekosten

Die Gesamtkosten von EGEE I belaufen sich auf 46 Millionen Euro ([Cord07]). Wie bereits ausgeführt, beinhalten diese Kosten weder die Kosten für die Hardware noch die für die Infrastruktur oder die Netzwerkanbindung. Stattdessen beziehen sich diese Kosten im Wesentlichen auf die anfallenden Personalkosten zum Betreiben des Grids mit bereits existierenden Ressourcen. Die Aufgaben des involvierten Personals wurden dabei in drei große Bereiche eingeteilt, die als Netzwerk-, Service- und gemeinsame Forschungsaktivitäten bezeichnet werden ([EGEE08a], siehe auch Abbildung 2-6). Zur Ermittlung der Softwarekosten im Falle von EGEE ist der letzte Bereich von Interesse, da er der Entwicklung und Integration der Middleware-Software dient. Gemäß [Jone05] entspricht dieser Bereich 24 Prozent des Budgets von EGEE I, mithin 11 Millionen Euro. Damit ist mit jährlichen Softwarekosten von etwa 5,5 Millionen Euro zu rechnen.

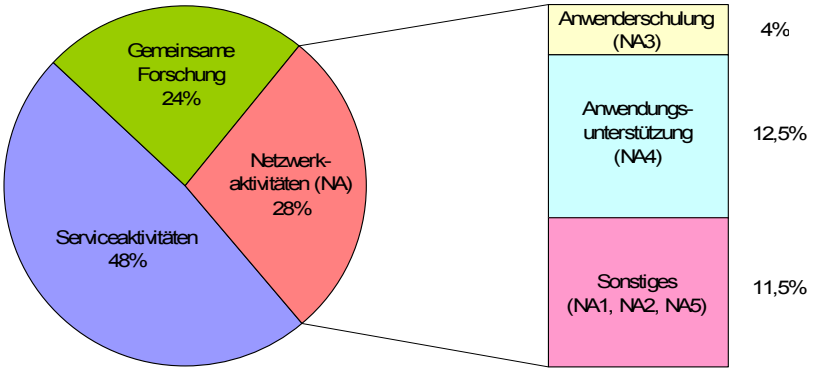

Abbildung 2-6: Budgetaufteilung in EGEE I ([Jone05])

Kosten von Administration, Service und Support

Zur Abschätzung der Kosten für Administration, Service und Support sind die Bereiche der Netzwerk- und der Service-Aktivitäten von Interesse. Letzterer Bereich dient dem Betreiben des Grids und ist folglich hier voll einzurechnen. Nach [Jone05] entfallen ca. 48 Prozent des Budgets auf diesen Bereich. Dies sind etwa 22 Millionen Euro – bzw. 11 Millionen Euro pro Jahr.

Etwas komplizierter ist die Betrachtung des Bereiches der Netzwerkaktivitäten, auf den 28 Prozent des Budgets entfallen. Dieser Bereich ist noch einmal unterteilt in weitere Teilaktivitäten NA1 bis NA5 (siehe Abbildung 2-6). NA1 dient dem Management und der Koordination sämtlicher Aktivitäten des EGEE ([EGEE08b]), während NA2 der Öffentlichkeitsarbeit dient ([EGEE08c]). Die Teilaktivität NA3 umfasst Tätigkeiten zur Anwenderschulung und -einführung ([EGEE08d]). NA4 dient der Suche nach geeigneten Anwendungen und auch der Unterstützung dieser ([EGEE08e]). Die letzte Teilaktivität, NA5, dient schließlich der Zusammenarbeit mit anderen Grid-Projekten ([EGEE08f]). Wenngleich die meisten dieser Aktivitäten auch in anderen Grid-Projekten in ähnlicher Form vorkommen und entsprechende Kosten verursachen, sind

diese jedoch nicht unbedingt dem Ressourcen-Anbieter zuzuschreiben. So gilt insbesondere für Anwenderschulungen und Hilfen für spezielle Anwendungen, dass diese üblicherweise explizit durch die Kunden zu bezahlen sind. Die entsprechenden Teilaktivitäten NA3 und NA4 werden daher hier nicht zu den Kosten für Administration, Service und Support hinzugerechnet. Gemäß [Jone05] entspricht NA3 4 Prozent des Budgets von EGEE I – und NA4 entspricht 12,5 Prozent des Budgets. Somit bleiben für NA1, NA2 und NA5 noch 11,5 Prozent von 46 Millionen Euro bzw. 5,3 Millionen Euro. Aufs Jahr gerechnet entspricht dies Kosten von 2,6 Millionen Euro.

Insgesamt ergeben sich damit für das EGEE-I-Projekt jährliche Kosten von etwa 13,6 Millionen Euro für Administration, Service und Support.

Kosten der Internetverbindungen

Das EGEE-Projekt nutzt für die Verbindung der Rechenzentren das europäische Forschungsnetz GEANT2 ([DANT06]). Die Kosten dieses Netzes sind jedoch nicht im Budget von EGEE I inbegriffen. Da die Kosten trotzdem den Ressourcen-Anbietern zuzurechnen sind, ist hier eine entsprechende Abschätzung der Kosten für diese Verbindungen erforderlich. Gemäß der Erkenntnisse aus Abschnitt 2.1.6 ergeben sich dabei Kosten zwischen etwa 0,09 € und 0,12 € pro CPU und Stunde, wenn die Verbindungen ausschließlich im Rahmen des EGEE verwendet werden. Dies ist beim EGEE aber nicht der Fall. So werden zwar die involvierten CPUs ausschließlich im Rahmen des EGEE-Projektes verwendet, nicht jedoch die Rechenzentren, in denen die entsprechenden Server stehen.

Dennoch ist für die im Rahmen von GEANT2 betriebenen Internetverbindungen explizit ausgeführt, dass die hohen Übertragungsraten insbesondere für neue Anwendungen (wie z. B. das Grid-Computing) benötigt werden ([DANT06]). Folglich können hier die Kosten für die Internet-Verbindungen auch nicht ignoriert werden. Mangels genauerer Angaben wird, basierend auf den Zahlen aus Abschnitt 2.1.6, mit der groben Schätzung gearbeitet, dass die Kosten für die notwendige Erhöhung der Übertragungskapazität nur bei etwa 50 Prozent entsprechender neuer Leitungen liegen. Mit dieser Annahme ergeben sich für EGEE I jährliche Kosten für die Internetverbindungen zwischen 4,6 und 6,1 Millionen Euro.

Gesamtkosten von EGEE I

Die ermittelten Kosten für EGEE I sind in Tabelle 2-5 zusammengefasst. Die Gesamtkosten entsprechen Kosten zwischen 0,27 € und 0,71 € pro CPU und Stunde, wenn davon ausgegangen wird, dass die Ressourcen das ganze Jahr über ausgelastet sind. Laut [Bird07] ist diese Annahme jedoch nicht annähernd erfüllt. Vielmehr lag die Auslastung nur bei etwa einem Drittel. Dies führt für die Zeiten, zu denen die Ressourcen tatsächlich genutzt wurden, zu Kosten zwischen 0,80 € und 2,12 €.[1]

[1] Diese Werte ergeben sich durch Multiplikation der für den Fall der Vollauslastung ermittelten Kosten mit dem Faktor 3, da nur für ein Drittel der CPU-Stunden entsprechende Grid-Nutzer existieren, denen die Kosten in Rechnung gestellt werden könnten. Die Multiplikation mit 3 ist dabei nicht ganz exakt, da zumindest die Kosten

Kostenfaktor	Kosten pro Jahr (€)
Hardware	780.000 – 42.000.000
Infrastruktur	1.900.000 – 3.900.000
Stromverbrauch der Server	1.100.000 – 1.800.000
Software	5.500.000
Administration, Service und Support	13.600.000
Verbindung zum Internet	4.600.000 – 6.100.000
Summe	**27.500.000 – 73.000.000**

Tabelle 2-5: Kosten im EGEE I

Die erhebliche Breite der ermittelten Spanne der Kosten ist im Wesentlichen der nur sehr groben Schätzung der Hardwarekosten geschuldet. Wie bereits ausgeführt, kann hier daher keine genauere Schätzung für die Kosten im Falle des EGEE abgegeben werden. Als Abschätzung für die möglichen Kosten in vergleichbaren Grids ist eine solche Spanne jedoch hilfreich.

2.2.2. EGEE II

Nach der Schätzung für EGEE I wird hier eine Abschätzung der Kosten für die zweite Projektphase durchgeführt. Der Hintergrund für diese zusätzliche Abschätzung ist der, dass in EGEE I möglicherweise besonders hohe Kosten entstanden sind, da das Projekt nicht zuletzt aufgrund der Größe einen gewissen Neuigkeitswert hatte. Aus diesem Grunde liefert die Abschätzung für die zweite Phase möglicherweise repräsentativere Werte. Die Ausführungen in diesem Abschnitt fallen gegenüber denen in Abschnitt 2.2.1 jedoch kürzer aus, da die meisten der Abschätzungen in analoger Weise, nur mit anderen Zahlen durchgeführt wurden. Da zum Zeitpunkt des Schreibens EGEE II jedoch noch nicht abgeschlossen war, basiert die hier durchgeführte Kostenschätzung nur auf den ersten 19 Monaten von EGEE II.

Kosten für die Server-Hardware

Zur Abschätzung der Hardware-Kosten wird wieder die durchschnittliche Zahl von CPUs im Grid benötigt. Die Entwicklung dieser Zahl ist in Abbildung 2-7 dargestellt. Die durchschnittliche Zahl von CPUs liegt bei 37.420. Unter Anwendung des gleichen Schätzverfahrens wie bei EGEE I (vgl. Abschnitt 2.2.1) ergibt sich für die Hardware eine Spanne der jährlichen Kosten von 2,5 bis zu 134 Millionen Euro. Diese Kosten schließen die Preise für zu ersetzende defekte Komponenten ein.

für den Stromverbrauch und für den Ersatz defekter Komponenten sich leicht verringern würde. Eine genauere Berechnung würde die Ergebnisse jedoch nur geringfügig verändern.

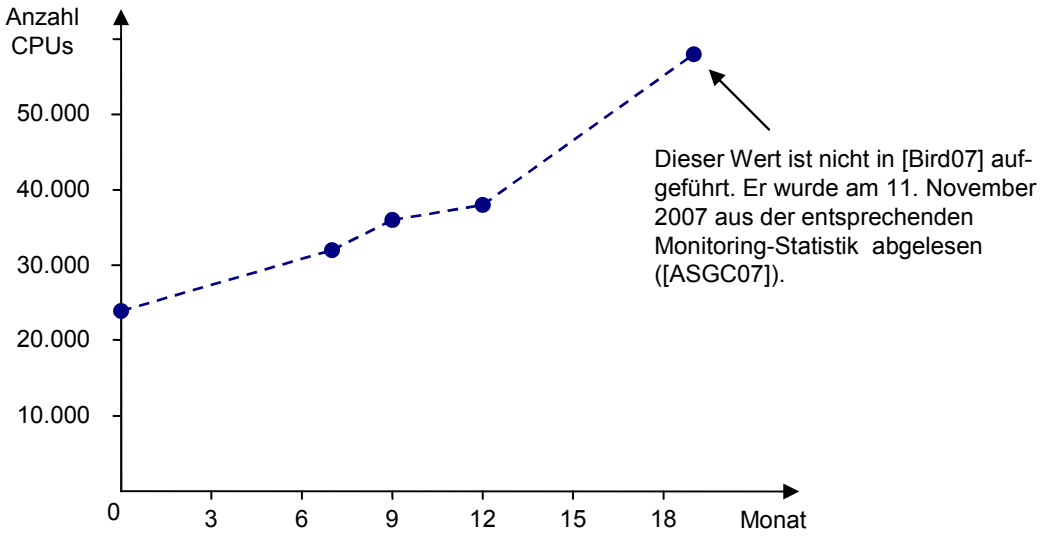

Abbildung 2-7: Rechen-Ressourcen in EGEE II ([Bird07])

Infrastrukturkosten

Mit analoger Rechnung wie beim EGEE I ergeben sich jährliche Infrastrukturkosten zwischen 6,2 und 12,3 Millionen Euro.

Stromverbrauch der Rechentechnik

Bei angenommenen Stromkosten zwischen 90 und 156 Euro pro CPU und Jahr (vgl. Abschnitt 2.2.1) sind jährliche Stromkosten von 3,4 bis 5,8 Millionen Euro anzusetzen.

Softwarekosten

Die Gesamtkosten von EGEE II belaufen sich auf 53 Millionen Euro ([Cord07a]). Die Entwicklung der Middleware-Software entspricht wiederum dem so genannten Bereich der gemeinsamen Forschungsaktivitäten. Auf diese entfielen in EGEE II nur noch 14 Prozent des Gesamtbudgets ([Loom07]), also 7,4 Millionen Euro. Dies entspricht jährlichen Kosten von 3,7 Millionen Euro für die Middleware-Software.

Kosten von Administration, Service und Support

Die Kosten für Administration, Service und Support finden sich auch beim EGEE II in den Bereichen Netzwerk- und Service-Aktivitäten wieder. Die Kosten von letzteren entsprechen etwa 54 Prozent des Budgets von EGEE II und sind vollständig bei der Abschätzung der hier betrachteten Kosten von Administration, Service und Support einzubeziehen.

Für die Netzwerkaktivitäten ist wiederum eine genauere Aufschlüsselung in die Teilaktivitäten erforderlich. Grundsätzlich ist die Unterteilung dabei genauso wie in EGEE I vorgenommen worden, lediglich die genauen Gewichtungen sind verändert. Auf die hier

relevanten Teilaktivitäten NA1, NA2 und NA5 entfallen in etwa 11 Prozent des Budgets von EGEE II, so dass für Administration, Service und Support mit insgesamt 65 Prozent des Budgets von EGEE II zu rechnen sind. Dies entspricht 34,5 Million Euro – bzw. 17,2 Millionen Euro jährlich.

Kosten der Internet-Verbindungen

Den gleichen Annahmen wie in Abschnitt 2.2.1 folgend ergeben sich für die Kosten der Internetverbindungen jährliche Kosten zwischen 14,6 und 19,4 Millionen Euro.

Gesamtkosten von EGEE II

Die Kosten des EGEE-II-Grids sind in Tabelle 2-6 zusammengefasst. Insgesamt ergibt sich für die jährlichen Kosten eine Spanne von 47,6 bis zu 192 Millionen Euro. Dies entspricht etwa 0,15 € bis 0,59 € pro CPU und Stunde bei angenommener Vollauslastung. Für die Annahme einer solchen vollständigen Auslastung gibt es jedoch keinen Grund. Auch wenn keine eindeutigen Zahlen zur durchschnittlichen Auslastung der Ressourcen von EGEE II gefunden werden konnten, so deuten aktuelle Daten aus der Monitoring-Statistik ([ASGC07]) auf eine in etwa gleich gebliebene Auslastung von einem Drittel hin. Diese berücksichtigend ergibt sich für die Zeiten der tatsächlichen Ressourcennutzung eine Spanne von 0,44 € bis zu 1,76 € pro CPU und Stunde.

Kostenfaktor	Kosten pro Jahr (€)
Hardware	2.500.000 – 134.000.000
Infrastruktur	6.200.000 – 12.300.000
Stromverbrauch der Server	3.400.000 – 5.800.000
Software	3.700.000
Administration, Service und Support	17.200.000
Verbindung zum Internet	14.600.000 – 19.400.000
Summe	**47.600.000 – 192.000.000**

Tabelle 2-6: Kosten im EGEE II

Beim Vergleich der ermittelten Kosten von EGEE I mit denen von EGEE II fällt auf, dass im zweiten Fall die Kosten pro CPU und Stunde tatsächlich niedriger sind. Wie bereits angesprochen, ist dies vermutlich darauf zurückzuführen, dass bei EGEE I noch eher Neuland betreten wurde als bei EGEE II. Für eine Abschätzung für aktuelle oder künftige Projekte dürften daher die Kosten von EGEE II eher geeignet sein.

2.2.3. Novartis

Ein aus dem kommerziellen Sektor stammendes Beispiel für ein Grid ist das von Novartis. Novartis ist ein weltweit agierendes Schweizer Pharmaunternehmen, das 2003 mit dem Grid-Computing begonnen hat. Das Ziel besteht darin, umfangreiche Simulationen zur Suche nach neuen Wirkstoffen für Medikamente durchzuführen. Zu diesem Zweck wurden zunächst 2700 PCs (mit Pentium-4-Prozessoren) eingebunden, die über ver-

schiedene Standorte verteilt waren (Basel, Wien, Cambridge (USA)). Auch wenn Planungen zur Erweiterungen dieses Grids bestanden (und vermutlich auch umgesetzt wurden), soll im Weiteren nur der Stand von 2003 betrachtet werden.

Kosten für die Server-Hardware

Die im Grid von Novartis eingesetzten Ressourcen waren Rechner, die ohnehin für die Arbeitsplätze der Mitarbeiter benötigt wurden. Dementsprechend sind die Anschaffungskosten hier nicht in die Kostenabschätzung einzubeziehen. Somit müssen nur die Kosten für die zu ersetzenden defekten Komponenten berücksichtigt werden. Zur Abschätzung dieser Kosten wird – wie auch bei der Abschätzung für das EGEE – davon ausgegangen, dass die CPUs, CPU-Kühler, Mainboards, Netzteile und Festplatten in gleicher Anzahl vertreten sind. Unter dieser Annahme ergeben sich die in Tabelle 2-7 dargestellten Ergebnisse.

Komponente	Anzahl im Grid von Novartis	Ersetzungskosten pro Jahr (€)
Intel Pentium 4	2.700	8.200 – 73.000
CPU-Kühler	2.700	10.000 – 27.000
Mainboard	2.700	9.600 – 59.000
Netzteil	2.700	5.900 – 24.000
Festplatte	2.700	1.400 – 11.000

Tabelle 2-7: Kosten für zu ersetzende Komponenten im Grid von Novartis

Wird wiederum eine zweijährige Gewährleistung angenommen, dann muss für eine Nutzungsdauer von drei Jahren nur mit einem Drittel dieser Kosten gerechnet werden. Bei einer Nutzungsdauer von fünf Jahren hingegen ist mit 60 Prozent der Kosten zu rechnen. Insgesamt ergibt sich damit für die hier betrachteten Kosten durch defekte Komponenten eine Spanne von etwa 12.000 € bis zu etwa 120.000 €.

Infrastrukturkosten

Aufgrund der Tatsache, dass in dem hier betrachteten Grid normale Arbeitsplatzrechner verwendet werden, die ohnehin angeschafft worden wären, ist von keinen zusätzlichen Kosten für die Infrastruktur auszugehen.

Stromverbrauch der Rechentechnik

Beim Stromverbrauch ist ebenfalls zu berücksichtigen, dass die Ressourcen nicht exklusiv für das Grid-Computing betrieben werden, sondern auch als Arbeitsplatzrechner genutzt werden. Dementsprechend ist zu ermitteln, um welchen Betrag sich die Stromkosten durch das Grid-Computing erhöhen (vgl. Abschnitt 2.1.3). Zu diesem Zweck ist zunächst eine Abschätzung vorzunehmen, welche Zeiten die Rechner auch ohne das Grid angeschaltet wären. Dazu kann für einen normalen Arbeitstag eine Nutzungsdauer von etwa 10 Stunden angesetzt werden, wenn man davon ausgeht, dass die Rechner nicht für die Arbeitspausen heruntergefahren werden. Rechnet man weiter mit etwa 220 Arbeitstagen im Jahr, so ergibt sich eine Gesamtzeit von 2200 Stunden, während der die

Rechner auch ohne Grid-Computing angeschaltet wären. Entsprechend der Ausführungen in Abschnitt 2.1.3 ergeben sich für diese Zeit pro CPU zusätzliche Stromkosten zwischen 12 € und 17 €, wenn ein Preis von 0,09 € pro kWh angesetzt wird. Für die restliche Zeit des Jahres (6560 Stunden) ist mit weiteren Kosten pro CPU im Bereich von 85 € bis zu 117 € zu errechnen.

Insgesamt ist damit für die Stromkosten pro CPU und Jahr eine Spanne von 97 € bis zu 134 € anzunehmen, was Kosten zwischen 260.000 € und 360.000 € für das gesamte Grid entspricht.

Softwarekosten

Im Novartis-Grid wurde die GridMP-Middleware von United Devices eingesetzt. Laut [Inte03] betrugen die Lizenzkosten 400.000 $. Entsprechend des hier angewendeten Umrechungskurses ergibt dies Kosten von etwa 285.000 €. Leider ist den Angaben nicht zu entnehmen, ob diese Kosten jährlich anfallen oder nur einmalig. Ferner ist für letzteren Fall unklar, wie lange die Software im Einsatz war. Aus diesen Gründen muss hier mit einer entsprechend großen Spanne gerechnet werden, um die möglichen Fälle abzudecken. Diese geht von einem Jahr bei jährlichen Lizenzkosten bis zu fünf Jahren als angenommene Obergrenze für eine sinnvolle Nutzungsdauer (vgl. Abschnitt 2.1.4). Insgesamt ist daher mit jährlichen Kosten für die Software zwischen 60.000 € und 285.000 € zu rechnen.

Kosten von Administration, Service und Support

Für die Kosten von Administration, Service und Support im Grid von Novartis ließen sich in der Literatur keine entsprechenden Angaben finden. Um dennoch zu einer sinnvollen Schätzung zu gelangen, wird ein Personalbedarf angenommen, der dem des EGEE-Projektes proportional ist. Es bleibt jedoch zu klären, welche der beim EGEE-Projekt einbezogenen Aktivitäten auch bei Novartis einzuberechnen sind. So dürften für die tatsächlich im Grid von Novartis aufgetretenen Kosten die Aktivitäten aus dem Bereich der Netzwerkaktivitäten im Grunde nicht unbedingt relevant sein, da das Grid nur Novartis-intern betrieben wird. Da es hier jedoch weniger um die konkreten Kosten im Falle des Novartis-Grids geht, sondern eher um möglichst repräsentative Werte für reale Grids, wird dennoch mit den vollen Kosten für Administration, Service und Support wie im EGEE-Projekt gerechnet. Im Falle von EGEE I handelt es sich dabei um jährlich 13,6 Millionen Euro für das Betreiben von 11.770 CPUs. Als proportionaler Wert für das Grid von Novartis ergeben sich damit 3,1 Millionen Euro. Eine analoge Rechnung für das EGEE II hingegen führt nur zu etwa 1,2 Millionen Euro pro Jahr für Administration, Service und Support.

Kosten der Internet-Verbindungen

Laut [Inte03] war der Anstieg der Netzwerkbelastung nur außerordentlich gering, so dass im Falle von Novartis keine Veränderung an der Verbindung zum Internet notwendig war – jedenfalls nicht wegen des eingeführten Grid-Computings. Wesentlicher Grund dafür dürfte jedoch die Art der Jobs sein, die bei Novartis anfallen und vermut-

lich ein sehr günstiges Verhältnis zwischen Rechenzeit und zu übertragenden Datenmengen aufweisen. Da dies jedoch im Grunde kein Merkmal des Rechen-Grids, sondern eines der ausgeführten Grid-Anwendungen ist, soll hier zum Vergleich auch der Fall betrachtet werden, dass in dem Grid von Novartis auch solche Anwendungen ausgeführt werden würden, die eine Verbesserung der Internetverbindungen erfordern würden. Den gleichen Annahmen wie beim EGEE folgend würden sich für diesen Fall zusätzliche Kosten im Bereich von 1,1 und 1,5 Millionen Euro pro Jahr ergeben.

Gesamtkosten des Grids von Novartis

Die in den vorherigen Abschnitten ermittelten Kosten für das Grid von Novartis sind in Tabelle 2-8 zusammengefasst. Die dabei ermittelten Gesamtkosten führen zu einem Bereich von 0,06 € bis zu 0,16 € pro CPU und Stunde für den Fall, dass keine zusätzlichen Kosten für die Internetverbindungen einberechnet werden. Für den anderen Fall ergibt sich eine entsprechende Spanne von 0,11 € bis zu 0,23 €. Die angegebenen Werte gelten dabei nur für den Fall einer vollständigen Auslastung der Rechner. Genaue Angaben zu dieser Auslastung sind jedoch in der Literatur nicht zu finden.

Kostenfaktor	Kosten pro Jahr (€)	
	ohne zusätzliche Netzwerkkosten	mit zusätzlichen Netzwerkkosten
Hardware	12.000 – 120.000	
Infrastruktur	–	
Stromverbrauch der Server	260.000 – 360.000	
Software	60.000 – 285.000	
Administration, Service und Support	1.200.000 – 3.100.000	
Verbindung zum Internet	–	1.100.000 – 1.500.000
Summe	**1.500.000 – 3.900.000**	**2.600.000 – 5.400.000**

Tabelle 2-8: Kosten im Novartis-Grid

2.3. Kommerzielle Angebote

Zum Vergleich und auch zur Prüfung der Plausibilität der hier für reale Projekte abgeschätzten Kosten werden in diesem Abschnitt zusätzlich zwei kommerzielle Angebote zur Nutzung von Rechen-Hardware vorgestellt. Konkret wird dabei auf die Angebote von Sun und Amazon eingegangen.

2.3.1. Angebot von Sun

Das Grid-Angebot von Sun wurde im März 2006 gestartet ([Comp06]). Begonnen wurde das Grid mit 5000 AMD-Opteron-Prozessoren, mittlerweile sind jedoch deutlich mehr Rechen-Ressourcen integriert. Die Ressourcen werden dabei ausdrücklich nicht eingesetzt, um Anwendungen von Sun auszuführen. Insbesondere laufen die Simula-

tionen für Suns Prozessorentwicklungen auf anderen Rechnern, da die entsprechende Software gar nicht für die x86-Architektur geschrieben wurde.

Die Nutzung des Sun-Grids ist so gestaltet, dass auf den Rechnern bereits Solaris 10 installiert ist (vgl. auch [Sun06b]). Die Grid-Programme müssen daher unter diesem Betriebssystem lauffähig sein. Für den Transport der auszuführenden Programme zu den Rechnern von Sun ist das Programmpaket zusammenzustellen, zu packen und übers Netz an Sun zu übertragen. Dann kann das entsprechende Programm ausgeführt werden, wobei dafür durchaus mehrere Prozesse erzeugt und auch auf verschiedene Prozessoren verteilt werden können. Nach dem Ende der Programmausführung kann von Seiten des Grid-Nutzers auf die Ergebnisse zurückgegriffen werden.

Als Preis für die Nutzung der Ressourcen ist ein US-Dollar pro CPU und Stunde angesetzt (0,71 € beim angesetzten Umtauschkurs von 1:1,4). Dieser Preis gilt zumindest für kurzfristige Buchungen. Bei längerfristigen Verträgen zur Nutzung dieser Ressourcen werden aufgrund der besseren Planbarkeit Rabatte von bis zu 50 Prozent gewährt. Zu zahlen ist für die beanspruchte Prozessorzeit. Dabei wird für jeden Job die Summe der Ausführungszeiten auf den genutzten Prozessoren gebildet und der Preis schließlich auf volle Dollar aufgerundet.

2.3.2. Angebot von Amazon

Amazon bietet mit der *Elastic Compute Cloude* ebenfalls Ressourcen an, auf denen rechenintensive Jobs ausgeführt werden können ([Amaz07], [Amaz07a]). Die einzelnen Systeme sind angegeben als äquivalent zu 1,7-GHz-x86-Prozessoren mit 1,75 GB Hauptspeicher, 160 GB Festplatte und einer 250-Mbps-Netzwerkverbindung. Die Ressourcen werden auch von Amazon selbst verwendet, um die täglich anfallenden Transaktionen abarbeiten zu können. Letztlich stellt damit Amazon also seine ohnehin angeschafften Ressourcen zur Verfügung, wenn sie zeitweise nicht benötigt werden.

Die Nutzung der Ressourcen läuft so ab, dass die Hardware unter vollständiger Kontrolle des Nutzers steht. Dies funktioniert so, dass die Rechner durch AMIs (Amazon Machine Images) konfiguriert werden. Diese AMIs beinhalten nicht nur die benötigten Anwendungsprogramme, sondern auch das Betriebssystem, wodurch die vollständige Beeinflussbarkeit der Ressource gewährleistet ist, die Ressource während der Zeit der Nutzung aber auch erst gebootet werden muss.

Das Preismodell von Amazon ist nicht ganz so einfach wie das von Sun. Es gibt einen Preis für die Nutzung eines Prozessors für eine Stunde. Dieser Preis beträgt momentan 0,10 $ (= 0,07 €), wobei die Nutzung jedes einzelnen Prozessors immer auf volle Stunden aufgerundet wird. Zusätzlich dazu fallen Kosten für den Transfer der Daten an, nämlich 0,10 $ pro GB an Eingabedaten und 0,18 $ pro GB an Ausgabedaten. Diese Angaben sind für einen Vergleich der Kosten mit anderen Grids aber nur bedingt hilfreich, weil dafür die Preise pro Stunde und CPU erforderlich sind. Um diese Preise abzuschätzen, werden „typische“ Werte für die Kommunikationskosten bei Grid-Jobs benötigt. Deshalb wird hier mit den Abschätzungen analog zu denen aus Abschnitt 2.2.1 gearbeitet. Demnach ist von zusätzlichen Kosten zwischen 0,04 € und 0,06 € pro CPU

und Stunde auszugehen. Insgesamt ergibt sich damit bei der Nutzung des Grids von Amazon ein Preisspanne von etwa 0,11 € bis zu 0,13 € für die Nutzung einer CPU für eine Stunde.

2.3.3. Vergleich der Marktpreise mit den abgeschätzten Kosten

Da die Betrachtung der Marktpreise u. a. auch deshalb durchgeführt wurde, um die erzielten Abschätzungen für die Kosten eines Ressourcen-Anbieters auf Plausibilität zu prüfen, ist hier ein Vergleich zwischen diesen Kosten und den Preisen notwendig. Zu diesem Zweck sind die Ergebnisse der vorangegangenen Abschnitte in Tabelle 2-9 zusammengefasst. Dabei sind die Projekte eingeteilt in solche, bei denen die Ressourcen primär einem anderen Zweck als dem Grid-Computing dienen, und solche, bei denen die Ressourcen exklusiv für das Grid-Computing verwendet werden.

Ressourcen-Nutzung	Grid-Projekt	Kosten pro Stunde und CPU (€)
gemeinsam (im Grid und für andere Zwecke)	Amazon	0,11 – 0,13
	Novartis	0,06 – 0,16 inkl. Internetanbindung: 0,11 – 0,23
exklusiv (nur im Grid)	Sun	0,71
	EGEE I	0,27 – 0,71 Berücksichtigung der Auslastung: 0,80 – 2,12
	EGEE II	0,15 – 0,59 Berücksichtigung der Auslastung: 0,44 – 1,76

Tabelle 2-9: Zusammenfassung der Kosten für die untersuchten Grids

Grundsätzlich ergeben sich für die kommerziellen Angebote von Sun und Amazon ähnliche Werte wir für die Kosten im Falle von EGEE und Novartis. Genauer wird dies in den folgenden beiden Unterabschnitten diskutiert.

Kosten des Grids von Novartis im Vergleich zu den Preisen von Amazon

Vergleicht man die Kosten des Grids von Novartis mit den Preisen des Angebots von Amazon, dann ergeben sich sehr ähnliche Werte. Dies ist im Grunde recht erstaunlich, weil die Kosten des Grids von Novartis für eine Vollauslastung ermittelt wurden, die weder bei Novartis noch bei Amazon sichergestellt ist. Wird diese jedoch nicht erreicht, dann ist auch mit höheren Kosten zu rechnen. Ferner beinhalten die veranschlagten Kosten des Grids von Novartis auch noch keine Gewinnspanne. Es stellt sich daher die Frage, warum die Preise von Amazon so erstaunlich niedrig sein können. Folgende Gründe erscheinen denkbar.

Es ist möglich oder sogar wahrscheinlich, dass die Personalkosten des Grids von Novartis für Administration, Service und Support zu hoch angesetzt worden sind. Für diese Kosten wurden einfach Werte eingesetzt, die proportional zu denen aus EGEE I bzw. EGEE II sind. In diesen Projekten sind jedoch Ressourcen vieler verschiedener Organisationen miteinander verbunden, was bereits prinzipiell eine aufwendigere Aufgabe darstellt als die Verbindung von Ressourcen einer einzelnen Organisation wie

bspw. der von Amazon oder Novartis. Ferner handelt es sich bei EGEE I und EGEE II um öffentlich finanzierte Projekte, die sich häufig nicht durch besonders hohe Effizienz auszeichnen.[1] Schließlich ist die obere Grenze für die angenommenen Kosten für Administration, Service und Support im Falle des Novartis-Grids von EGEE I abgeleitet worden, für welches sich bereits in Abschnitt 2.2.2 angedeutet hat, dass die Kosten durch die Neuartigkeit des Projekts vermutlich außerordentlich hoch liegen.

Weiterhin ist zu berücksichtigen, dass die Abschätzung der Kosten für das Grid von Novartis im Wesentlichen auf Angaben zu Kosten in Europa beruht. Beim Storage Grid von Amazon (S3) gibt es bspw. einen expliziten Unterschied in der Preisgestaltung zwischen den USA und dem europäischen Raum. Dabei liegen die Kosten in letzterem um etwa 20 Prozent über denen in den USA, was auf verschiedene Gründe zurückgeführt wird, u. a. auch auf den Energiepreis ([Amaz08], [PCWe07]).

Kosten von EGEE I und II im Vergleich zu den Preisen von Sun

Beim Vergleich der geschätzten Kosten für EGEE I und EGEE II mit den Preisen von Sun ergibt sich bei Annahme einer vollständigen Auslastung der Ressourcen, dass die Preise von Sun über den ermittelten Kosten der EGEE-Projekte liegen. Berücksichtigt man jedoch die niedrige Auslastung in den EGEE-Projekten, so kommt man auf eine Spanne für die Kosten, in die zwar der Preis von Sun hineinfällt, der aber eher am unteren Ende der Spanne liegt. Damit stellt sich auch hier wieder die Frage, wo dann der Raum für die nötige Gewinnspanne bleibt. Letztlich sind für den vergleichsweise niedrigen Preis von Sun folgende Erklärungen denkbar.

Zum Ersten ist nicht gesagt, dass die Auslastung der Ressourcen des Grids von Sun ebenfalls nur bei einem Drittel liegt. So erreicht bspw. Sun laut [Schn06] in seinen eigenen Rechenzentren (die nicht für das Grid-Computing verwendet werden) Auslastungswerte von etwa 95 Prozent. Auch wenn keine Informationen zur Auslastung der Ressourcen im Sun-Grid vorliegen, erscheint so zumindest eine Auslastung deutlich oberhalb von einem Drittel als nicht unwahrscheinlich.

Zum Zweiten ist auch hier wieder das Argument anzuführen, dass die Kosten für Administration, Service und Support im Falle von Sun vermutlich niedriger liegen als in den EGEE-Projekten. Die Gründe dafür sind die gleichen wie im Falle von Amazon (insbesondere der geringere organisationsübergreifende Abstimmungsaufwand und die möglicherweise effizientere Organisationsstruktur).

Schließlich sei daran erinnert, dass die Abschätzung für die Kosten der Hardware zu sehr großen Spannen führte (vgl. Abschnitt 2.1.1), die nicht weiter eingegrenzt werden konnten. Es ist bspw. denkbar, dass Sun aufgrund des großen Umfangs seiner Bestellungen Rabatte auf die üblichen Preise bekommt.

---

[1] Dies ist nicht nur der persönliche Eindruck des Autors, sondern auch die übliche Annahme in der wirtschaftswissenschaftlichen Literatur. Vergleiche hierzu bspw. [Schw05] (*„Begriffe wie Management, Marketing, Effizienzorientierung in Verbindung mit «NPO» sind weit davon entfernt, Allgemeingut zu sein, geschweige denn das entsprechende Handeln.“*, S. 59) oder [Behr99] (Gegenüberstellung von Markt- und Zentralverwaltungswirtschaft).

2.4. Ergebnisse und Schlussfolgerungen

In den vorangegangenen Abschnitten wurden für verschiedene Szenarien von Rechen-Grids die entstehenden Kosten untersucht. Es wurde eine Aufschlüsselung nach Kostenfaktoren durchgeführt und für diese wurden jeweils praxisnahe Werte ermittelt. Dabei kristallisierten sich die folgenden Faktoren als die wichtigsten heraus, die hier von der Bedeutung her in absteigender Reihenfolge aufgeführt sind:

- Server-Hardware
- Administration, Service und Support
- Internet-Anbindung
- Infrastruktur

Für die konkrete Größe dieser Kostenfaktoren ist es dabei von entscheidender Bedeutung, ob die Grid-Ressourcen ausschließlich für das Grid-Computing verwendet werden oder aber mit höherer Priorität einem anderen, primären Einsatzzweck zur Verfügung stehen. Dabei ergibt sich für die aufgeführten Faktoren eine unterschiedlich große Abhängigkeit. So sind die Kosten von Administration, Service und Support weitgehend unabhängig davon, ob die Ressourcen ausschließlich für Grid-Anwendungen verwendet werden. Im Gegensatz dazu entstehen bei der nur zusätzlichen Ausführung von Grid-Anwendungen auf bereits vorhandenen Ressourcen fast keine zusätzlichen Kosten für die Server-Hardware und für die Infrastruktur. Auch die Kosten für die Internetanbindung reduzieren sich in diesem Falle deutlich.

Als Konsequenz ergibt sich, dass zeitweise zur Verfügung stehende Ressourcen, die nicht primär dem Grid-Computing dienen, deutlich günstiger angeboten werden können als solche Ressourcen, die ausschließlich für das Grid-Computing zur Verfügung stehen. Dieser Unterschied wurde sowohl bei den durchgeführten Abschätzungen für reale, aber nicht allgemein zugängliche Grid-Projekte deutlich als auch bei der Analyse der Marktpreise entsprechender kommerzieller Angebote (vgl. Tabelle 2-9). Gerade dieser große Unterschied in den Kosten ist die Motivation für die in der vorliegenden Arbeit in Betracht gezogene Verwendung von Einzelrechnern, die primär nicht dem Grid-Computing dienen, sondern bspw. Arbeitsplatzrechner sind. Dass es diesbezüglich eine ganze Reihe geeigneter Anwendungen gibt, wurde bereits im Einleitungskapitel ausführlich dargestellt. Es wurde jedoch auch darauf hingewiesen, dass für eine bessere Nutzbarkeit solcher nur zeitweilig und unzuverlässig zur Verfügung stehender Ressourcen Prognosen der zu erwartenden freien Rechenkapazitäten hilfreich sein könnten. Um solche Prognosen geht es in den folgenden Kapiteln dieser Arbeit.

3. Prognosegröße und Bewertung der Prognosegüte

In den ersten beiden Kapiteln dieser Arbeit wurde einerseits herausgearbeitet, warum es lohnenswert sein kann, die freien Rechenkapazitäten von Einzelrechnern für rechenintensive Jobs verwenden zu wollen. Andererseits wurde auf das Problem hingewiesen, dass die übrig bleibenden, zur Ausführung von Grid-Jobs zur Verfügung stehenden freien Rechenkapazitäten starken Schwankungen unterliegen. Dies hat den in Abschnitt 1.3 angeführten Vorschlag der Prognose der zu erwartenden freien Rechenkapazitäten motiviert. Vor der Diskussion möglicher Prognoseverfahren stellen sich jedoch zwei Fragen. Zum einen ist zu klären, was genau eine solche Prognose für Informationen liefern soll. Zum anderen stellt sich das Problem der Bewertung der Informationen, also das Problem der Bewertung der Prognosegüte. Diese beiden Punkte werden in diesem Kapitel diskutiert.

3.1. Bestimmung der vorherzusagenden Größe

3.1.1. Was sind freie Rechenkapazitäten?

Die im Rahmen der vorliegenden Arbeit betrachteten freien Rechenkapazitäten sollen von der insgesamt zur Verfügung stehenden Rechenkapazität gerade dem Anteil entsprechen, der für die Ausführung von Grid-Jobs genutzt werden könnte. Es stellt sich die Frage, wie groß dieser Anteil ist bzw. wie er abgegrenzt werden kann.

Eine Sichtweise ist die, auf einem Rechner nur dann Grid-Jobs auszuführen, wenn momentan keine Nutzung dieses Rechners vorliegt. Eine solche Sichtweise wurde bspw. im Condor-Projekt ([Litz88]) eingenommen. Bei diesem Projekt wurde ein Hintergrund-Job unterbrochen, sobald der lokale Nutzer aktiv wurde. Auf diese Weise werden aber viele ungenutzte Prozessorzyklen verschenkt, da heutige Rechner auch während der Benutzung durch lokale Nutzer selten ausgelastet sind. Eine weniger restriktive, aber ähnliche Vorgehensweise wäre daher, Hintergrund-Prozesse laufen zu lassen, solange die Last des Systems einen bestimmten Schwellwert (bspw. 10 %) nicht überschreitet. Diese Variante ist auf der linken Seite der Abbildung 3-1 veranschaulicht, wobei der durch die primären Anwender momentan nicht genutzte Anteil der Rechenkapazität durch die dicke blaue (bzw. schwarze) Linie dargestellt ist. Für die Grid-Jobs kann dann der grün (bzw. grau) eingefärbte Anteil der Rechenkapazität genutzt werden, während die weiß eingefärbten Anteile unterhalb der dicken blauen (bzw. schwarzen) Kurve verfallen würden. Um letzteres zu vermeiden, gibt es eine weitere Variante, bei der sämtliche freien Prozessorzyklen als verfügbar betrachtet werden, also auch dann, wenn ein Rechner durch den lokalen Nutzer bereits zu einem beträchtlichen Teil beschäftigt wird. Ein Grid-Job dürfte demnach im Hintergrund

laufen und die sonst verfallende Rechenkapazität nutzen. Diese Sichtweise ist auf der rechten Seite der Abbildung 3-1 veranschaulicht.

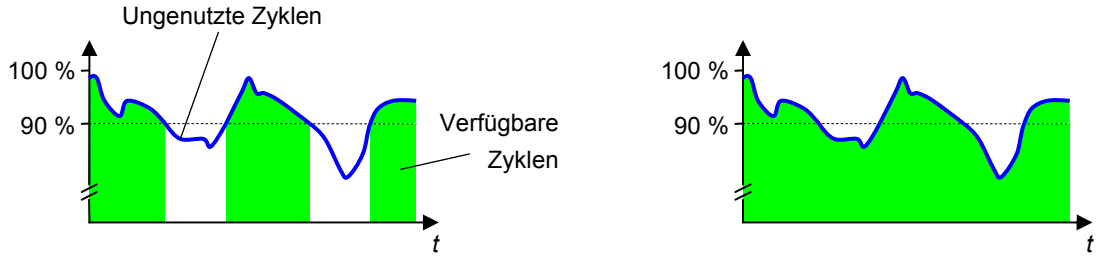

Abbildung 3-1: Mögliche Sichtweisen zur Verfügbarkeit der ungenutzten Prozessorzyklen

In allen Varianten müsste darauf geachtet werden, dass die primären Nutzer der bereitgestellten Ressourcen nicht in ihrer Arbeit gestört werden. Auf den ersten Blick besonders kritisch erscheint dabei die zuletzt genannte Variante. Allerdings stellen weder I/O- noch CPU-Spitzen ein grundsätzliches Problem dar, da in diesen Fällen die Grid-Anwendung lediglich pausieren muss, um die primären Anwendungen nicht zu stören. Zur Überprüfung, inwieweit ein solches Szenario praktisch realisierbar ist, wurden entsprechende Tests mit realen Systemen durchgeführt. Bei den untersuchten Linux- und Unix-Systemen stellte sich heraus, dass auch Prozesse mit der niedrigsten Priorität immer noch Rechenzeit bekommen, selbst wenn höher priorisierte Prozesse diese Rechenzeit ebenfalls hätten nutzen können (vgl. Anhangsabschnitte A.2 und A.3). Unter Windows hingegen zeigte sich, dass ein höher priorisierter Prozess tatsächlich nicht nennenswert in der Abarbeitung beeinträchtigt wird (vgl. Anhangsabschnitt A.1). In (subjektiven) Tests bei der interaktiven Arbeit am eigenen Rechner konnte ebenfalls keine Beeinträchtigung durch Prozesse mit niedriger Priorität festgestellt werden. Damit kann zumindest bei entsprechender Implementierung des Schedulings eine spürbare Beeinträchtigung der primären Anwender ausgeschlossen werden. Aus Sicht des Autors ist davon auszugehen, dass bei entsprechend größerer Verbreitung des Grid-Computings bspw. auch Linux-Systeme ein solches Scheduling-Verhalten zumindest optional anbieten würden.

Etwas problematischer ist die Auslastung des Speicherplatzes. Falls dieser für eine bestimmte Zeit knapp wird, müsste der Grid-Job so schnell wie möglich ausgelagert werden. Aber auch das ist in vielen Fällen ohne große Beeinträchtigung der Nicht-Grid-Anwendungen möglich, nämlich immer dann, wenn die meisten Speicherseiten des Grid-Jobs nur zu Lesezwecken verwendet werden (wie es z. B. beim in Abschnitt 1.2.3 vorgestellten Rendering der Fall wäre). Diese Speicherseiten bräuchten lediglich als verfügbar markiert zu werden, so dass ein neu gestartetes Anwendungsprogramm unverzüglich die entsprechenden Seiten bekommen könnte. Erst beim Wiederanlauf des Grid-Jobs würde es zu entsprechendem Zusatzaufwand kommen, durch den aber nur die Abarbeitung des Grid-Jobs verzögert würde.

Da also die zuletzt genannte, auf der rechten Seite von Abbildung 3-1 dargestellte Variante einerseits realisierbar ist, ohne die primären Nutzer signifikant zu stören, andererseits aber den Grid-Nutzern die größten Vorteile bietet, wird im Weiteren von dieser Variante ausgegangen.

3.1.2. Definition von K_{free}

Nachdem grundsätzlich festgelegt wurde, welche Prozessorzyklen als verfügbar gelten, wird nun genauer definiert, welche Größe durch das Prognoseverfahren vorhergesagt werden soll. Wichtig ist, dass eine Prognose dieser Größe für die in Abschnitt 1.3.1 genannten Einsatzzwecke hilfreich ist.

Da es in der vorliegenden Arbeit vor allem um länger laufende Jobs geht, ist bei der Prognose weniger die freie Rechenkapazität zu einem bestimmten Zeitpunkt als in einem bestimmten Zeitraum relevant. Für die genannten Einsatzzwecke ist dabei meist der sich unmittelbar anschließende Zeitraum von Interesse, und nur für diesen Zeitraum werden in dieser Arbeit Prognosen abgegeben. Dabei ist nicht die in diesem Zeitraum durchschnittlich verfügbare Rechenkapazität wichtig, sondern die zusammenhängend nutzbare Kapazität ausgehend vom Startzeitpunkt. Wenn also ein Scheduler entscheiden muss, auf welchem Rechner ein bestimmter Job gestartet werden soll, dann wäre es für diese Entscheidung günstig zu wissen, welcher der in Frage kommenden Rechner den gegebenen Job mit möglichst hoher Wahrscheinlichkeit ausführen und auch erfolgreich beenden kann. Dafür ist vom Zeitpunkt des Starts des Jobs die freie Rechenkapazität relevant, die vom Job ununterbrochen (also ohne Ausschalten des Rechners) verwendet werden könnte.

Die freie Rechenkapazität bis zum nächsten Ausschalten zu beurteilen ist jedoch nur dann sinnvoll, wenn für die Job-Ausführung ein akzeptabler Zeitrahmen eingehalten wird. So ist bspw. ein Rechner, der zwar einen Job zu Ende rechnen würde, dies aber aufgrund entsprechender Auslastung erst in einigen Tagen täte, in vielen Fällen nicht empfehlenswert, wenn es andere Rechner gibt, welche den Job innerhalb einiger Stunden ausführen können. Aus diesem Grunde ist es sinnvoll, nur dann den Zeitraum bis zum nächsten Abschalten zu betrachten, wenn der Zeitpunkt des Abschaltens nicht zu weit in der Ferne liegt. Anderenfalls sollte der Zeitraum anderweitig begrenzt werden.

Diese Überlegungen führten letztlich zu der Größe $K_{\mathrm{free},i}$, wie sie in der folgenden Definition 3-1 eingeführt wird. Im Weiteren wird diese Größe zum Teil auch verkürzend mit K_{free} bezeichnet, wenn der Wert von i gerade nicht relevant ist. Mit dieser Definition kann $K_{\mathrm{free},i}(t)$ offenbar nur Werte aus dem Intervall [0;100] annehmen. Der tatsächliche Wert ist dabei i. Allg. zum Zeitpunkt t noch nicht bekannt, da ggf. Informationen über die Rechnerauslastung aus dem Zeitraum von t bis $(t + t_{\mathrm{span}})$ zu berücksichtigen sind (vgl. Abbildung 3-2). Dies ist auch der Grund, aus dem zum Zeitpunkt t noch eine Prognose für $K_{\mathrm{free},i}(t)$ erforderlich ist.

Definition 3-1:

Die Größe $\boldsymbol{K_{\text{free},i}(t)}$ gibt für einen Rechner die freie Rechenkapazität an, die von einem Zeitpunkt t bis zum nächsten Stopp des Systems und innerhalb der nächsten i Stunden zur Verfügung steht. Angegeben ist dabei, wie viel Prozent der theoretisch maximal möglichen Rechenkapazität zur Verfügung stehen. Formal ist die Größe wie folgt definiert:

$$K_{\text{free},i}(t) = \frac{100}{t_{\max}} \cdot \int_{t}^{t+t_{\text{span}}} K(x) \cdot dx \qquad \text{mit} \qquad t_{\text{span}} = \min\{t_{\max}, t_{\text{on}}\} \qquad \text{und} \qquad t_{\max} = i\,\text{h}$$

Die Größen $t_{\max}$, t_{on} und $K(x)$ haben dabei die folgenden Bedeutungen:

- $t_{\max}$ ist der längstmögliche Zeitraum (i Stunden), über den die freie Rechenkapazität betrachtet wird. Bleibt der Rechner noch länger an, werden Rechenkapazitäten in der Zeit nach $(t + t_{\max})$ nicht berücksichtigt.
- t_{on} gibt die Zeitdauer an, die ein Rechner (ausgehend vom Zeitpunkt t) angeschaltet ist. Für einen zum Zeitpunkt t ausgeschalteten Rechner gilt $t_{\text{on}} = 0$.
- $K(x)$ gibt für einen Zeitpunkt x den relativen Anteil der Rechenkapazität eines Rechners an, der verfügbar (frei) wäre.

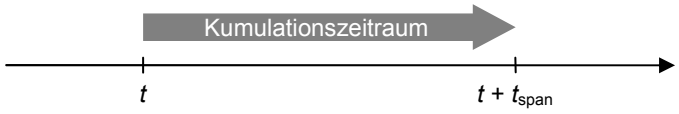

Abbildung 3-2: Kumulationszeitraum für $K_{\text{free},i}(t)$

Es bleibt zu klären, was genau bzgl. $K_{\text{free},i}(t)$ prognostiziert werden soll. Der vielleicht naheliegendste Gedanke ist der entsprechende Erwartungswert. Allerdings wäre zumindest für die Auswahl der Rechner beim Scheduling eine Wahrscheinlichkeitsverteilung für die Größe K_{free} eine deutlich wertvollere Information. Denn hat man einen Job, für den man die benötigte Rechenleistung ungefähr abschätzen kann, dann sollte man einen Rechner nehmen, der diese Rechenleistung mit hoher Wahrscheinlichkeit erbringen wird. Der Erwartungswert ist zur Abschätzung dieser Wahrscheinlichkeit nur bedingt geeignet. Beispielsweise könnte ein hoher Erwartungswert von K_{free} durch wenige besonders große Werte zustande kommen, so dass die Wahrscheinlichkeit einer zu geringen verbleibenden freien Rechenkapazität dennoch recht hoch ist. Veranschaulicht ist das Problem in Abbildung 3-3 für zwei Verteilungen und eine für einen bestimmten Job benötigte Rechenleistung *min*. Obwohl die rot (links) dargestellte Verteilung mit ihrem Erwartungswert μ_1 deutlich oberhalb von *min* liegt, wird die benötigte Kapazität nur mit einer Wahrscheinlichkeit von etwa 50 Prozent erreicht. Anders liegt der Fall bei der grün (rechts) dargestellten Verteilung, deren Erwartungswert μ_2 nur knapp oberhalb

von *min* liegt. Dennoch wird bei dieser Verteilung die benötigte Kapazität mit 100 Prozent Wahrscheinlichkeit erreicht. Aufgrund dieser mangelnden Aussagekräftigkeit des Erwartungswertes soll hier versucht werden, die Wahrscheinlichkeitsverteilung für die Größe $K_{\text{free},i}$ vorherzusagen.

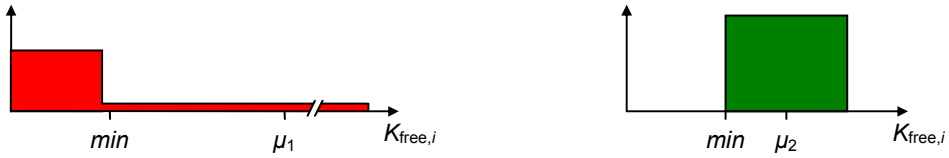

Abbildung 3-3: Mangelnde Aussagekräftigkeit des Erwartungswertes

3.2. Maße zur Beurteilung der Prognosegüte

Ein wichtiger Punkt bei der Untersuchung möglicher Verfahren zur Prognose einer bestimmten Größe ist der, die Güte der Prognosen einzuschätzen. In der vorliegenden Arbeit ist dabei insbesondere zu beurteilen, wie gut die von verschiedenen Verfahren erzeugten Prognosen für die in Abschnitt 1.3.1 angegebenen Einsatzzwecke geeignet sind. Gerade diese Einschätzung der Prognosequalität ist bei der hier gewünschten Vorhersage von Wahrscheinlichkeitsverteilungen nicht trivial.

Das Problem wird offensichtlich bei der Betrachtung der zur vorliegenden Arbeit ähnlichsten anderen Arbeit ([Wyck98], ausführliche Darstellung in Abschnitt 5.2). Diese hatte das Ziel, Wahrscheinlichkeitsverteilungen für die verbleibenden Idle-Zeiten vorherzusagen. Zu diesem Zweck wurde aus den innerhalb eines Monats auf einem Rechner beobachteten Idle-Zeiten eine empirische Häufigkeitsverteilung gebildet und diese als Grundlage für die Prognosen im darauf folgenden Monat verwendet (Abbildung 3-4).

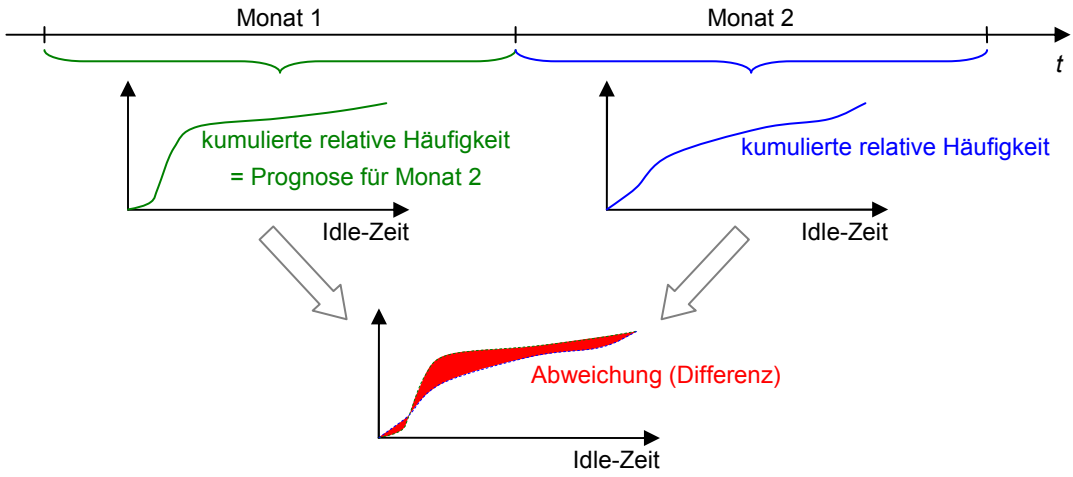

Abbildung 3-4: Prognosebewertung in [Wyck98]

Die Eignung dieser Herangehensweise wurde durch einen Vergleich (bzw. die Differenzbildung) dieser Häufigkeitsverteilung mit der für den folgenden Monat gemessenen Häufigkeitsverteilung eingeschätzt. Ein solches Vorgehen liefert jedoch nur dann eine sinnvolle Aussage zur Prognosegüte, wenn die Wahrscheinlichkeitsverteilungen der Idle-Zeiten für die Zeit eines ganzen Monats als konstant angenommen werden können. Eine solche Verteilung müsste also insbesondere unabhängig vom Wochentag und von der Tageszeit sein, was wenig realistisch erscheint.

Das Ziel der folgenden Diskussion besteht daher darin Maße zu finden, die trotz der nicht voraussetzbaren Stationarität des zugrunde liegenden stochastischen Prozesses sinnvolle Aussagen über die Qualität der Prognosen treffen. Zu bewerten ist, inwieweit die Prognosen den „realen“ Verteilungen[1] möglichst nahe kommen. Dabei können die zu unterschiedlichen Zeitpunkten vorherzusagenden realen Verteilungen nicht als gleich angenommen werden. Dies impliziert, dass selbst im Nachhinein jeder realen Verteilung nur ein einziger Wert (nämlich der Messwert der Größe K_{free}) zugeordnet werden kann (vgl. Abbildung 3-5). Damit fehlt aber auch die Voraussetzung für die unmittelbare Anwendung von Standard-Verfahren wie z. B. dem χ^2-Test oder dem Kolmogorov-Smirnov-Anpassungstest, da diese Tests jeweils zwei Verteilungen vergleichen, hier jedoch nur eine Verteilung vorliegt und ein Messwert.[2] Da in der Literatur auch keine anderen auf das hier skizzierte Problem passenden Ansätze der Beurteilung zu finden sind, wird in den folgenden Abschnitten diskutiert, wie dennoch auf sinnvolle Weise die Genauigkeit der vorhergesagten Verteilungen beurteilt werden kann.

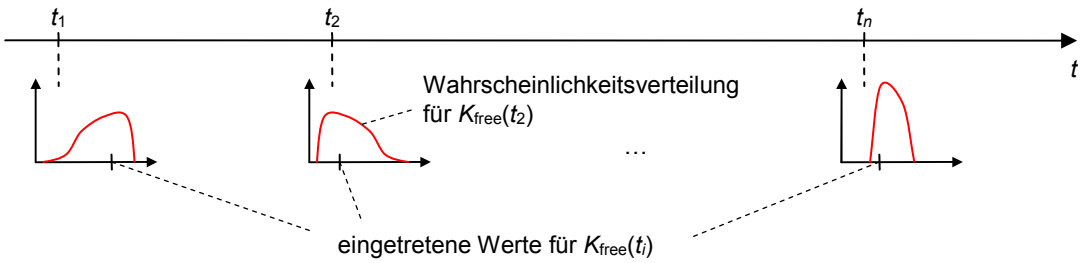

Abbildung 3-5: Wahrscheinlichkeitsverteilungen und Messwerte zu verschiedenen Zeitpunkten

[1] Diese „realen“ Verteilungen sei diejenigen, die aufgrund der bis zum jeweiligen Prognosezeitpunkt eingetretenen Ereignisse und der bestehenden Zusammenhänge zu kommenden Werten den möglichen eintretenden Werten entsprechende Wahrscheinlichkeiten zuordnet. Ein Verfahren, dass diese realen Verteilungen vorhersagen könnte, wäre optimal.

[2] Die beiden genannten Testverfahren liefern zunächst ohnehin nur eine Aussage, ob die angenommene (bzw. vorhergesagte) Verteilung als zutreffend angenommen werden kann oder nicht. Man könnte jedoch möglicherweise auf der Basis dieser Tests Kriterien für den Vergleich zweier Prognoseverfahren ableiten. Dies wird hier jedoch aufgrund der ohnehin vorhandenen Probleme nicht weiter diskutiert.

3.2.1. Durchschnittliche Einhaltung der vorhergesagten Wahrscheinlichkeiten

Ein erster Ansatz zur Beurteilung der Prognosen orientiert sich an den in Abschnitt 1.3.1 angesprochenen verbindlichen Zusagen, dass gewünschte Rechenkapazitäten mit bestimmten Wahrscheinlichkeiten zur Verfügung stehen. Die Korrektheit der versprochenen (aber nicht direkt messbaren) Wahrscheinlichkeiten wird in der Praxis meist durch die real beobachtbaren Häufigkeiten geprüft: Macht ein Verfahren in n Fällen die Zusicherungen $Z_{p,i}$ ($1 \le i \le n$), dass mit einer Wahrscheinlichkeit p jeweils eine Rechenkapazität von mindestens $x_{p,i}$ zur Verfügung steht, dann sollte auch in etwa $p \cdot n$ Fällen die jeweilige Rechenkapazität erreicht oder überschritten werden, d. h.:

$$\left.\begin{array}{ll} Z_{p,1}: & P\left(K_{\text{free}}(t_1) \ge x_{p,1}\right) = p \\ Z_{p,2}: & P\left(K_{\text{free}}(t_2) \ge x_{p,2}\right) = p \\ \vdots & \vdots \\ Z_{p,n}: & P\left(K_{\text{free}}(t_n) \ge x_{p,n}\right) = p \end{array}\right\} \quad \Rightarrow \quad \left|\left\{i \,\middle|\, K_{\text{free}}(t_i) \ge x_{p,i},\, i \in \{1,\ldots,n\}\right\}\right| \approx p \cdot n \qquad \textbf{(3-1)}$$

Die Eigenschaft eines Verfahrens, die Forderung (3-1) für beliebige p aus dem Intervall [0;1] zu erfüllen, sei im weiteren als **Genauigkeit im Durchschnitt** bezeichnet.

Problematisch an der Überprüfung dieser Eigenschaft ist zunächst, dass sich für die von den Grid-Nutzern gewünschten Rechenkapazitäten i. d. R. nicht immer die gleiche Wahrscheinlichkeit ergeben wird. Um dennoch die in dieser Arbeit diskutierten Prognosen hinsichtlich der Erfüllung obiger Forderung (3-1) bewerten zu können, wird von den durch die Prognosen für die einzelnen Zeitpunkte t_i jeweils vorhergesagten kumulierten Verteilungsfunktionen F_i ausgegangen, aus denen sich die Zusagen $Z_{p,i}$ ableiten lassen (vgl. Abbildung 3-6).

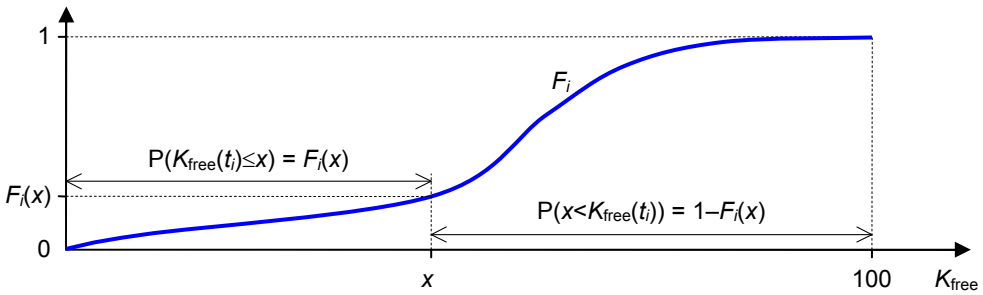

Abbildung 3-6: Bedeutung einer für einen bestimmten Zeitpunkt t_i prognostizierten kumulierten Verteilungsfunktion F_i

Für eine solche kumulierte Verteilungsfunktion F_i gilt per Definition:

$$F_i(x) = P\left(K_{\text{free}}(t_i) \le x\right)$$

$$\Leftrightarrow \quad 1 - F_i(x) = 1 - P\left(K_{\text{free}}(t_i) \le x\right)$$

$$\Leftrightarrow \quad 1 - F_i(x) = P\left(K_{\text{free}}(t_i) > x\right) \tag{3-2}$$

Für stetige Funktionen F_i ergibt sich eine recht nahe liegende Variante der Bewertung solcher kumulierter Verteilungsfunktionen. Aus diesem Grunde wird im folgenden Unterabschnitt zunächst diese Stetigkeit vorausgesetzt. Die Verallgemeinerung, in der dann auch Unstetigkeitsstellen betrachtet werden, wird in Abschnitt 3.2.1.2 diskutiert.

3.2.1.1. Stetige kumulierte Verteilungsfunktionen

Für den Fall der Stetigkeit von F_i in x ist Formel (3-2) äquivalent zu:

$$1 - F_i(x) = P\left(K_{\text{free}}(t_i) \geq x\right)$$

Folgerichtig könnte die Zusage $Z_{p,i}$ genau dann abgegeben werden, wenn die folgende Bedingung erfüllt ist (vgl. Formel (3-1)):

$$1 - F_i(x_{p,i}) = p$$

Diese Bedingung ist gleichbedeutend mit:

$$(1-p) = F_i(x_{p,i})$$

Ein Einsetzen dieser Bedingung in Formel (3-1) führt zu folgender Forderung:

$$\left.\begin{array}{l}(1-p) = F_1(x_{p,1}) \\ (1-p) = F_2(x_{p,2}) \\ \quad\vdots \\ (1-p) = F_n(x_{p,n})\end{array}\right\} \quad \Rightarrow \quad \left|\left\{i \middle| K_{\text{free}}(t_i) \geq x_{p,i},\, i \in \{1,\ldots,n\}\right\}\right| \approx p \cdot n \tag{3-3}$$

Mit der angenommenen Stetigkeit der F_i gibt es dabei für jedes i ($1 \leq i \leq n$) wenigstens ein $x_{p,i}$, mit dem die Bedingung $(1–p) = F_i(x_{p,i})$ erfüllt ist. Aufgrund der aufsteigenden Monotonie kumulierter Verteilungsfunktionen müsste bei einer korrekten Vorhersage F_i ferner gelten:

$$K_{\text{free}}(t_i) \geq x_{p,i} \quad \Leftrightarrow \quad F_i\left(K_{\text{free}}(t_i)\right) \geq F_i\left(x_{p,i}\right) \tag{3-4}$$

Die Äquivalenz der beiden Ungleichungen müsste dabei auch für nicht streng monoton steigende Funktionen F_i gelten. Veranschaulicht ist dies in Abbildung 3-7. Dort gibt es einen konstanten Bereich, weil laut F_i Werte im Intervall $(x_1;x_4]$ ausgeschlossen werden. Demnach würde dann für die Paare

$$(x_a; x_b) \quad \text{mit} \quad a,b \in \{1,2,3,4\} \quad \text{und} \quad a \neq b$$

die Beziehung

$$x_a \geq x_b \quad \Leftrightarrow \quad F_i\left(x_a\right) \geq F_i\left(x_b\right)$$

nicht mehr gelten. Allerdings dürfte es entsprechend der gemachten Vorhersage die Werte x_2, x_3 und x_4 gar nicht geben, so dass kein Widerspruch zur Formel (3-4) besteht.[1]

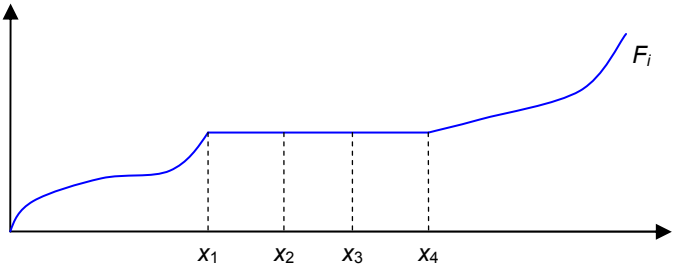

Abbildung 3-7: Verteilungskurve F_i mit konstantem Intervall $(x_1;x_4]$

Damit lässt sich die in Formel (3-3) aufgestellte Forderung umschreiben in:

$$\left.\begin{array}{l}(1-p)=F_1(x_{p,1})\\(1-p)=F_2(x_{p,2})\\\quad\vdots\\(1-p)=F_n(x_{p,n})\end{array}\right\} \Rightarrow \left|\left\{i\middle|F_i\left(K_{\text{free}}(t_i)\right)\geq F_i\left(x_{p,i}\right), i\in\{1,\ldots,n\}\right\}\right| \approx p\cdot n \qquad \textbf{(3-5)}$$

Aufgrund der bereits festgestellten Existenz der einzelnen $x_{p,i}$, mit denen die Bedingung $(1–p) = F_i(x_{p,i})$ erfüllt ist, kann die Forderung aus Formel (3-5) vereinfacht werden zu:

$$\left|\left\{i\middle|F_i\left(K_{\text{free}}(t_i)\right)\geq (1-p), i\in\{1,\ldots,n\}\right\}\right| \approx p\cdot n$$

Dies ist für stetige F_i gleichbedeutend mit:

$$\left|\left\{i\middle|F_i\left(K_{\text{free}}(t_i)\right)< (1-p), i\in\{1,\ldots,n\}\right\}\right| \approx (1-p)\cdot n$$

bzw. $$\frac{\left|\left\{i\middle|F_i\left(K_{\text{free}}(t_i)\right)< p, i\in\{1,\ldots,n\}\right\}\right|}{n} \approx p \qquad \textbf{(3-6)}$$

Da diese Forderung für jedes p aus dem Intervall [0;1] ableitbar ist, wird mit der Genauigkeit im Durchschnitt letztlich eine Gleichverteilung der Werte von $F_i(K_{\text{free}}(t_i))$ über das Intervall [0;1] gefordert. Je gleichmäßiger diese Verteilung ist, umso genauer stimmen die beobachtbaren relativen Häufigkeiten mit den vorhergesagten Wahrscheinlichkeiten überein, so dass die Nichteinhaltung der in Abschnitt 1.3.1 angesprochenen verbindlichen Zusagen unwahrscheinlicher wird. Dies ist gleichbedeutend damit, dass

[1] Streng genommen könnte die Bedingung (3-4) verletzt werden, wenn die Werte lediglich für ein offenes Intervall der Form $(x_1;x_4)$ ausgeschlossen würden, weil dann zwar bei einer stetigen Verteilung $F(x_1) \geq F(x_4)$ gelten würde, nicht aber $x_1 \geq x_4$. Dies stellt hier jedoch aus zwei Gründen kein wirkliches Problem dar. Zum einen wären die Auswirkungen auf die im Folgenden abgeleiteten Maße letztlich verschwindend gering. Zum anderen sagt keine der im Weiteren diskutierten Prognoseverfahren Verteilungen voraus, bei denen lediglich für ein offenes Intervall das Auftreten von Werten ausgeschlossen wird.

auch entsprechende negative Konsequenzen (wie z. B. Vertragsstrafen oder die Abwanderung von Kunden) mit geringerer Wahrscheinlichkeit eintreten.

3.2.1.1.1. Das Maß $M_{avg,cont}$

Zur Einschätzung der Genauigkeit im Durchschnitt wird für die eingetretenen Werte x_i bestimmt, welche kumulierten Wahrscheinlichkeiten $F_i(x_i)$ für diese Werte vorhergesagt wurden. Veranschaulicht ist dies im linken Teil der Abbildung 3-8 für die aufgetretenen Werte 5, 16 und zweimal 9. Der übersichtlicheren Darstellung wegen ist dabei nur eine Funktion F_i abgebildet. (Die Prognoseverteilung wäre hier also für vier verschiedene Zeitpunkte gleich.) Aus den für die n eingetretenen Werten bestimmten kumulierten Wahrscheinlichkeiten wird dann eine Funktion gebildet, welche die Häufigkeitsverteilung dieser Wahrscheinlichkeitswerte darstellt (vgl. die rote (stufenförmige) Kurve im rechten Teil der Abbildung 3-8).[1] Die Idealform dieser Häufigkeitsverteilung im Intervall [0,1] stellt dabei nach Formel (3-6) die Kurve $f(x) = x$ dar, welche in Abbildung 3-8 grün (gestrichelt) dargestellt ist. Die (für endliche n zwangsläufig vorhandene) Abweichung von dieser Kurve ist in der Abbildung als gelbe (bzw. graue) Fläche dargestellt.

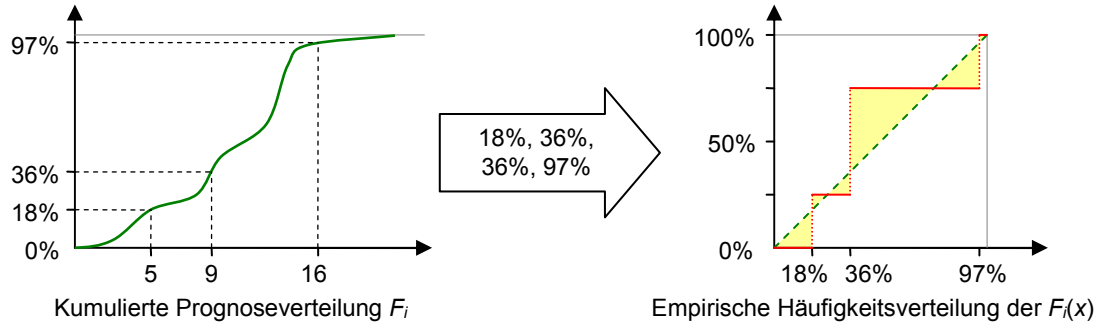

Abbildung 3-8: Die Bildung des Maßes $M_{avg,cont}$

Als Maß $M_{avg,cont}$ für die Güte der Prognosen wird die Abweichung zwischen der gemessenen Häufigkeitsverteilung der Wahrscheinlichkeitswerte und der Funktion $f(x) = x$ im Intervall [0,1] verwendet. Bei diesem Maß $M_{avg,cont}$ gilt eine Prognose als um so besser, je kleiner der Wert des Maßes ist. Die formale Definition lautet wie folgt:

[1] Hier und im Rest der Arbeit sei mit der Häufigkeitsverteilung immer die *kumulierte* Verteilung der *relativen* Häufigkeiten gemeint.

Definition 3-2:

Ein Prognoseverfahren habe für die Zeitpunkte t_1 bis t_n jeweils stetige kumulierte Wahrscheinlichkeitsverteilungen F_i prognostiziert. Gemessen wurden für die jeweiligen Zeiten die Werte x_i. Dann berechnet sich **das Maß $M_{\text{avg,cont}}$** wie folgt:

$$M_{\text{avg,cont}} = \int_0^1 \left| h_{\text{cont}}(x) - x \right| dx \qquad \text{mit} \qquad h_{\text{cont}}(x) = \frac{\left| \left\{ F_i(x_i) \middle| F_i(x_i) \le x, i \in \{1,\ldots,n\} \right\} \right|}{n}$$

3.2.1.1.2. Prüfung einzelner Perzentile

Das Maß $M_{\text{avg,cont}}$ gibt letztlich an, wie gut die vorhergesagten Wahrscheinlichkeitsverteilungen im Durchschnitt eingehalten werden. Dazu wird die Gleichverteilung der für die eingetretenen Werte x_i prognostizierten kumulierten Wahrscheinlichkeiten $F_i(x_i)$ über dem Intervall [0;1] geprüft. Ist man jedoch nur an bestimmten Perzentilen der Wahrscheinlichkeitsverteilung interessiert (bspw. weil im Vertrag vereinbart wurde, dass die vorausgesagte freie Rechenkapazität auch in wenigstens 95 Prozent der Fälle zur Verfügung stehen muss), dann ist eine gesonderte Untersuchung dieser Perzentile wünschenswert.

In dieser Arbeit werden daher zusätzlich zur Verwendung des Maßes $M_{\text{avg,cont}}$ gesonderte Untersuchungen zu den Vorhersagen mit 80%iger, 95%iger und 99%iger Wahrscheinlichkeit durchgeführt. Entsprechend obiger Ausführungen zu Abbildung 3-6 beziehen sich die Untersuchungen folglich auf die Einhaltung des 20%-, 5%- und 1%-Perzentils. Es wird dabei geprüft, für welchen Anteil der eintretenden Werte x_i die prognostizierten kumulierten Wahrscheinlichkeiten $F_i(x_i)$ in das entsprechende Intervall fallen. Idealerweise sollten dies bei den angesprochenen Perzentilen gerade etwa 20, 5 oder 1 Prozent der Werte sein. Die auftretende Abweichung $\Delta_{\text{cont}}(p)$ ist wie folgt definiert:

Definition 3-3:

Eine Prognoseverfahren habe für die Zeitpunkte t_1 bis t_n jeweils stetige kumulierte Wahrscheinlichkeitsverteilungen F_i prognostiziert. Zu den jeweiligen Zeiten gemessen wurden die Werte x_i. Die **Abweichung des Perzentils p** wird dann mit **$\Delta_{\text{cont}}(p)$** bezeichnet und wie folgt definiert:

$$\Delta_{\text{cont}}(p) = \frac{\left| \left\{ F_i(x_i) \middle| F_i(x_i) \le p, i \in \{1,\ldots,n\} \right\} \right|}{n} - p$$

Mit den ausgewählten Perzentilen kann somit ein Aussage darüber getroffen werden, inwieweit potentielle Zusagen, dass ein Rechner noch mit mindestens 80, 95 oder 99 Prozent Wahrscheinlichkeit eine bestimmte Aufgabe zu Ende rechnen kann, eingehalten werden.

3.2.1.2. Unstetige kumulierte Verteilungsfunktionen

In der bisher geführten Diskussion zur Bewertung der Genauigkeit im Durchschnitt ist von der Stetigkeit der prognostizierten kumulierten Verteilungsfunktionen ausgegangen worden. Es stellt sich daher die Frage, welche Konsequenzen Unstetigkeitsstellen haben und wie mit ihnen umgegangen werden sollte. Für die in dieser Arbeit betrachteten Verteilungsfunktionen sind dabei nur Sprünge der Art relevant, dass ein bestimmter Wert eine besonders große, nicht infinitesimal kleine Wahrscheinlichkeit besitzt. Als Beispiel ist dies in Abbildung 3-9 für den Wert x_{jump} dargestellt, der eine Wahrscheinlichkeit von 50 Prozent besitzt, während alle anderen Punkte nur eine (maximal) infinitesimal kleine Wahrscheinlichkeit besitzen. Bei einer solchen Prognoseverteilung kann die prognostizierte kumulierte Verteilungsfunktion F_i keine Werte aus dem Intervall (p_1,p_2) ergeben. Auf der anderen Seite müsste jedoch der Fall $F_i(K_{\text{free}}(t_i)) = p_2$ mit besonders großer Wahrscheinlichkeit eintreten – im hier skizzierten Fall mit einer Wahrscheinlichkeit von 50 Prozent.

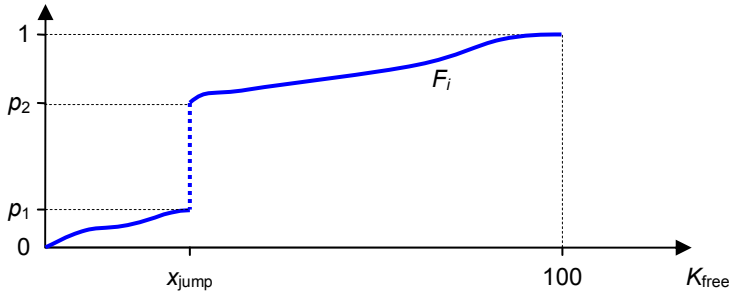

Abbildung 3-9: Unstetigkeitsstelle von F_i

Um dennoch auf die durch $M_{\text{avg,cont}}$ sowie $\Delta_{\text{cont}}(p)$ bewertete Gleichverteilung zu kommen, kann man darüber nachdenken, die Prognose $F_i(x_{\text{jump}})$ so zu bewerten, als wenn sie gleichmäßig über das Intervall $(p_1,p_2]$ verteilt wäre. Eine nahe liegende Lösung besteht dabei darin, bei einer solchen Sprungstelle in der prognostizierten kumulierten Verteilung den in die Bewertungsfunktionen einfließenden Wert p_2 durch einen zufällig aus dem Intervall $(p_1,p_2]$ gewählten Wert zu ersetzen. Dementsprechend wäre wieder die bei Stetigkeit erwartete Gleichverteilung der prognostizierten Werte im Bereich [0;1] zu erwarten. Die entsprechende Transformation t_{cont}, welche eine kumulierte Verteilungsfunktion in eine stetige Zufallsgröße umwandelt, sei wie folgt definiert:

Definition 3-4:

Die **Transformation $t_{\mathbf{cont}}$** wandelt eine kumulierte Verteilungsfunktion F in eine stetige Zufallsgröße um und ist wie folgt definiert:

$$t_{\text{cont}}(F,x) = \begin{cases} F(x) & \text{falls } \lim_{z \to x}\big(F(z)\big) \text{ existiert} \\ rand\big(F(x) - P(x), F(x)\big) & \text{sonst} \end{cases}$$

Dabei liefert $rand(x,y)$ eine zufällig aus dem Intervall $(x,y]$ gewählte Zahl, wobei jede der Zahlen aus diesem Intervall gleichwahrscheinlich ist. $P(x)$ ist die gemäß $F(x)$ dem Wert x zugeordnete Wahrscheinlichkeit, d. h.:

$$P(x) = F(x) - \lim_{z \uparrow x} F(z)$$

Mit Hilfe dieser Transformation t_{cont} werden die oben definierten Maße $M_{\text{avg,cont}}$ und $\Delta_{\text{cont}}(p)$ angepasst, um für unstetige kumulierte Verteilungsfunktionen angewendet werden zu können:

Definition 3-5:

Ein Prognoseverfahren habe für die Zeitpunkte t_1 bis t_n jeweils kumulierte Wahrscheinlichkeitsverteilungen F_i prognostiziert. Gemessen wurden für die jeweiligen Zeiten die Werte x_i. Dann berechnet sich **das Maß $M_{\mathbf{avg}}$** wie folgt:

$$M_{\text{avg}} = \int_0^1 \left|h(x) - x\right| dx \quad \text{mit} \quad h(x) = \frac{\left|\left\{t_{\text{cont}}(F_i, x_i) \middle| t_{\text{cont}}(F_i, x_i) \le x,\ i \in \{1,\ldots,n\}\right\}\right|}{n}$$

Definition 3-6:

Eine Prognoseverfahren habe für die Zeitpunkte t_1 bis t_n jeweils kumulierte Wahrscheinlichkeitsverteilungen F_i prognostiziert. Zu den jeweiligen Zeiten gemessen wurden die Werte x_i. Die **Abweichung des Perzentils p** wird dann mit **$\Delta(p)$** bezeichnet und wie folgt definiert:

$$\Delta(p) = \frac{\left|\left\{t_{\text{cont}}(F_i, x_i) \middle| t_{\text{cont}}(F_i, x_i) \le p,\ i \in \{1,\ldots,n\}\right\}\right|}{n} - p$$

Bei zu Vergleichszwecken durchgeführten Messungen mit den Maßen M_{avg} sowie $\Delta(p)$ auf der einen und $M_{\text{avg,cont}}$ sowie $\Delta_{\text{cont}}(p)$ auf der anderen Seite ergaben sich nur Unterschiede in Nachkommastellen. Da dies jedoch nicht für alle praxisrelevanten Fälle sichergestellt ist, wurde in der vorliegenden Arbeit ausschließlich mit M_{avg} sowie $\Delta(p)$ gearbeitet, um die Genauigkeit im Durchschnitt bewerten zu können.

3.2.2. Punktgenauigkeit der vorhergesagten Wahrscheinlichkeiten

Die im vorigen Abschnitt vorgeschlagenen Varianten der Prognosebewertung sind geeignet um zu prüfen, inwieweit die vorhergesagten Wahrscheinlichkeiten mit den messbaren relativen Häufigkeiten übereinstimmen. Diese Übereinstimmung ist für die Einhaltung von Dienstgütevereinbaren wichtig und führte zur Forderung der „Genauigkeit im Durchschnitt“. Für die Auswahl des für einen Job am besten geeigneten Rechners oder für die Wahl des zeitlichen Abstands zwischen aufeinander folgenden Checkpoints sollte jedoch ein Prognoseverfahren gewählt werden, das nicht nur im Durchschnitt genau ist. Es ist mindestens genauso wichtig Vorhersagen zu haben, die für die einzelnen Zeitpunkte möglichst genau sind.

Das Problem lässt sich am besten an einem Beispiel illustrieren (vgl. Abbildung 3-10). Es sei so, dass zu unterschiedlichen Zeitpunkten zwei verschiedene reale Gleichverteilungen vorliegen. Zur einen Hälfte liegen die Werte im Intervall $[a,(a+b)/2]$, zur anderen Hälfte im Intervall $[(a+b)/2,b]$. Die Prognose hingegen bleibe einfach konstant. Mit der oben vorgeschlagenen Beurteilung der Prognosegüte mittels M_{avg} würde die Prognose als perfekt gelten, da im Durchschnitt die Verteilung stimmt. Eine Prognose hingegen, welche die wechselnden realen Verteilungen ungefähr (aber eben nicht genau) vorhersagen würde, würde hingegen schlechter abschneiden, obwohl sie zu den einzelnen Zeitpunkten eigentlich genauer ist.

Abbildung 3-10: Konstante Prognosen bei wechselnden Verteilungen

Auch der umgekehrte Fall führt zum gleichen Problem und ist in Abbildung 3-11 dargestellt. Hier gibt es eine konstante reale Verteilung, die Prognosen jedoch wechseln zwischen zwei Verteilungen. Obwohl die Prognosen offensichtlich nicht gut sind (die rosa (bzw. hellgrauen) Flächen zeigen die Bereiche, die gar nicht eintreten können), würden sie nach dem oben beschriebenen Verfahren als perfekt eingeschätzt werden. Eine genauere Prognose, welche bspw. eine Gleichverteilung zwischen 4 und 11 voraussagt, würde hingegen schlechter abschneiden.

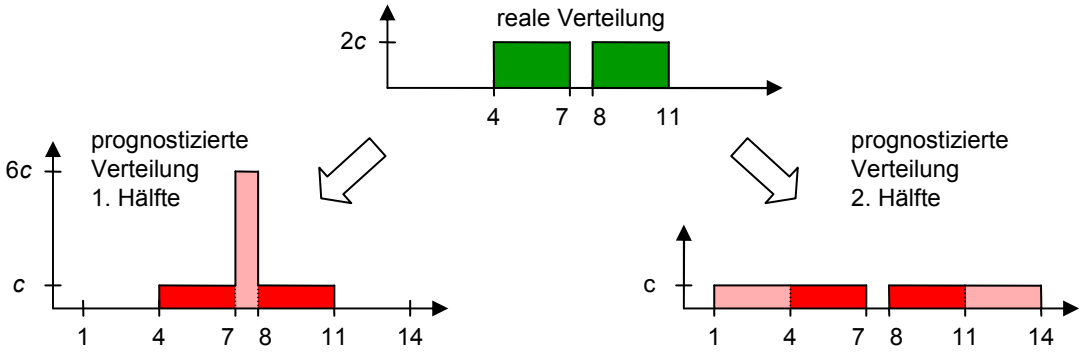

Abbildung 3-11: Wechselnde Prognosen bei konstanter Verteilung

Aufgrund der genannten Probleme ist es erforderlich, neben der Genauigkeit im Durchschnitt auch die **Punktgenauigkeit** zu fordern. Eine Vorhersagemethode wird dabei hier als punktgenau bezeichnet, wenn die für die verschiedenen Zeitpunkte t_i $(1 \leq i \leq n)$ jeweils vorausgesagten Wahrscheinlichkeitsverteilungen mit den realen Verteilungen der entsprechenden Zeitpunkte übereinstimmen. Mit der realen Verteilung für einen Zeitpunkt t sei dabei jene gemeint, welche sich bei Kenntnis aller bis zum Zeitpunkt t gewinnbaren Informationen ergibt.[1]

Naturgemäß ist auch bezüglich der Punktgenauigkeit nicht zu erwarten, dass reale Vorhersageverfahren diese erfüllen können. Folgerichtig geht es wiederum um die Messung, wie weit entfernt die Vorhersagemethode von dieser Eigenschaft ist.

3.2.2.1. Idee für ein Maß zur Einschätzung der Punktgenauigkeit

Um die Punktgenauigkeit eines Vorhersageverfahrens einschätzen zu können, wird ein Maß benötigt, welches mit dem einzigen realen Wert, der für jede der realen Verteilungen messbar ist (vgl. Abbildung 3-5), einen Rückschluss auf die Güte der Prognose zieht. Es ist also eine Funktion f gesucht, die gerade für die optimale Prognose ein Extremum hat, wobei hier im Weiteren o. B. d. A. von einem Minimum ausgegangen wird. Für suboptimale Prognosen sollten sich demnach um so höhere Werte ergeben, je stärker die Prognosen von der realen Verteilung abweichen. Ein wesentliches Problem an dieser Stelle ist, dass die reale Verteilung gerade nicht bekannt ist (vgl. Ausführungen am Anfang des Kapitels) – sie beeinflusst lediglich die Werte, die zu den Zeitpunkten gemessen werden, für die die Verteilung gilt.

Die Idee zur Lösung des Problems besteht darin, gerade diesen Einfluss auszunutzen, um eine Zufallsfunktion zu konstruieren, deren Erwartungswert mit der oben genannten Funktion f übereinstimmt. Hat man eine solche Zufallsfunktion, kann man auch die Punktgenauigkeit zweier Prognoseverfahren vergleichen, indem man diese Zufalls-

[1] Der in der Definition verwendete Begriff der realen Verteilungen lässt sich nur schwer genauer definieren, insbesondere weil diese realen Verteilungen sich nicht näher bestimmen lassen (vgl. Diskussion am Anfang des Kapitels).

funktion auf die zu den einzelnen Zeitpunkten einer Zeitreihe gehörenden Werte und die dazugehörigen Prognosen anwendet und die Ergebnisse schließlich für die Zeitreihe aufsummiert. Für das bessere Verfahren müsste sich demnach eine kleinere Summe ergeben.[1] Wie eine solche Zufallsfunktion aussehen kann, wird im Weiteren diskutiert.

3.2.2.2. Betrachtung für zwei Werteklassen

Die im Folgenden vorgeschlagene Funktion zur Beurteilung der Punktgenauigkeit wird von jeder prognostizierten Verteilung nur endlich viele, diskrete Werte prüfen. Damit wird der kontinuierliche Wertebereich implizit zu einem diskreten vereinfacht. Jeder der diskreten Werte steht für ein Intervall des ursprünglichen Wertebereichs. Die Grenzen der Intervalle werden dabei unabhängig von der prognostizierten Verteilung gewählt. Auf den Grund für diese Forderung wird am Ende dieses Abschnitts noch kurz eingegangen.

Veranschaulicht ist die Vereinfachung in der folgenden Abbildung 3-12. Es werden die kumulierten Wahrscheinlichkeiten für drei Werte geprüft, durch die letztlich die vier Intervalle (Klassen) I_1 bis I_4 entstehen. Nicht geprüft werden die kumulierten Wahrscheinlichkeiten an den Rändern des theoretisch möglichen Wertebereichs. Diese Prüfung ist deshalb nicht nötig, weil die entsprechenden Wahrscheinlichkeiten immer 0 bzw. 1 sein müssen – und weil der theoretische Wertebereich bekannt ist.[2] Deshalb ist im rechten Teil der Abbildung für die Klasse I_4 die kumulierte Wahrscheinlichkeit auch rosa (heller) dargestellt, um zu kennzeichnen, dass sie nicht überprüft werden muss.

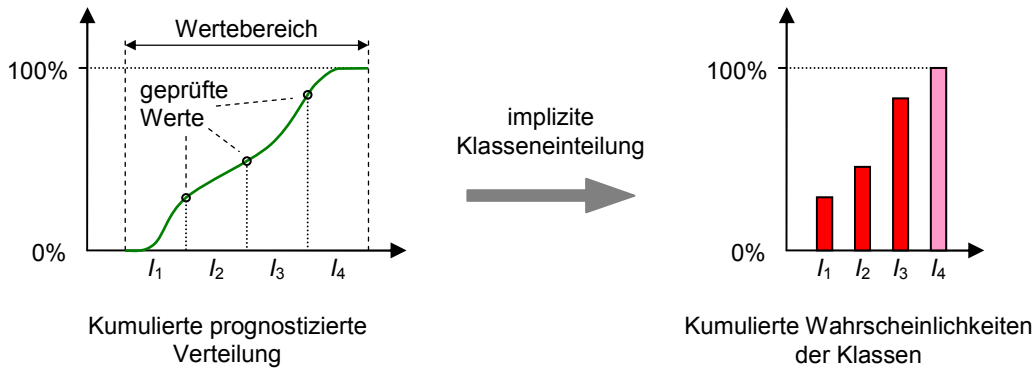

Abbildung 3-12: Implizite Klasseneinteilung bei der Bewertung der Punktgenauigkeit

[1] Offensichtlich kommt es also auch hier letztlich zu einer Durchschnittsbildung. Allerdings erfolgt hier zunächst eine Bewertung der einzelnen(!) Prognosen und aus diesen Bewertungen wird dann der Durchschnitt gebildet. Im Gegensatz dazu wird bei den in der Literatur vorgeschlagenen Verfahren (siehe Diskussion am Anfang von Abschnitt 3.2) keinerlei Bewertung der einzelnen Vorhersagen vorgenommen. Stattdessen werden einerseits die Vorhersagen und andererseits die schließlich beobachteten Messwerte gesammelt. Die Bewertung erfolgt dann durch den Vergleich dieser beiden Mengen, was für eine Prüfung der Punktgenauigkeit nicht zielführend ist.

[2] Bei der hier untersuchten Größe K_{free} ist der theoretisch mögliche Wertebereich immer [0;100].

Für den Anfang sei eine Einteilung in zwei Klassen durchgeführt, eine Verallgemeinerung auf n Klassen befindet sich im nächsten Abschnitt. In Abbildung 3-13 sind für eine solche Einteilung die kumulierte reale Verteilung (mit dem Parameter q) und die kumulierte prognostizierte Verteilung (mit dem Parameter p) dargestellt.[1]

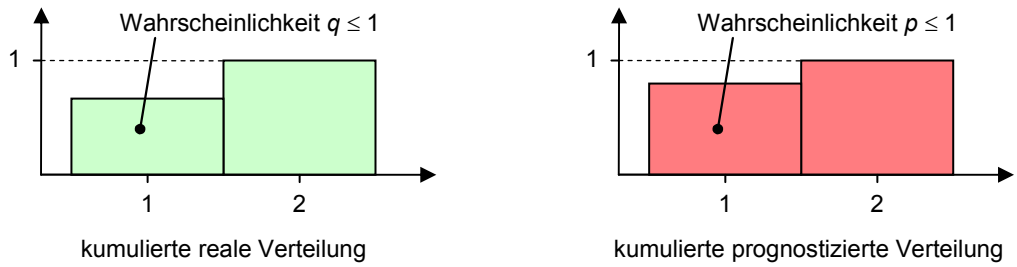

Abbildung 3-13: Einteilung des Wertebereichs in zwei Klassen

Dann wird nach einer von der Prognose p abhängigen Funktion f_q gesucht, die für $p = q$ minimal ist und mit stärker werdender Abweichung immer größer wird. (Für jedes mögliche q würde es also eine extra Funktion geben.) Um auf eine solche Funktion zu kommen, wird zunächst von einer geeigneten Ableitung ausgegangen, die gerade für $p = q$ null wird:

$$f_q'(p) = 2p - 2q$$

Der Faktor 2 ist gewählt, um eine „schönere“ Stammfunktion zu erhalten:

$$f_q(p) = p^2 - 2q \cdot p$$

Dies wäre eine mögliche Funktion, die aber (wie bereits angedeutet) ohne Kenntnis von q nicht berechnet werden kann. Unter Nutzung des im Nachhinein ermittelbaren Messwertes ergibt sich jedoch mit der folgenden Funktion $s_f(p)$ ein erwartungstreuer Schätzer:

$$s_f(p) = \begin{cases} p^2 - 2p & \text{falls Messwert ins Intervall 1 fällt} \\ p^2 & \text{sonst} \end{cases}$$

Die Erwartungstreue dieses Schätzers folgt aus der Berechnung des Erwartungswertes für eine diskrete Zufallsgröße X. Nimmt diese mögliche Werte von x_1 bis x_n mit jeweiligen Wahrscheinlichkeiten w_1 bis w_n an, dann berechnet sich der Erwartungswert gemäß der folgenden Formel:

[1] Auf den Grund für die Nutzung der kumulierten Verteilungen wird bei der Übertragung auf n Werteklassen eingegangen. Für die Diskussion einer zweiwertigen Verteilung ist dieser Punkt zunächst unerheblich; eine Verwendung der nicht-kumulierten Verteilungen würde hier zu den gleichen Ergebnissen führen.

$$E(X) = \sum_{i=1}^{n} w_i x_i \qquad \textbf{(3-7)}$$

Für den hier vorgeschlagenen Schätzer $s_f(p)$ ergibt sich der Erwartungswert daher wie folgt:

$$E\left(s_f(p)\right) = q \cdot \left(p^2 - 2p\right) + (1-q) \cdot p^2 = p^2 - 2q \cdot p$$

Da der vorgeschlagene Schätzer erwartungstreu ist und weil $f_q''(p) = 2$ gilt, ist der Erwartungswert dieses Schätzers minimal für $p = q$ (und zwar nur unter dieser Bedingung).[1]

3.2.2.3. Betrachtung für n Werteklassen

Um die Verteilung genauer beurteilen zu können, reicht eine Einteilung in zwei Klassen von Werten nicht aus. Daher soll im Folgenden ein Maß entwickelt werden, welches für beliebig viele Klassen geeignet ist. Dazu wird vom Ansatz für zwei Klassen ausgegangen. Zunächst soll jedoch an dieser Stelle begründet werden, warum das Maß anhand der kumulierten Verteilung konstruiert wird. Dazu sei die folgende Abbildung 3-14 betrachtet.

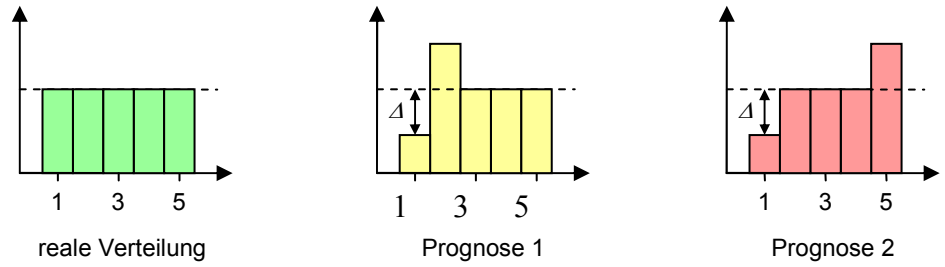

Abbildung 3-14: Berücksichtigung der Ordnung bei der Beurteilung der Prognosegüte

Die reale Verteilung sei eine Gleichverteilung über fünf Werteklassen. Es gibt zwei verschiedene Prognosen für diese Verteilung. Beide Prognosen unterschätzen die Wahrscheinlichkeit der Klasse 1 um ein bestimmtes Δ und überschätzen dafür die Wahrscheinlichkeit einer anderen Klasse. Während Prognose 1 die Wahrscheinlichkeit der Klasse 2 überschätzt, überschätzt Prognose 2 die der Klasse 5. Betrachtet man die Abweichungen der Wahrscheinlichkeiten der einzelnen Werteklassen getrennt, dann sind die beiden Prognosen gleichwertig.

Allerdings entsprechen die hier eingeführten Klassen aneinandergrenzenden Intervallen des ursprünglichen Wertebereichs, so dass es auch eine natürliche Ordnungsrelation für diese Klassen gibt. Damit sollte die Prognose 1 aus Abbildung 3-14 gegenüber der

[1] An dieser Stelle sollte die Bedeutung der Forderung, dass die Einteilung der Klassen unabhängig von der Prognose zu erfolgen hat, klar geworden sein. So wurde q als feste Wahrscheinlichkeit für die erste Klasse verwendet. Dies wäre nicht möglich gewesen bei einer von der Prognose abhängigen Klasseneinteilung.

Prognose 2 als besser gelten. Genau aus diesem Grund wird hier mit kumulierten Verteilungsfunktionen gearbeitet, was in Abbildung 3-15 angedeutet ist. Ziel ist es dann, die Abweichungen zwischen der realen kumulierten Verteilungsfunktion und deren Schätzung zu bewerten – wobei größere Abweichungen durch größere Werte gekennzeichnet werden sollen.

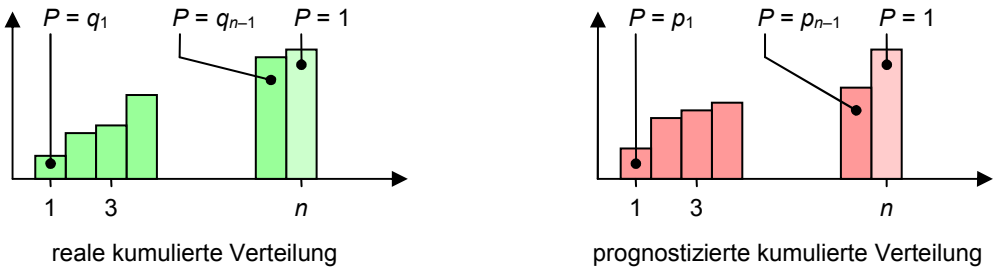

Abbildung 3-15: Kumulierte Verteilungsfunktion für n Klassen

Wenn der Ansatz für zwei Werteklassen auf n Werteklassen übertragen werden soll, ergibt sich zunächst, dass nicht mehr nur noch ein Parameter existiert, sondern (n-1) unabhängige Parameter zu berücksichtigen sind (vgl. Abbildung 3-15). Zur Vereinfachung der Schreibweise seien dabei die Einzelparameter q_1 bis q_{n-1} bzw. p_1 bis p_{n-1} zu den (n-1)-dimensionalen Vektoren q und p zusammengefasst. Dann ist wiederum eine Funktion f_q zu finden, die für die optimale Prognose minimal ist und um so größer wird, je größer die Abweichung ist. Ausgegangen wird erneut von der (hier partiellen) Ableitung nach einem p_i ($1 \le i \le n-1$):

$$\frac{\partial f_q}{\partial p_i} = 2p_i - 2q_i \qquad \text{und} \qquad \frac{\partial^2 f_q}{\partial^2 p_i} = 2$$

Es wird deutlich, dass für eine solche partielle Ableitung die Stammfunktion ein Minimum für $p_i = q_i$ erreichen würde (wenn nur p_i variiert würde). Analog zum Fall mit zwei Werteklassen ergibt sich eine entsprechende Funktion f_q:

$$f_q(p) = \sum_{i=1}^{n-1} {p_i}^2 - \sum_{i=1}^{n-1} 2q_i \cdot p_i$$

Ein dazu passender erwartungstreuer Schätzer ist $s_f(p)$:

$$s_f(p) = \begin{cases} \sum_{i=1}^{n-1} {p_i}^2 - 2\sum_{i=k}^{n-1} p_i & \text{falls Messwert in Klasse } k \text{ fällt } (1 \le k < n) \\ \sum_{i=1}^{n-1} {p_i}^2 & \text{sonst} \end{cases} \qquad \textbf{(3-8)}$$

Als Erwartungswert für diesen Schätzer ergibt sich entsprechend der allgemeinen Berechnungsformel für den Erwartungswert diskreter Zufallsgrößen (Formel (3-7)):

$$E\left(s_f(p)\right) = q_1 \cdot \left(\sum_{i=1}^{n-1} {p_i}^2 - 2\sum_{i=1}^{n-1} p_i \right) + \sum_{k=2}^{n-1} \left[\left(q_k - q_{k-1}\right) \cdot \left(\sum_{i=1}^{n-1} {p_i}^2 - 2\sum_{i=k}^{n-1} p_i \right) \right] + \left(1 - q_{n-1}\right) \cdot \sum_{i=1}^{n-1} {p_i}^2$$

$$\rightarrow \qquad E\left(s_f(p)\right) = \sum_{i=1}^{n-1} {p_i}^2 - \sum_{i=1}^{n-1} 2q_i \cdot p_i$$

Dies entspricht gerade der Funktion $f_q(p)$, womit der Erwartungswert für die bestmögliche Prognose minimal ist.

3.2.2.4. Vorliegen einer gemeinsamen Intervallskala?

Dass das vorgeschlagene Maß für die optimale Prognose das einzige Minimum hat, ist noch nicht ausreichend. Darüber hinaus sollte auch gewährleistet sein, dass nicht-optimale Prognosen in einer plausiblen Reihenfolge sortiert werden. Weil darüber hinaus die Werte für unterschiedliche Zeitpunkte aufsummiert werden sollen, ist auch das Vorliegen einer gemeinsamen Intervallskala wünschenswert, was im Folgenden untersucht wird. Es werden dabei nicht die Schätzer betrachtet, sondern die Funktionen f_q für die jeweiligen Werte von q. Da die Schätzer gerade für diese Funktionen erwartungstreu sind und viele Werte aufsummiert werden sollen, ist die Summe der mit dem Schätzer berechneten Werte mit hoher Wahrscheinlichkeit sehr dicht bei der Summe der eigentlichen Funktionswerte von f_q.

Für die Diskussion relevant ist die Bewertung der optimalen Prognose, also der realen Verteilung. Für diese ergibt sich die folgende Bewertung:

$$f_q(q) = \sum_{i=1}^{n-1} {q_i}^2 - \sum_{i=1}^{n-1} 2q_i \cdot q_i = -\sum_{i=1}^{n-1} {q_i}^2$$

Bildet man die Differenz zwischen einer anderen Prognose p und der Optimalprognose q, dann erhält man:

$$f_q(p) - f_q(q) = \sum_{i=1}^{n-1} {p_i}^2 - \sum_{i=1}^{n-1} 2q_i \cdot p_i + \sum_{i=1}^{n-1} {q_i}^2 = \sum_{i=1}^{n-1} \left(p_i - q_i\right)^2$$

Für jede Prognose p ergibt die Funktion $f_q(p)$ also einen Wert, der um die Summe der quadrierten Abweichungen der einzelnen Parameter p_i von den entsprechenden q_i höher ist als der Funktionswert für die optimale Prognose. Man kann nun die Meinung vertreten, dass diese Summe der Quadrate der Abweichungen ein proportionales Maß für die Gesamtabweichung der Schätzung ist.[1] Folgt man dieser Ansicht, dann erzeugt das

[1] Diese Ansicht lässt sich nicht logisch begründen oder gar beweisen. Sie wird jedoch in vielen Fällen verwendet, u. a. bei der Regression nach der Methode der kleinsten Quadrate. Es sei aber explizit erwähnt, dass man die Ansicht nicht teilen muss. Ist man diesbezüglich anderer Auffassung, müsste man ein anderes Maß für die Beurteilung der vorhergesagten Wahrscheinlichkeitsverteilung nutzen.

vorgeschlagene Maß Werte auf einer Intervallskala. Es handelt sich dabei sogar um eine gemeinsame(!) Intervallskala für unterschiedliche Zeitpunkte (bzw. unterschiedliche Werte von q), da die Differenz zur Bewertung der optimalen Prognose immer gerade die Summe der quadrierten Abweichungen der einzelnen Parameter ist. Somit können die Werte von unterschiedlichen Zeitpunkten bedenkenlos aufaddiert werden, wobei die kleinere Summe für das bessere Prognoseverfahren steht.

3.2.2.5. Unabhängigkeit von der Sortierrichtung

Eine weitere sich stellende Frage ist die, inwieweit die Einschätzung der Prognosegüte unabhängig davon ist, ob man die Klassen auf- oder absteigend sortiert. Die Unabhängigkeit sollte an sich gegeben sein, da es i. Allg. keinen plausiblen Grund gibt, warum die eine oder die andere Richtung die einzig mögliche wäre. Die entsprechende Entscheidung sollte dann jedoch keinen Einfluss auf die relative Güte verschiedener Prognoseverfahren haben.

Um diese Unabhängigkeit von der Sortierrichtung zu untersuchen, werden für die von rechts kumulierte Verteilungsfunktion die realen Wahrscheinlichkeiten r_i und die geschätzten Wahrscheinlichkeiten s_i eingeführt, was in der folgenden Abbildung 3-16 für die r_i und $n = 4$ dargestellt ist:

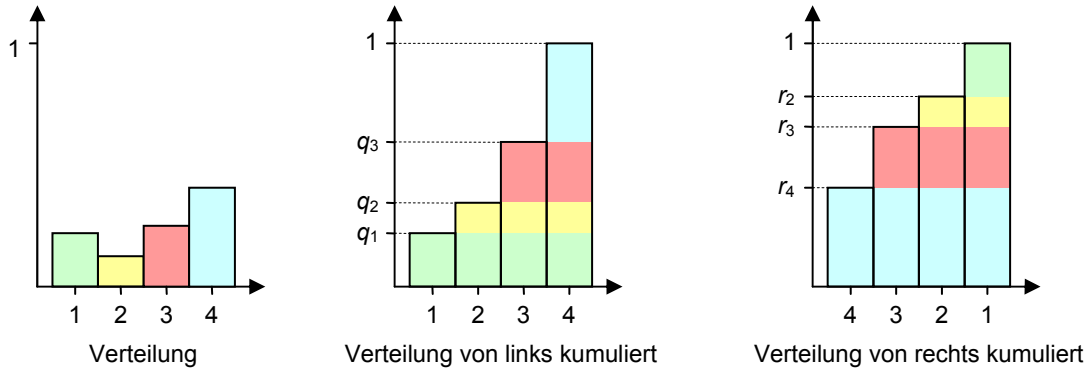

Abbildung 3-16: Bildung der kumulierten Verteilungsfunktion aus unterschiedlichen Richtungen

Die genauen Definitionen für r_i und s_i sind:

$$r_i = 1 - q_{i-1} \quad \text{und} \quad s_i = 1 - p_{i-1}$$

Die einzelnen Werte r_2 bis r_n bzw. s_2 bis s_n seien dabei zu jeweils (n-1)-dimensionalen Vektoren r und s zusammengefasst. Für die Kumulierung von rechts ergibt sich damit die Funktion $g_r(s)$, die nach dem gleichen Schema wie $f_q(p)$ arbeitet und folgendes Aussehen hat:

$$g_r(s) = \sum_{i=1}^{n-1} s_{n-i+1}^2 - \sum_{i=1}^{n-1} 2r_{n-i+1}s_{n-i+1}$$
$$= \sum_{i=2}^{n} s_i{}^2 - \sum_{i=2}^{n} 2r_i \cdot s_i$$

Ersetzt man in dieser Darstellung die Vektoren s und r durch die entsprechenden Terme mit p und q, dann erhält man:

$$g_r(s) = \sum_{i=2}^{n} \left[(1-p_{i-1})^2 - 2(1-q_{i-1}) \cdot (1-p_{i-1})\right]$$
$$= \sum_{i=1}^{n-1} \left[(1-2p_i + p_i{}^2) - (2-2q_i) + (2p_i - 2q_i p_i)\right] = \sum_{i=1}^{n-1} \left[p_i{}^2 - 2q_i p_i - 1 + 2q_i\right]$$
$$= f_q(p) - \sum_{i=1}^{n-1} (2q_i - 1)$$

Die Differenz zwischen $f_q(p)$ und $g_r(s)$ berechnet sich somit wie folgt:

$$f_q(p) - g_r(s) = \sum_{i=1}^{n-1} (2q_i - 1)$$

Diese Differenz wird i. Allg. von null abweichen. Sie ist aber unabhängig von den Prognosen, da sie nur von den q_i und nicht von den p_i oder s_i abhängt. Insbesondere ist damit die Differenz zwischen den Funktionswerten für die optimale und für eine suboptimale Prognose unabhängig davon, ob die Klassen auf- oder absteigend sortiert werden, denn:

$$f_q(q) - g_r(r) = f_q(p) - g_r(s) = \sum_{i=1}^{n-1} (2q_i - 1)$$

$$\rightarrow \quad f_q(p) - f_q(q) = g_r(s) - g_r(r) \qquad \textbf{(3-9)}$$

Da hier nur eine Intervallskala vorliegt, ist mit Formel (3-9) die gewünschte Unabhängigkeit von der Sortier- bzw. Kumulierrichtung nachgewiesen.

3.2.2.6. Das Maß M_{exp}

Abschließend sei das Maß M_{exp} endgültig definiert:

Definition 3-7:

Für die Zeitpunkte t_1 bis t_m gebe es eine Folge von Zufallsvariablen (X_t), deren gemeinsamer Wertebereich eingeteilt ist in die Intervalle I_1 bis I_n. Die zu den Zeitpunkten t_1 bis t_m eingetretenen (d. h. gemessenen) Werte seien mit x_1 bis x_m bezeichnet.

Das Prognoseverfahren V sagt für die einzelnen Zeitpunkte jeweils kumulierte Wahrscheinlichkeiten vorher. Für jeden Zeitpunkt t_j wird dabei ein Vektor p_j von (n-1) Wahrscheinlichkeiten $p_{j,1}$ bis $p_{j,n\text{-}1}$ vorhergesagt, für die im Idealfall gelten würde:

$$p_{j,i} = \sum_{k=1}^{i} P\left(X_{t_j} \in I_k\right)$$

Bei der Feststellung des Ereigniswertes x_j wird der vorhergesagte Vektor p_j wie folgt bewertet:

$$b(p_j, x_j) = \begin{cases} \sum_{i=1}^{n-1} p_{j,i}^2 - 2\sum_{i=k}^{n-1} p_{j,i} + \sum_{i=k}^{n-1} 1 & \text{falls } x_j \in I_k \ (1 \le k < n) \\ \sum_{i=1}^{n-1} p_{j,i}^2 & \text{sonst} \end{cases}$$

Für die Bewertung des verwendeten Prognoseverfahrens V bzgl. der vorhergesagten und gemessenen m Werte wird schließlich **das Maß M_{exp}** wie folgt definiert:

$$M_{\text{exp}} = \frac{\sum_{j=1}^{m} b(p_j, x_j)}{m}$$

Offensichtlich weicht die in Definition 3-7 verwendete Funktion b etwas von dem in Formel (3-8) eingeführten Schätzer s_f ab. So liefert die Funktion b je nach getroffenem Intervall k einen um (n–k) höheren Wert als s_f. Die bereits festgestellten wünschenswerten Eigenschaften der Bewertungsfunktion bleiben dadurch aber unberührt. Im Gegensatz zu s_f liefert die Funktion b jedoch für die Prognose, welche der getroffenen Klasse k die volle Wahrscheinlichkeit zuordnet, als Ergebnis eine 0. Für alle anderen Prognosen werden durch b positive Werte geliefert. Auf diese Weise gibt es bei b im Gegensatz zu s_f eine harte Schranke für die optimalen Ergebnisse, da negative Ergebnisse nicht auftreten können. Deshalb liegt dem Maß M_{exp} auch b und nicht s_f zu Grunde.

Es sei jedoch darauf hingewiesen, dass der Wert 0 dennoch kein Nullpunkt im Sinne einer Verhältnisskala ist, weil für ein Prognoseverfahren, welches regelmäßig die Bewertung 0 erhält, Informationen aus der Zukunft notwendig wären, ein solches Verfahren also nicht erreichbar ist. Aufgrund der nicht ermittelbaren „realen“ Verteilungen

sind daher aber leider keine Aussagen darüber möglich, um wie viel Prozent die Prognosefehler mit einem bestimmten Verfahren verringert werden können.

Noch nicht festgelegt wurde die Zahl der Intervalle, also die Größe des Parameters n (vgl. Definition 3-7). Ein größeres n sorgt für eine genauere Beurteilung der Genauigkeit der Prognosen, mit einem kleineren n ist die Berechnung des Maßes M_{exp} weniger aufwendig. In den im Rahmen der vorliegenden Arbeit durchgeführten empirischen Untersuchungen wurde stets mit einem Wert von $n = 100$, also mit 100 Intervallen gearbeitet.

3.2.3. Zusammenfassung

Zusammenfassend sei festgestellt, dass im Folgenden die Vorhersageverfahren nach zwei Kriterien beurteilt werden sollen. Zum einen wird die Genauigkeit im Durchschnitt beurteilt. Zu diesem Zweck wird das Maß M_{avg} eingesetzt und es wird zusätzlich mittels $\Delta(p)$ die Einhaltung wichtiger Perzentile gemessen. Zum anderen wird unter Verwendung des Maßes M_{exp} die Punktgenauigkeit beurteilt. Ein gutes Prognoseverfahren sollte nach Möglichkeit bzgl. beider Kriterien gute Resultate liefern, da mit den beiden Kriterien zwei verschiedene wünschenswerte Eigenschaften geprüft werden.[1]

Das Besondere an den hier vorgeschlagenen Bewertungskriterien ist, dass sie auch bei stochastischen Prozessen mit wechselnden realen Verteilungen anwendbar sind. In anderen Arbeiten zur Bewertung von prognostizierten Verteilungen wird hingegen typischerweise (implizit) die Unveränderlichkeit dieser Verteilungen angenommen. Dies ist insbesondere in dem zur vorliegenden Arbeit ähnlichsten, in Kapitel 1 genauer beschriebenen Ansatz aus [Wyck98] der Fall.

[1] Man könnte einwenden, dass ein punktgenaues Verfahren auch automatisch im Durchschnitt genau ist. Allerdings geht es hier nicht um Ja/Nein-Fragen, ob die Punkt- oder Durchschnittsgenauigkeit erreicht ist, da diese Fragen von vornherein schon mit Nein beantwortet werden könnten. Vielmehr geht es darum, wie weit ein Verfahren von diesen wünschenswerten Eigenschaften entfernt ist. Das Maß M_{exp} eignet sich jedoch nicht zur Einschätzung der Genauigkeit im Durchschnitt – und umgekehrt eignet sich das Maß M_{avg} auch nicht zur Einschätzung der Punktgenauigkeit.

4. Verfahren zur Prognose freier Rechenkapazitäten

In den vorangegangenen Kapiteln wurde motiviert, warum freie Rechenkapazitäten prognostiziert werden sollten, welche Informationen diese Prognose liefern sollen und mit welchen Maßen die Güte dieser Informationen bewertet werden kann. Darauf aufbauend wird in diesem Kapitel auf die eigentliche Untersuchung möglicher Prognoseverfahren eingegangen. Zu diesem Zweck wird zunächst in Abschnitt 4.1 die für den empirischen Vergleich gewählte Vorgehensweise erläutert und begründet. Die Darstellung und Entwicklung möglicher Prognoseverfahren ist Gegenstand von Abschnitt 4.2. Abgeschlossen wird das Kapitel in Abschnitt 4.3 mit einer Übersicht über die Ergebnisse, wie sie mit dem dargelegten Vorgehen für die in Betracht gezogenen Prognoseverfahren erzielt wurden.

4.1. Vorgehen zum Vergleich der Prognoseverfahren

Die relativen Vorzüge zwischen den im Rahmen der vorliegenden Arbeit untersuchten Prognoseverfahren freier Rechenkapazitäten lassen sich nicht theoretisch herleiten bzw. beweisen. So besteht bei all diesen Verfahren die Annahme darin, dass sich das in der Vergangenheit beobachtete Verhalten in ähnlicher Weise auch in der Zukunft einstellen wird. Jedoch ist die Grundannahme des ähnlichen Verhaltens in Zukunft und Vergangenheit nicht beweisbar. Daher kann es nur darum gehen zu prüfen, inwieweit die Prognosen bei realen Rechnern zutreffend sind. Folglich kommt für eine Beurteilung der verschiedenen Prognoseverfahren nur ein empirischer Vergleich in Frage. Die für diesen Vergleich gewählte Vorgehensweise ist Gegenstand dieses Abschnitts.

4.1.1. Verwendung von Log-Dateien

Zum empirischen Vergleich der Prognoseverfahren wurden Log-Dateien genutzt, die Informationen über den zeitlichen Verlauf der freien Rechenkapazitäten verschiedener Rechner beinhalteten. Für die Nutzung dieser Log-Dateien sprechen im Wesentlichen zwei wichtige Gründe:

- Es wird mit Daten realer Rechner gearbeitet. Die dargestellten Ergebnisse beruhen also nicht auf willkürlichen Annahmen bzgl. der typischen Auslastung von Rechnern, sondern auf in der Praxis beobachtetem Verhalten.
- Die Nutzung bereits erhobener Log-Dateien erlaubt es, unterschiedliche Prognoseverfahren unter *exakt* denselben Bedingungen miteinander zu vergleichen.

Unterteilung der Log-Dateien

Ein wichtiger Punkt bei der Verwendung empirischer Daten zur Entwicklung und Bewertung irgendwelcher Verfahren ist die Unterteilung der Daten in zwei Gruppen. Die erste Gruppe von Daten wird für die Anpassungsphase benötigt (vgl. Abbildung 4-1). In dieser Phase wird nach geeigneten Verfahren gesucht und es wird versucht, die in Frage kommenden Verfahren durch geeignete Anpassungen zu optimieren. Grundlage für dieses Vorgehen sind die Daten der genannten ersten Gruppe, die ggf. Rückschlüsse auf vorzunehmende Änderungen zulassen, deren Erfolg dann wiederum anhand der Daten dieser ersten Gruppe gemessen werden kann. Ergebnis der Anpassungen sind schließlich Verfahren, die gegenüber alternativen Herangehensweisen Vorteile bieten sollen.

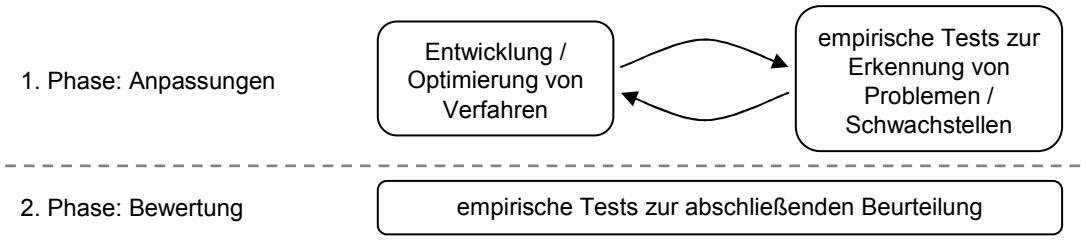

Abbildung 4-1: Phasen der Anpassung und der Bewertung

Die Beurteilung, ob die vorgeschlagenen Verfahren tatsächlich gegenüber anderen Verfahren vorteilhaft sind, lässt sich jedoch nicht anhand der Daten dieser ersten Gruppe vornehmen (vgl. bspw. [Abe01], Abschnitt 1.4). Das Problem ist eine möglicherweise auftretende Überanpassung (engl.: overfitting). Schließlich wird in der Anpassungsphase absichtlich nach Verfahren gesucht, die für die zur Verfügung stehenden Daten möglichst gute Werte liefern. Dabei besteht die Gefahr, dass diese Verfahren nur für die den Anpassungen zu Grunde liegenden Daten gute Ergebnisse liefern, für andere, typischerweise auftretende Daten aber gerade nicht. Eine Überanpassung lässt sich jedoch nicht entdecken, wenn die abschließende Untersuchung mit den gleichen Daten wie bei der Anpassung durchgeführt wird. Ein Beispiel einer Überanpassung ist in Abbildung 4-2 dargestellt.[1]

Aus den genannten Gründen muss die abschließende Beurteilung der in Frage kommenden Verfahren anhand anderer Daten erfolgen. Erst wenn diese Daten die Ergebnisse aus der Anpassungsphase bestätigen, ist dies ein Beleg für die Vorteilhaftigkeit der vorgeschlagenen Verfahren. Dementsprechend wurden die einzelnen im Rahmen dieser

[1] In dem Beispiel ist die empirische Bestimmung der Zeitdauern des freien Falls im Vakuum dargestellt. Die schwarzen Punkte stellen die Messwerte dar, welche mit gewissen Fehlern behaftet sind. Die grüne (dicke) Linie ist eine Trendlinie mit relativ einfachem Verlauf. Sie kommt den exakten Werten schon sehr nahe, obwohl sie von den meisten gemessenen Werten mehr oder weniger deutlich abweicht.

Die rote (dünne) Linie hat eine deutlich kompliziertere Form, durch die sie die aufgenommenen Messwerte exakt trifft. Dennoch gibt sie den tatsächlichen Verlauf der exakten Fallhöhen weniger gut wieder. Somit handelt es sich bei dieser Linie um eine Überanpassung.

Arbeit verwendeten Log-Dateien entweder für die Anpassungsphase oder für die anschließende Bewertungsphase eingesetzt.

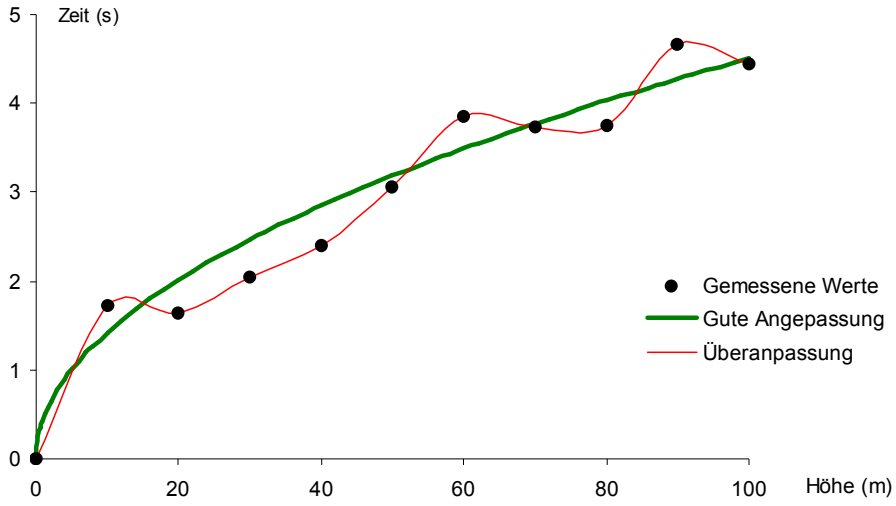

Abbildung 4-2: Mögliche Anpassungen von Trendlinien für Messwerte der Zeitdauern des freien Falls

4.1.2. Erstellung eigener Log-Dateien

In Ermangelung der ausreichenden Verfügbarkeit öffentlich zugänglicher und für die hier untersuchte Problemstellung geeigneter Log-Dateien wurden im Rahmen der vorliegenden Arbeit auf verschiedenen Rechnern über längere Zeiträume (viele Monate lang) Daten zur Auslastung von CPU und Arbeitsspeicher erhoben. Für die Erhebung dieser Daten wurden Tools verwendet, die im Wesentlichen von den entsprechenden Betriebssystemen (Windows, Linux, Solaris) zur Verfügung gestellt werden. Genauer beschrieben ist dies in den folgenden Unterabschnitten.

Erstellung von Log-Dateien unter Windows

Unter dem Betriebssystem Windows (Windows 2000 bzw. Windows XP) gibt es den Performance Monitor. Mit diesem ist es möglich, entsprechende System-Daten zu protokollieren, während das System aktiv ist. Dabei werden in regelmäßigen Abständen die aktuellen Daten in entsprechende Log-Dateien geschrieben. Für die hier vorherzusagenden freien Rechenkapazitäten waren besonders die Prozessorauslastung und die Menge des verfügbaren Hauptspeichers interessant. Als Zeitabstand zwischen aufeinander folgenden Messungen wurde 10 Sekunden eingestellt, um möglichst genaue Werte zu bekommen. Das Logging startet automatisch beim Hochfahren des Systems und ist auch nicht an das Einloggen irgendwelcher Nutzer gebunden. Bei jedem Systemstart wird eine neue Datei für die Log-Daten angelegt. Zur Konfiguration der Protokollierung sind Administrator-Rechte auf dem entsprechenden Rechner erforderlich.

Erstellung von Log-Dateien unter Unix und Linux

Neben Windows-Rechnern wurden auch Maschinen untersucht, die mit Unix (Solaris) oder Linux betrieben werden. Bei diesen Betriebssystemen lässt sich die Protokollierung zwar nicht ganz so einfach einrichten wie unter Windows, dafür ist es aber sogar ohne Administrator- bzw. Root-Rechte möglich.

Ausgangspunkt ist das Tool `top`, welches in regelmäßigen Abständen Werte des Systems messen und ausgeben kann. Ebenso wie bei den Windows-Rechnern wurde dieser Abstand auf 10 Sekunden festgelegt. Die Ausgabe von `top` erfolgt normalerweise auf die Konsole, kann aber auch in eine Datei umgeleitet werden, was für die hier durchgeführte Protokollierung ausgenutzt wurde. Damit blieb nur noch das Problem des automatischen Startens beim Hochfahren des Systems. Dieses Problem wurde mit entsprechenden Cron-Jobs gelöst. Zu diesem Zweck wurde ein Script geschrieben, das jede Minute gestartet wird. Dieses Script prüft, ob das Schreiben in die Log-Datei bereits aktiviert wurde – wenn nicht, dann wird dies unverzüglich nachgeholt. Auf diese Weise wurde auch bei den Unix- und Linux-Maschinen erreicht, dass für jede Periode des Angeschaltetseins eines Rechners eine Datei angelegt wurde, welche die entsprechenden Auslastungswerte beinhaltet.

Rechner, für die eigene Log-Dateien erstellt wurden

Für die Erstellung eigener Log-Dateien musste auf zur Verfügung stehende Rechner zurückgegriffen werden. Dies waren zum Teil Privatrechner, vor allem aber Rechner der BTU Cottbus. Bei letzteren handelte es sich teilweise um Bürorechner, größtenteils jedoch um Rechner in Computerpools.

Für die Anpassungsphase wurde zunächst eine kleine Gruppe von nur vier Rechnern verwendet. Es wurde absichtlich eine solch kleine Gruppe gewählt, da die Menge an zur Verfügung stehenden Rechnern begrenzt war und die für die abschließende Überprüfung der Ergebnisse verwendete zweite Gruppe um so größer sein kann, je weniger Rechner in der Anpassungsphase verwendet werden. Ferner war es durch die geringe Zahl von Rechnern möglich, in vertretbarer Zeit eine Vielzahl von möglichen Verfahren zu testen. Durch die geringe Größe der Gruppe ist die Repräsentanz jedoch stark eingeschränkt. Um dennoch sinnvolle Schlüsse ziehen zu können, wurde auf eine möglichst große Heterogenität der Gruppe Wert gelegt. Diese Heterogenität sollte helfen einzuschätzen, inwieweit die relative Vorteilhaftigkeit bestimmter Verfahren unabhängig von den konkreten Charakteristika der Nutzung der Rechner ist. Diese Unabhängigkeit ist wichtig, weil im Rahmen dieser Arbeit nach Verfahren gesucht wurde, die auch für andere Rechner zu Verbesserungen führen würden. Gesucht wurde daher nach Optimierungen, die nach Möglichkeit die Resultate für alle Rechner der Gruppe verbessern. Eine Auflistung und Beschreibung der Rechner der ersten Gruppe kann Tabelle 4-1 entnommen werden.

Die Rechner der zweiten Gruppe sind in Tabelle 4-2 beschrieben. Für die überprüfenden Messungen wurde diese Gruppe noch einmal weiter unterteilt, wobei auf eine möglichst große Homogenität der Unterggruppen geachtet wurde. Die Auswertung erfolgt dann

für die einzelnen Untergruppen getrennt (vgl. Abschnitt 4.3). Das Ziel der Einteilung in homogene Untergruppen bestand darin zu prüfen, ob die vorgeschlagenen Prognoseverfahren tatsächlich relativ allgemeingültig zu Verbesserungen führen. Bei einer Betrachtung des durchschnittlichen Verhaltens über alle Rechner könnten sonst leicht evt. vorhandene Nachteile der vorgeschlagenen Verfahren bei bestimmten Rechnern durch die bei anderen Rechnern vorhandenen Vorteile verdeckt werden.

Rechner	Betriebssystem	Prozessor	Hauptspeicher (MB)
Bassanio	Windows 2000	AMD Athlon XP 1800+	512
Capulet	Red Hat 4.1.1-51 (Fedora Kernel 2.6.20-1)	2 x Intel Xeon 2,66 GHz	2.048
Ep	Solaris 9, 64 Bit	4 x UltraSPARC-II 450MHz (Sun Ultra 80)	1.024
Lap	Windows XP Professional	Intel Celeron 2,2 GHz	768

Tabelle 4-1: Für die Anpassungsphase verwendete Rechner

Untergruppe	Rechner	Betriebssystem	Prozessor	Hauptspeicher
Pool-Rechner	Pool R310 Linux (20 Rechner)	Red Hat 3.4.4-2 (Fedora Kernel 2.6.12-1)	AMD Athlon 64 3500+	2.048 MB
	Pool R310 Windows (20 Rechner)	Windows XP	AMD Athlon 64 3500+	2.048 MB
	Pool R411 (6 Rechner)	Red Hat 4.1.1-51 (Fedora Kernel 2.6.20-1)	2 x Intel Xeon 2,66 GHz	2.048 MB
PCs	Enterprise	Windows XP	AMD Athlon 64 X2 Dual Core Processor 4200+	2.048 MB
	Han	Windows 2000	AMD Athlon 1,4 GHz	1.024 MB
	Amidala	Windows XP	AMD Athlon 64 X2 Dual Core Processor 3800+	2.048 MB
	Executor	Windows XP	AMD Athlon 64 X2 Dual Core Processor 3800+	2.048 MB
Server	Mouse	Solaris 9, 64 Bit	2 x UltraSPARC-III+, 900 MHz (Sun Fire 280R)	4.096 MB

Tabelle 4-2: Für die überprüfenden Messungen verwendete Rechner

Anzumerken ist, dass die Unterteilung im Falle der zweiten und dritten Untergruppe zu Gruppengrößen führt, die für sich genommen für eine Auswertung zu klein sind. Allerdings geht es hier darum Verfahren zu finden, die möglichst universell funktionieren. Dementsprechend sollen diese Untergruppen vor allem dazu dienen, das Gesamtergebnis zu überprüfen und weniger dazu, für die einzelnen Untergruppen unterschiedliche Schlussfolgerungen zu ziehen. Insofern führt die vorgenommene Aufteilung also zu vorsichtigeren Schlussfolgerungen als eine Gesamtbetrachtung aller Rechner der zweiten Gruppe. Darüber hinaus ist anzumerken, dass für die einzelnen Rechner die Daten über über besonders lange Zeiträume von vielen Monaten erhoben wurden. Im

Falle des Servers Mouse wurde die Protokollierung sogar über einen Zeitraum von fast drei Jahren durchgeführt, so dass die Zeitreihe in vier (immer noch sehr lange) Zeitreihen aufgeteilt werden konnte. Wie weiter unten noch deutlich werden wird, sind diese langen Zeiträume günstig, damit die Prognosegüte anhand vieler verschiedener Prognosen geprüft werden kann (und nicht nur anhand einer einzelnen Prognose pro Rechner).

4.1.3. Nutzung fremder Log-Dateien

Neben den bereits beschriebenen Rechnern, für welche die Log-Dateien extra angelegt wurden, stellte sich die Frage, ob auch öffentlich verfügbare Informationen zu den Workloads von Rechnern für die Überprüfung der Verfahren genutzt werden könnten. Leider gibt es im Internet nur wenige solcher Datensätze, die darüber hinaus für das hier untersuchte Problem aus unterschiedlichen Gründen problematisch sind.

Eines der verfügbaren Archive ([Dind98]) enthält die Daten, wie sie in der in [Dind00] beschriebenen Untersuchung der kurzfristigen Vorhersagbarkeit der Verfügbarkeit von Rechnern verwendet wurden. Das Archiv umfasst die Daten von 38 Rechnern. Da es bei der angeführten Untersuchung nur um die Vorhersage für wenige Sekunden ging, mussten die entsprechenden Log-Daten für diese Rechner nur für wenige Tage erhoben werden. Damit sind die entsprechenden Informationen aber leider für die in der vorliegenden Arbeit durchgeführten Untersuchungen nicht hilfreich.

Das vielleicht bekannteste Archiv für Last-Informationen ist das Parallel Workloads Archive ([Feit05]). In diesem befinden sich für eine Reihe von Massenparallelrechnern Informationen über die auf diesen Maschinen gestarteten Jobs. Zu diesen Informationen zählen bspw. die Startzeiten der Jobs, die Wartezeiten, die Anzahl der verwendeten Prozessoren usw. Auch dieses Archiv ist für die vorliegende Arbeit nicht hilfreich. Zum einen geht es hier vor allem auch darum, die verfügbaren Rechenkapazitäten bis zum nächsten Ausschalten vorherzusagen. Entsprechende Informationen über den Zustand der einzelnen Maschinen bzw. Prozessoren (ob sie an- oder ausgeschaltet sind) gehen aus den Archiv-Daten nicht hervor. Zum anderen ist anzumerken, dass die meisten der Datensätze nur Informationen über wenige Tage liefern, was für die Bewertung der hier untersuchten Vorhersagemethoden nicht ausreichend ist.

Eine sehr umfangreiche Ermittlung von Daten ist in [Long95] beschrieben. Demnach wurde Mitte der 90iger Jahre für etwa 1000 Internet-Rechner ermittelt, zu welchen Zeiten diese verfügbar sind und zu welchen nicht. Zu diesem Zweck wurden die Rechner in relativ kurzen Abständen immer wieder angesprochen, um die Verfügbarkeit zu überprüfen. Daraus abgeleitet wurden die Zeitspannen der ununterbrochenen Verfügbarkeit. Die hier gesuchte Größe K_{free} lässt sich leider aus den Zeitspannen nicht exakt ableiten. Allerdings kann man eine Abschätzung von K_{free} vornehmen, indem man für die Zeiten, zu denen die Rechner angeschaltet waren, eine praxisnahe Auslastung dieser Rechner annimmt. Entsprechend diesbezüglicher Angaben in der Literatur ([Andr02], [HP07], [Röhr04], [Schä07]) und der Beobachtungen an den eigenen Rechnern erscheint eine Auslastung um die 10 Prozent realistisch. Natürlich lässt sich

einwenden, dass die Daten schon relativ alt sind und außerdem die Abschätzungen für die Größe K_{free} recht grob sind, da sie nur auf den Zeiten des Angeschaltetseins basieren, nicht aber auf Messungen der tatsächlichen Auslastung. Aus zwei Gründen wurden die Daten trotzdem verwendet. Zum einen geht es hier nur um zusätzliche Simulationen neben denen, die auf den selbst erstellten (und für den hier verfolgten Zweck informativeren) Log-Dateien basieren. Zum anderen können die Daten aus [Long95] wertvolle Hinweise liefern um einzuschätzen, ob die in der vorliegenden Arbeit gemachten Vorschläge zur Prognoseerstellung allgemein zu Verbesserungen führen oder nur bei wenigen, speziellen Nutzungsprofilen.[1]

4.1.4. Auswertung der Log-Dateien

Um in Frage kommende Prognoseverfahren vergleichen zu können, mussten die erstellten Log-Dateien schließlich ausgewertet werden. Zu diesem Zweck wurde mittels Java ein Tool entwickelt, mit dem diese Auswertung durchgeführt werden kann. Der grundlegende Aufbau des Tools folgt dem Pipes-and-Filters-Muster ([Busc00]), so wie es in der folgenden Abbildung 4-3 dargestellt ist.

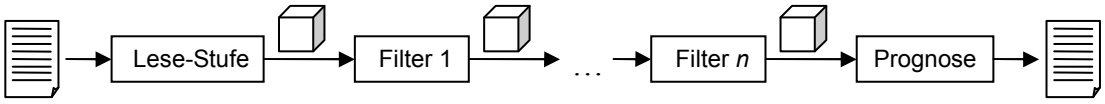

Abbildung 4-3: Verarbeitung der Log-Dateien

Zunächst werden die Informationen aus den Log-Dateien durch die Lese-Stufe in einzelne Datenobjekte umgewandelt (in der Grafik als Würfel dargestellt), die jeweils die Informationen für einen Zeitpunkt besitzen. Für die unterschiedlichen Betriebssysteme gibt es dabei jeweils angepasste Lese-Stufen, da auch die Log-Dateien einen jeweils unterschiedlichen Aufbau haben. Die von den Lese-Stufen erzeugten Objekte sind jedoch bereits unabhängig vom genutzten Betriebssystem.

Die Objekte werden dann durch verschiedene Filterstufen weiter aufgearbeitet, bevor die eigentliche Prognose durchgeführt wird. Die Auswahl, die Reihenfolge und das Verhalten dieser Filterstufen ist in dem erwähnten Tool leicht über entsprechende Einstellungen in der Konfigurationsdatei veränderbar. Die für die Messungen letztlich verwendete Konfiguration ist im Folgenden beschrieben.

Behandlung von Zeiten des Ausgeschaltetseins

Die erste Filterstufe ist dafür verantwortlich, für die Zeiten, zu denen die Rechner ausgeschaltet waren, zusätzliche Datenobjekte einzufügen, damit die zur nächsten Filterstufe gelangenden Datenobjekte jeweils einen konstanten zeitlichen Abstand von

[1] An dieser Stelle möchte ich Prof. Jim Plank und Prof. Darrell Long dafür danken, mir freundlicherweise die erhobenen Daten zur Verfügung gestellt bzw. deren Verwendung gestattet zu haben.

10 Sekunden aufweisen (vgl. Abschnitt 4.1.2). Durch das Einfügen dieser Objekte ist die weitere Verarbeitung einfacher. Da ein ausgeschalteter Rechner nicht für Grid-Jobs nutzbar ist, weisen die zusätzlichen eingefügten Objekte eine freie Rechenkapazität von 0 aus. Man könnte zwar argumentieren, dass im Rahmen des Grid-Computings genutzte Einzelrechner nicht mehr zum Ende der Phasen der primären Nutzung abgeschaltet würden und daher entgegen den Angaben der Log-Dateien ununterbrochen zur Verfügung stehen könnten. Aus verschiedenen Gründen wird diese Sichtweise hier jedoch nicht vertreten.

So wurde in Kapitel 1 auf den Stromverbrauch von Rechnern eingegangen. Es wurde deutlich (vgl. Abbildung 2-2), dass heutige Rechner auch im Idle-Zustand einen beträchtlichen Energieverbrauch aufweisen. Dies bedeutet, dass nicht unerhebliche Kosten entstehen, wenn Rechner einfach angelassen werden, um für Grid-Jobs zur Verfügung zu stehen, die dann aber vielleicht gar nicht kommen. Neben dem zusätzlichen Energiebedarf verursachen laufende Rechner auch Lärm, der zumindest in einem von Personen genutzten Zimmer nicht unbedingt akzeptiert wird, nur um auf *möglicherweise* eintreffende Grid-Jobs zu warten. Diese Lärmbelästigung dürfte für privat genutzte PCs relevant sein – aber auch für Rechner in einem gemeinsam genutzten Büro.

Ferner ist es nicht immer sinnvoll, alle zur Verfügung stehenden Rechner in einem Netz von Einzelrechnern für Grid-Jobs zu verwenden. Das ist vor allem dann der Fall, wenn die Grid-Jobs eine relativ umfangreiche Netzwerkkommunikation benötigen, da in diesem Fall das Netzwerk bzw. die Internet-Verbindung ggf. so stark ausgelastet wäre, dass die laufenden Grid-Jobs stark ausgebremst würden. Unter solchen Voraussetzungen wäre es wenig sinnvoll, die vorhandenen Rechner nur deshalb anzulassen, um auf Grid-Jobs zu warten.

Ein weiteres Problem ergibt sich bei Rechnern, auf denen zwei Betriebssysteme parallel installiert sind (bspw. Linux und Windows), zwischen denen nur durch einen Neustart gewechselt werden kann. Von den in dieser Arbeit untersuchten Rechnern weist eine ganze Reihe eine solche Konfiguration auf. Bei einem solchen Rechner kann bei einer bspw. unter Windows laufenden Grid-Software nicht davon ausgegangen werden, dass der Rechner zu Zeiten, zu denen Windows nicht läuft, für das Grid-Computing genutzt werden kann.

Schließlich ist es i. Allg. schwierig festzustellen, warum ein Rechner ausgeschaltet wurde. Auch die im Rahmen der vorliegenden Arbeit erstellten Log-Dateien geben darüber keine Auskunft. Neben dem Ausschalten zum Ende der eigentlichen Nutzungsdauer kommen auch Soft- oder Hardwarefehler in Frage. Ebenso sind gelegentlich Neustarts notwendig, um die Systemsoftware oder die Hardware zu aktualisieren.

Aufgrund der genannten Probleme werden in der vorliegenden Arbeit nur ungenutzte Rechenkapazitäten zu solchen Zeiten gezählt, zu denen eine Maschine auch angeschaltet war. Zu allen anderen Zeiten wird die Maschine als nicht verfügbar gewertet.

Weitere Filterstufen

Eine weitere Filterstufe wurde eingeführt, um ungenutzte Prozessorzyklen nur dann als verfügbar anzusehen, wenn gleichzeitig eine gewisse Mindestmenge an Speicher vorhanden war. Schließlich könnte ein Grid-Job wenig mit den Prozessorzyklen anfangen, wenn der Programmcode und die benötigten Daten sich nicht im Hauptspeicher befinden können. Für die hier durchgeführten Tests wurde jeweils eine Mindestmenge von wenigstens 256 MB freiem Speicher gefordert, andere Grenzen wurden im Rahmen dieser Arbeit nicht untersucht. Es wurde also insbesondere nicht untersucht, inwieweit andere Grenzen zu anderen Ergebnissen führen.[1] Für all die Datenobjekte, in denen weniger freier Hauptspeicher angegeben war, wurde die Menge an verfügbaren Prozessorzyklen auf 0 gesetzt.

Eine weitere Filterstufe war nötig, um die verfügbaren Rechenzyklen zu kumulieren. Schließlich war den Log-Dateien zunächst für die einzelnen Zeitpunkte nur zu entnehmen, wie ausgelastet der Rechner zu diesen Zeitpunkten war. Entsprechend der in Definition 3-1 eingeführten Größe $K_{\text{free},i}$ waren jedoch die freien zusammenhängenden Rechenkapazitäten in den nächsten i Stunden zu betrachten. Bei den Messungen wurde für i mit Werten von 6, 12 und 24 gearbeitet, so dass die maximalen Zeitspannen der Kumulation 6, 12 oder 24 Stunden lang waren. Die Kumulation hatte zur Folge, dass ein einem bestimmten Zeitpunkt t zugeordnetes Datenobjekt O_t bereits Informationen aus dem Zeitraum von t bis zu i Stunden später besaß, so wie es in Abbildung 4-4 dargestellt ist.

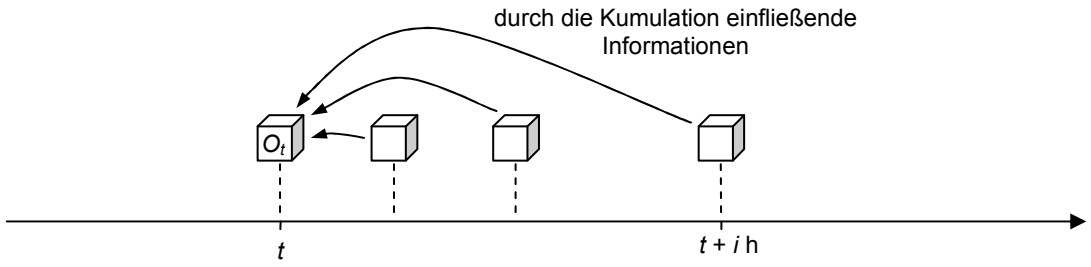

Abbildung 4-4: Mögliche Informationen aus der Zukunft in einem Datenobjekt O_t

Schließlich wurde zum Zwecke einer effizienten Bearbeitung in der letzten Filterstufe eine Zusammenfassung der Werte durchgeführt, so dass ein einzelnes Datenobjekt nicht mehr nur noch die Eigenschaften von 10 Sekunden repräsentiert, sondern die einer Stunde.

[1] Es ist allerdings anzumerken, dass es hinsichtlich der Gesamtmenge an Hauptspeicher pro Rechner erhebliche Unterschiede gab. Damit ergaben sich mit der grundsätzlich geforderten Menge von 256 MB an freiem Hauptspeicher bereits unterschiedlich große relative Anteile an gefordertem Hauptspeicher, die jedoch nicht zu sichtbaren Unterschieden hinsichtlich der Prognosegüte führten. Dies ist ein Indiz (aber kein Beweis) dafür, dass bei der Wahl anderer Mindestmengen an Hauptspeicher sich die Ergebnisse nicht grundlegend ändern würden.

Prognoseerstellung und -bewertung

Erst nach der so erfolgten Aufbereitung der Werte fand schließlich die Prognoseerstellung und -bewertung statt. Diese lief so ab, dass die Zeitreihe mit den kumulierten verfügbaren Prozessorzyklen als Ausgangspunkt genommen wurde, so wie sie durch die zusammengefassten Datenobjekte repräsentiert wurde. Mit den untersuchten Prognoseverfahren wurden dann zu den einzelnen Zeitpunkten der Zeitreihe jeweils Prognosen erstellt. Dabei musste darauf geachtet werden, dass nur Informationen verwendet wurden, die zum Zeitpunkt der Prognoseerstellung auch bereits vorlagen. Zur Erklärung dieses Punktes sei noch einmal auf Abbildung 4-4 verwiesen. So enthält das dargestellte Datenobjekt O_t Informationen von Zeitpunkten, die bis zu i Stunden in der Zukunft liegen. Aus diesem Grunde kann dieses Objekt erst bei Prognoseerstellungen ab ($t + i$ h) eingesetzt werden.

Schließlich wurden die für einen Rechner und ein bestimmtes i zu den einzelnen Zeitpunkten erstellten Prognosen mit den tatsächlich gemessenen Rechenkapazitäten verglichen und entsprechend der in Abschnitt 3.2 eingeführten Maße bewertet. Auf diese Weise wurde für M_{exp} und M_{avg} jeweils ein Wert ermittelt. Ebenso ergab sich auch für jedes untersuchte p jeweils ein Messwert für $|\Delta(p)|$. Eingeflossen sind dabei aber nur jene Zeitpunkte, zu denen der Rechner auch angeschaltet war. Der Grund dafür ist der, dass zu Zeiten des Ausgeschaltetseins die Größe K_{free} per Definition den Wert 0 annimmt, eine spezielle Prognose also gar nicht erforderlich ist.

Offensichtlich erlauben bei dieser Vorgehensweise längere Zeitreihen zuverlässigere Bewertungen, da bei einer langen Zeitreihe die Bewertung nicht nur von der (mehr oder weniger zufälligen) Güte einer einzelne Prognosen abhängt, sondern von der Güte vieler, für unterschiedliche Zeitpunkte abgegebenen Prognosen. Dies ist auch der Grund, aus dem nur solche Log-Dateien verwendet wurden, die Informationen über mehrere Monate hinweg beinhalten.

4.1.5. Zusammengefasste Darstellung der Messergebnisse

Die Ergebnisse für die im folgenden Abschnitt 4.2 diskutierten Varianten werden in diesem Kapitel jeweils durch entsprechende Balkendiagramme dargestellt. Die zugrunde liegenden genauen Werte sind in Anhang D dokumentiert. In diesem sind die mit der skizzierten Vorgehensweise ermittelten Ergebnisse für die einzelnen Gruppen von Rechnern zusammengefasst dargestellt, wobei für die verschiedenen Prognoseverfahren jeweils die Durchschnittswerte von M_{exp}, von M_{avg} sowie von $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ gebildet wurden. Der Durchschnittsberechnung zugrunde liegen die pro Rechner und pro Wert von i ermittelten Einzelwerte, wobei für i Werte von 6, 12 und 24 gewählt wurden. Veranschaulicht ist diese Vorgehensweise in Abbildung 4-5. Bei n Rechnern erfolgt die Mittelwertbildung demnach für jede der fünf Messgrößen aus $3 \cdot n$ Einzelwerten.

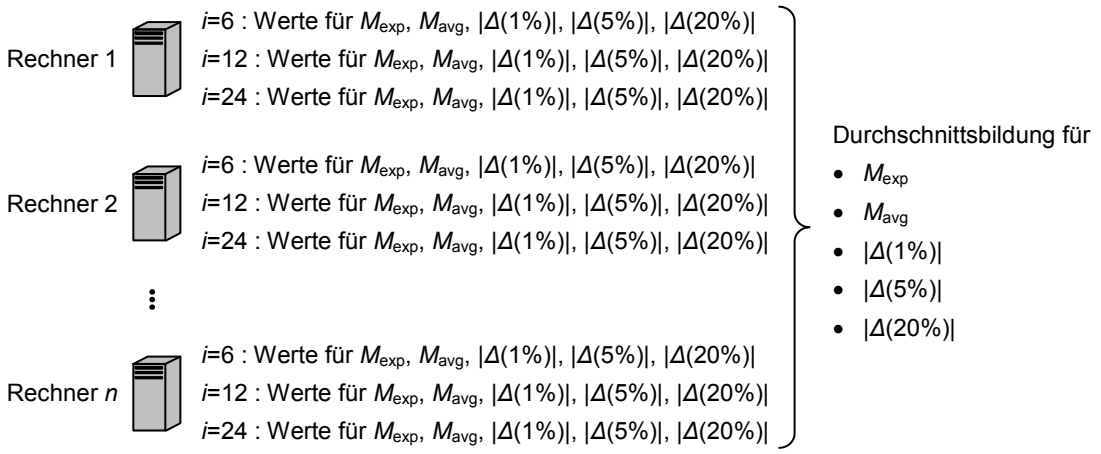

Abbildung 4-5: Zusammengefasste Darstellung der Messergebnisse nach Durchschnittsbildung

4.2. Prognoseverfahren für K_{free}

Mittels der in den vorangegangenen Abschnitten vorgestellten Vorgehensweise sollen nun potentielle Vorhersageverfahren miteinander verglichen werden. Die grundlegende Annahme ist dabei, dass die in der Vergangenheit beobachteten Häufigkeitsverteilungen entsprechende Rückschlüsse auf die künftigen Wahrscheinlichkeiten zulassen. Für die genaue Ableitung der Wahrscheinlichkeiten sind jedoch zumindest zwei Punkte zu berücksichtigen. Zum einen ist bei jeder Prognose zu entscheiden, welche Werte aus der Vergangenheit berücksichtigt werden sollen. Dieser Punkt wird im folgenden Unterabschnitt 4.2.1 diskutiert. Zum anderen stellt sich die Frage, welche Schlüsse aus den ausgewählten Werten der Vergangenheit zu ziehen sind. Dieser Punkt wird ausführlich in den Abschnitten 4.2.2 und 4.2.3 besprochen.

4.2.1. Zuordnung der Messwerte zu Häufigkeitsverteilungen

Bei der Auswahl der für eine Prognose zu berücksichtigenden gemessenen Werte der Größe K_{free} ist zum einen eine Entscheidung über den Zeitraum zu treffen. Diesbezüglich wurden im Rahmen der vorliegenden Arbeit keine größeren Untersuchungen zur Optimierung vorgenommen. Stattdessen werden die Werte aus den letzten zehn Wochen berücksichtigt, da sich diese Zeitspanne in anfänglichen Tests als vergleichsweise vorteilhaft erwies. Im Gegensatz zur Vorgehensweise in [Wyck98] werden die Werte also nicht aus einem einmal festgelegten Zeitraum genommen, sondern aus einem sich kontinuierlich verschiebenden Zeitraum.

Neben der Wahl des Zeitraums stellt sich die Frage, ob man alle zu einem Rechner in den letzten zehn Wochen ermittelten Werte für K_{free} verwenden sollte. Alternativ ist es denkbar, verschiedene Gruppen bzw. Häufigkeitsverteilungen zu bilden und die einzelnen Werte nach bestimmten Kriterien diesen Häufigkeitsverteilungen zuzuordnen. Veranschaulicht sind die beiden Varianten in Abbildung 4-6: In der oberen Hälfte ist

dargestellt, wie aus allen Messwerten (Striche an der Abszisse) eine einzelne Häufigkeitsverteilung gebildet werden kann (die Treppenfunktion). In der unteren Hälfte hingegen sind die gleichen Messwerte drei verschiedenen Häufigkeitsverteilungen zugeordnet worden, die sich mehr oder weniger stark unterscheiden. Für die Prognose zu einem bestimmten Zeitpunkt wird dann diejenige dieser Häufigkeitsverteilungen genutzt, in die schließlich auch der Messwert eingeordnet werden müsste.

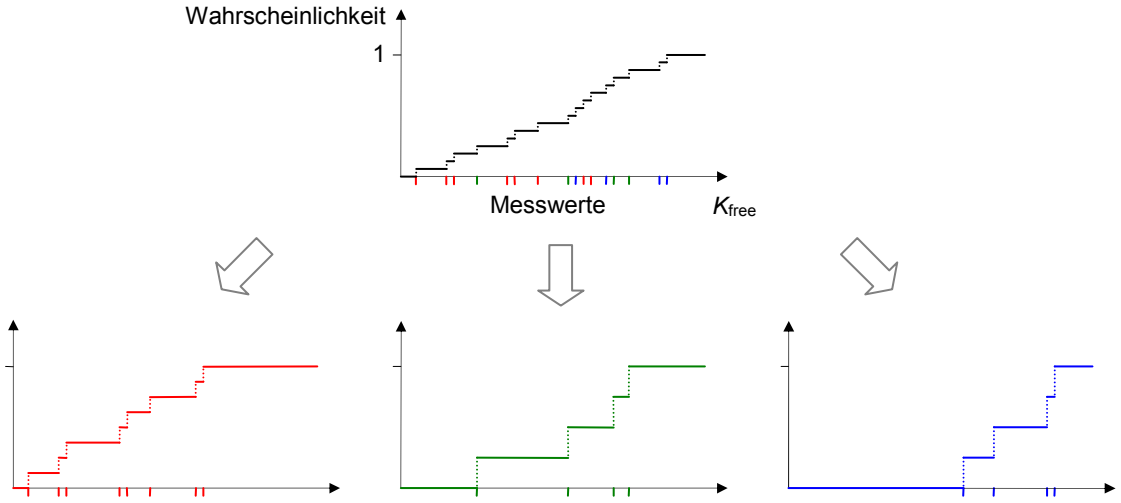

Abbildung 4-6: Aufteilung der Messwerte auf mehrere Häufigkeitsverteilungen

Voraussetzung für eine solche Aufteilung der Messwerte sowie die damit notwendige Auswahl einer der Verteilungen für die Prognose ist ein geeignetes Klassifizierungskriterium. Ganz allgemein ist eine Aufteilung auf verschiedene Häufigkeitsverteilungen nur sinnvoll, wenn sich diese deutlich voneinander unterscheiden. Anderenfalls ist eine Aufteilung eher nachteilig, da so pro Verteilung weniger Werte zur Verfügung stehen und sich daher Ausreißer stärker auswirken und auch insgesamt die Verteilung weniger repräsentativ ist. Auf in Frage kommende Klassifizierungskriterien wird in Abschnitt 4.2.1.2 genauer eingegangen.

4.2.1.1. Nutzung einer einzelnen Häufigkeitsverteilung

Vor der Betrachtung geeigneter Kriterien zur Unterscheidung verschiedener empirischer Häufigkeitsverteilungen soll zunächst auf Varianten mit nur einer einzigen Häufigkeitsverteilung eingegangen werden. Dies ist auch deshalb sinnvoll, um bei den komplexeren Varianten einen Vergleich zu haben und so ggf. entscheiden zu können, ob die Varianten mit mehreren Häufigkeitsverteilungen überhaupt Vorteile bringen.

4.2.1.1.1. C_{simple}: Ansatz mit einer einzigen Häufigkeitsverteilung

Die einfachste Variante besteht darin, aus den in den letzten Wochen gemessenen Werten für K_{free} nur eine einzige empirische Häufigkeitsverteilung zu bilden. Die so gewonnene Häufigkeitsverteilung wird schließlich als Vorhersage für die reale Ver-

teilung genutzt. Dieses Klassifizierungskriterium (das im Grunde genommen gar keines ist) sei im Weiteren mit C_{simple} bezeichnet.

4.2.1.1.2. Ansatz aus [Wyck98]

Der Ansatz von [Wyck98] (vgl. Abschnitt 5.2) nutzt für die Prognose eine einzige Häufigkeitsverteilung pro Rechner, welche die Längen der Idle-Zeiten eines Monats beinhaltet. Für die Vorhersage der noch verbleibenden Idle-Zeit k wird dann aber (im Gegensatz zur gerade genannten Variante C_{simple}) mit bedingten Wahrscheinlichkeiten gearbeitet. Bei bereits verstrichener Idle-Zeit z ergibt sich die Wahrscheinlichkeit $P_{=k}$, dass die noch verbleibenden Idle-Zeit gleich k ist, gemäß der folgenden Formel, wenn X die Verteilung aller vollständigen Idle-Zeiten ist:

$$P_{=k} = \frac{P(X = k + z)}{P(X \geq z)} \qquad \textbf{(4-1)}$$

Während [Wyck98] nur die Idle-Zeiten eines Rechners betrachtet, werden in der vorliegenden Arbeit sämtliche ungenutzten Prozessorzyklen als verfügbar angesehen (vgl. Abschnitt 3.1.1). Dies impliziert, dass auch anteilige Nutzungen eines Rechners möglich sind, so dass die reine Zeitdauer einer zusammenhängenden Nutzbarkeit eines Rechners noch nichts über die Leistungsfähigkeit in diesem Zeitraum aussagt. Dies ist der Grund, aus dem in der vorliegenden Arbeit nicht Zeiträume, sondern freie Rechenkapazitäten vorhergesagt werden (vgl. Abschnitt 3.1.2).

Da die Angabe der freien Rechenkapazität ihrerseits wiederum nichts über den Zeitrahmen aussagt, werden nur Rechenkapazitäten innerhalb von i Stunden berücksichtigt. Diese Begrenzung führt dazu, dass die in [Wyck98] vorgeschlagene Vorgehensweise mit den bedingten Wahrscheinlichkeiten hier nicht anwendbar ist. Das Problem lässt sich an einem einfachen Beispiel erläutern, in dem ein Rechner jede Stunde konstant die gleiche Rechenkapazität von $K_{1\text{h}}$ liefert. Die empirische Verteilung von $K_{\text{free},i}$ würde dann für den Wert $i \cdot K_{1\text{h}}$ eine Wahrscheinlichkeit von 100 % ausweisen. Nach einer Laufzeit des Rechners von einer Stunde würde sich analog zu obiger Formel (4-1) zur bedingten Wahrscheinlichkeit für $K_{\text{free},i}$ aber entsprechend Formel (4-2) nur noch eine verbleibende Rechenkapazität von $(i–1) \cdot K_{1\text{h}}$ ergeben. Die Größe Y stehe dabei für die Verteilung aller Werte von $K_{\text{free},i}$ und $P_{=(i–1)\cdot K_{1\text{h}}}$ sei die Wahrscheinlichkeit, dass die noch verbleibende Rechenkapazität gleich $(i–1) \cdot K_{1\text{h}}$ ist.

$$P_{=(i-1)\cdot K_{1\text{h}}} = \frac{P\left(Y = (i-1)\cdot K_{1\text{h}} + K_{1\text{h}}\right)}{P(Y \geq K_{1\text{h}})} = \frac{P\left(Y = i \cdot K_{1\text{h}}\right)}{P(Y \geq K_{1\text{h}})} = \frac{1}{1} = 1 \qquad \textbf{(4-2)}$$

Nach einer Stunde betrüge $K_{\text{free},i}$ aber nicht $(i–1) \cdot K_{1\text{h}}$, sondern wiederum $i \cdot K_{1\text{h}}$, da die Rechenkapazität einer weiteren Stunde mitgezählt werden könnte. Aufgrund dieses Problems wird der Ansatz aus [Wyck98] hier nicht weiter verfolgt.

4.2.1.2. Nutzung mehrerer Häufigkeitsverteilungen

Als Alternative zur Variante C_{simple} bleibt somit die Verwendung eines echten Klassifizierungskritieriums für die in Abbildung 4-6 dargestellte Zuordnung der Messwerte zu Häufigkeitsverteilungen. Mögliche Kriterien werden in diesem Abschnitt diskutiert.

4.2.1.2.1. C_{uptime}: Unterscheidung nach der bisherigen Laufzeit des Rechners

Ein mögliches Unterscheidungskriterium, das hier C_{uptime} genannt wird, ist die bisherige Laufzeit des Rechners. Nach diesem Kriterium werden für unterschiedliche bisherige Laufzeiten getrennte empirische Verteilungen ermittelt. Da man nicht unendlich viele solcher Verteilungen verwalten kann, müssen ähnliche Laufzeiten zusammengefasst werden. Im Rahmen dieser Arbeit wurde mit einer Einteilung in 50 Verteilungen V_1 bis V_{50} experimentiert. Eine Verteilung V_i (mit $1 \le i \le 49$) steht dabei für bisherige Laufzeiten des Rechners von mindestens (i-1), aber weniger als i Stunden. Die Verteilung V_{50} steht für Laufzeiten von mindestens 49 Stunden. Mit einer solchen Einteilung können Laufzeiten von bis zu zwei Tagen unterschieden werden.

Vom Effekt her führt dieses Kriterium zu einem ähnlichen Ansatz wie dem aus [Wyck98], da in diesem die Prognosewahrscheinlichkeiten in Abhängigkeit von der bisherigen Idle-Zeit des Rechners berechnet wurden. Es wurde also eine signifikante Abhängigkeit zwischen bisheriger Idle-Zeit und verbleibender Idle-Zeit unterstellt. Analog wird bei Anwendung des Unterscheidungskritieriums C_{uptime} auf eine Abhängigkeit zwischen der bisherigen Laufzeit des Rechners und der freien Rechenkapazität innerhalb der nächsten Stunden gesetzt.

4.2.1.2.2. $C_{\text{hourOfWeek}}$: Unterscheidung der Stunden einer Woche

Während mit C_{uptime} eine Abhängigkeit von der bisherigen Laufzeit des Rechners berücksichtigt wird, soll mit $C_{\text{hourOfWeek}}$ die Abhängigkeit von der Zeit relativ zum Wochenanfang berücksichtigt werden. Die zugrunde liegende Annahme ist die, dass zum einen die einzelnen Tage typische Verläufe der Nutzung besitzen, zum anderen mit unterschiedlichen Verläufen an unterschiedlichen Wochentagen zu rechnen ist. Mit dem Unterscheidungskriterium C_{uptime} wird für jede Stunde der Woche eine andere Häufigkeitsverteilung geführt, so dass es insgesamt 168 verschiedene Häufigkeitsverteilungen gibt.

4.2.1.2.3. $C_{\text{hourOfDay}}$: Unterscheidung der Stunden eines Tages

Die Annahme bei $C_{\text{hourOfWeek}}$ ist die, dass unterschiedliche Tage signifikant verschiedene Nutzungen der Rechner ergeben. Wesentlicher Nachteil dieser Variante ist jedoch, dass sich eine große Zahl empirischer Verteilungen ergibt, so dass jede einzelne nur relativ wenige Werte der letzten Zeit enthalten kann. Für einen Mittelweg könnte es sinnvoll sein, nicht mehr für jeden einzelnen Wochentag eine extra Verteilung zu haben, sondern nur noch die Wochenenden vom Rest der Woche zu unterscheiden. Es wird also unterstellt, dass die Nutzung an den Tagen von Montag bis Freitag relativ ähnlich ist und ferner sich die Nutzung am Samstag nur unwesentlich von der am Sonntag unterscheidet. Dementsprechend ergeben sich 48 verschiedene empirische Verteilungen,

wenn für jede Stunde eines Tages eine extra Verteilung angelegt wird. Das so beschriebe Kriterium zur Unterscheidung der Häufigkeitsverteilungen sei im Weiteren $C_{\text{hourOfDay}}$ genannt.

4.2.1.2.4. C_{phase}: Das Kriterium C_{uptime} mit Unterscheidung von Tagesabschnitten

In [Wyck98] ist festgestellt worden, dass die Wahrscheinlichkeitsverteilungen aus unterschiedlichen Tagesabschnitten sich deutlicher voneinander unterscheiden als solche des gleichen Tagesabschnittes. Dabei wurden die Tage in wenige Abschnitte (ca. 4) eingeteilt und für diese Abschnitte getrennte empirische Verteilungen ermittelt. Ein ähnlicher Ansatz soll auch hier verfolgt werden, wobei mit folgenden vier Zeitspannen gearbeitet wird: 8 – 12 Uhr, 12 – 18 Uhr, 18 – 23 Uhr und 23 – 8 Uhr. Dieses Kriterium wird mit C_{phase} bezeichnet.

4.2.1.3. Zusammenfassung

Mit den Varianten C_{simple}, C_{uptime}, $C_{\text{hourOfWeek}}$, $C_{\text{hourOfDay}}$ und C_{phase} wurden verschiedene denkbare Kriterien zur Unterscheidung von empirischen Häufigkeitsverteilungen eingeführt. Eine empirische Bewertung dieser Kriterien ist jedoch an dieser Stelle noch nicht möglich, weil die Zuordnung zu empirischen Häufigkeitsverteilungen nur der erste Schritt zur Prognoseerstellung ist. Der zweite, in den folgenden Abschnitten thematisierte Schritt besteht darin, aus einer empirischen Häufigkeitsverteilung eine Prognose zu erzeugen.

4.2.2. Transformation einer Häufigkeitsverteilung in eine Prognose

Das Problem der Ableitung einer Prognose aus einer empirischen Häufigkeitsverteilung, die entsprechend des gewählten Unterscheidungskriteriums gebildet wurde, ist in der folgenden Abbildung 4-7 veranschaulicht: Links stellt die grüne (stufenförmige) Kurve die ermittelte Häufigkeitsverteilung dar, rechts die rote (stetige) Kurve eine möglicherweise daraus abgeleitete Wahrscheinlichkeitsverteilung. Für diese Ableitung sind unterschiedliche Herangehensweisen denkbar, von denen eine ganze Reihe im Folgenden untersucht wird. Dabei werden zunächst entsprechende Varianten beschrieben, bevor in Abschnitt 4.2.2.4 in entsprechenden Diagrammen die Ergebnisse gegenübergestellt werden. Eine Darstellung der Messwerte in Tabellenform findet sich überdies in Anhangsabschnitt D.1. Die Ergebnisse werden schließlich zeigen, dass die hier diskutierte, in der Literatur jedoch meist ignorierte Transformation einen größeren Einfluss auf die Prognosegüte hat als die oben diskutierten Klassifizierungskriterien.

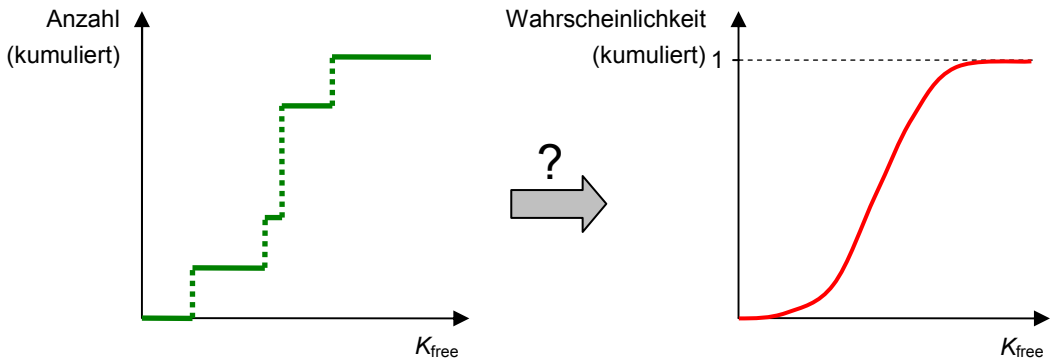

Abbildung 4-7: Transformation der empirischen Häufigkeitsverteilung in eine prognostizierte Wahrscheinlichkeitsverteilung

4.2.2.1. Annahme einer bestimmten Form der Verteilung

Beim ersten hier diskutierten Ansatz wird eine bestimmte Form der Verteilung angenommen, z. B. eine Exponential- oder eine Normalverteilung. Dargestellt ist dieser Ansatz auf der linken Seite der Abbildung 4-8 für eine gestrichelt dargestellte gemessene Verteilung, an die eine Normalverteilung angepasst wurde. Ein gegensätzlicher, erst in späteren Abschnitten diskutierter Ansatz hingegen verzichtet auf die Annahme einer bestimmte Standardverteilung. Stattdessen wird mit empirischen Häufigkeitsverteilungen gearbeitet, die entsprechend umfangreich beschrieben werden. Veranschaulicht ist dieser zweite Ansatz auf der rechten Seite der Abbildung 4-8, wobei dort die Verteilung durch ein Histogramm mit 10 Klassen beschrieben wird. Eine ähnliche Herangehensweise findet sich bspw. in [Wyck98].

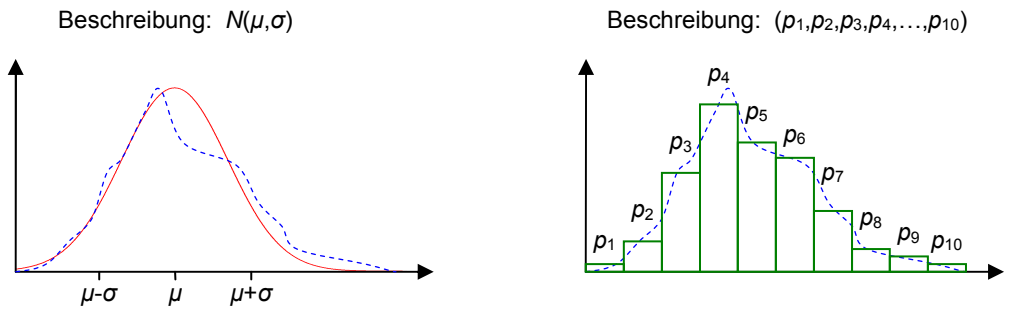

Abbildung 4-8: Mögliche Ansätze für Prognoseverfahren

Das Problem der ersten Variante ist, dass die Form der gewählten Standardverteilung zur realen Verteilung passen muss. In den seltensten Fällen gehören die realen Ver-

teilungen jedoch wirklich den angenommenen Verteilungsmodellen an.[1] Somit werden bei deren Verwendung zusätzliche Fehler eingeführt. Ferner ist es fragwürdig, dass für die hier untersuchte Größe $K_{\text{free},i}$ eine einzelne Verteilungsform angenommen werden kann, die der realen Verteilung auch nur ähnelt. Die bspw. in [Nurm03] für gut geeignet befundenen Hyperexponential- und Weibull-Verteilungen gelten nur für Datenmengen, welche die Werte unterschiedlicher Rechner zusammenfassen. Für einzelne Rechner ist von anderen und auch sehr unterschiedlichen Formen der Verteilungsfunktionen auszugehen.[2]

Doch selbst wenn für die Größe $K_{\text{free},i}$ eine solche Standardverteilung gefunden werden könnte, bliebe die Frage nach den Vorteilen ihrer Verwendung. Der i. Allg. wesentliche Vorteil ist darin zu sehen, dass sich die Verteilung auch mit wenigen Parametern beschreiben lässt. Man braucht bspw. bei einer Weibull-Verteilung nur zwei Parameter, während man ohne Kenntnis der Form der Verteilung i. d. R. eine ganze Reihe von Parametern benötigt, um die Verteilung zu beschreiben. Letzteres ist insbesondere dann von Nachteil, wenn Verteilungen in Simulationen generiert werden sollen, da in diesem Fall die festgestellten Verteilungen typischerweise auch zu skalieren sind. Dies geht für feste Modelle mit wenigen Parametern meist recht einfach. Für empirische Häufigkeitsverteilungen hingegen ist eine solche Skalierung problematisch, wenn man die charakteristischen Eigenschaften erhalten möchte, da diese bei einer empirisch gewonnenen Verteilung i. Allg. erst einmal gar nicht bekannt sind. Dieser Punkt der Generierung bzw. Skalierung von Verteilungen ist in der vorliegenden Arbeit jedoch irrelevant.

Für die hier gewünschte Vorhersage von Wahrscheinlichkeiten wäre eher relevant, die Charakterisierung von Verteilungen durch wenige Parameter zu nutzen, um die möglicherweise vorhandenen statistischen Zusammenhänge zwischen den zu verschiedenen Zeitpunkten gehörenden Verteilungen zu modellieren. Auf diese Weise könnten u. U. bekannte Zeitreihenmodelle (wie bspw. die ARMA-Modelle, vgl. [Schl99]) auf die einzelnen Parameter der Verteilung angewendet werden. Doch dazu müsste die Frage beantwortet werden, wie die Zeitreihe mit den einzelnen Werten von $K_{\text{free},i}$ in eine Zeitreihe mit Wahrscheinlichkeitsverteilungen bzw. empirischen Häufigkeitsverteilungen umgewandelt werden kann. Wie bereits in Abschnitt 3.2 diskutiert (und in Abbildung 3-5 dargestellt) wurde, ist zu den einzelnen Zeitpunkten jeweils immer nur ein Wert messbar, nicht aber die Verteilung selbst. Aus diesem Grund ist jedoch die angedeutete Variante mit der Kombination von Zeitreihenmodellen mit Wahrscheinlichkeiten kaum denkbar.

---

[1] Beispielsweise wurde in [Nurm03] festgestellt, dass die zur Modellierung empfohlenen hyperexponentiellen und Weibull-Verteilungen bei genügend langen Zeitreihen durch entsprechende Verteilungstests stets als die realen Verteilungen ausgeschlossen wurden.

[2] Beispielsweise werden Arbeitsplatzrechner häufig genau für die Arbeitszeit angeschaltet, laufen also normalerweise vielleicht zwischen 8 und 12 Stunden. Werden die Rechner gelegentlich doch über Nacht laufen gelassen, so ergeben sich gleich wieder Laufzeiten von über 30 Stunden. Eine solche Verteilung wird durch übliche Standardverteilungen jedoch nicht beschrieben. Darüber hinaus können andere Rechner davon sehr verschiedene Nutzungsweisen aufweisen, so dass ein allgemeines Modell fragwürdig ist.

Insgesamt erscheint der Ansatz mit der Verwendung standardisierter Verteilungsmodelle somit als wenig erfolgversprechend, weshalb in dieser Arbeit dem zweiten, auf die Annahme einer bestimmten Form der Verteilung verzichtenden Ansatz mehr Raum eingeräumt wird. Da der Ansatz mit der Annahme einer bestimmten Form der Verteilung aber u. a. in [Nurm04] verfolgt wurde, wurden zur Überprüfung der hier vorgebrachten Argumentation zusätzliche Experimente mit den in [Nurm03] vorgeschlagenen Hyperexponential-Verteilungen durchgeführt. Dabei kam ein Ansatz zum Einsatz, der sich an [Nurm04] orientierte. In diesem Ansatz wird das in obiger Abbildung 4-7 skizzierte Problem der Transformation so gelöst, dass die in den letzten Wochen aufgetretenen Werte von $K_{\text{free},i}$ gesammelt werden, um an diese Werte eine zweistufige hyperexponentielle Verteilung anzupassen. Zur Anpassung wird auf die bereits in [Nurm04] erwähnte EMpht-Software zurückgegriffen, die in [Olss98] näher beschrieben ist. Das auf diese Weise über den Umweg einer hyperexponentiellen Verteilung arbeitende Transformationsverfahren sei im Weiteren mit $T_{\text{hyperexponential}}$ bezeichnet.

4.2.2.2. Dichte-basierte Transformationen

Wird nicht von einer bestimmten Form der Verteilung ausgegangen, gibt es zumindest zwei grundsätzliche Möglichkeiten zur Bildung der Prognose. Die erste, in diesem Abschnitt diskutierte Variante versucht zunächst die Dichtefunktion zu schätzen, um aus dieser die kumulierte Verteilungsfunktion abzuleiten.

4.2.2.2.1. Histogramm-basierte Transformation

Histogramme sind die ältesten und wohl auch am häufigsten verwendeten Hilfsmittel zur Schätzung der Dichtefunktion ([Rinn03]). Der Ansatz ist auch in den meisten Statistikbüchern beschrieben (z. B. [Ecke00], [Mann01]). An dieser Stelle wird mit der Betrachtung dieses Ansatzes begonnen, weil die zur vorliegenden Arbeit ähnlichste ([Wyck98]) ebenfalls Histogramme verwendet.

Bei der Histogramm-basierten Transformation wird der Wertebereich in (üblicherweise gleichgroße) Intervalle eingeteilt. Für jedes dieser Intervalle wird die Anzahl der aufgetretenen Werte gezählt. Ausgehend von den ermittelten Anzahlen setzt man für jedes Intervall eine entsprechende Gleichverteilung so an, dass sich insgesamt für alle Intervalle zusammen eine Wahrscheinlichkeit von 100 % ergibt. Graphisch veranschaulicht ist dieses Verfahren in Abbildung 4-9 für eine Einteilung des Wertebereichs in vier Klassen.

Ein entscheidender Aspekt des beschriebenen Ansatzes ist die Wahl der Intervallanzahl. Für diese Wahl gibt es verschiedene Empfehlungen. Hier wird der Regel von Sturges (vgl. [Ecke00], [Mann01]) gefolgt, welche die Intervallanzahl c für eine Häufigkeitsverteilung mit n Werten wie folgt vorschlägt:

$$c \approx 1 + 3{,}3 \cdot \log_{10} n$$

Die Histogramm-basierte Transformation mit der Intervallanzahl entsprechend der Regel von Sturges wird im Weiteren T_{histo} genannt.

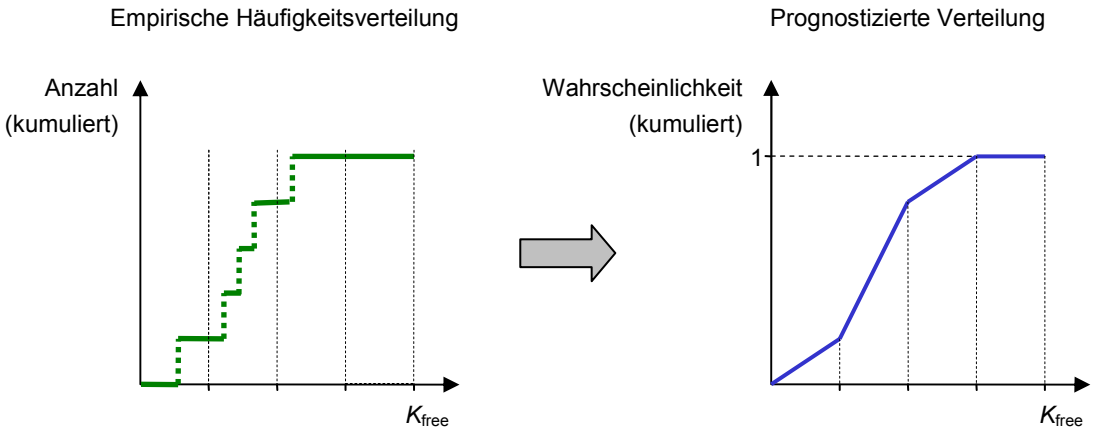

Abbildung 4-9: Histogramm-basierte Transformation T_{histo}

4.2.2.2.2. Kernschätzer

Ein wesentliches Problem der Histogramm-basierten Transformation liegt darin, dass die angenommene Gleichverteilung in den einzelnen Intervallen des Wertebereichs oftmals nicht gegeben ist. Für diesen Fall wird in der Literatur häufig der Einsatz von Kernschätzern empfohlen ([Hart99], [Rinn03]).

Der grundlegende Ansatz bei der Verwendung von Kernschätzern ist der, zunächst die Dichtefunktion empirisch zu bestimmen. Die dafür angewendete Formel hat die folgende allgemeine Form:

$$f_{K,n} = \frac{1}{n \cdot b} \sum_{i=1}^{n} K\left(\frac{x - x_i}{b}\right)$$

Dabei ist n die Anzahl der Beobachtungsdaten, die x_i sind die Werte der Beobachtungsdaten. b ist ein im Prinzip frei wählbarer Parameter, dessen Bedeutung gleich noch deutlich werden wird. Die Funktion K ist die verwendete Kernfunktion, welcher der Ansatz den Namen verdankt, $f_{K,n}$ ist die Abschätzung der Dichtefunktion. Die typischerweise aufgestellten Anforderungen an die Kernfunktion K sind:

(1) $K(u) \geq 0$

(2) $K(u) = K(-u)$

(3) $\int_{-\infty}^{\infty} K(u)\,\mathrm{d}u = 1$

Die Bedingungen (1) und (3) sind offensichtlich erforderlich, damit sich mit der oben angegebenen Formel für $f_{K,n}$ tatsächlich eine Dichtefunktion ergibt. Bedingung (2) wird eher aus Plausibilitätsgründen aufgeführt, über die man sich allerdings streiten kann. Auf jeden Fall aber lassen die Bedingungen Raum für eine ganze Reihe möglicher

Kernfunktionen, wie bspw. die in Abbildung 4-10 dargestellte Dreieck- oder Rechteckfunktion.[1]

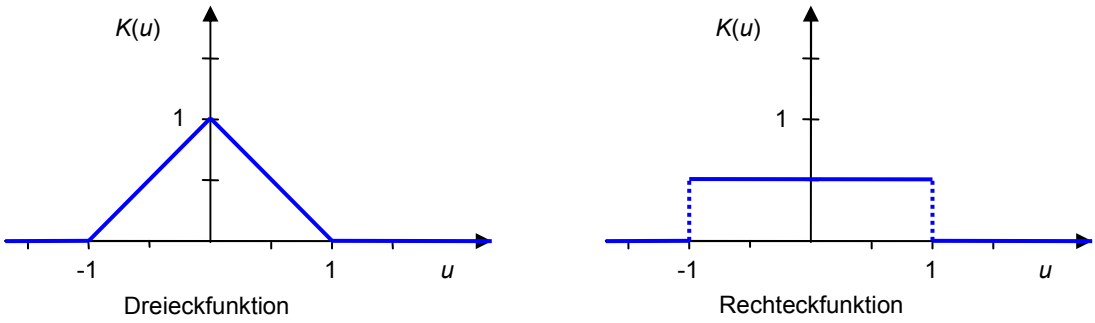

Abbildung 4-10: Dreieck- und Rechteckfunktion als mögliche Kernfunktionen $K(u)$

Der Sinn der für $f_{K,n}$ angegebenen Schätzungsformel für die Dichtefunktion besteht darin, die relative Häufigkeit $1/n$ eines einzelnen aufgetretenen Beobachtungswertes auf ein umgebendes Intervall zu verteilen, um so die Annahme der Gleichverteilung in bestimmten Intervallen aufgeben zu können. Im Grunde genommen müsste man dabei zur Bestimmung der Dichtefunktion für jeden möglichen Wert u den Funktionswert $f_{K,n}(u)$ ermitteln. Da dies aufgrund der unendlich großen Zahl von Werten für u nicht möglich ist, behilft man sich damit, den Funktionswert $f_{K,n}(u)$ in bestimmten Abständen zu berechnen. Die Dichtefunktion wird dann bspw. durch entsprechende Polygonzüge an die ermittelten Funktionswerte angepasst. Ausgehend von dieser Dichtefunktion müsste man dann noch die kumulierte Verteilungsfunktion abschätzen, wenn man an dieser interessiert ist.

An dieser Stelle wird offensichtlich, dass das Verfahren recht kompliziert, letztlich also ziemlich rechenintensiv ist. Da die Vorhersagen aber für die verbesserte Auslastung der einzelnen Rechner (und nicht zur zusätzlichen Auslastung) gedacht sind, werden die Kernschätzer im Rahmen der vorliegenden Arbeit nicht näher betrachtet. Es wird daher in den folgenden Abschnitten nach anderen Varianten der Transformation von der empirischen Häufigkeitsverteilung zur Prognose der Wahrscheinlichkeitsverteilung gesucht.

4.2.2.3. Direkte Schätzung der kumulierten Verteilungsfunktion

Die Nutzung von Schätzungen der Dichtefunktion zur Schätzung der kumulierten Verteilungsfunktion (CDF) ist zunächst recht intuitiv. Dies dürfte auch der Hauptgrund für die häufige Verwendung von Histogrammen zur Ableitung der CDF sein. Mit der Schätzung der Dichtefunktion durch Histogramme oder Kernschätzer ist jedoch immer eine Glättung der empirischen Häufigkeitsverteilung verbunden, was zu zusätzlichen

[1] Der in der Formel für $f_{K,n}$ angegebene Parameter b ist also letztlich dafür da, eine bestimmte gewählte Grundform zu strecken bzw. zu stauchen, um schließlich die gewünschte Kernfunktion zu erhalten.

Prognosefehlern führt. Für die hier gesuchte CDF ist eine vorherige Schätzung der Dichtefunktion aber gar nicht erforderlich. Daher werden im Folgenden Varianten diskutiert, welche die empirische Häufigkeitsverteilung direkt in die CDF transformieren.

4.2.2.3.1. 1:1-Transformation

Eine erste und besonders einfache Möglichkeit zur direkten Ableitung der Prognose der Verteilung besteht darin, die empirische Verteilungsfunktion F_{emp} zu nutzen. Diese Verteilungsfunktion ist in vielen Statistikbüchern beschrieben (z. B. [Pest98], [Rinn03]) und für n beobachtete Werte v_1, v_2, …, v_n wie folgt definiert:

$$F_{emp}(v) = \frac{\#\{i : v_i \le v\}}{n}$$

Die Nutzung dieser Funktion (welche sich praktisch 1:1 aus der empirischen Häufigkeitsverteilung ergibt) als Prognose für die Wahrscheinlichkeitsverteilung wird im Weiteren als $T_{1:1}$ bezeichnet.

4.2.2.3.2. Transformationsvariante $T_{\text{fullRange}}$

Die bisher diskutierten Varianten der Transformation sind in unveränderter Form der Literatur entnommenen worden. Bei den Tests hat sich jedoch herausgestellt, dass diese Varianten alle große Prognosefehler aufweisen, die sich insbesondere an der sehr ungenauen Einhaltung der vorhergesagten Perzentile erkennen lassen, weshalb nach besseren Transformationsvarianten gesucht wurde.

Zur Ableitung einer ersten Variante, die im Weiteren mit $T_{\text{fullRange}}$ bezeichnet wird, wurde die Menge der n in der empirischen Häufigkeitsverteilung enthaltenen Werte v_1, v_2, …, v_n als aufsteigend sortierte Liste betrachtet. Um die Diskussion einfach und verständlich zu halten, sei im Folgenden von der vereinfachten Annahme ausgegangen, dass alle Werte paarweise verschieden sind.[1] Somit ergeben sich (n+1) verschiedene Positionen, an denen ein neuer Wert v in die Liste einsortiert werden könnte. Ohne weitere Informationen über die Verteilung gibt es dabei keinen Grund, eine dieser Positionen als wahrscheinlicher als eine andere zu betrachten, so dass sich insgesamt die folgende Annahme ergibt:

$$P(v < v_1) = P(v_i < v < v_{i+1}) = P(v > v_n) = \frac{1}{n+1} \qquad (1 \le i < n)$$

Es gibt also (n+1) verschiedene Intervalle, in die der nächste Wert v fallen könnte, wobei für jedes dieser Intervalle die gleiche Wahrscheinlichkeit angenommen wird. Für das linke und das rechte Intervall existiert dabei jeweils nur eine durch einen aufgetretenen Wert festgelegte Grenze. Die andere Intervallgrenze kann jedoch durch die entsprechende theoretische Grenze des möglichen Wertebereichs festgelegt werden, was

[1] Die notwendigen Modifikationen für den Fall gleicher Werte sind nicht besonders kompliziert und weitgehend offensichtlich, würden die folgenden Erläuterungen und Definitionen jedoch erheblich aufblähen und damit ziemlich unübersichtlich machen.

hier auch so getan wird. Der erste der beiden begrenzenden Werte ist einfach 0, der zweite Wert ist die maximale Rechenkapazität des Rechners, welche innerhalb der maximalen Zeit t_{max} erreichbar ist. (Bezogen auf die hier diskutierte Größe K_{free} ist die obere Grenze gleich 100.) Wird die untere Grenze v_0 und die obere Grenze v_{n+1} bezeichnet, dann ergibt sich die folgende Annahme der Wahrscheinlichkeiten:

$$P(v_i < v < v_{i+1}) = \frac{1}{n+1} \qquad (0 \le i \le n)$$

Da für die Verteilung innerhalb der Intervalle keine weiteren Informationen zur Verfügung stehen, ist die Annahme von Gleichverteilungen sowohl sinnvoll als auch einfach, weshalb diese Annahme bei der Transformation $T_{fullRange}$ getroffen wird. Der wesentliche Unterschied zur Variante T_{histo} besteht dann darin, dass die Intervallgrenzen nicht im Voraus festgelegt werden, sondern durch die in der jüngeren Vergangenheit aufgetretenen Werte von K_{free} bestimmt sind. Eine graphische Darstellung der Variante $T_{fullRange}$ findet sich in Abbildung 4-11.

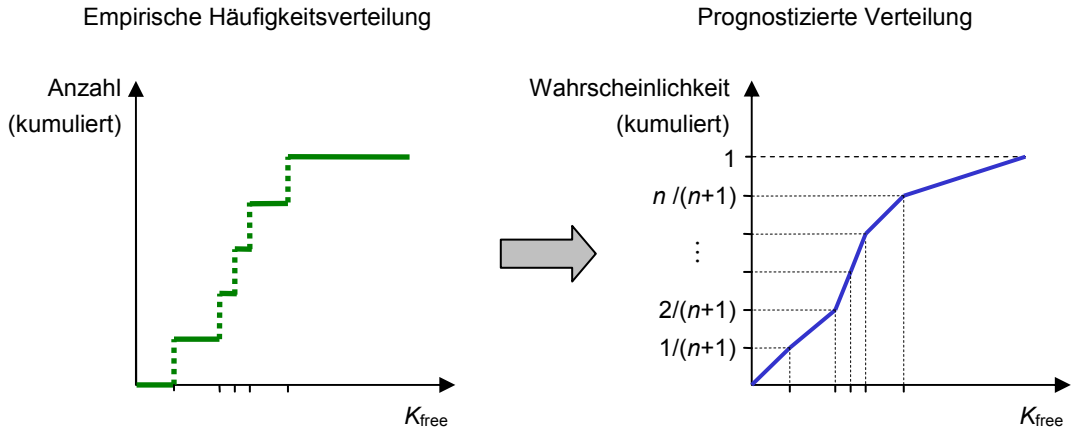

Abbildung 4-11: Transformationsvariante $T_{fullRange}$

4.2.2.3.3. Transformationsvariante $T_{dynRange}$

In der Transformationsvariante $T_{fullRange}$ werden sowohl die untere Grenze des ersten Intervalls als auch die obere Grenze des letzten Intervalls aus den theoretischen Grenzen des Wertebereichs abgeleitet. Die Bestimmung dieser Grenzen ist zwar kein Problem, allerdings kann der praktisch auftretende Wertebereich deutlich kleiner als der theoretische Wertebereich sein. Dies gilt insbesondere für die obere Grenze. So bleiben manche Rechner praktisch nie die maximale Zeit t_{max} ununterbrochen angeschaltet oder sie haben immer eine signifikante Grundlast.

Aus diesem Grunde wurde mit der im Folgenden diskutierten Variante $T_{dynRange}$ versucht, den praktisch relevanten Wertebereich dynamisch zu erfassen: Statt bei der Modellierung der Verteilung den theoretisch möglichen Wertebereich zugrunde zu legen, wird von dem minimalen Intervall ausgegangen, in dem alle in letzter Zeit aufgetretenen Werte liegen. Der Einfachheit halber wurde für diese „letzte Zeit“ der

gleiche Zeitraum genutzt wie für die Ermittlung der empirischen Häufigkeitsverteilungen.[1] Die Variante T_{dynRange} arbeitet dann analog zur Variante $T_{\text{fullRange}}$, nur dass die obere Grenze v_{n+1} des Wertebereiches dynamisch bestimmt wird.[2]

4.2.2.4. Vergleich der Ergebnisse

Eine (anhand der vier Testrechner) vorgenommene erste vergleichende Bewertung der vorgestellten Transformationsverfahren ergibt sich aus Abbildung 4-12 und Abbildung 4-13; die jeweils dargestellte Größe ist unter den Säulen angegeben. Die Säulen repräsentieren dabei die Mittelwerte, die aus den in Anhangsabschnitt D.1 dokumentierten Werten aller getesteten Unterscheidungskriterien gebildet wurden.

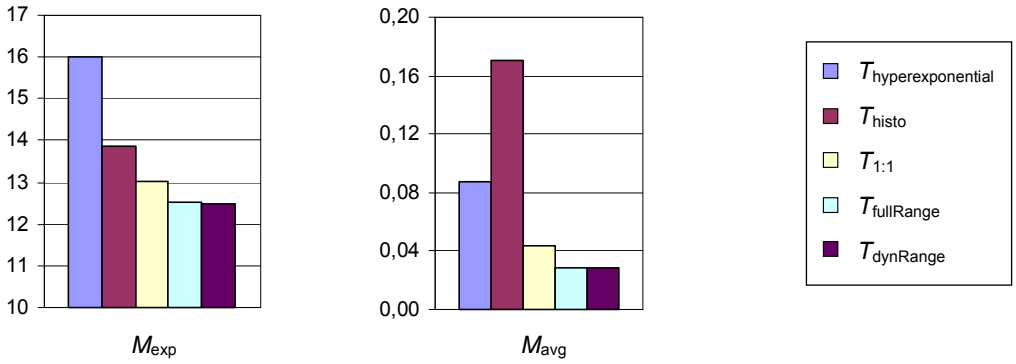

Abbildung 4-12: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg}

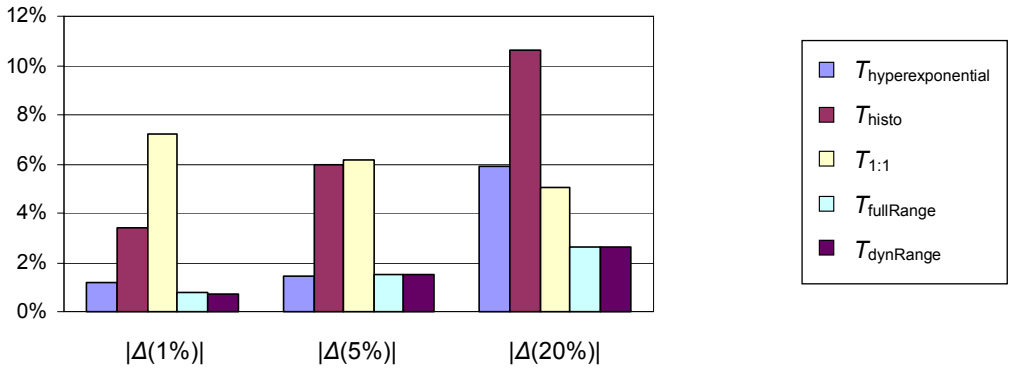

Abbildung 4-13: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$

[1] Genau genommen wird aus Gründen der Effizienz der Implementierung diese Zeitspanne nicht so genau eingehalten. Das genaue Vorgehen ist in Anhangsabschnitt B.1 erläutert.

[2] Auf eine dynamische Bestimmung der unteren Grenze wird hier verzichtet, da eine verbleibende Rechenkapazität von 0 immer möglich ist, da jeder Rechner auch mal ausfällt. In empirischen Tests ergaben sich bei einer dynamischen Bestimmung der unteren Grenze auch schlechtere Ergebnisse als mit der hier vorgestellten Variante.

Es wird deutlich, dass die beiden Verfahren $T_{\text{fullRange}}$ und T_{dynRange} für die vier Testrechner sowohl hinsichtlich der Punktgenauigkeit als auch bzgl. der Genauigkeit im Durchschnitt am besten abschneiden. Als offensichtlich für diese Rechner überhaupt nicht geeignet stellte sich die Annahme des Vorliegens einer Hyperexponential-Verteilung heraus. Zusätzlich dazu ist anzumerken, dass die notwendigen Berechnungen zur Ermittlung einer möglichst gut geeigneten hyperexponentiellen Verteilung durch das iterative Annäherungsverfahren sehr aufwendig sind, was auch bei den Simulationsdurchläufen deutlich wurde, die mit $T_{\text{hyperexponential}}$ um ein Vielfaches langsamer waren als mit den anderen betrachteten Verfahren. Der hohe Aufwand ist neben der geringen Prognosegenauigkeit ein weiterer Grund, den Ansatz $T_{\text{hyperexponential}}$ nicht zu verwenden.

Neben den Transformationsverfahren wurden auch die in Abschnitt 4.2.1 diskutierten Unterscheidungskriterien miteinander verglichen, wobei die Mittelwerte hier über die fünf getesteten Transformationsverfahren gebildet wurden. Die ermittelten Werte sind in Abbildung 4-14 und Abbildung 4-15 dargestellt.

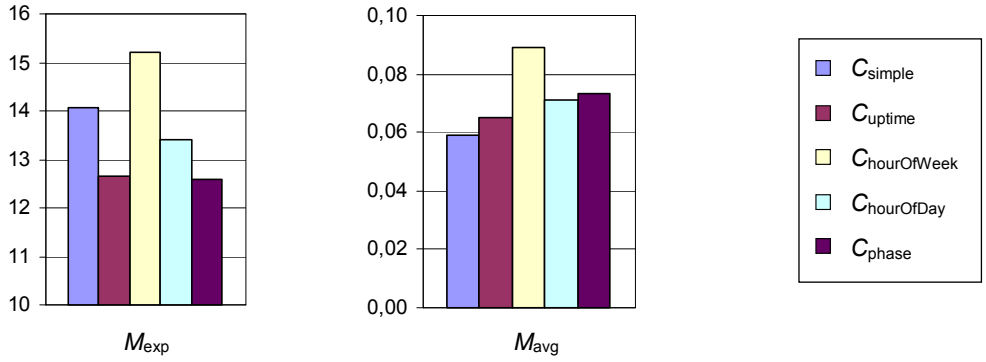

Abbildung 4-14: Ergebnisse der Klassifizierungskriterien bzgl. M_{exp} und M_{avg}

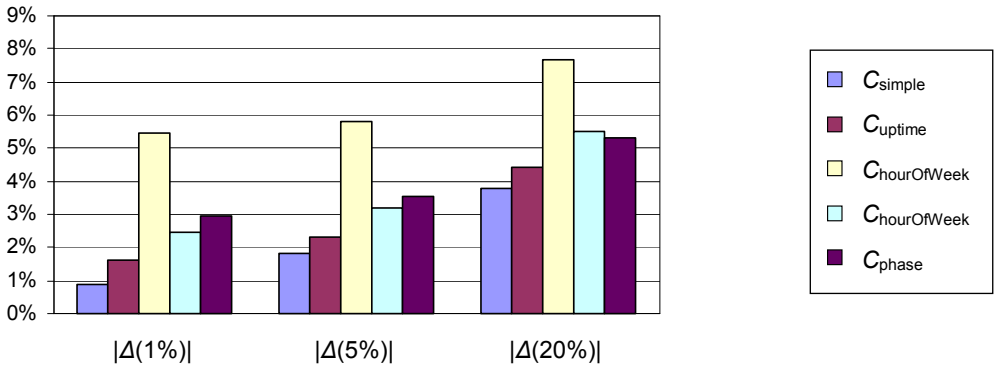

Abbildung 4-15: Ergebnisse der Klassifizierungskriterien bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$

Die Ergebnisse weisen keines der Klassifizierungskriterien als eindeutig bestes aus, vergleichsweise gut schneidet C_{uptime} ab: Es liegt bzgl. M_{exp} in etwa auf dem Niveau von C_{phase} und bzgl. M_{avg} ist es für die vier Testrechner nur etwas schlechter als C_{simple}. Die Ergebnisse zur mit M_{avg} gemessenen Genauigkeit im Durchschnitt spiegeln sich dabei auch in den Messungen für die untersuchten Perzentile wider. Auffällig schlecht schneidet das Kriterium $C_{hourOfWeek}$ ab, was so nicht von vornherein zu erwarten war.

4.2.3. Adaptive Korrektur der Prognosen

Von den bisher diskutierten Möglichkeiten der Transformation schneiden die Varianten $T_{fullRange}$ und $T_{dynRange}$ sowohl hinsichtlich der Punktgenauigkeit als auch bzgl. der Genauigkeit im Durchschnitt am besten ab. Im Folgenden soll es um eine Maßnahme zur weiteren Verbesserung der Prognosegüte gehen, wobei sich die Betrachtungen auf $T_{fullRange}$ beschränken, da dieses Verfahren etwas einfacher als $T_{dynRange}$ ist und die Ergebnisse dieser beiden Verfahren nahezu identisch sind. Ausgangspunkt der Überlegungen sind die zu den einzelnen Zeitpunkten mittels $T_{fullRange}$ erstellten Prognosen der kumulierten Verteilungen (vgl. Abbildung 4-16). Die ebenfalls eingezeichneten Werte für K_{free} sind dabei zum Zeitpunkt der Prognose noch nicht bekannt und können daher erst im Nachhinein gemessen werden (vgl. Abbildung 3.2).

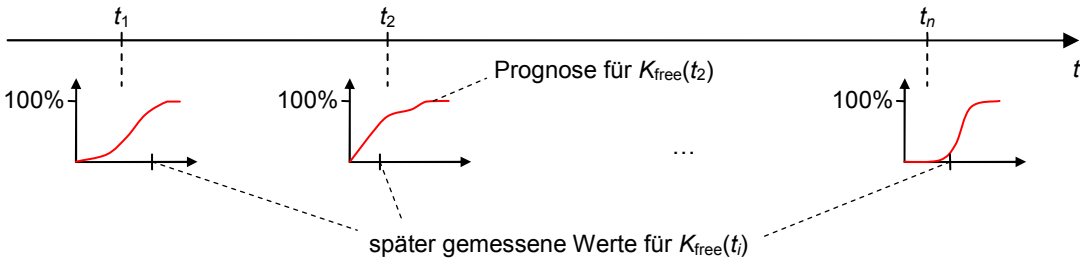

Abbildung 4-16: Prognostizierte Wahrscheinlichkeitsverteilungen und im Nachhinein gemessene Werte zu verschiedenen Zeitpunkten

Ist aber schließlich der Wert von K_{free} zu einem bestimmten Zeitpunkt bekannt, kann mittels der ursprünglich erstellten Prognosen eine kumulierte Wahrscheinlichkeit zugeordnet werden. Dargestellt ist diese Zuordnung in Abbildung 4-17 für den Zeitpunkt t_2. Die kumulierten Wahrscheinlichkeitswerte zu den einzelnen Zeitpunkten werden gesammelt und sollten in etwa eine Gleichverteilung zwischen 0 und 100 Prozent ergeben (vgl. Diskussion in Abschnitt 3.2.1). In der Praxis wird es zu mehr oder weniger deutlichen Abweichungen kommen, so wie es bspw. in Abbildung 4-18 dargestellt ist: Die (nicht gleichmäßig zwischen 0 und 100 Prozent verteilten) Punkte auf der Abszisse sind die für die einzelnen Zeitpunkte im Nachhinein ermittelten kumulierten Wahrscheinlichkeitswerte (vgl. Abbildung 4-17). Aus diesen lässt sich analog zum Verfahren $T_{fullRange}$ (mit dem theoretisch möglichen Wertebereich von 0 % bis 100 %, vgl. Abschnitt 4.2.2.3.2) eine Verteilungskurve H_{cum} ableiten, die in Abbildung 4-18 als

dicke gepunktete Linie eingezeichnet ist. Die ideale Kurve wäre $H_{ideal}(x) = x$ und ist als gestrichelte Linie dargestellt.

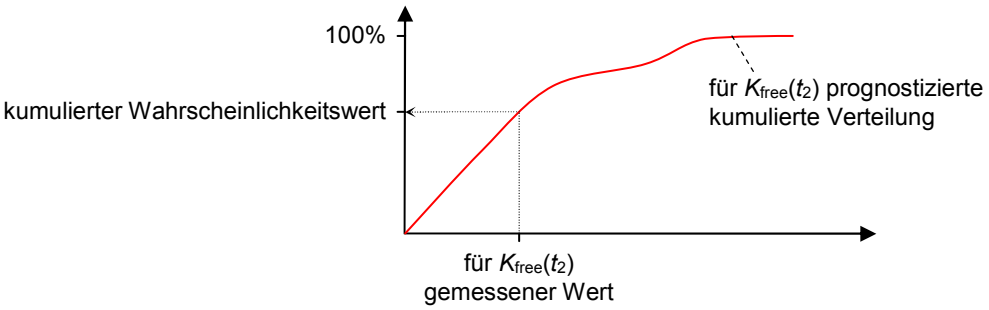

Abbildung 4-17: Zuordnung des ursprünglich prognostizierten kumulierten Wahrscheinlichkeitswertes zum gemessenen Wert für K_{free}

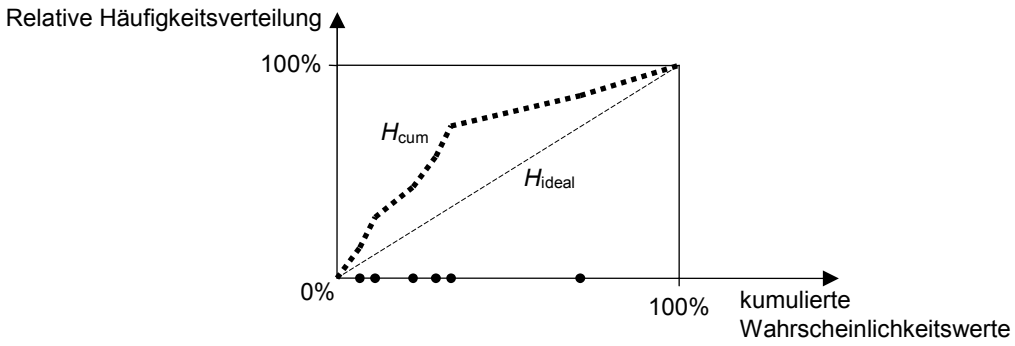

Abbildung 4-18: Verteilung der ermittelten kumulierten Wahrscheinlichkeitswerte

Aus der ermittelten Verteilungskurve H_{cum} könnte sich dann bspw. ergeben, dass für 50 Prozent der gemessenen Werte von K_{free} kumulierte Wahrscheinlichkeiten von weniger als 25 % vorhergesagt worden sind. Wenn dies auf einen systematischen Fehler zurückzuführen ist, sollte eine Prognose von 25 % auf 50 % korrigiert werden. Angedeutet ist dieser Ansatz in Abbildung 4-19: Links ist die eigentliche Prognoseverteilung für K_{free}, wie sie mit $T_{fullRange}$ aus den Werten der Vergangenheit abgeleitet worden sein könnte. Bei einer Anfrage der Wahrscheinlichkeit, dass K_{free} kleiner oder gleich einem bestimmten Wert v ist, würde mit der prognostizierten Verteilung ein Wert p ermittelt werden, der in der Darstellung bei 25 % liegt. Für diesen Wert kann unter zu Hilfenahme der rechts dargestellten Verteilungskurve H_{cum} ermittelt werden, für wie viele der bisher gemessenen Werte von K_{free} eine Wahrscheinlichkeit kleiner oder gleich p vorhergesagt wurden. In der Abbildung gilt dies für $p_{corr} = 50$ % der bisherigen Vorhersagen. Denkbar wäre jetzt, dass die Prognose für v nicht p liefert, sondern p_{corr}. Ebenso ist es denkbar, dass irgendein Wert aus dem Intervall $[p;p_{corr}]$ gewählt wird. Auf entsprechende Varianten wird in den folgenden Unterabschnitten eingegangen. Die

Ergebnisse der Messungen sind wiederum in Anhangsabschnitt D.1 in Tabellenform angegeben und in Abschnitt 4.2.3.4 durch Diagramme zusammengefasst.

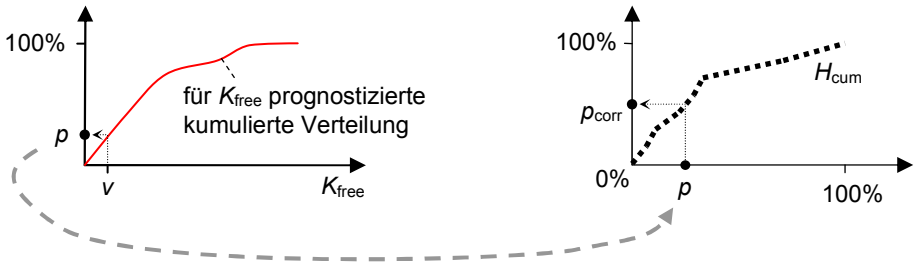

Abbildung 4-19: Adaptive Korrektur der Verteilungsprognose

4.2.3.1. Vollständige Korrektur (A_{full})

Zunächst einmal soll die vorgeschlagene Idee auf relativ einfache Weise umgesetzt werden. Dazu wird – wie gerade beschrieben – für die bereits diskutierten Verfahren jeweils die Verteilung der zu den eingetretenen Werten von K_{free} gehörenden kumulierten Wahrscheinlichkeiten gemessen. Die Sammlung der Werte in H_{cum} erfolgt für denselben Zeitraum, für den auch die empirischen Verteilungen aufgebaut werden (also für die letzten 10 Wochen). Im Gegensatz zu den empirischen Häufigkeitsverteilungen wird jedoch kein Unterscheidungskriterium angewandt – es gibt also nur eine Verteilung H_{cum}. Diese könnte dabei die unkorrigierten Prognosewerte beinhalten oder die bereits korrigierten Werte. In den Tests hat sich letztere Variante als vorteilhaft erwiesen, so dass im Weiteren auch ausschließlich diese betrachtet wird.

Als Prognosewahrscheinlichkeit wird dann bei der hier betrachteten Variante A_{full} der korrigierte Wert p_{corr} genommen, so wie er laut Abbildung 4-19 ermittelt werden kann. Die sich mit dieser Prognosevariante für die vier Testrechner erzielten Ergebnisse sind in Tabelle D-6 zusammengefasst.

4.2.3.2. Anteilige Korrektur ($A_{partial}$)

Wie bereits angesprochen, muss die Prognose nicht unbedingt durch den nach dem oben beschriebenen Verfahren gewonnenen Wert p_{corr} ersetzt werden. Alternativ bietet es sich bspw. an, aus den beiden Werten p und p_{corr} ein gewichtetes arithmetisches Mittel zu bilden, um die Prognosewahrscheinlichkeit $p_{forecast}$ wie folgt zu bestimmen:

$$p_{forecast} = f_{corr} \cdot p_{corr} + (1 - f_{corr}) \cdot p$$

Der Faktor f_{corr} ist dabei das Korrekturgewicht und kann Werte aus dem Intervall [0;1] annehmen. Die Variante der Verwendung eines Gewichtes $0 < f_{corr} < 1$ sei hier $A_{partial}$ genannt. Der Hintergedanke dabei ist der, nach Möglichkeit eine Überkorrektur zu vermeiden. Zu diesem Zweck wurde in den empirischen Untersuchungen mit verschiedenen Werten des Korrekturgewichts f_{corr} experimentiert, die Ergebnisse für $f_{corr} = 0{,}5$ sind in Tabelle D-7 zusammengefasst.

4.2.3.3. Variable Korrektur ($A_{variable}$)

Die Idee der Variante $A_{partial}$ war es, eine Überkorrektur zu vermeiden und so ein gegenüber A_{full} besseres Prognosekorrekturverfahren zu haben. In den Messungen konnte jedoch mit den getesteten Korrekturfaktoren keine Verbesserung erzielt werden. Alternativ wäre eine Korrekturvariante denkbar, die gerade dann besonders stark korrigiert, wenn eine erhöhte Wahrscheinlichkeit für einen großen Fehler vorliegt. Dies ist naturgemäß dann der Fall, wenn die empirisch gewonnene Verteilung nur wenige Werte enthält, da dann zum einen Ausreißer einen besonders starken Einfluss haben, zum anderen die Interpolationsfehler besonders groß sind. Es müsste also die Korrektur bei einer Verteilung mit wenigen Werten besonders stark sein – und bei einer Verteilung mit mehr Werten schwächer ausfallen. Eine entsprechende dynamische Bestimmung des Korrekturgewichts f_{corr} ist bspw. mit folgender Formel möglich:

$$f_{corr} = f_{min} + \frac{1 - f_{min}}{\max\{1, count\}}$$

Dabei ist f_{min} die untere Schranke für das Korrekturgewicht und sollte im Intervall [0;1] liegen. Bei *count* handelt es sich um die Anzahl der Werte in der empirisch gewonnenen Verteilung. Auf diese Weise liegt das dynamische Korrekturgewicht f immer im Intervall $[f_{min},1]$. Die Variante mit einem solchen dynamischen Korrekturgewicht sei im Folgenden mit $A_{variable}$ bezeichnet; Tabelle D-8 zeigt die Ergebnisse für $f_{min} = 0{,}5$.

4.2.3.4. Ergebnisse

Die Ergebnisse der drei vorgestellten Korrekturverfahren sind in Abbildung 4-20 und Abbildung 4-21 den Ergebnissen der Variante $T_{fullRange}$ gegenübergestellt. Es zeigt sich, dass die Genauigkeit im Durchschnitt erheblich verbessert werden konnte, ohne die Punktgenauigkeit der Vorhersagen negativ zu beeinflussen. Die besten Ergebnisse wurden dabei mit A_{full} erzielt. Bei den Korrekturverfahren $A_{partial}$ und $A_{variable}$ wurde mit verschiedenen Werten für die Parameter f_{corr} bzw. f_{min} gearbeitet, ohne jedoch die Ergebnisse der Variante A_{full} zu erreichen. In den Diagrammen sind daher nur exemplarisch die Ergebnisse für $f_{corr} = f_{min} = 0{,}5$ dargestellt.

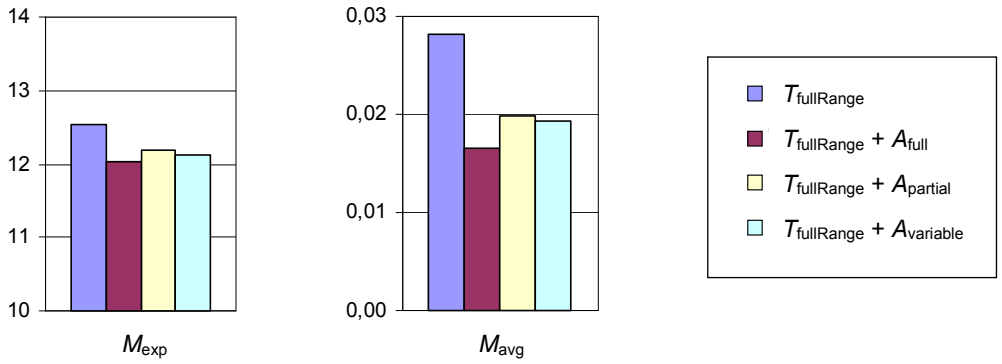

Abbildung 4-20: Ergebnisse der adaptiven Korrekturverfahren bzgl. M_{exp} und M_{avg}

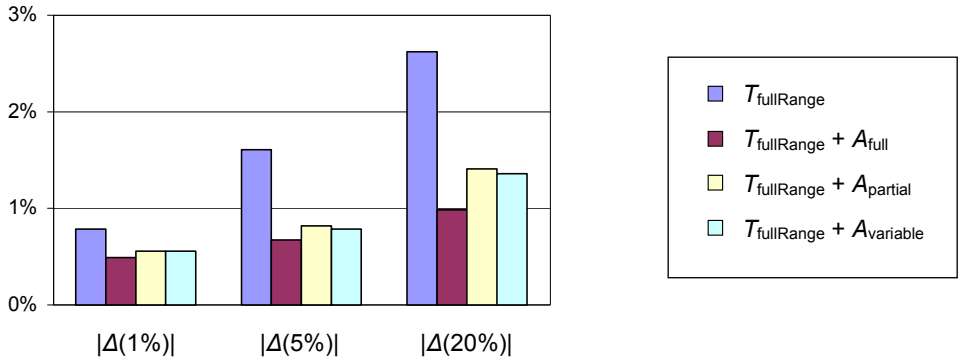

Abbildung 4-21: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$

4.2.4. Verworfene Ansätze

Neben den bereits vorgestellten Verfahren wurden auch noch andere Ansätze getestet, mit denen jedoch keine weiteren Verbesserungen erzielt werden konnten. Zur Vervollständigung der Dokumentation werden die wichtigsten dieser Ansätze im Folgenden kurz skizziert, eine ausführliche Beschreibung findet sich in Anhang A.

4.2.4.1. Variable Unterscheidung der Häufigkeitsverteilungen

In Abschnitt 4.2.1 wurden unterschiedliche Ansätze diskutiert, um die messbaren Werte auf verschiedene Häufigkeitsverteilungen aufzuteilen. Es stellte sich das grundsätzliche Problem, dass eine Häufigkeitsverteilung möglichst spezifisch für eine bestimmte Situation sein sollte, andererseits aber auch nicht zu wenige Werte enthalten darf. Die Zahl der sinnvollerweise zu unterscheidenden Häufigkeitsverteilungen hängt dabei vom jeweiligen Rechner ab. Beispielsweise werden beim Kriterium C_{uptime} 50 verschiedene Klassen unterschieden, für manche Rechner wäre jedoch eine Einteilung in weniger Klassen sinnvoller. Statt also nach einem bestimmten Klassifizierungskriterium stets eine feste Zahl verschiedener Häufigkeitsverteilungen zu unterscheiden, könnten es sinnvoll sein, manche dieser Verteilungen (situationsabhängig) zusammenzufassen. Zu diesem Zweck wurde mit Verfahren zur automatischen Clusterbildung experimentiert.

Das Prinzip ist in Abbildung 4-22 für die sechs Häufigkeitsverteilungen V_1 bis V_6 dargestellt, die zu drei Clustern zusammengefasst wurden. Die Konsequenz ist dabei, dass aus allen in einen Cluster fallenden Werten eine einzige Häufigkeitsverteilung gebildet wird. In Abbildung 4-22 würden bspw. alle zu den Verteilungen V_1 bis V_3 gehörenden Werte eine einzige Häufigkeitsverteilung bilden, die den Cluster 1 repräsentiert.

In den empirischen Tests wurde mit zahlreichen Varianten zur Bildung der Cluster gearbeitet, eine genaue Beschreibung des Vorgehens kann Abschnitt C.1 entnommen werden. Tendenziell am besten abgeschnitten hat das dort beschriebene Verfahren $C_{uptime,clustered}$, das ausgehend vom Klassifizierungskriterium C_{uptime} eine Cluster-Bildung

durchführt. Letztlich konnten jedoch gegenüber den anderen bisher betrachteten Verfahren keine entscheidenden Verbesserungen erzielt werden.

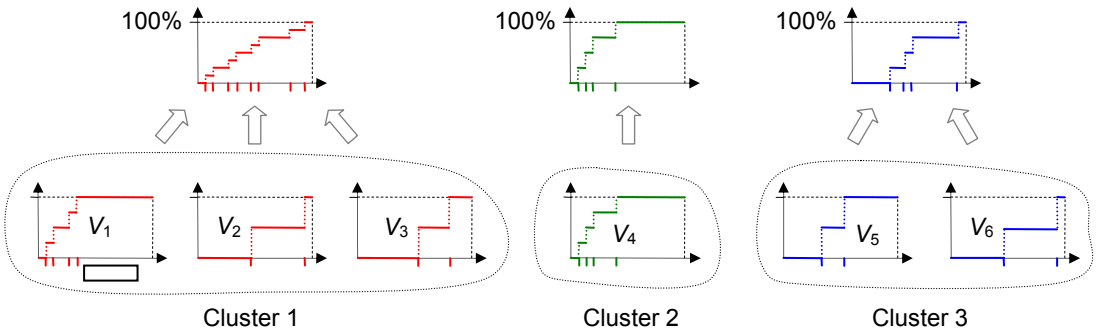

Abbildung 4-22: Clusterbildung für empirische Häufigkeitsverteilungen

Es stellt sich die Frage, was die Ursache für den ausbleibenden Erfolg der Cluster-Verfahren ist. Aus Sicht des Autors ist das Hauptproblem das Fehlen geeigneter Maße zur Bestimmung der zu vereinigenden Cluster. So ist es zunächst plausibel, bevorzugt solche Wahrscheinlichkeitsverteilungen in einem Cluster zu vereinigen, die einander besonders ähnlich sind. Es können jedoch nicht die realen Wahrscheinlichkeitsverteilungen, sondern nur die davon mehr oder weniger stark abweichenden empirischen Häufigkeitsverteilungen gemessen werden. Die Vereinigung dieser Häufigkeitsverteilungen verbessert jedoch gerade dann die Prognose, wenn die beiden realen Verteilungen zueinander ähnlich sind und die empirischen Häufigkeitsverteilungen von diesen realen Verteilungen in gegensätzlicher Richtung stark abweichen. In diesem Fall können sich die gegensätzlichen Fehler zum Teil ausgleichen. In Abbildung 4-23 ist ein solches Beispiel dargestellt: V_1 und V_2 sind die kumulierten Verteilungsfunktionen der beiden realen Wahrscheinlichkeitsverteilungen und in der Abbildung gepunktet bzw. gestrichelt dargestellt. Die entsprechenden stufenförmigen empirischen Häufigkeitsverteilungen sind mit H_1 und H_2 bezeichnet. Sie ergeben sich aus den ermittelten Messwerten, die durch Striche an den Abszissen angedeutet sind. Bei der Vereinigung von H_1 und H_2 entsteht die im unteren Teil der Abbildung dargestellte Verteilung H_{1+2}, die offensichtlich näher an den realen Verteilungsfunktionen V_1 und V_2 liegt als die ursprünglichen Häufigkeitsverteilungen H_1 und H_2; die Verwendung von H_{1+2} anstelle von H_1 und H_2 wäre hier also tatsächlich vorteilhaft.

Bei starken Abweichungen in gegensätzlicher Richtung sind sich die empirischen Häufigkeitsverteilungen aber trotz ähnlicher realer Verteilungen nicht mehr unbedingt besonders ähnlich. Schaut man sich Abbildung 4-23 an, wird die Vereinigung der empirischen Häufigkeitsverteilungen um so sinnvoller, je weiter diese auseinanderdriften. Dies spricht eher für die Vereinigung von Häufigkeitsverteilungen mit großer Distanz, was zum Teil auch im Rahmen der empirischen Untersuchen versucht wurde. Unähnliche Häufigkeitsverteilungen sind aber oft darauf zurückzuführen, dass die realen Verteilungen tatsächlich stark voneinander abweichen. In diesem Fall sollte wiederum keine Vereinigung der Häufigkeitsverteilungen stattfinden.

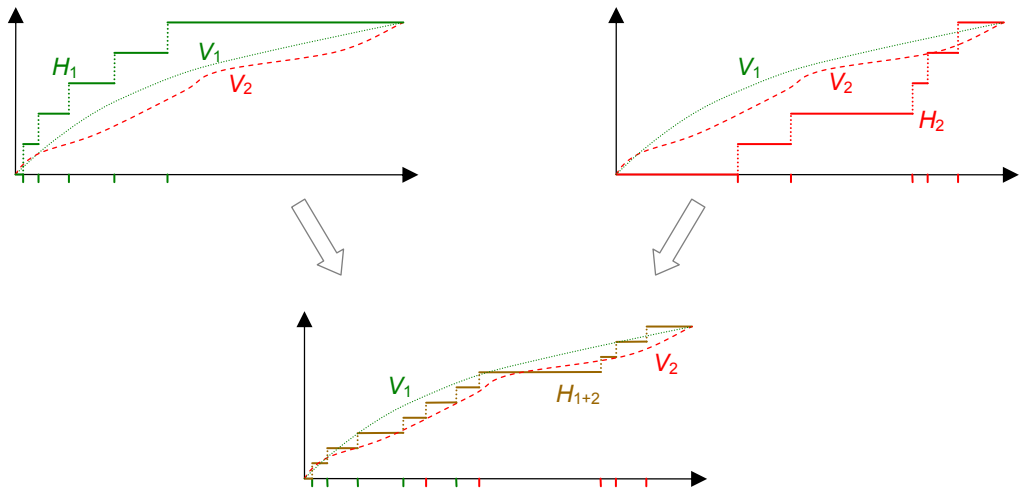

Abbildung 4-23: Reale Verteilungen V_1 und V_2 und empirische Häufigkeitsverteilungen H_1 und H_2 sowie die kombinierte Häufigkeitsverteilung H_{1+2}

Zusammenfassend bleibt daher festzuhalten, dass die Ähnlichkeit von empirischen Häufigkeitsverteilungen nicht geeignet ist, um Rückschlüsse dahingehend zu ziehen, ob die Verteilungen zu vereinigen sind oder nicht. Dies dürfte der Hauptgrund für den ausgebliebenen Erfolg der Cluster-Verfahren sein.

4.2.4.2. Mischverfahren

Ein Problem mit den (i. Allg. empfehlenswerten) Klassifizierungskriterien C_{uptime} und C_{phase} besteht darin, dass sie für manche Rechner die Prognose gegenüber C_{simple} verbessern können, für andere Rechner hingegen eher verschlechtern. Aus diesem Grunde war mit den im vorigen Abschnitt beschriebenen Clusterverfahren bereits versucht worden, die Klassifizierungskriterien automatisch anzupassen. Da dies jedoch zu keinen befriedigenden Ergebnissen führte, wurde mit den im Folgenden beschriebenen Mischverfahren versucht, von den in Frage kommenden Klassifizierungskriterien automatisch das günstigste auszuwählen. Die Idee besteht dabei darin, parallel Prognosen mit verschiedenen Klassifizierungskriterien zu erstellen. Für die Entscheidung, welche dieser erstellten Prognosen letztlich abgegeben wird, werden die Ergebnisse der letzten Zeit herangezogen; es wird also das Ergebnis desjenigen Prognoseverfahrens ausgewählt, das in der letzten Zeit am besten bewertet worden ist.

Offensichtlicher Nachteil einer solchen Mischvariante gegenüber den anderen bisher diskutierten Verfahren mit festem Klassifizierungskriterium ist, dass mehrere Prognosen berechnet und bewertet werden müssen; es ergibt sich also ein erhöhter Aufwand zur Erstellung der letztlich gesuchten Prognose. Um diesen Aufwand in Grenzen zu halten, wurden die Tests nur mit solchen Verfahren durchgeführt, die intern genau zwei Prognosen erzeugen, also mit genau zwei verschiedenen Unterscheidungskriterien arbeiten. Aus der Vielzahl von Kombinationen verschiedener Unterscheidungskriterien lieferten

die Varianten $C_{uptime,simple,2}$ und $C_{phase,simple,2}$ noch die besten Ergebnisse. Die Variante $C_{uptime,simple,2}$ kombiniert die beiden Unterscheidungskriterien C_{uptime} und C_{simple}, während die Variante $C_{phase,simple,2}$ die beiden Kriterien C_{phase} und C_{simple} kombiniert. Eine genaue Definition der beiden Verfahren findet sich in Anhangsabschnitt C.2.

Doch auch die Ergebnisse der Mischverfahren führten für die Gruppe mit den vier Rechner gegenüber den entsprechenden reinen Klassifizierungskriterien C_{uptime} und C_{phase} nicht zu den erhofften Verbesserungen, eine genauere Darstellung der Messergebnisse findet sich in Abschnitt C.2. Berücksichtigt man den erhöhten Aufwand zur Erstellung mehrerer möglicher Prognosen, sprechen diese Ergebnisse somit gegen die Verwendung der angesprochenen Mischverfahren. Die in Abschnitt 4.3 aufgeführten Ergebnisse bestätigen die hier gezogen Schlussfolgerung.

4.2.5. Zusammenfassung des Vorgehensweise

An dieser Stelle soll die Vorgehensweise, wie sie entsprechend der Ergebnisse mit den vier Testrechnern empfehlenswert zu sein scheint, zusammengefasst dargestellt werden (vgl. Abbildung 4-24).

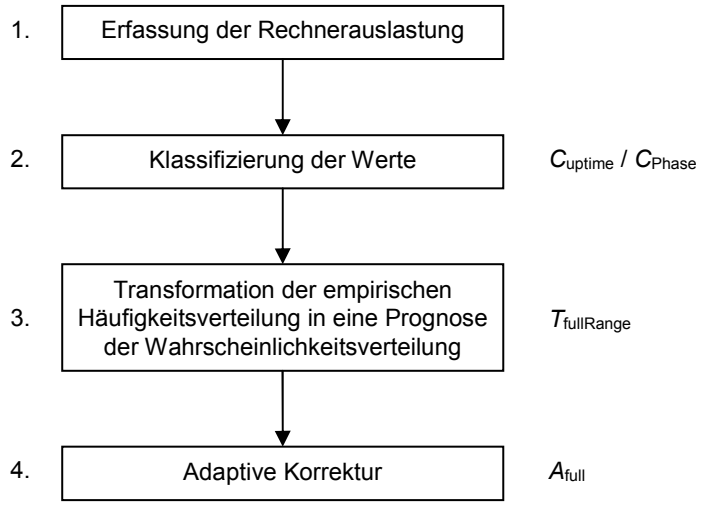

Abbildung 4-24: Übersichtsdarstellung des empfehlenswerten Vorgehens zur Ableitung einer Prognose

Der erste Schritt ist die Erfassung der Auslastung der einzelnen Rechner. Die entsprechenden Daten, aus denen schließlich die Messwerte für K_{free} abzuleiten sind, können lokal auf dem Rechner erhoben und auch dort gespeichert werden.

Im zweiten Schritt bietet sich eine Klassifizierung der Werte an, wobei die Ergebnisse kein eindeutig bestes Kriterium liefern. Vergleichsweise günstig scheint die Klassifizierung nach der Laufzeit des Rechners zu sein, ggf. mit einer weiteren Unterteilung entsprechend der Phase eines Tages. In Abhängigkeit von der gewählten Klassifizierung ergeben sich entsprechende empirische Häufigkeitsverteilungen der Größe K_{free}.

Im dritten Schritt sind die empirischen Häufigkeitsverteilungen, die naturgemäß nur diskrete Werte enthalten, in Wahrscheinlichkeitsverteilungen zu überführen. Für diese Überführung wurden verschiedene Varianten miteinander verglichen, wobei sich die Variante $T_{\text{fullRange}}$ als besonders geeignet herausgestellt hat.

Schließlich ist es zur Verbesserung der Genauigkeit im Durchschnitt noch empfehlenswert, eine Korrektur der sich mit $T_{\text{fullRange}}$ ergebenen Wahrscheinlichkeitsverteilung vorzunehmen. Für diese Maßnahme zeigte sich die adaptive Korrekturvariante A_{full} als besonders wirkungsvoll.

Die hier abgegebene Empfehlung zur Erstellung der Prognosen basiert zunächst auf den Ergebnissen von lediglich vier Testrechnern. Für eine Überprüfung des Nutzens der für die Stufen 2 bis 4 vorgeschlagenen Varianten ist jedoch eine umfangreichere Prüfung anhand anderer Daten erforderlich. Diese Prüfung ist Gegenstand des folgenden Abschnitts.

4.3. Messergebnisse

Nachdem in den bisherigen Abschnitten dieses Kapitels auf verschiedene mögliche Verfahren zur Prognose der freien Rechenkapazitäten eingegangen wurde, sollen nun die Ergebnisse mit den vier Testrechnern anhand anderer Daten überprüft werden. Zu diesem Zweck werden die Verfahren an den in Abschnitt 4.1.2 beschriebenen Untergruppen der zweiten Gruppe von Rechnern getestet.

Um die Kombinationen aus den oben diskutierten Verfahren möglichst kurz und trotzdem eindeutig referenzieren zu können, wird im Weiteren die folgende Schreibweise gewählt:

$$K(t,a[,c])$$

Bei dieser Schreibweise steht t für die angewendete Transformation. Der Parameter a gibt die verwendete adaptive Korrektur an; durch ein Minus-Zeichen wird angezeigt, dass keine Korrektur zum Einsatz kam. Der optionale dritte Parameter (c) steht für das zum Einsatz gekommene Klassifizierungskriterium. Das Kürzel $K(T_{\text{fullRange}},\text{-},C_{\text{uptime}})$ steht also bspw. für die Kombination aus der Transformationsvariante $T_{\text{fullRange}}$ und dem Klassfikationskriterium C_{uptime}.

4.3.1. Rechner in Computer-Pools

4.3.1.1. Windows-Rechner eines Studentenpools

Die erste der Untergruppen besteht aus den Rechnern des Studentenpools R310 der BTU Cottbus (vgl. Abschnitt 4.1.2). Auf diesen ist zum einen Windows installiert, zum anderen Linux. Hier soll es ausschließlich um die verfügbaren Zeiten unter Windows gehen. Die entsprechenden Ergebnisse sind in Tabelle D-11 zusammengefasst.

Ein Vergleich der verschiedenen Transformationsverfahren ist in Abbildung 4-25 und Abbildung 4-26 enthalten. Es zeigt sich, dass die Kombination $K(T_{\text{fullRange}},A_{\text{full}})$ am besten abschneidet, wenngleich der Vorsprung gegenüber den Alternativen nicht ganz so groß ist wie in der Anpassungsphase. Insbesondere liefert hier die Transformationsvariante $T_{\text{hyperexponential}}$ relativ gesehen bessere Ergebnisse als in der Anpassungsphase.

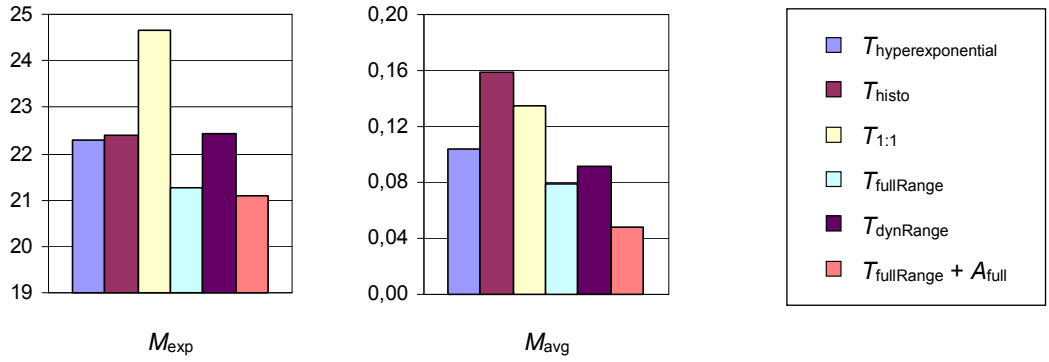

Abbildung 4-25: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg} für den Studenten-Pool R310 unter dem Betriebssystem Windows

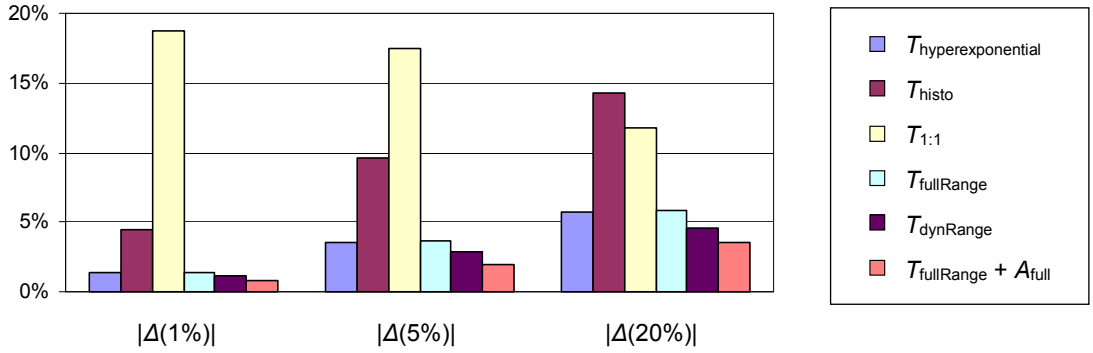

Abbildung 4-26: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für den Studenten-Pool R310 unter dem Betriebssystem Windows

In Abbildung 4-27 und Abbildung 4-28 sind die Ergebnisse der verschiedenen Klassifizierungskriterien einander gegenüber gestellt. Dabei zeigen die Abbildungen nicht die durchschnittlichen Ergebnisse über alle Transformationsverfahren, sondern die Ergebnisse für $K(T_{\text{fullRange}},A_{\text{full}})$, da mit dieser Variante die besten Ergebnisse erzielt wurden und diese folglich auch zum Einsatz kommen sollte. Eine (hier nicht dargestellte) Betrachtung der Durchschnittswerte über alle diskutierten Transformationsverfahren liefert jedoch sehr ähnliche Relationen zwischen den verschiedenen Klassifizierungskriterien. Insgesamt am besten schneidet das Kriterium C_{uptime} ab.

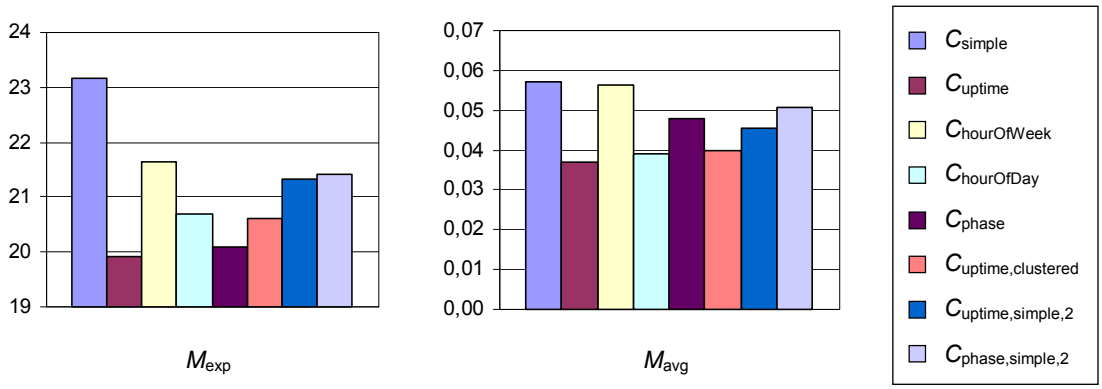

Abbildung 4-27: Vergleich der untersuchten Klassifizierungskriterien mittels M_{exp} und M_{avg} für den Studenten-Pool unter dem Betriebssystem Windows

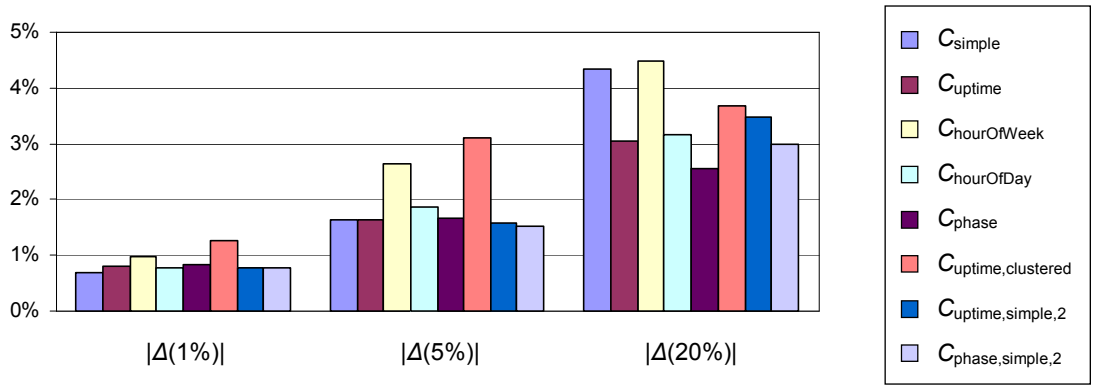

Abbildung 4-28: Vergleich der untersuchten Klassifizierungskriterien mittels $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für den Studenten-Pool unter dem Betriebssystem Windows

4.3.1.2. Linux-Rechner eines Studentenpools

Als nächstes wurde auf den gleichen Rechnern wie in Abschnitt 4.3.1.1 gemessen, wie sich die einzelnen Prognoseverfahren unter Linux verhielten.

Für die getesteten Transformationsverfahren sind die Ergebnisse in Abbildung 4-29 und Abbildung 4-30 zum Vergleich gegenübergestellt, die genauen Zahlenwerte befinden sich in Tabelle D-12. Im hier getesteten Fall ergeben sich wieder ähnlich große Unterschiede wie in der Anpassungsphase, wobei $K(T_{fullRange},A_{full})$ gegenüber den meisten anderen Varianten deutliche Vorteile hat. Lediglich bzgl. der Punktgenauigkeit führt hier $K(T_{dynRange},-)$ zu einem etwas besseren Ergebnis.

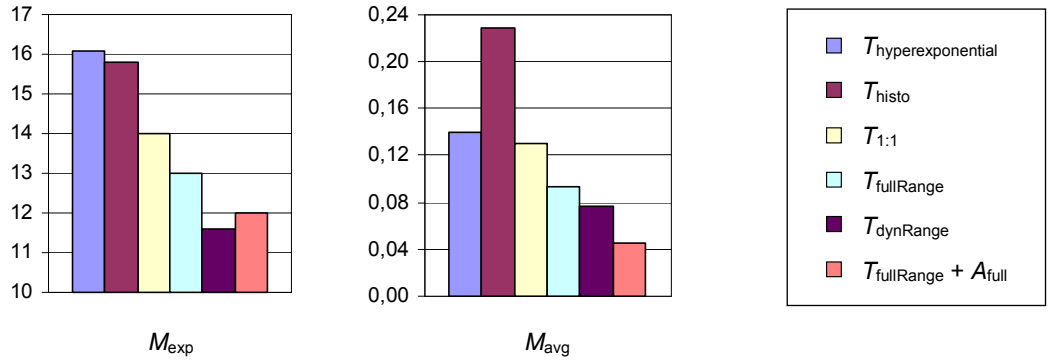

Abbildung 4-29: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg} für den Studenten-Pool unter dem Betriebssystem Linux

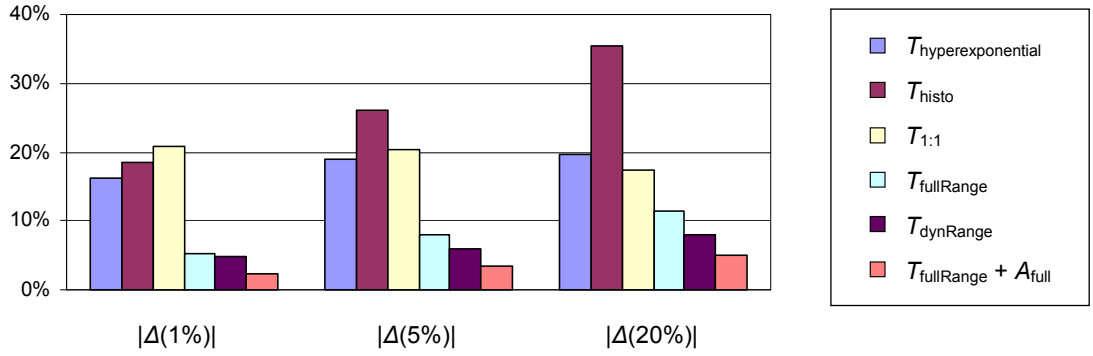

Abbildung 4-30: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für den Studenten-Pool unter dem Betriebssystem Linux

Gegenüberstellungen der verschiedenen Klassifizierungskriterien finden sich in Abbildung 4-31 und Abbildung 4-32. Wiederum liegen diesem Vergleich nicht die Mittelwerte über alle Transformationsverfahren zu Grunde, sondern die Ergebnisse der Variante $K(T_{fullRange},A_{full})$. Für die hier getestete Gruppe gibt es kein Klassifizierungskriterium, das besser als alle anderen ist, sondern mehrere mit ähnlichen Ergebnissen. Zu diesen Kriterien gehört auch C_{uptime}, während bspw. das Kriterium C_{phase}, welches unter Windows für die gleiche Gruppe von Rechnern noch zu relativ guten Ergebnissen geführt hatte, hier zu den schlechtesten Varianten zählt.

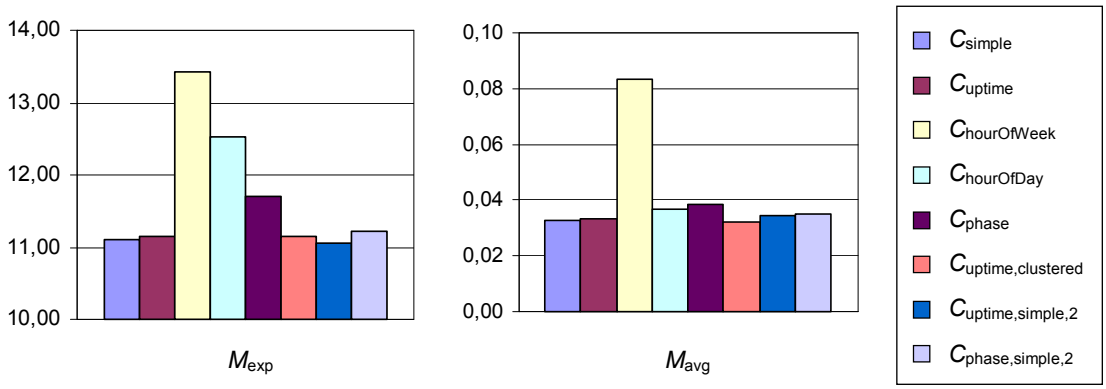

Abbildung 4-31: Vergleich der untersuchten Klassifizierungskriterien mittels M_{exp} und M_{avg} für den Studenten-Pool unter dem Betriebssystem Linux

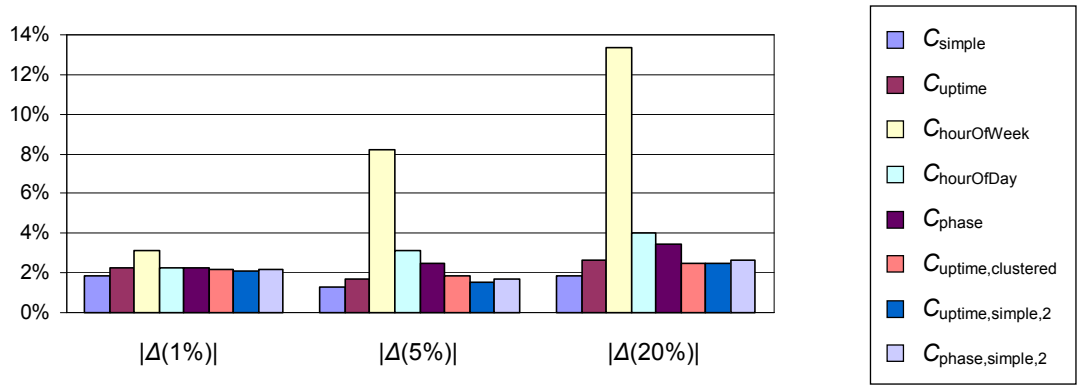

Abbildung 4-32: Vergleich der untersuchten Klassifizierungskriterien mittels $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für den Studenten-Pool unter dem Betriebssystem Linux

4.3.1.3. Linux-Rechner im Pool R411

Im Pool des Lehrstuhls Rechnernetze (Pool R411) wurden weitere Untersuchungen anhand der dort befindlichen Linux-Rechner durchgeführt. Die gemessenen Ergebnisse sind in Tabelle D-13 aufgeführt, graphische Gegenüberstellungen der verschiedenen Transformationsvarianten finden sich in Abbildung 4-33 und Abbildung 4-34. Offenbar schneidet auch hier die Variante $K(T_{fullRange},A_{full})$ am besten ab.

Gegenüberstellungen der verschiedenen Klassifizierungskriterien (bei Verwendung der Transformationsvariante $K(T_{fullRange},A_{full})$) finden sich in Abbildung 4-35 und Abbildung 4-36. Insgesamt schneidet erneut C_{uptime} mit am besten ab, wenngleich es hinsichtlich der Genauigkeit im Durchschnitt durch $C_{hourOfDay}$ übertroffen wird.

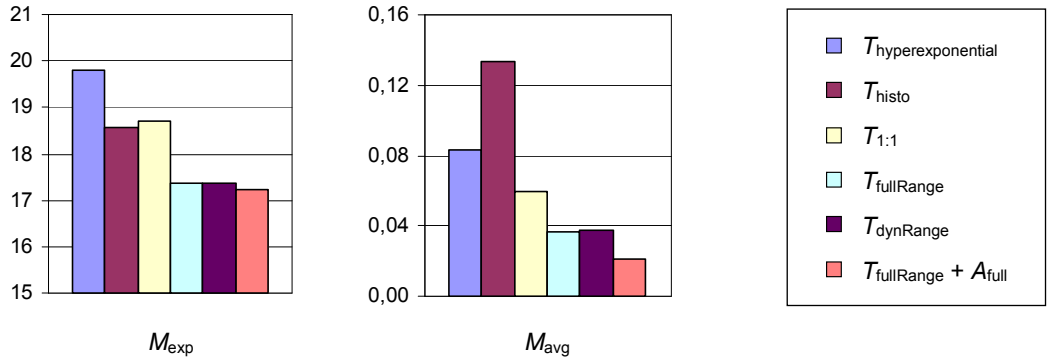

Abbildung 4-33: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg} für den Lehrstuhl-Pool unter dem Betriebssystem Linux

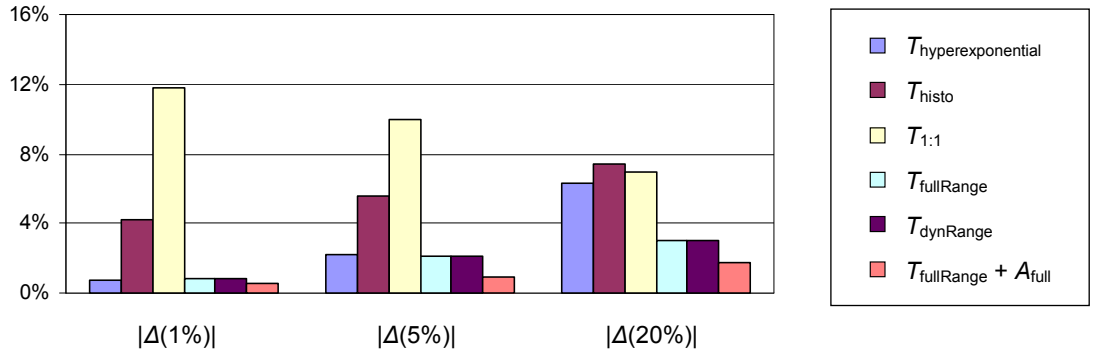

Abbildung 4-34: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für den Lehrstuhl-Pool unter dem Betriebssystem Linux

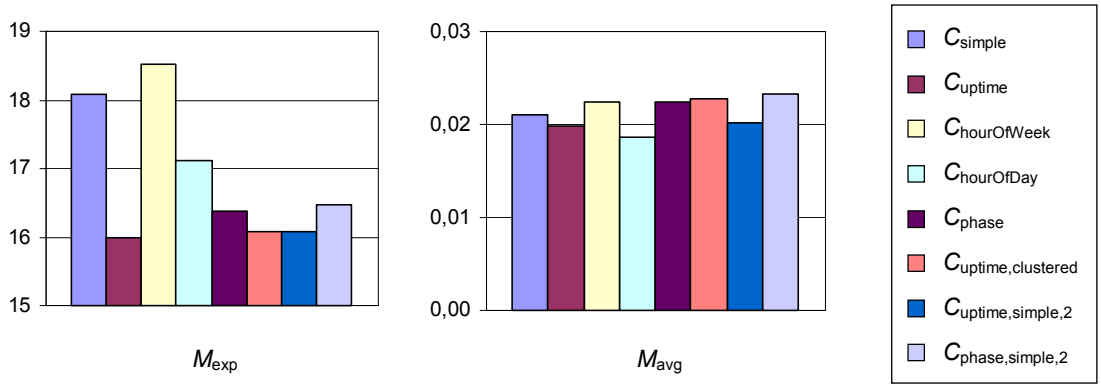

Abbildung 4-35: Vergleich der untersuchten Klassifizierungskriterien mittels M_{exp} und M_{avg} für den Lehrstuhl-Pool unter dem Betriebssystem Linux

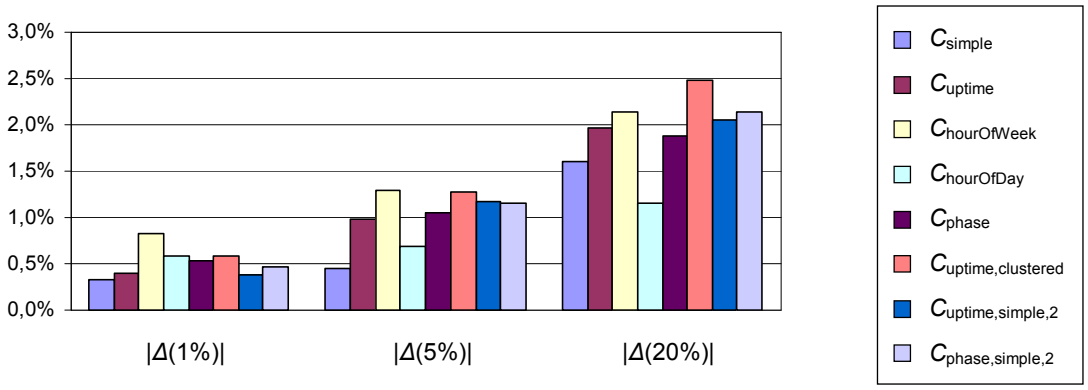

Abbildung 4-36: Vergleich der untersuchten Klassifizierungskriterien mittels $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für den Lehrstuhl-Pool unter dem Betriebssystem Linux

4.3.2. PCs

In einer weiteren Gruppe wurden die untersuchten Rechner zusammengefasst, die als PCs (d. h. im privaten oder beruflichen Bereich personengebunden) eingesetzt wurden. In die Untersuchung flossen dabei vier Rechner ein, für die die Ergebnisse in Tabelle D-14 aufgelistet sind. Abbildung 4-37 und Abbildung 4-38 stellen den Vergleich zwischen den verschiedenen Transformationsverfahren graphisch dar. Wiederum weist $K(T_{\text{fullRange}},A_{\text{full}})$ die höchste Genauigkeit im Durchschnitt auf und liegt auch bzgl. der Punktgenauigkeit vor den der Literatur entnommenen Verfahren $T_{\text{hyperexponential}}$, T_{histo} und $T_{1:1}$.

In Abbildung 4-39 und Abbildung 4-40 sind die Ergebnisse für die untersuchten Klassifizierungskriterien dargestellt, wobei wiederum die Variante $T_{\text{fullRange}}$ in Verbindung mit der adaptiven Korrektur A_{full} zu Grunde liegt. Ein bestes Klassifizierungskriterium gibt es hier nicht, da die bzgl. der Punktgenauigkeit besten Kriterien bzgl. der Genauigkeit im Durchschnitt am schlechtesten abschneiden.

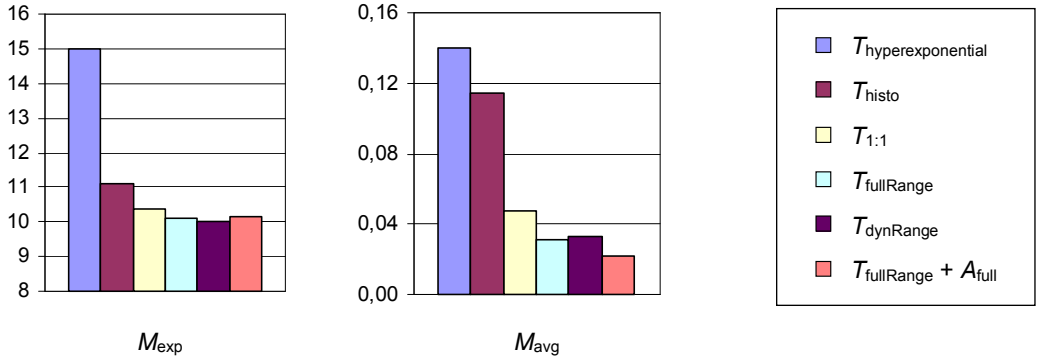

Abbildung 4-37: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg} für die untersuchten PCs

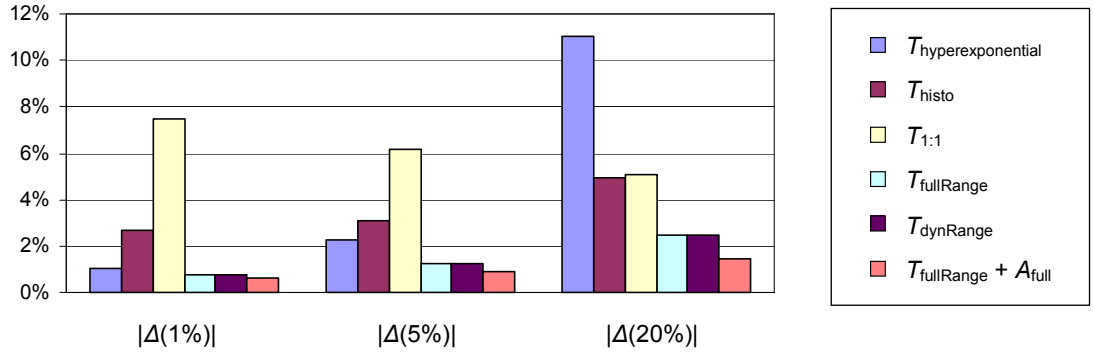

Abbildung 4-38: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für die untersuchten PCs

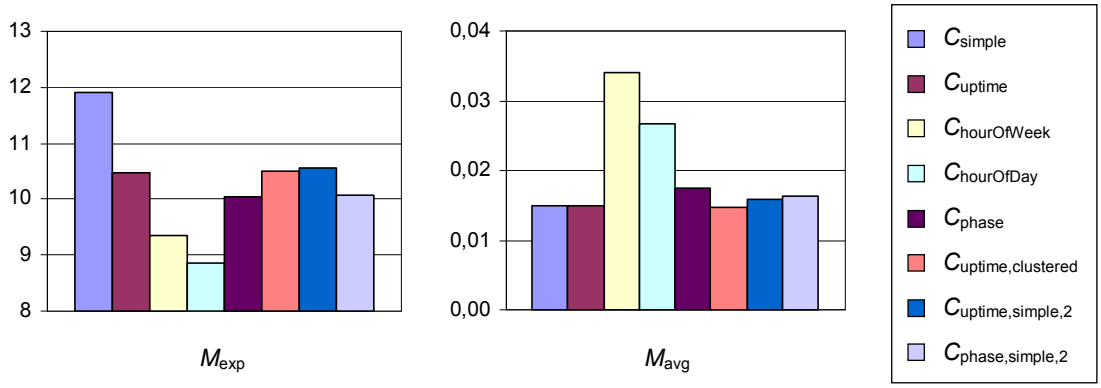

Abbildung 4-39: Vergleich der untersuchten Klassifizierungskriterien mittels M_{exp} und M_{avg} für die untersuchten PCs

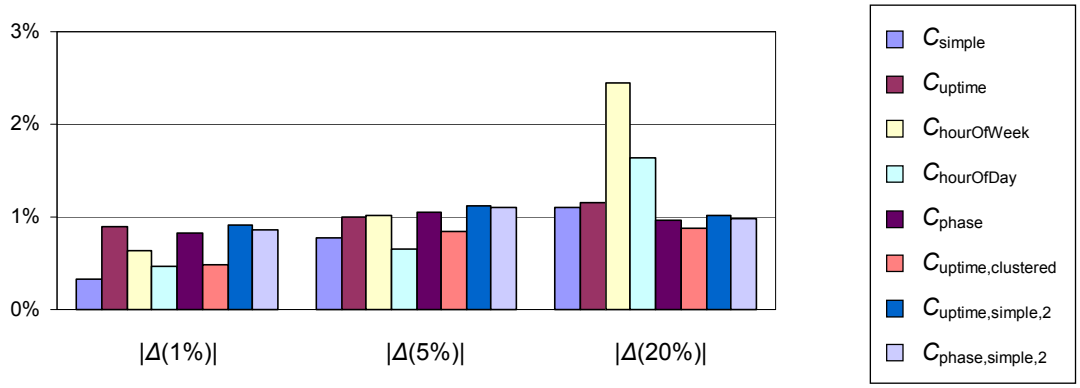

Abbildung 4-40: Vergleich der untersuchten Klassifizierungskriterien mittels $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für die untersuchten PCs

4.3.3. Untersuchte Server-Maschine

Neben den Pool-Rechnern und den PCs wurden ferner zwei als Server betriebene Rechner in die Untersuchungen einbezogen. Da einer dieser beiden Server bereits in der Anpassungsphase verwendet wurde, bleibt nur der Rechner Mouse übrig, der über einen Zeitraum von drei Jahren beobachtet wurde. Dieser Zeitraum wurde aus Gründen der besseren Parallelisierbarkeit der Simulationsdurchläufe in drei etwa 8-monatige Phasen und den Rest eingeteilt, so dass für jeden Zeithorizont (6, 12 oder 24 Stunden) mit vier verschiedenen, einander nicht überlappenden Zeitreihen gearbeitet wurde. Die ermittelten Ergebnisse sind in Tabelle D-15 aufgelistet. Graphische Gegenüberstellungen der verschiedenen Transformationsverfahren liefern Abbildung 4-41 und Abbildung 4-42. Es wird deutlich, dass auch hier die Variante $K(T_{\text{fullRange}},A_{\text{full}})$ am besten abschneidet, und zwar sowohl hinsichtlich der Punktgenauigkeit als auch hinsichtlich der Genauigkeit im Durchschnitt.

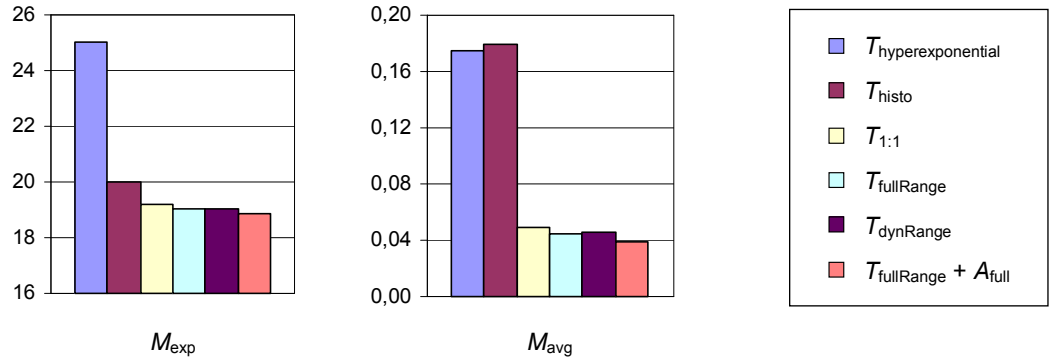

Abbildung 4-41: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg} für die untersuchte Server-Maschine

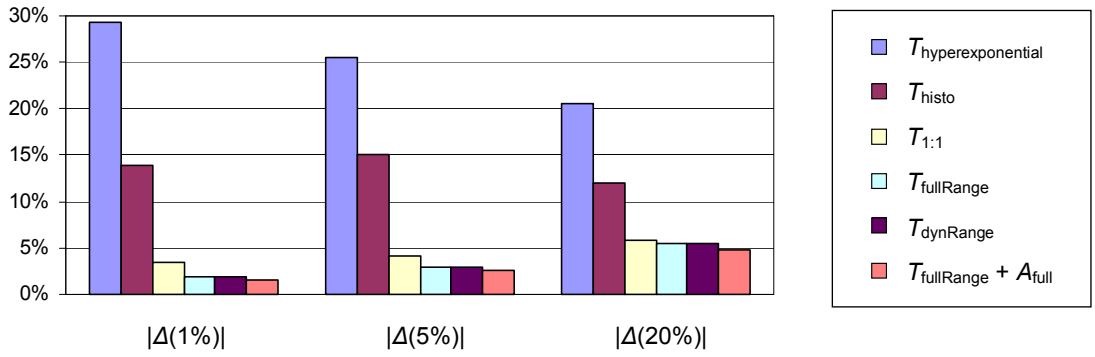

Abbildung 4-42: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für die untersuchte Server-Maschine

Neben den Transformationsverfahren wurden auch hier die Klassifizierungskriterien miteinander verglichen, die graphische Darstellung der Ergebnisse liefern Abbildung 4-43 und Abbildung 4-44. Wiederum gibt es kein klar bestes Kriterium. C_{uptime} zählt zwar erneut zu den vergleichsweise günstigen Kriterien, $C_{\text{uptime,clustered}}$ schneidet jedoch etwas besser ab.

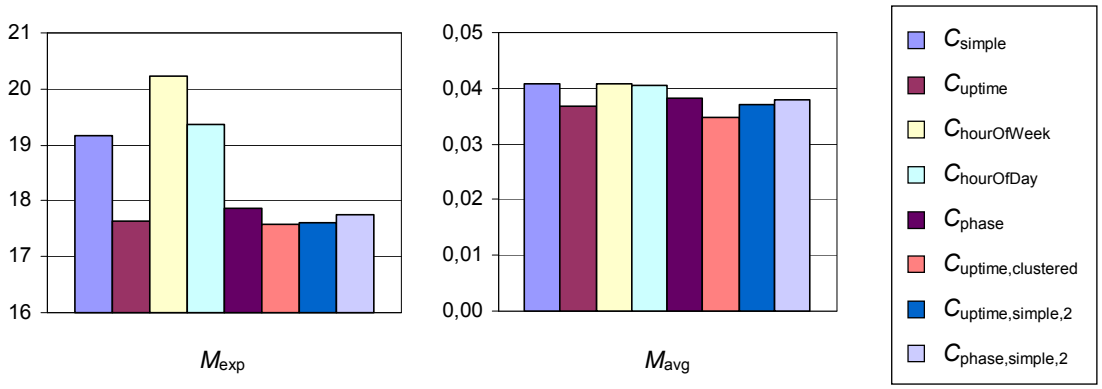

Abbildung 4-43: Vergleich der untersuchten Klassifizierungskriterien mittels M_{exp} und M_{avg} für die untersuchte Server-Maschine

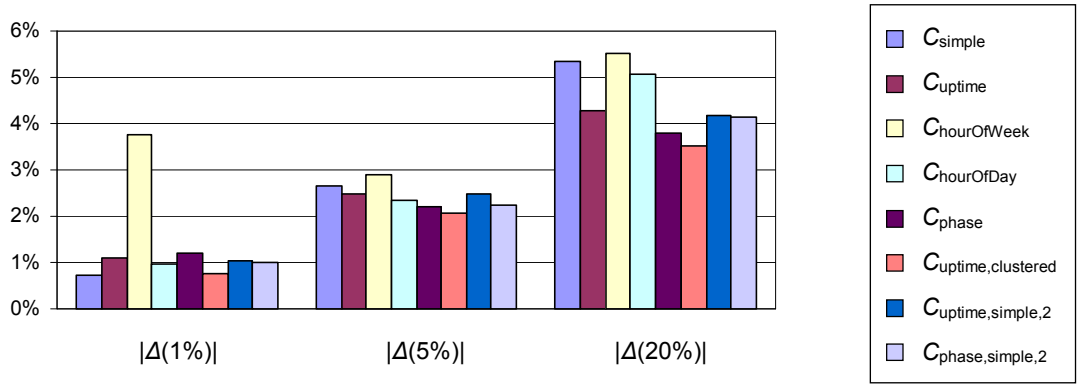

Abbildung 4-44: Vergleich der untersuchten Klassifizierungskriterien mittels $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für die untersuchte Server-Maschine

4.3.4. Rechner aus der Untersuchung in [Long95]

Schließlich wurden Simulationen mit den Rechnerdaten aus der Untersuchung in [Long95] durchgeführt. Für die meisten Rechner standen Daten aus etwa einem halben Jahr zur Verfügung. Für die Simulationen wurden jene Rechner nicht betrachtet, für die nur Informationen aus einem Zeitraum von weniger als 12 Wochen existierten. Ferner wurden die Daten von offensichtlich fehlerhaften Informationen bereinigt. Übrig blieben die Daten von 992 Rechnern, für die Untersuchungen mit allen Transformationsverfahren außer $T_{\text{hyperexponential}}$ durchgeführt wurden. Eine Untersuchung mit $T_{\text{hyperexponential}}$

war leider aufgrund des hohen Rechenaufwandes nicht für alle 992 Rechner möglich, da dieses Transformationsverfahren einen gegenüber allen anderen Verfahren deutlich höheren Berechnungsaufwand verursacht.

Um $T_{\text{hyperexponential}}$ dennoch in den Vergleich einbeziehen zu können, wurde eine zufällige Auswahl von 20 verschiedenen Rechnern getroffen, anhand derer die Vergleichsuntersuchungen durchgeführt wurden. Im Weiteren sind nur die Ergebnisse für diese 20 Rechner dargestellt. Die Relationen zwischen den von $T_{\text{hyperexponential}}$ verschiedenen Transformationsverfahren waren dabei aber sehr ähnlich zu denen bei der Betrachtung der 992 Rechner.

Für die bereits in Abschnitt 4.1.3 angesprochene, notwendige Ableitung einer Näherung für die Größe K_{free} aus den Informationen über die Zeiten des Angeschaltetseins wurden die zufälligen Auslastungswerte für die einzelnen Zeitpunkte nach dem in Abschnitt B.2 angegebenen Verfahren erzeugt. Mit diesem Verfahren werden realistische durchschnittliche Auslastungswerte erreicht und auch die Größe der Schwankungen ist praxisnah. Grundsätzliches und mit den gegebenen Daten nicht zu lösendes Problem ist jedoch, dass über das Vorhandensein regelmäßiger Schwankungen (z. B. in Abhängigkeit von der Tageszeit oder von der Zeitdifferenz zum letzten Systemstart) nur gemutmaßt werden kann. Im verwendeten Verfahren zur Generierung der Auslastungswerte wurde daher gänzlich auf die Erzeugung solcher regelmäßigen Schwankungen verzichtet, was jedoch Klassifizierungskriterien benachteiligt, die solche Regelmäßigkeiten erwarten (dazu später mehr).

Mit den aus den zufälligen Auslastungswerten für K_{free} abgeleiteten Werten wurden schließlich die experimentellen Untersuchungen wie mit den anderen Rechnerdaten durchgeführt, die Messergebnisse sind in Tabelle D-16 aufgelistet. Ein graphischer Vergleich der verschiedenen Transformationsverfahren ist in Abbildung 4-45 und Abbildung 4-46 zu finden. Die Dargestellungen deuten auch hier darauf hin, dass die Kombination $K(T_{\text{fullRange}},A_{\text{full}})$ insgesamt die besten Ergebnisse liefert.

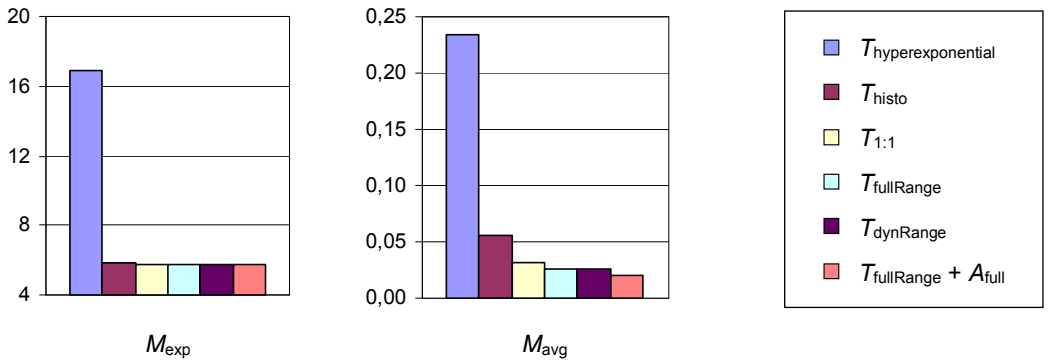

Abbildung 4-45: Ergebnisse der Transformationsverfahren bzgl. M_{exp} und M_{avg} für die untersuchten Rechner aus [Long95]

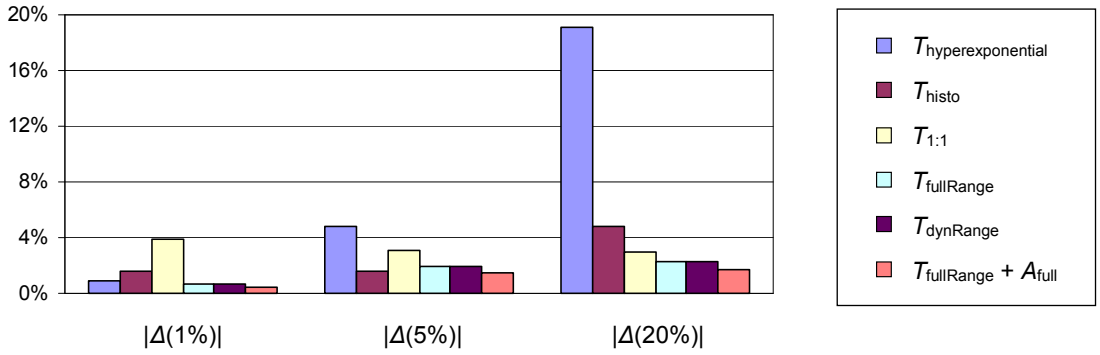

Abbildung 4-46: Ergebnisse der Transformationsverfahren bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für die untersuchten Rechner aus [Long95]

Ein Vergleich der Klassifizierungskriterien (Abbildung 4-47, Abbildung 4-48) liefert keine klare Empfehlung für das zu wählende Kriterium. Im Vergleich zu den anderen Gruppen von Rechnern, für welche die Auslastungswerte nicht zufällig erzeugt werden mussten, schneidet die Variante C_{simple} hier jedoch sehr gut ab. Die dafür plausibelste Erklärung dürfte das oben bereits erwähnte Problem sein, dass bei der Generierung der Auslastungswerte keine regelmäßigen Schwankungen erzeugt werden, die jedoch Grundlage für alle Kriterien außer C_{simple} sind. Insofern dürfte zumindest der Vergleich der Klassifizierungskriterien anhand der Daten aus [Long95] nicht besonders aussagekräftig sein.

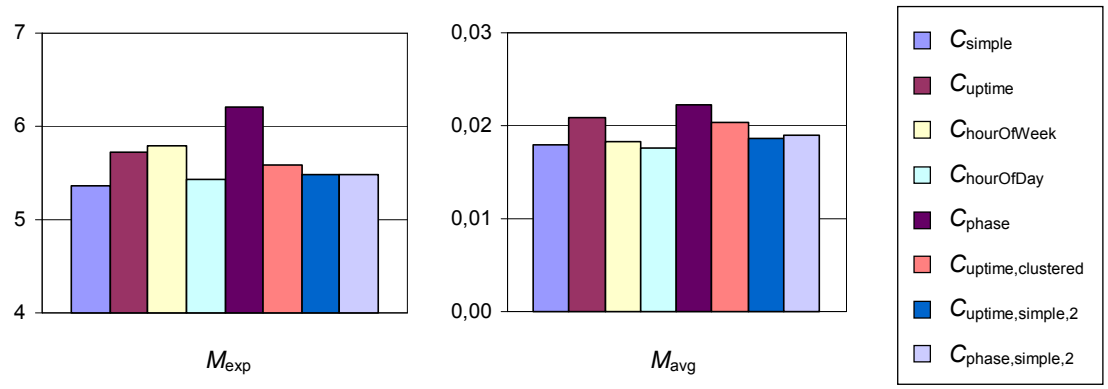

Abbildung 4-47: Vergleich der untersuchten Klassifizierungskriterien mittels M_{exp} und M_{avg} für die untersuchten Rechner aus [Long95]

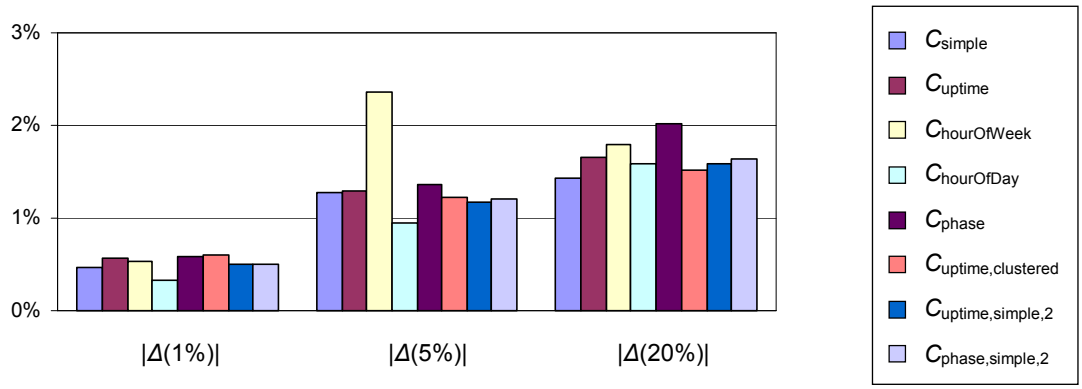

Abbildung 4-48: Vergleich der untersuchten Klassifizierungskriterien mittels $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ für die untersuchten Rechner aus [Long95]

4.3.5. Signifikanztests

Die mit den einzelnen Gruppen von Rechnern durchgeführten Tests bestätigen die wesentlichen Ergebnisse der Anpassungsphase. Demnach schneidet von den untersuchten Transformationsverfahren $K(T_{\text{fullRange}},A_{\text{full}})$ am besten ab. Unter den Klassifizierungskriterien gibt es keinen eindeutigen Sieger, allerdings gehört C_{uptime} meist zu den besten Varianten, während andere Varianten entweder zum Teil oder auch immer deutlich schlechtere Ergebnisse liefern. Da diese Ergebnisse (insbesondere die Vorteilhaftigkeit von $K(T_{\text{fullRange}},A_{\text{full}})$ gegenüber anderen Transformationsverfahren) so bei allen untersuchten Rechnergruppen zu beobachten war, ist dies bereits ein Indiz dafür, dass die beobachteten Relationen relativ allgemeingültig sind.

Zur Prüfung der Signifikanz der Ergebnisse soll hier jedoch eine zusätzliche Prüfung durchgeführt werden. Dabei stellen sich im Wesentlichen zwei Fragen:

F1) Ist die Transformationsvariante $K(T_{\text{fullRange}},A_{\text{full}})$ tatsächlich besser als bisher angewendete Transformationsvarianten?

F2) Ist die Kombination $K(T_{\text{fullRange}},A_{\text{full}},C_{\text{uptime}})$ tatsächlich besser als bisher angewendete Kombinationen?

Da es um einen Vergleich der in dieser Arbeit vorgeschlagenen Prognoseverfahren mit in der Literatur bereits vorgeschlagenen Verfahren geht, ist insbesondere ein Vergleich mit den Transformationen $T_{\text{hyperexponential}}$, T_{histo} und $T_{1:1}$ von Interesse. Zusätzlich dazu wird aber auch ein Vergleich zu den anderen in der vorliegenden Arbeit betrachteten Verfahren vorgenommen.

4.3.5.1. Vorgehensweise mittels Bootstrap-Verfahren

Die sicherste Methode, um die Siginifikanz von Messergebnissen zu prüfen, ist eine Überprüfung der Ergebnisse an vielen verschiedenen, voneinander unabhängigen Stichproben. In gewisser Weise wurde dies hier durch die getrennte Untersuchung der verschiedenen Rechnergruppen auch getan. Da die einzelnen Gruppen jedoch zum Teil

zu klein waren, soll eine zusätzliche, alternative Herangehensweisen zur Einschätzung der Signifikanz der Ergebnisse zum Einsatz kommen, bei der aber keine weiteren (hier auch nicht vorhandenen) Daten zur Auslastung von Rechnern erforderlich sind.

Eine solche Variante ist die Bootstrap-Methode ([Good99], [Good05]), bei der aus der vorhandenen Menge von erhobenen Daten neue Stichproben gebildet werden. Die Bildung einer neuen Stichprobe aus der erhobenen Menge von n Werten erfolgt dabei durch das n-malige zufällige Auswählen von Werten aus der ermittelten Stichprobe, wobei die Werte auch mehrfach ausgewählt werden können. (Es erfolgt also ein Ziehen mit Zurücklegen.) Auf diese Weise können viele verschiedene Stichproben erzeugt werden, die (vermutlich) statt der ursprünglich ermittelten Stichprobe hätten auftreten können. Für die Anzahl künstlich zu bildender Stichproben wird in [Good05] ein Wert von etwa 1000 empfohlen, weshalb in der vorliegenden Arbeit mit diesem Wert gearbeitet wurde, der auch zu stabilen Ergebnissen führte.

Für jede der künstlich erzeugten Stichproben erfolgt eine Überprüfung, ob sich das mit der ursprünglichen Stichprobe erzielte Ergebnis bestätigt. Für ein signifikantes Ergebnis wird i. Allg. erwartet, dass sich das Ergebnis in 95 % der Fälle bestätigt, für ein hochsignifikantes Ergebnis in wenigstens 99 % der Fälle. Voraussetzung ist dabei, dass die einzelnen, in der ursprünglichen Stichprobe enthaltenen Werte voneinander unabhängig sind.

Für die hier durchgeführte Untersuchung wird dabei von den für die Messungen genutzten Rechnern der Gruppe 2 (vgl. Tabelle 4-2) ausgegangen. Es handelt sich somit um 26 verschiedene Pool-Rechner (wobei auf 20 dieser zwei verschiedene Betriebssysteme installiert waren, für die getrennte Prognosen gemacht wurden), um einen Server, vier PCs, insgesamt also um 31 verschiedene Rechner. Zusätzlich dazu werden die 20 Rechner aus der Untersuchung von [Long95] berücksichtigt, so dass insgesamt 51 Rechner einbezogen werden können. Aufgrund der in Abschnitt 4.3.4 diskutierten Probleme mit den Daten aus [Long95] werden jedoch zwei verschiedene Untersuchungen durchgeführt, einmal für alle 51 Rechner und einmal nur für die 31 Rechner, für die vollständige Daten vorliegen (also ohne die Rechner aus [Long95]).

In beiden Fällen ist die Zahl der zugrunde liegenden Zeitreihen größer als die Zahl der Rechner, da drei verschiedene Zeithorizonte von 6, 12 und 24 Stunden unterschieden wurden und außerdem für verschiedene Betriebssysteme auch verschiedene Zeitreihen vorliegen. Ferner wurden beim Server Mouse die Daten in vier Zeitreihen aufgeteilt, um die Simulationen besser parallelisieren zu können. Da die auf dem gleichen Rechner ermittelten unterschiedlichen Datensätze jedoch nicht als voneinander unabhängig angenommen werden können, wurde hier bei der Bildung einer Bootstrap-Stichprobe die Auswahl eines einzelnen Datensatzes so durchgeführt, dass bei n Rechnern zunächst mit einer Wahrscheinlichkeit von jeweils $1/n$ einer der Rechner ausgewählt wurde – und für diesen dann unter den zur Verfügung stehenden Zeitreihen zufällig gewählt wurde. Konsequenterweise wurden also Stichproben der Größe 51 bzw. 31 gewählt, da nur die Daten unterschiedlicher Rechner als voneinander unabhängig angenommen werden können.

Ein Problem der gerade beschriebenen Verwendung der Bootstrap-Methode ist, dass die ermittelbaren Prozentwerte nicht unverzerrt sind. Aus diesem Grunde wurde die BC_a-Methode entwickelt, welche ebenfalls Bootstrap-Stichproben bildet, die Ableitung der Prozentwerte aber anders durchführt und dadurch die Verzerrung korrigiert, was zu erheblich besseren Schätzungen führt ([Good99]). Zur Überprüfung der Ergebnisse mit der oben erläuterten Bootstrap-Methode wurde daher zusätzlich eine Überprüfung mittels der BC_a-Methode durchgeführt.

4.3.5.2. Ergebnisse für alle 51 Rechner

Vergleich der Transformationsverfahren

Zunächst fand zur Beantwortung der Frage F1 (vgl. S. 111) ein Vergleich zwischen der Transformation $K(T_{\text{fullRange}},A_{\text{full}})$ und alternativen Varianten statt. Bei diesem Vergleich ist zum einen interessant, ob $K(T_{\text{fullRange}},A_{\text{full}})$ besser als die der Literatur entnommenen Verfahren ist, zum anderen, ob $K(T_{\text{fullRange}},A_{\text{full}})$ besser als die in dieser Arbeit zusätzlich betrachteten Verfahren ist. Im ersten Fall geht es also um den Vergleich mit den Transformationen $T_{\text{hyperexponential}}$, T_{histo} und $T_{1:1}$, im zweiten Fall zusätzlich um den Vergleich mit $T_{\text{fullRange}}$ und T_{dynRange} (jeweils ohne adaptive Korrektur). Die Vergleiche wurden einerseits anhand des Maßes M_{exp}, andererseits anhand des Maßes M_{avg} durchgeführt.

Die in Tabelle 4-3 und Tabelle 4-4 dargestellten Ergebnisse basieren auf den Mittelwerten der Ergebnisse der untersuchten Klassifizierungskriterien. Zusätzlich dazu sind in Klammern die bei den mit den einzelnen Klassifizierungskriterien durchgeführten Vergleichen ermittelten Minimalwerte angegeben. Die Prozentzahlen stellen dabei jeweils den Anteil der künstlich erzeugten Stichproben dar, für den $K(T_{\text{fullRange}},A_{\text{full}})$ gegenüber dem jeweils untersuchten alternativen Transformationsverfahren besser abgeschnitten hat. So lässt sich bspw. aus Tabelle 4-3 ablesen, dass $K(T_{\text{fullRange}},A_{\text{full}})$ im Vergleich zu $K(T_{\text{dynRange}},\text{-})$ bzgl. der Punktgenauigkeit signifikant besser abschneidet, wenn man die Mittelwerte von M_{exp} über alle betrachteten Klassifizierungskriterien vergleicht, da diese in 98,5 % der Fälle bei $K(T_{\text{fullRange}},A_{\text{full}})$ niedriger als bei $K(T_{\text{dynRange}},\text{-})$ waren. Auch bei einer Einzelbetrachtung der getesteten Klassifizierungskriterien schneidet die Variante $K(T_{\text{fullRange}},A_{\text{full}})$ gegenüber $K(T_{\text{dynRange}},\text{-})$ signifikant besser ab, da $K(T_{\text{fullRange}},A_{\text{full}})$ für jedes getestete Kriterium in wenigstens 96,0 % der Fälle den niedrigeren Wert aufwies.

<table>
<tr><th colspan="3">Transformationen aus der Literatur</th><th colspan="2">Eigene Transformationsverfahren</th></tr>
<tr><th>$T_{\text{hyperexponential}}$</th><th>$T_{\text{histo}}$</th><th>$T_{1:1}$</th><th>$T_{\text{fullRange}}$</th><th>$T_{\text{dynRange}}$</th></tr>
<tr><td>100,0 %
(100,0 %)</td><td>100,0 %
(99,5 %)</td><td>100,0 %
(97,6 %)</td><td>99,9 %
(94,5 %)</td><td>98,5 %
(96,0 %)</td></tr>
<tr><td colspan="3">100,0 %
(≥ 97,6 %)</td><td colspan="2">≥ 98,5 %
(≥ 94,5 %)</td></tr>
<tr><td colspan="5">≥ 98,5 %
(≥ 94,5 %)</td></tr>
</table>

Tabelle 4-3: Signifikanzwerte anhand des Maßes M_{exp}

Transformationen aus der Literatur			Eigene Transformationsverfahren	
$T_{\text{hyperexponential}}$	T_{histo}	$T_{1:1}$	$T_{\text{fullRange}}$	T_{dynRange}
100,0 %	100,0 %	100,0 %	100,0 %	100,0 %
100,0 %			100,0 %	
100,0 %				

Tabelle 4-4: Signifikanzwerte anhand des Maßes M_{avg}

Insgesamt deuten die dargestellten Ergebnisse darauf hin, dass die Kombination $K(T_{\text{fullRange}},A_{\text{full}})$ Prognosen mit signifikant höherer Punktgenauigkeit liefert als die der Literatur entnommenen Verfahren. Auch gegenüber den anderen eigenen Transformationsverfahren sind die Ergebnisse signifikant besser, wenn man von der einzigen Ausnahme absieht, nämlich der Kombination $K(T_{\text{fullRange}},A_{\text{full}},C_{\text{simple}})$, für welche mit 94,5% die 95%-Grenze knapp verfehlt wurde. Mit allen anderen Kriterien (zu denen auch das scheinbar relativ vorteilhafte Kriterium C_{uptime} gehört) wurde jedoch die 95-Prozent-Grenze erreicht. Bezüglich der Genauigkeit im Durchschnitt ist in allen Fällen eine hochsignifikante Verbesserung festgestellt worden.

Die Ausführung der BC_a-Methode verstärkt die hier mit der Bootstrap-Methode ermittelten Ergebnisse sogar, da sie die Kombination $K(T_{\text{fullRange}},A_{\text{full}})$ gegenüber allen anderen Transformationsverfahren als hochsignifikant besser einschätzt – sowohl hinsichtlich der Genauigkeit im Durchschnitt als auch hinsichtlich der Punktgenauigkeit.

Vergleich des Gesamtverfahrens

Für die letztlich zu erstellende Prognose ist das hier als vorteilhaft erscheinende Transformationsverfahren mit einem geeigneten Klassifizierungskriterium zu kombinieren. Dabei lieferte in den bisherigen Untersuchungen jeweils C_{uptime} vergleichsweise gute Ergebnisse. Aus diesem Grunde wurde schließlich entsprechend der Fragestellung F2 (vgl. S. 111) geprüft, inwieweit die Kombination $K(T_{\text{fullRange}},A_{\text{full}},C_{\text{uptime}})$ gegenüber den anderen möglichen Kombinationen aus Transformationsverfahren und Klassifizierungskriterium vorteilhaft ist, wofür entsprechende paarweise Vergleiche durchgeführt wurden. Die Ergebnisse sind in Tabelle 4-5 und Tabelle 4-6 aufgeführt, wobei für jedes Transformationsverfahren der Signifikanzwert desjenigen Klassifizierungskriteriums angegeben ist, das am häufigsten besser abgeschnitten hat als die hier untersuchte Kombination $K(T_{\text{fullRange}},A_{\text{full}},C_{\text{uptime}})$. Das entsprechende Klassifizierungskriterium ist dann in Klammern angegeben. Wie man also bspw. aus Tabelle 4-5 ablesen kann, wurde beim Vergleich mit möglichen, die Transformationsvariante T_{histo} nutzenden Verfahren festgestellt, dass unabhängig vom dabei verwendeten Klassifizierungskriterium die Variante $K(T_{\text{fullRange}},A_{\text{full}},C_{\text{uptime}})$ bzgl. der Punktgenauigkeit in mindestens 99,9 % der Fälle besser abschneidet, wobei mit $K(T_{\text{histo}},\text{-},C_{\text{uptime,simple,2}})$ der niedrigste Wert gemessen wurde.

Transformationen aus der Literatur			Eigene Transformationsverfahren		
$T_{hyperexponential}$	T_{histo}	$T_{1:1}$	$T_{fullRange}$	$T_{dynRange}$	$T_{fullRange} + A_{full}$
100,0 %	≥ 99,9 % ($C_{uptime,simple,2}$)	100,0 %	≥ 95,9 % ($C_{uptime,clustered}$)	≥ 97,0 % ($C_{uptime,clustered}$)	≥ 28,7 % ($C_{uptime,clustered}$)
≥ 99,9 %			≥ 97,0 %		
≥ 97,0 %					

Tabelle 4-5: Signifikanzwerte anhand des Maßes M_{exp}

Transformationen aus der Literatur			Eigene Transformationsverfahren		
$T_{hyperexponential}$	T_{histo}	$T_{1:1}$	$T_{fullRange}$	$T_{dynRange}$	$T_{fullRange} + A_{full}$
100,0 %	100,0 %	100,0 %	100,0 %	100,0 %	≥ 38,7 % ($C_{uptime,clustered}$)
100,0 %			100,0 %		
100,0 %					

Tabelle 4-6: Signifikanzwerte anhand des Maßes M_{avg}

Zusammenfassend lässt sich für die untersuchten Rechner sagen, dass die Kombination $K(T_{fullRange},A_{full},C_{uptime})$ mit hochgradiger Signifikanz besser ist als die aus der Literatur entnommenen Verfahren. Im Vergleich mit weiteren eigenen, nicht adaptiv korrigierten Verfahren ist die Genauigkeit im Durchschnitt ebenfalls hochgradig signifikant verbessert, bzgl. der Punktgenauigkeit besteht eine signifikante Verbesserung. Lediglich bzgl. der Kombination anderer Klassifizierungskriterien mit $T_{fullRange}$ und A_{full} ergibt sich keine signifikante Verbesserung. Dies deutet wiederum darauf hin, dass es unter den untersuchten Klassifizierungskriterien kein eindeutig bestes gibt.

Die Anwendung der BC_a-Methode bestätigte die Ergebnisse der Bootstrap-Methode.

4.3.5.3. Ergebnisse bei Exklusion der Rechner aus [Long95]

Vergleich der Transformationsverfahren

Die Ergebnisse der Signifikanzuntersuchungern für die 31 Rechner, für die vollständige Daten vorlagen, werden hier nach dem gleichen Schema präsentiert wie für die 51 Rechner im vorherigen Abschnitt, die entsprechenden Informationen zum Vergleich zwischen den Transformationsverfahren befinden sich in Tabelle 4-7 und Tabelle 4-8. Demnach liefert das Transformationsverfahren $K(T_{fullRange},A_{full})$ gegenüber allen anderen untersuchten Verfahren Prognosen mit signifikant höherer Punktgenauigkeit und hochsignifikant höherer Genauigkeit im Durchschnitt. Die Ausführung der BC_a-Methode bestätigte die Ergebnisse der Bootstrap-Methode.

Transformationen aus der Literatur			Eigene Transformationsverfahren	
$T_{hyperexponential}$	T_{histo}	$T_{1:1}$	$T_{fullRange}$	$T_{dynRange}$
100,0 % (100,0 %)	100,0 % (99,0 %)	100,0 % (97,8 %)	99,7 % (95,5 %)	99,4 % (95,6 %)
100,0 % (≥ 97,8 %)			≥ 99,4 % (≥ 95,5 %)	
≥ 99,4 % (≥ 95,5 %)				

Tabelle 4-7: Signifikanzwerte anhand des Maßes M_{exp}

Transformationen aus der Literatur			Eigene Transformationsverfahren	
$T_{hyperexponential}$	T_{histo}	$T_{1:1}$	$T_{fullRange}$	$T_{dynRange}$
100,0 %	100,0 %	100,0 %	100,0 %	100,0 %
100,0 %			100,0 %	
100,0 %				

Tabelle 4-8: Signifikanzwerte anhand des Maßes M_{avg}

Vergleich des Gesamtverfahrens

Der bei den Signifikanzprüfungen durchgeführte Vergleich zwischen den Gesamtverfahren lieferte die in Tabelle 4-9 und Tabelle 4-10 dargestellten Ergebnisse, welche die Kombination $K(T_{fullRange},A_{full},C_{uptime})$ gegenüber allen Varianten aus der Literatur sowohl hinsichtlich der Punktgenauigkeit als auch hinsichtlich der Genauigkeit im Durchschnitt als hochsignifikant besser ausweisen. Auch gegenüber den eigenen Transformationsvarianten sind die Ergebnisse signifikant besser, wenn man die Variante $K(T_{fullRange},A_{full})$ außen vorlässt. Es ist also wiederum festzustellen, dass es kein klar besten Klassifizierungskriterium gibt, die Verwendung von C_{uptime} in Verbindung mit $K(T_{fullRange},A_{full})$ aber insgesamt empfehlenswert ist. Die Ergebnisse der BC_a-Methode bestätigen auch hier die Ergebnisse der Bootstrap-Methode.

Transformationen aus der Literatur			Eigene Transformationsverfahren		
$T_{hyperexponential}$	T_{histo}	$T_{1:1}$	$T_{fullRange}$	$T_{dynRange}$	$T_{fullRange}$ + A_{full}
100,0 %	100,0 %	100,0 %	≥ 97,7 % ($C_{uptime,clustered}$)	≥ 98,7 % ($C_{uptime,clustered}$)	≥ 49,0 % ($C_{uptime,clustered}$)
100,0 %			≥ 97,7 %		
≥ 97,7 %					

Tabelle 4-9: Signifikanzwerte anhand des Maßes M_{exp}

Transformationen aus der Literatur			Eigene Transformationsverfahren		
$T_{hyperexponential}$	T_{histo}	$T_{1:1}$	$T_{fullRange}$	$T_{dynRange}$	$T_{fullRange} + A_{full}$
100,0 %	100,0 %	100,0 %	100,0 %	100,0 %	≥ 48,8 % ($C_{uptime,clustered}$)
100,0 %			100,0 %		
100,0 %					

Tabelle 4-10: Signifikanzwerte anhand des Maßes M_{avg}

4.3.6. Fazit

Zusammenfassend lässt sich festhalten, dass die in der Anpassungsphase abgeleitete und in Abschnitt 4.2.5 beschriebene Vorgehensweise sich tatsächlich als besonders empfehlenswert herausgestellt hat. Zur besseren Nachvollziehbarkeit ist diese Vorgehensweise hier noch einmal in Abbildung 4-49 veranschaulicht.

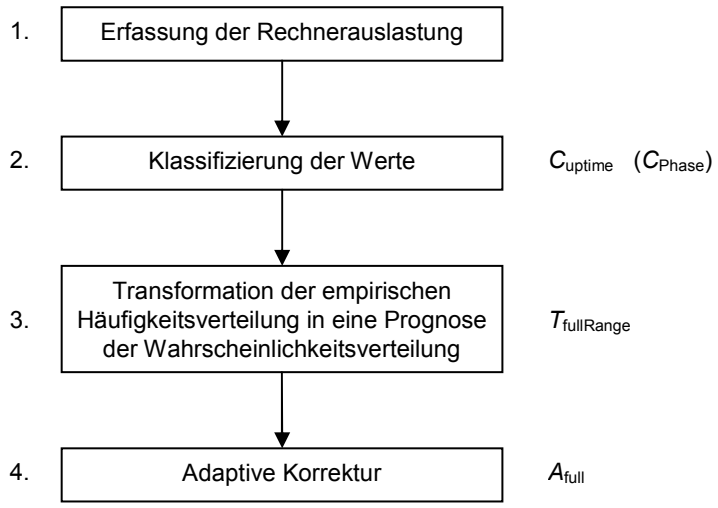

Abbildung 4-49: Übersichtsdarstellung des empfehlenswerten Vorgehens zur Ableitung einer Prognose

So wurde zum einen die große Bedeutung der Transformationsstufe bestätigt. Es wurde festgestellt, dass das in dieser Arbeit vorgeschlagene Transformationsverfahren $T_{fullRange}$ in Verbindung mit der adaptiven Korrektur A_{full} gegenüber allen der Literatur entnommenen und auch gegenüber den anderen in dieser Arbeit untersuchten Transformationsverfahren signifikant oder sogar hochsignifikant besser abschneidet. Dies gilt sowohl für die Genauigkeit im Durchschnitt als auch für die Punktgenauigkeit der Vorhersagen.

Zum anderen hat sich wie auch schon in der Anpassungsphase kein eindeutig bestes Klassifizierungskriterium ergeben. Vergleichsweise gut schnitten jedoch wiederum die Kriterien C_{uptime} und C_{Phase} ab, wobei sich die Betrachtungen auf ersteres konzentrierten. Demzufolge kann für das für die Prognosen anzuwendende Gesamtverfahren die Kombination aus $K(T_{fullRange},A_{full})$ mit dem Kriterium C_{uptime} empfohlen werden, da diese Kombination gegenüber allen anderen der Literatur entnommenen und außerdem auch

gegenüber den Kombinationen mit $T_{\text{fullRange}}$ bzw. T_{dynRange} signifikant oder sogar hochsignifikant besser abschneidet. Wesentliche Grundlage dafür ist sicherlich die Vorteilhaftigkeit der Kombination $K(T_{\text{fullRange}},A_{\text{full}})$, aber auch die Tatsache, dass das Kriterium C_{uptime} allgemein relativ gut geeignet ist.

5. Verwandte Arbeiten

In der Literatur gibt es bereits eine Reihe von Arbeiten, die von ihrer Zielsetzung her Ähnlichkeiten mit der hier untersuchten Vorhersage der Verfügbarkeit von Einzelrechnern haben. Auf diese Arbeiten wird hier ausführlich eingegangen, um die entscheidenden Unterschiede aufzeigen zu können. Nicht näher betrachtet werden allgemeine Arbeiten zum Scheduling beim Grid-Computing. Zwar kann die hier untersuchte Vorhersagbarkeit der Verfügbarkeit von Einzelrechnern (auch) zur Optimierung des Schedulings dienen, das Scheduling selbst ist jedoch nicht Gegenstand der vorliegenden Arbeit. Somit erübrigt sich auch ein Vergleich mit anderen Arbeiten zu diesem Thema, sofern in diesen nicht auch an Vorhersagen zur Verfügbarkeit gearbeitet wurde.

5.1. Abschätzung von Zeiten des Ausgeschaltetseins

In **[Bolo00]** ist untersucht worden, inwieweit der freie Speicherplatz auf den Festplatten innerhalb eines Netzes von Arbeitsplatzrechnern genutzt werden kann, um ein verteiltes Dateisystem aufzubauen. Das Problem besteht dabei vor allem darin, dass die Daten trotz des willkürlichen Ausschaltens solcher Maschinen verfügbar bleiben müssen. Der Ansatz ähnelt somit dem Grid-Computing, wie es in dieser Arbeit betrachtet wird – mit dem Unterschied, dass es nicht um die Ausnutzung freier Rechenzyklen, sondern um die Ausnutzung freien Speicherplatzes geht.

Hier von Interesse ist die Analyse der Zeiten des Ausgeschaltetseins (der *Auszeiten*) der einzelnen Rechner. Für die Gruppe der untersuchten Rechner, die allesamt zur Microsoft Corporation gehörten, wurde eine Verteilung gefunden, wie sie in Abbildung 5-1 durch die schwarze, stark oszillierende Linie dargestellt ist. Die entsprechende kumulierte Verteilungsfunktion ist grau dargestellt. Näherungsweise konnte sie durch eine Kombination aus zwei Normalverteilungen und einer Gamma-Verteilung beschrieben werden. Diese Näherung ist in der Abbildung 5-1 als gestrichelte schwarze Linie dargestellt. Die Prognose, wie lange ein einzelner Rechner noch ausgeschaltet bleibt, wird dann mittels bedingter Wahrscheinlichkeiten ermittelt, indem die für die kumulierte Verteilungsfunktion ermittelte Näherung verwendet wird. Geht man von der Dichtefunktion f dieser Näherung aus, dann ergibt sich bei einer bereits beobachteten Auszeit t_p für die noch verbleibende Auszeit der Erwartungswert t_f wie folgt:[1]

$$t_f = \int_{t_p}^{\infty} \left(\left[\int_{t_p}^{\infty} f(y)dy \right]^{-1} \cdot f(x) \cdot x \cdot dx \right) - t_p = \frac{\int_{t_p}^{\infty} f(x) \cdot x \cdot dx}{\int_{t_p}^{\infty} f(y)dy} - t_p$$

[1] In [Bolo00] ist eine andere, nach Meinung des Autors der vorliegenden Arbeit fehlerhafte Formel angegeben.

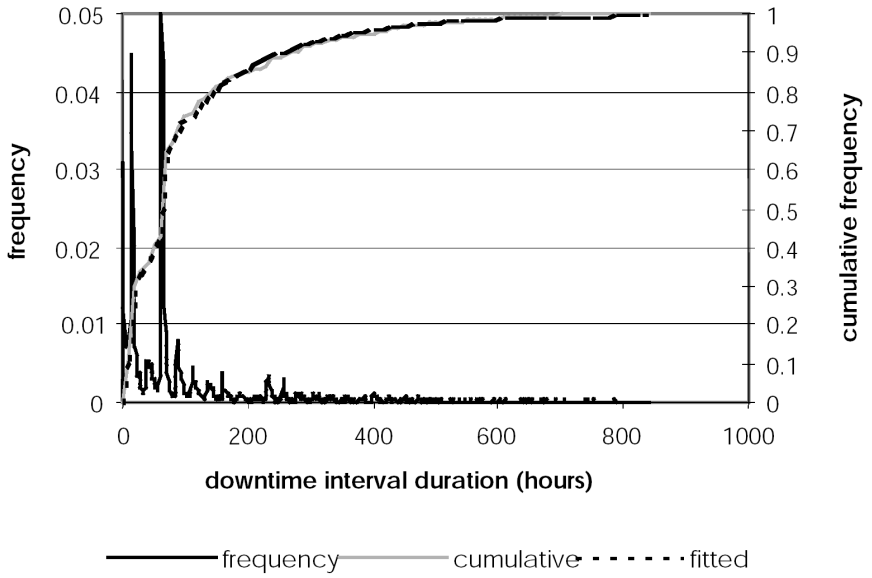

Abbildung 5-1: In [Bolo00] gefundene Verteilung der Auszeiten der untersuchten Rechner (englische Beschriftung, da die Abbildung unverändert aus [Bolo00] entnommen wurde)

Die entsprechende Kurve für unterschiedliche t_p, so wie sie in [Bolo00] ermittelt wurde, ist in Abbildung 5-2 dargestellt. Die aus dieser Kurve ableitbaren Erwartungswerte für die noch verbleibende Auszeit sind für das in [Bolo00] diskutierte System von Interesse, weil auf ausgeschalteten Rechnern liegende Daten erst wieder mit dem Hochfahren verfügbar werden.

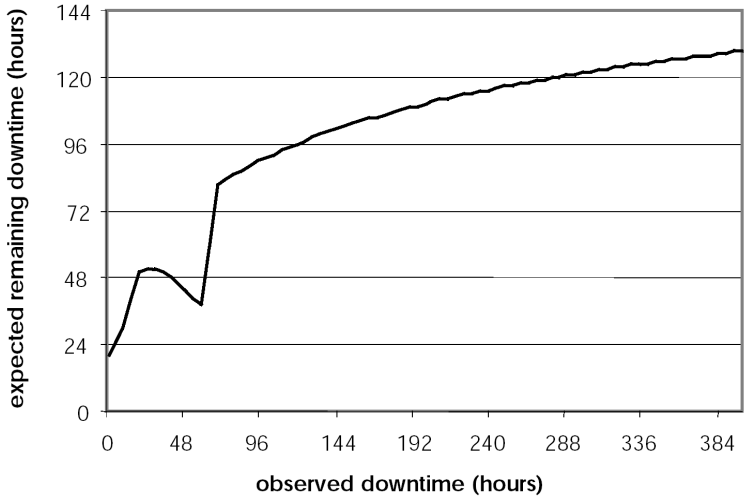

Abbildung 5-2: In [Bolo00] ermittelte Kurve für die Erwartungswerte der noch verbleibenden Auszeit in Abhängigkeit der bisherigen Länge der aktuellen Auszeit (englische Beschriftung, da die Abbildung unverändert aus [Bolo00] entnommen wurde)

Von der Prognosegröße und vom Einsatzzweck abgesehen ist der wesentliche Unterschied zu dem in der vorliegenden Arbeit verfolgten Ansatz der, dass die Vorhersagen in [Bolo00] nicht auf individuellen Daten der einzelnen Rechner basieren. Nur durch die aggregierte Betrachtung der Daten aller Rechner war es möglich, eine bestimmte Form der Verteilung zu ermitteln, die für einzelne Rechner so nicht zu erwarten ist. Für das mit der vorliegenden Arbeit u. a. verfolgte Ziel der Unterstützung der Auswahl geeigneter Rechner für bestimmte Jobs sind jedoch individuelle Vorhersagen zwingend erforderlich. Aus diesem Grunde wurden in der vorliegenden Arbeit andere Ansätze verfolgt (vgl. Diskussion in Abschnitt 4.2.2.1).

5.2. Abschätzung von Zeiten der Verfügbarkeit

Inwieweit sich die freien Rechenkapazitäten normaler Workstations vorhersagen lassen, wird in **[Mutk92]** diskutiert. Das Ziel bei dieser Arbeit bestand darin, ein Scheduling zu ermöglichen, welches vorgegebene Fristen für die Abarbeitung der abgegebenen Jobs einhält. Der grundsätzliche Ansatz ist der, die in der Vergangenheit beobachteten freien Rechenkapazitäten auf die Zukunft zu übertragen. Die Untersuchungen bezogen sich auf Zeithorizonte von unter einer Stunde bis zu mehreren Tagen. Es stellte sich heraus, dass eine Unterscheidung bestimmter Phasen eines Tages sinnvoll ist und ebenso eine getrennte Behandlung der Wochenenden, eine Unterscheidung der einzelnen Wochentage hingegen nicht. Im Unterschied zur vorliegenden Arbeit bezogen sich die Prognosen jedoch auf Gruppen von Rechnern, so dass dieser Ansatz bspw. nicht hilfreich ist, um für einen bestimmten Job einen passenden Rechner auszuwählen. Ferner lieferten die Prognosen lediglich Erwartungswerte für die Zahl der verfügbaren Workstations und keine Wahrscheinlichkeitsverteilungen, wie sie von den in der vorliegenden Arbeit diskutierten Prognoseverfahren erzeugt werden.

Auch Nurmi et al. beschreiben in **[Nurm03]** eine Analyse der Zeiten der Verfügbarkeit von Workstations. Zugrunde lagen der Untersuchung drei große Gruppen von Rechnern. Mit der Analyse wurden verschiedene Ziele verfolgt. Zum einen wurde angedeutet, dass die Ergebnisse letztlich für das Scheduling und auch für die Bestimmung geeigneter Intervalle zwischen Checkpoints hilfreich sein könnten. Das Hauptziel bestand jedoch darin, Simulationen von Grids mit realistischeren Modellen durchführen zu können. Dementsprechend konzentrierte sich die Analyse auf die Bestimmung der Form der statistischen Verteilung der Zeiten der Verfügbarkeit. Es wurde festgestellt, dass Hyperexponential- und Weibull-Verteilungen relativ gut geeignet sind, während Exponential- und Pareto-Verteilungen eher ungeeignet zur Beschreibung der typischerweise auftretenden Zeiten sind. Nicht untersucht wurde hingegen, inwieweit sich Tages- oder Wochenzyklen erkennen lassen. Ferner beziehen sich auch hier die vorgeschlagenen Verteilungsformen nicht auf einzelne Rechner, sondern auf Gruppen von Rechnern. Dies lässt jedoch keine Rückschlüsse auf die Verteilungen auf den einzelnen Rechnern zu, was aber bspw. für die gewünschte Unterstützung bei der Auswahl geeigneter Rechner notwendig wäre. Wie in Abschnitt 4.2.2.1 diskutiert wurde, erscheint eine allgemein anwendbare Verteilungsform unrealistisch; die im Rahmen der

vorliegenden Arbeit durchgeführten Untersuchungen sprechen zumindest gegen die Eignung der in [Nurm03] empfohlenen Hyperexponentialverteilungen.

Aufbauend auf der genannten Veröffentlichung gehen die gleichen Autoren in **[Brev04]** darauf ein, inwieweit sich Angaben zu den Längen der verfügbaren Zeiten einzelner Workstations machen lassen. Konkret wurde auf die 5%-Quantile eingegangen, für die nach unteren Schranken auf einem Konfidenzniveau von 95 % gesucht wurde. Es wurden verschiedene Verfahren geprüft, von denen das so genannte Binomialverfahren die mit Abstand besten Ergebnisse lieferte. Ausgangspunkt dieses Verfahrens ist die Erfassung der n Zeitdauern x_i der Verfügbarkeit innerhalb eines bestimmten Zeitintervalls. Diese Zeitdauern lassen sich als eine Stichprobe auffassen, wobei davon ausgegangen wird, dass die einzelnen Werte x_i voneinander unabhängig sind. Damit lässt sich aus diesen Werten eine Binomialverteilung $B(0{,}05;n)$ mit einzelnen Werten b_i ableiten, für die gilt:

$$b_i = \begin{cases} 1 & \text{falls } x_i < X_{5\%} \\ 0 & \text{sonst} \end{cases}$$

$X_{5\%}$ bezeichnet dabei das (nicht bekannte) 5%-Quantil der realen Verteilung der möglichen Zeitdauern der Verfügbarkeit. Um für $X_{5\%}$ eine untere Schranke auf einem Konfidenzniveau von 95 % anzugeben, wurde die maximale Anzahl k ermittelt, für die gilt:

$$P\big(B(0{,}05;n) \le k\big) \le 5\%$$

Diese Ungleichung ist gleichbedeutend mit:

$$P\big(B(0{,}05;n) > k\big) = P\big(B(0{,}05;n) \ge k+1\big) \ge 95\%$$

Sortiert man die Werte x_1 bis x_n aufsteigend und bezeichnet sie in der dann entstehenden Reihenfolge mit $x_{(1)}$ bis $x_{(n)}$, dann müsste der Wert $x_{(k+1)}$ mit wenigstens 95% Wahrscheinlichkeit noch innerhalb des Quantils $X_{5\%}$ liegen. Somit ist $x_{(k+1)}$ auf dem Konfidenzniveau von 95 % als untere Schranke für $X_{5\%}$ nutzbar, wobei in [Brev04] ohne ersichtlichen Grund mit $x_{(k)}$ gearbeitet wurde.

Zur Bestimmung des maximalen Wertes k kann von der Formel für Binomialverteilungen ausgegangen werden:

$$P\big(B(0{,}05;n) \le k\big) = \sum_{j=0}^{k} \binom{n}{j} \cdot (1-0{,}05)^{n-j} \cdot 0{,}05^{j}$$

Entsprechend der empirischen Ergebnisse aus [Brev04] lag $x_{(k)}$ meist sehr nahe an $X_{5\%}$. Wie jedoch bereits in [Brev04] angesprochen wurde, müsste für eine Nutzbarkeit dieser Schranken für die Vorhersage der noch verbleibenden Zeit der Verfügbarkeit eines Rechners der beschriebene Ansatz modifiziert werden. So wird bei der ermittelten unteren Schranke bisher nicht die bereits vergangene Zeit der aktuellen Verfügbarkeitsperiode berücksichtigt. Dies wäre zwar bei einer Exponentialverteilung irrelevant, gemäß [Nurm03] liegt eine solche jedoch gerade nicht vor. Ferner wurde in [Brev04]

nicht untersucht, inwieweit sich weitere Informationen (wie z. B. die aktuelle Zeit des Tages) zur Verbesserung der Abschätzung einsetzen lassen.

Aus Sicht des Autors handelt es sich bei dem in [Brev04] vorgeschlagenen Vorgehen um einen interessanten Ansatz zur Bestimmung unterer Schranken für ein einzelnes Quantil der Verteilung der Zeiten der Verfügbarkeit. Aus den in Abschnitt 3.1.2 genannten Gründen wurde in der vorliegenden Arbeit jedoch versucht, die vollständige Verteilung abzuschätzen, was mit dem Verfahren aus [Brev04] nicht möglich ist. Die dort erfolgte Abschätzung eines einzelnen Quantils liefert nur einen einzelnen Wert, geht vom Umfang der Information her also nicht über die Abschätzung des Erwartungswertes hinaus. Aus diesem Grunde wurde hier nicht auf dem Ansatz aus [Brev04] aufgebaut.

In einer weiteren Veröffentlichung von Nurmi et al. (**[Nurm04]**) wurde untersucht, inwieweit sich Prognosen für die noch verbleibende Zeit der Verfügbarkeit abgeben lassen, um geeignete Zeitpunkte für Checkpoints von Grid-Jobs zu ermitteln. Es ging dabei insbesondere um Prognosen in einer Umgebung, in der die Ressourcen nicht exklusiv zur Verfügung stehen. Getestet wurden die Verfahren in einer Condor-Umgebung ([Litz88]). Für die Erstellung der Prognosen wurde auf den Daten von einem halben Jahr aufgebaut, aufgrund derer für die einzelnen Rechner Verteilungen abgeleitet wurden. Konkret wurde versucht, an die erhobenen Messwerte Exponential-, Weibull- und Hyperexponentialverteilungen mit zwei bzw. drei Komponenten (Phasen) anzupassen. Von den getesteten Varianten schnitten die Hyperexponentialverteilungen am besten ab. Eine Begründung für die Annahme solcher Standardverteilungen wird jedoch nicht gegeben, alternative Ansätze ohne die Annahme einer bestimmten Form der Verteilung werden weder diskutiert noch in empirischen Vergleichen gegenübergestellt. Wie bereits angemerkt wurde, kann die Eignung solcher Standardverteilungen aber nicht aus den entsprechenden Ergebnissen für Gruppen von Rechnern ([Nurm03]) abgeleitet werden. Die im Rahmen der vorliegenden Arbeit vorgenommene empirische Prüfung bzgl. der Eignung solcher standardisierten Verteilungen für einzelne Rechner spricht klar gegen diesen Ansatz.

Mit dem Network Weather Service (NWS, vgl. **[Wols98]**, **[Wols99]**, **[Wols00]**) wird ebenfalls versucht, die Verfügbarkeit von Rechnern vorauszusagen.[1] Erzeugt werden Prognosen mit einem Zeithorizont von maximal einer halben Minute. Die Werte für die zu erwartende Verfügbarkeit der Rechner werden dabei mit unterschiedlichen Verfahren ermittelt: Zum einen werden der Mittelwert und der Median der Auslastungswerte der letzten Zeit gebildet und als Vorhersage genutzt. Zum anderen wird alternativ versucht, mittels autoregressiver Zeitreihenmodelle (vgl. [Schl99]) die Entwicklung der Werte vorherzusagen. Da keines der genannten Verfahren grundsätzlich bessere Ergebnisse liefert, wird die Auswahl dynamisch getroffen. Zu diesem Zweck werden Prognosen mit allen aufgeführten Verfahren durchgeführt – und jeweils mit den schließlich gemes-

[1] Neben der Verfügbarkeit des Rechners wird zusätzlich auch versucht, die Verfügbarkeit der Netzwerkverbindungen vorherzusagen. Eine solche Vorhersage kann sehr hilfreich sein, ist jedoch nicht Gegenstand der vorliegenden Arbeit.

senen Werten verglichen. Das Verfahren, das zuletzt den kleineren Fehler erzeugt hat, wird für die Prognose des nächsten Wertes eingesetzt.

Auch **[Yang03]** beschäftigt sich mit der Prognose der Verfügbarkeit von CPUs innerhalb der nächsten (maximal 40) Sekunden. In dem vorgestellten Verfahren wird versucht, die Tendenz (Entwicklung) der CPU-Last zu erfassen und für die Prognose des nächsten Wertes zu verwenden. Dabei wurde insbesondere versucht zu vermeiden, dass beim Erreichen von Umkehrpunkten die Extrapolation deutlich über das Ziel hinausschießt. Dies wurde im Wesentlichen dadurch erreicht, dass die vorhergesagte Veränderung nicht nur vom aktuellen Niveau und der gemessenen Tendenz, sondern auch von der Wahrscheinlichkeit abhängt, dass demnächst ein Umkehrpunkt erreicht wird. Die Wahrscheinlichkeit wurde dabei über die erfasste Wahrscheinlichkeitsverteilung vorangegangener Umkehrpunkte abgeschätzt. Laut Aussage der Autoren schnitt das Verfahren in empirischen Vergleichen mit dem bereits erwähnten NWS eindeutig besser ab.

Eine weitere Untersuchung zur Vorhersagbarkeit der CPU-Auslastung im kurzfristigen Bereich (bis zu einer halben Minute) ist in **[Dind00]** beschrieben. Dabei wurden AR-, MA-, ARMA-, ARIMA- und ARFIMA-Modelle (vgl. [Schl99]) bzgl. ihrer Eignung für die Prognose miteinander verglichen. Die genannten Zeitreihenmodelle wurden ferner mit noch einfacheren Modellen verglichen. In den schließlich durchgeführten empirischen Untersuchungen, in denen das in [Dind99] vorgestellte Framework verwendet wurde, erwiesen sich die AR-Modelle (genauer gesagt die AR(16)-Modelle) als am geeignetsten: Einerseits können diese Modelle effizient berechnet werden, andererseits lieferte kein anderes Modell signifikant bessere Prognoseergebnisse.

Alle drei zuletzt genannten Ansätze (NWS, [Yang03], [Dind00]) unterscheiden sich dabei in den gleichen wesentlichen Punkten von der vorliegenden Arbeit. Der erste deutliche Unterschied ist der Zeithorizont. In den angeführten Arbeiten ging es ausschließlich um kurzfristige Vorhersagen (wenige Sekunden, maximal fünf Minuten), während hier längerfristige Vorhersagen (im Bereich von Stunden) diskutiert wurden. Ferner ist anzumerken, dass in den genannten Arbeiten an der Prognose einzelner (Erwartungs-)Werte gearbeitet wurde, während in der vorliegenden Arbeit aus den in Abschnitt 3.1.2 angeführten Gründen Vorhersagen der Wahrscheinlichkeitsverteilungen von Interesse waren. Für die dabei neu hinzukommenden Probleme zeigen die genannten Arbeiten jedoch keine Lösungsmöglichkeiten auf, weshalb nicht auf diesen Arbeiten aufgebaut wurde und stattdessen gänzlich andere Ansätze gewählt wurden.

Auch in **[Zhou05]** wurde untersucht, wie die zu erwartende künftige Verfügbarkeit der einzelnen Rechner für die Auswahl geeigneter Rechner zur Ausführung von Jobs genutzt werden kann. Im Gegensatz zur vorliegenden Arbeit wird jedoch keine echte Vorhersage der Verfügbarkeit für die einzelnen Ressourcen in [Zhou05] versucht. Stattdessen wird für jeden einzelnen Rechner eine Zeitzone angegeben; eine besonders gute Verfügbarkeit wird dann für die Nachtstunden angenommen. Die Überprüfung dieser Annahme erfolgte jedoch nicht anhand realer Daten, sondern ausschließlich unter Nutzung synthetischer Verhaltensprofile, für deren Praxisnähe keine Anhaltspunkte existieren. Auf Grundlage dieser synthetischen Profile erfolgte schließlich ein Vergleich

verschiedener Scheduling- bzw. Migrationsverfahren, die ihrerseits in der vorliegenden Arbeit nicht näher betrachtet werden.

Ebenfalls auf die zu erwartende Verfügbarkeit eingegangen wurde in **[Kond05]**. Dabei wurden für die einzelnen Rechner die Längen der Idle-Zeiten betrachtet, wie sie in der Vergangenheit aufgetreten sind. Zu diesem Zweck wurde für Zeitpunkte im Abstand von 10 Sekunden die jeweils verbleibende Idle-Zeit gemessen und aus den Messwerten eine Verteilung gebildet. Diese Verteilung bildete die Grundlage für die Prognosen, ob Jobs mit Ausführungszeiten von 5, 15 oder 35 Minuten ausgeführt werden können. Die unter Nutzung dieser Prognosen erzielten Verbesserungen beim Scheduling waren jedoch nur gering. Aus diesem Grunde erscheint ein Vergleich mit dem in der vorliegenden Arbeit präsentierten Ansatz angebracht. Zunächst ist festzuhalten, dass durch die Bildung einer einzigen Verteilung pro Rechner die Vorteile möglicher Unterscheidungskriterien nicht genutzt wurden. Faktisch wurde also ausschließlich mit der hier so bezeichneten Variante C_{simple} gearbeitet. Ferner ist auch nicht versucht worden, die Ableitung der Prognoseverteilung aus der gemessenen empirischen Häufigkeitsverteilung zu optimieren. Die in der vorliegenden Arbeit diskutierten Transformationsverfahren sind in [Kond05] also nicht betrachtet worden. Schließlich – und das ist der gravierendste Unterschied – wurde in [Kond05] die empirische Häufigkeitsverteilung auf den Mittelwert (bzw. Erwartungswert) reduziert, also nur dieser für die Optimierung des Schedulings eingesetzt. Wie jedoch in Abschnitt 3.1.2 dargelegt wurde, ist dieser Mittelwert nicht die eigentlich benötigte Information, um einen für die Job-Ausführung geeigneten Rechner auszuwählen.

Der zur vorliegenden Arbeit ähnlichste Ansatz ist in **[Wyck98]** beschrieben. In diesem geht es darum, ungenutzte Zeiten von Workstations für Grid-Jobs zu verwenden, wobei die Dauer dieser Zeiten (Idle-Zeiten) vorausgesagt werden soll. Die Workstations zählen dabei als genutzt, sobald die Last eine (sehr niedrige) Schwelle übersteigt oder aber ein lokaler Nutzer interaktiv arbeitet. Wie auch in der vorliegenden Arbeit werden nicht nur Erwartungswerte vorausgesagt, sondern Wahrscheinlichkeitsverteilungen. Die Grundidee besteht darin, für jeden Rechner eine empirische Häufigkeitsverteilung zu ermitteln, welche die Längen der Idle-Zeiten eines Monats beinhaltet. Für die Vorhersage der noch verbleibenden Idle-Zeit wird dann diese empirische Verteilung verwendet, wobei mit bedingten Wahrscheinlichkeiten für die Ermittlung der noch verbleibenden Zeit gearbeitet wird. Veranschaulicht ist der Ansatz in Abbildung 5-3, in der die empirische Häufigkeitsverteilung wie in [Wyck98] zu einem Histogramm zusammengefasst ist. Zunächst gibt es das vollständige Histogramm (blaue bzw. helle Balken). Ist der Rechner bereits eine Weile im Idle-Zustand (bspw. 6 Zeiteinheiten lang), dann können für die aktuelle Idle-Periode alle kürzeren Zeitspannen ausgeschlossen werden. Somit verteilen sich deren ursprüngliche Wahrscheinlichkeiten auf die anderen möglichen Zeitspannen. In der Abbildung ist dies durch die roten (dunklen) Balken dargestellt.

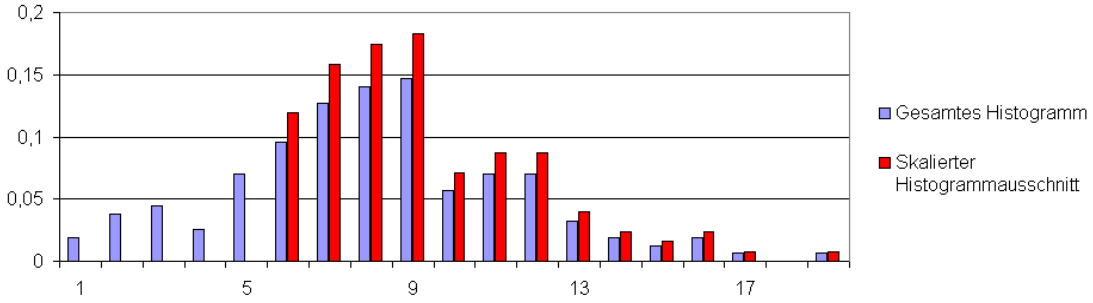

Abbildung 5-3: Ermittlung der Wahrscheinlichkeitsverteilung aus dem Histogramm in [Wyck98]

Für die Prognose der verbleibenden Zeit muss dann noch die bisher verstrichene Idle-Zeit z abgezogen werden. Somit ergibt sich die Wahrscheinlichkeit $P_{\geq k}$, dass die noch verbleibende Idle-Zeit mindestens so groß wie k ist, gemäß der folgenden Formel, wenn X die Verteilung aller vollständigen Idle-Zeiten ist:

$$P_{\geq k} = \frac{P(X \geq k + z)}{P(X \geq z)}$$

Eine wesentliche Erkenntnis der in [Wyck98] beschriebenen Untersuchungen war, dass Verteilungsdaten aus der Vergangenheit der gleichen Station nützliche Hinweise auf die künftigen Idle-Zeiten liefern, unterschiedliche Stationen jedoch deutlich größere Unterschiede bzgl. ihrer Verwendungscharakteristik aufweisen, so dass die Nutzung einer einzelnen (gemeinsamen) Verteilung für alle Stationen wenig sinnvoll erscheint. Weiterhin wurde in [Wyck98] untersucht, ob die Zeit des Tages bei der Erstellung von Prognosen berücksichtigt werden sollte, für unterschiedliche Abschnitte des Tages also unterschiedliche empirische Häufigkeitsverteilungen verwendet werden sollten. Für die meisten der Rechner wurde eine solche Unterscheidung als sinnvoll eingestuft.

Trotz des grundsätzlich ähnlichen Ansatzes ergeben sich im Vergleich zur vorliegenden Arbeit eine Reihe relevanter Unterschiede. Der erste ist die Vorhersagegröße selbst. Während in [Wyck98] die Längen der Idle-Zeiten vorhergesagt wurden, sollen hier die zusammenhängenden freien Rechenkapazitäten innerhalb der nächsten Stunden vorhergesagt werden. Damit können die Ergebnisse aus [Wyck98] nicht einfach auf die hier untersuchten Prognosen übertragen werden – nicht zuletzt wegen der sehr unterschiedlichen Zeithorizonte. Ein noch bedeutenderer Unterschied liegt im Vorgehen zur Bewertung der Prognosen (vgl. Abschnitt 3.2). In [Wyck98] wurde (implizit) angenommen, dass der der Zeitreihe mit den Idle-Zeiten zugrunde liegende stochastische Prozess die Eigenschaft der Stationarität besitzt, die zu unterschiedlichen Zeitpunkten eines Monats gehörenden Wahrscheinlichkeitsverteilungen der Idle-Zeiten also gleich sind. Für die Richtigkeit dieser Annahme gibt es jedoch keinen Anhaltspunkt, eine entsprechende Diskussion ist in [Wyck98] nicht enthalten. Ohne die Voraussetzung der Stationarität ergibt sich jedoch eine deutlich kompliziertere Bewertung der Prognosen, worauf ausführlich in Abschnitt 3.2 der vorliegenden Arbeit eingegangen wurde.

Schließlich wird in [Wyck98] das Problem der Ableitung der Prognose aus den gewonnenen empirischen Verteilungen nicht diskutiert. Wie in dieser Arbeit gezeigt wurde (vgl. Abschnitte 4.2.2 und 4.2.3), hat dieser Punkt allerdings erhebliche Auswirkungen auf die Genauigkeit der Prognosen.

5.3. Abschätzung des Aufwandes von Jobs

Die im vorigen Abschnitt 5.2 vorgestellten Arbeiten liefern Ansätze für die Abschätzung der Verfügbarkeit von Ressourcen. Solche Ansätze sind jedoch weitgehend wertlos, wenn die Komplexität der auszuführenden Jobs nicht abgeschätzt werden kann. Auch wenn sich die vorliegende Arbeit nicht näher mit solchen Abschätzungen beschäftigt, sollen hier der Vollständigkeit und der Abgrenzung wegen entsprechende Ansätze aus der Literatur dennoch kurz vorgestellt werden.

In **[Puru04]** ist ein System beschrieben, das die Abschätzung des Aufwandes für einzelne Jobs dadurch vornimmt, dass zunächst der Job nur auf einem kleinen Teil der abzuarbeitenden Datenmenge arbeitet. Aus der benötigten Zeit wird dann auf den Gesamtaufwand geschlossen, was naturgemäß nur bei solchen Anwendungen gut funktioniert, deren Aufwand proportional zur Datenmenge ist. Die anschließende Abarbeitung der restlichen Daten findet auf Maschinen des Grids statt, die aufgrund ihrer aktuellen Last und der vom Nutzer vorgegeben Zeitgrenzen ausgewählt werden.

Der in **[Desp01]** als FAST (Fast Agent's System Timer) bezeichnete Ansatz geht insofern über den gerade beschriebenen hinaus, als dass die Steigerung des Aufwandes nicht mehr proportional zum Umfang der Daten sein muss. Stattdessen wird mit komplexeren Angaben zur Ausführungsdauer und zum Speicherbedarf gearbeitet. Einerseits ist es dabei möglich, für die verwendeten Routinen explizit entsprechende Angaben zu machen. Andererseits kann auch versucht werden, die Komplexität automatisch zu bestimmen. Dazu werden (auf jedem in Frage kommenden Rechner) Testläufe durchgeführt. Aus den Messergebnissen wird dann mittels polynomialer Regression eine funktionale Beschreibung der Komplexität der einzelnen Routinen abgeleitet. Diese Vorgehensweise setzt jedoch voraus, dass sich der Aufwand der Routinen relativ gut und einfach aus den Eingabeparametern ableiten lässt. Ferner ist eine solche Vermessung auch nur sinnvoll, wenn die Routinen auf den gleichen Rechnern immer wieder ausgeführt werden sollen, da die Ermittlung der Komplexitätsklasse für eine einzelne Ausführung viel zu aufwendig wäre. Die schließlich ermittelten Prognosen für die Ausführungsdauer und den Speicherbedarf werden mit dem bereits erwähnten Network Weather Service (vgl. Abschnitt 5.2) kombiniert, um mit entsprechenden Informationen über die aktuelle Situation im Netz die aus mehreren Teilen bestehenden Jobs optimal auf Maschinen verteilen zu können.

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

Der Ausgangspunkt für die vorliegende Arbeit war die in der Literatur zu findende Idee der organisationsübergreifenden gemeinsamen Nutzung von Rechen-Ressourcen für unterschiedlichste Zwecke, um die Auslastung dieser Ressourcen zu verbessern. Die wesentliche Motivation dieser gemeinsamen Nutzung, die auch als Grid-Computing bezeichnet wird, ist dabei die Einsparung von Kosten. Da bisher keine Untersuchungen der entstehenden Kosten verfügbar sind, wurden diese in Kapitel 2 näher analysiert und quantifiziert. Beim Vergleich unterschiedlicher Formen des Grid-Computings wurde deutlich, dass die Variante der Nutzung von ohnehin vorhandenen, für einen anderen primären Zweck beschafften Ressourcen tatsächlich *bedeutende* Kostenfaktoren minimiert oder sogar ganz eliminiert. In den meisten Fällen handelt sich bei den nicht ausgelasteten Ressourcen um Einzelrechner, die bspw. in Büros stehen, in Computer-Pools oder auch in Privathaushalten.

Als wesentliches Problem der Nutzung solcher Einzelrechner wurde die stark schwankende Verfügbarkeit der freien Rechenkapazität identifiziert. Gerade bei der Ausführung länger laufender Jobs (die an sich für die Ausführung im Grid prädestiniert sind) machen es diese Schwankungen schwierig, eine sinnvolle Auswahl der Rechner zu treffen, auf denen die Jobs laufen sollen. Auch vertragliche Zusicherungen über die Wahrscheinlichkeit der erfolgreichen Job-Ausführung sind zunächst nicht möglich. Zur Linderung dieses Problems ist daher die Vorhersage der in der nächsten Zeit auf den einzelnen Rechnern zu erwartenden freien Rechenkapazitäten ein denkbarer Ansatz. Die Untersuchung und Entwicklung geeigneter Verfahren für solche Prognosen war das Hauptziel der vorliegenden Arbeit.

Vor der eigentlichen Diskussion möglicher Verfahren war zu klären, was eigentlich das genaue Ergebnis einer Prognose sein solle. Dazu musste zunächst eine sinnvolle Größe gefunden werden, die für die Nutzer der Prognosen auch hilfreich ist. Im Rahmen dieser Arbeit wurde mit der Größe $K_{\text{free},i}$ gearbeitet, die die kumulierte Rechenkapazität vom aktuellen Zeitpunkt bis zum nächsten Stopp des Systems bzw. innerhalb von i Stunden liefert. Eine solche Größe ist sinnvoll, um für einen Job mit bestimmtem Rechenbedarf einschätzen zu können, ob er innerhalb einer bestimmten Frist fertiggestellt werden kann. Aus in der Arbeit dargelegten Gründen ist von dieser Größe $K_{\text{free},i}$ aber weniger der Erwartungswert als die vollständige Wahrscheinlichkeitsverteilung von Interesse, weshalb die Prognosen auch solche Verteilungen liefern sollten.

Mit der Entscheidung zur Prognostizierung der Wahrscheinlichkeitsverteilungen ergab sich das Problem der Bewertung der Prognosen. Das Hauptproblem bestand dabei vor allem darin, dass die zu den einzelnen Zeitpunkten vorhergesagten Wahrscheinlichkeitsverteilungen den entsprechenden realen Verteilungen so nahe wie möglich kommen sollten, diese realen Verteilungen aber gerade nicht feststellbar sind. Als Ausweg wurde die Nutzung der Maße M_{exp}, M_{avg} und $\Delta(p)$ vorgeschlagen, die einerseits die Punkt-

genauigkeit und andererseits die Genauigkeit im Durchschnitt der gemachten Prognosen bewerten, wobei ein Prognoseverfahren hinsichtlich beider Kriterien gut abschneiden sollte. Die Entwicklung und Nutzung dieser Maße zur Bewertung der Prognoseverfahren ist ein wesentlicher Unterschied zu bisher in der Literatur zu findenden Arbeiten, die bei der Bewertung von Prognosen für zeitlich veränderliche Größen stillschweigend die Stationarität der den untersuchten Zeitreihen zugrunde liegenden stochastischen Prozesse unterstellten, obwohl diese höchst zweifelhaft ist.

Mit den so entwickelten Maßen wurde schließlich eine gegenüber bisherigen Ansätzen deutlich detailliertere Analyse möglicher Vorgehensweisen der Prognose vorgenommen. Ausgehend vom grundlegenden Ansatz, die zukünftigen freien Rechenkapazitäten aus den in der Vergangenheit beobachteten freien Rechenkapazitäten abzuleiten, wurden die dafür wesentlichen Schritte identifiziert: die Klassifizierung der Werte zur Bildung verschiedener Häufigkeitsverteilungen und die (einer Interpolation ähnelnde) Transformation einer solcher empirischen Häufigkeitsverteilung in die abzugebende Prognose. Dem Autor sind keine Untersuchungen bekannt, die in ähnlichem Umfang mögliche Klassifizierungskriterien vergleichen. Ferner wird in anderen Arbeiten der zweite Schritt (die Transformation) schlicht ignoriert und einfach irgendein, meist nicht näher aufgeführtes Verfahren verwendet. Wie sich jedoch herausgestellt hat, hat gerade auch diese Transformation erheblichen Einfluss auf die Genauigkeit der Prognose.

In den umfangreichen empirischen Untersuchungen, die anhand von auf realen Rechnern erhobenen Log-Dateien durchgeführt wurden, erwies sich letztlich keines der untersuchten Klassifizierungskriterien als den anderen überlegen. Auch die unternommenen Versuche der automatischen Optimierung des Klassifizierungskriteriums führten zu keiner Verbesserung. Das Kriterium C_{uptime}, das die Messwerte nach dem Zeitabstand zum letzten Systemstart klassifiziert und in ähnlicher Form auch schon in anderen Arbeiten verwendet wurde, erwies sich jedoch in Verbindung mit dem als empfehlenswert festgestellten Transformationsverfahren als relativ günstig, insbesondere waren die Ergebnisse auch für unterschiedlichste Rechner immer relativ gut.

Eine eindeutige Empfehlung gab es hingegen beim anzuwendenden Transformationsverfahren, für das die in der vorliegenden Arbeit entwickelte Kombination aus $T_{\text{fullRange}}$ und der adaptiven Korrektur A_{full} anzuwenden ist. In den empirischen Untersuchungen erwies sich diese Kombination gegenüber allen anderen (und insbesondere gegenüber den der Literatur entnommenen) Varianten sowohl hinsichtlich der Punktgenauigkeit der Vorhersagen als auch hinsichtlich der Genauigkeit im Durchschnitt als signifikant besser. In Verbindung mit dem bereits erwähnten, allgemein recht guten Klassifizierungskriterium C_{uptime} konnte dabei auch beim gesamten Prognoseverfahren eine signifikante Verbesserung gegenüber allen anderen, $T_{\text{fullRange}}$ oder A_{full} nicht nutzenden Varianten festgestellt werden. Als überhaupt nicht geeignet hat sich die in manch anderen Arbeiten empfohlene Annahme hyperexponentieller Verteilungen herausgestellt. Hauptgrund für diese abweichende Feststellung dürfte sein, dass die meisten Arbeiten von der Verteilung der über verschiedene Rechner aggregierten Daten ausgehen.

Wesentliche Beiträge dieser Arbeit sind somit die erwähnte Analyse der Kosten beim Grid-Computing, die erstmals den möglichen Kostenvorteil bei der Ausnutzung nicht ausgelasteter Rechner quantifiziert. Ferner wurde mit den eingeführten Maßen eine Möglichkeit entwickelt, um Vorhersagen von Wahrscheinlichkeitsverteilungen bei nicht stationären Zeitreihen beurteilen zu können. Schließlich lieferte diese Arbeit auf der Basis der entwickelten Maße und der durchgeführten empirischen Untersuchungen ein verbessertes Verfahren zur Ableitung von Prognosen für die zu erwartenden freien Rechenkapazitäten.

6.2. Ausblick

Aufbauend auf diesen Ergebnissen gibt es eine Reihe von interessanten Fragestellungen, die in direktem Zusammenhang stehen, in der vorliegenden Arbeit jedoch nicht betrachtet wurden.

Erweiterte Kostenanalyse

Die in Kapitel 2 beschriebene Untersuchung der Kosten des Grid-Computings konzentrierte sich auf die Kosten der Ressourcen-Anbieter. Wenngleich die zusammengetragenen Informationen bereits einen deutlichen Mehrwert gegenüber bisher in der Literatur zu findenden Angaben darstellen, sind für eine fundiertere Beurteilung der Zweckmäßigkeit des Grid-Computings neben den hier betrachteten Kosten der Ressourcen-Anbieter auch die der Ressourcen-Nutzer einzubeziehen. Dabei sind insbesondere die veränderten Aufwände für die Anwendungsentwicklung, die Kosten für die Internet-Anbindung des Nutzers und die Dienstgüte in den Kalkulationen zu berücksichtigen.

Suche nach anderen Indikatoren für die Vorhersage

Das Hauptziel dieser Arbeit war es, realistische Prognosen über die zukünftigen freien Rechenkapazitäten zu erzeugen. Untersucht wurden dabei Ansätze, aus den in der Vergangenheit beobachteten freien Rechenkapazitäten Rückschlüsse auf die zukünftigen freien Rechenkapazitäten zu ziehen. Gewählt wurde dieser Ansatz, weil er am erfolgversprechendsten erschien. Ein Vergleich mit alternativen Ansätzen, die ganz andere Indikatoren als die früheren freien Rechenkapazitäten verwenden, wurde jedoch nicht durchgeführt. Es wäre daher zu untersuchen, welchen alternativen messbaren Daten als Indikator für die zu erwartende freie Rechenkapazität zu gebrauchen sein könnten. Denkbar sind z. B. die aktiven Nutzer oder auch die laufenden Prozesse.

Nutzung der Vorhersagen

Die vorliegende Arbeit konzentrierte sich auf die eigentliche Erzeugung möglichst genauer Prognosen der künftigen freien Rechenkapazitäten. Es wurde zwar analysiert, für welche Anwendungszwecke diese Prognosen hilfreich sein könnten, um daraus auch abzuleiten, welche Informationen solche Prognosen liefern müssen und wie die

Genauigkeit der Prognosen gemessen werden kann. Nicht untersucht wurde jedoch, wie die gemachten Prognosen letztlich gewinnbringend angewendet werden können. Beispielsweise ist es bei der Auswahl eines Rechners für einen bestimmten Job nicht unbedingt die optimale Vorgehensweise, einfach den Rechner mit der höchsten Wahrscheinlichkeit der erfolgreichen Jobausführung zu nehmen, da vielleicht der zweitbeste Rechner für den konkreten Job auch noch fast genauso gut wäre und daher der beste Rechner für andere Jobs frei bleiben könnte. Ähnliche Probleme stellen sich bspw. auch bei der angesprochenen Festlegung der Checkpoint-Intervalle, die sich aber ohnehin nicht direkt aus den Vorhersagen ergeben.

Verfügbarkeitsabschätzung mit einem Startzeitpunkt in der Zukunft

Ein Einsatzzweck für Prognosen der freien Rechenkapazitäten, der in Abschnitt 1.3.1 noch nicht genannt wurde, liegt im möglichen Börsenhandel für Rechenleistung. Dieser mag auf den ersten Blick vielleicht weit hergeholt erscheinen, insbesondere deshalb, weil es zur Zeit noch keine solche Börse gibt. Allerdings könnte bei einer stärkeren Verbreitung des Grid-Computings eine solche Börse sinnvoll werden. Zur Veranschaulichung der Ausführungen soll auf eine ähnliche Börse, und zwar die Strombörse, vergleichend eingegangen werden. Dies ist auch insofern ein nahe liegender Vergleich, weil eine häufig genannte Vision des Grid-Computings die Analogie zum Stromnetz ist (vgl. [Fost98]).

In einem Stromnetz ergeben sich dabei für den Endanwender kaum größere Probleme; zur Nutzung des Stroms ist das entsprechende Gerät meist einfach über die Steckdose anzuschließen. Die zuverlässige Bereitstellung des Stroms entsprechend der zeitlich variierenden Anforderungen ist allerdings ein komplexes Unterfangen. Zum Ausgleich dieser zeitlichen Schwankungen genügt es nicht, einfach die vorhandenen Kraftwerke kontinuierlich laufen zu lassen. Stattdessen müssen Kraftwerke an- und abgeschaltet werden, was je nach Typ mehr oder weniger gut möglich ist. Eine vollständige langfristige Planung von Bedarf und Stromangebot ist jedoch nicht möglich, u. a. auch, weil beide Größen vom Wetter abhängen und Wetterprognosen nur im kurzfristigen Bereich zuverlässig sind. Aus diesem Grunde muss ggf. kurzfristig entschieden werden, welche Kraftwerke am nächsten Tag laufen sollen – und zu welchen Zeiten. Da sowohl das Anfahren als auch das Betreiben von Kraftwerken aber natürlich Geld und Ressourcen kostet, die man besser spart, wenn der Bedarf nicht vorhanden ist, muss es möglich sein, auch kurzfristig mit dem Strom handeln zu können. Zu diesem Zweck gibt es an den Strombörsen den so genannten Spotmarkt, an dem der Strom für den nächsten Tag gehandelt wird (vgl. [EEX05], [Stew05]).[1]

Ein ähnliches Vorgehen wie am Spotmarkt der Strombörsen ist auch für den Handel mit Rechenkapazitäten für den nächsten Tag denkbar. Wiederum besteht der Vorteil des kurzfristigen Handelns darin, dass evt. der Bedarf (der vielleicht sogar angemeldet ist),

[1] Um Missverständnissen vorzubeugen, sei an dieser Stelle darauf hingewiesen, dass an diesem Spot-Markt nicht annähernd die vollständige Menge des für den nächsten Tag benötigten Stroms gehandelt wird. Ein großer Teil des Bedarfs wird über langfristige Lieferbeziehungen geregelt. Dies ließe sich analog auf den Handel mit Rechenkapazitäten übertragen.

aber natürlich auch das Angebot besser abgeschätzt werden kann. Zumindest für letzteres ist anzumerken, dass geeignete Verfahren zur Abschätzung des Angebotes noch fehlen. Zum Betreiben einer solchen „Rechenkapazitäten-Börse“ müssten also zunächst entsprechende Prognoseverfahren entwickelt werden. Der wesentliche Unterschied zu den in dieser Arbeit diskutierten Prognoseverfahren liegt dabei im Startzeitpunkt für die betrachteten freien Rechenkapazitäten. Dieser würde beim Handel an der Börse i. d. R. in der Zukunft liegen, während in dieser Arbeit lediglich Prognosen für den unmittelbar beginnenden Zeitraum gemacht wurden (vgl. Abbildung 3-2).

Anhang A – Effekt von niederprioritären Prozessen

In diesem Anhang sind die Messergebnisse aufgeführt, die den Effekt von Prozessen mit niedriger Priorität beschreiben. Diese Messwerte dienen dazu zu prüfen, inwieweit Grid-Anwendungen im Hintergrund laufen können, ohne den eigentlichen Nutzer der Ressource übermäßig zu beeinträchtigen. Zur Prüfung dieser Effekte wurden Tests auf verschiedenen Systemen durchgeführt. Bei diesen Tests wurde im Vordergrund ein Berechnung gestartet, deren Dauer gemessen wurde. Die gleiche Berechnung wurde dann erneut gestartet, aber mit einem rechenintensiven Hintergrundprozess niedrigerer Priorität. Schließlich wurde die benötigte Zeit verglichen. Um die Messungen nicht zu stark von statistischen Ausreißern abhängig zu machen, wurden die Messungen jeweils 20 mal wiederholt.

A.1. Windows 2000

Zunächst wurden Messungen unter dem Betriebssystem Windows 2000 durchgeführt, im Rechner arbeitete ein Athlon XP 1800+ mit 500 MB RAM. Die entsprechenden Messergebnisse sind in Tabelle A-1 dargestellt:

	unbelastet (Dauer)	Mit Hintergrundprozess	
		Dauer	Abweichung
Minimum	4406 ms	4422 ms	0,4 %
Maximum	4422 ms	4500 ms	1,8 %
Durchschnitt	4416 ms	4478 ms	1,4 %

Tabelle A-1: Effekt von Hintergrundprozessen unter Windows 2000

Offensichtlich haben unter Windows Prozesse mit höherer Priorität absoluten Vorrang, so dass die Belastung durch einen Hintergrundprozess bei unter 2 % liegt.

A.2. Unix (Sun Solaris)

Anschließend wurden Messungen auf einer Sun Workstation (Sun Ultra-5_10) mit einem Prozessor, 128 MB Hauptspeicher und dem Betriebssystem Solaris in der Version 5.9 vorgenommen. Die Ergebnisse sind in der folgenden Tabelle A-2 dargestellt:

	unbelastet (Dauer)	Hintergrundprozess (nice)		Hintergrundprozess (nice 19)	
		Dauer	Abweichung	Dauer	Abweichung
Minimum	19.370 ms	29.478 ms	52,2 %	22.541 ms	16,4 %
Maximum	24.055 ms	34.748 ms	44,5 %	23.668 ms	-1,6 %
Durchschnitt	20.075 ms	31.141 ms	55,1 %	22.934 ms	14,2 %

Tabelle A-2: Effekt von Hintergrundprozessen unter Solaris 5.9

Wie den Messwerten deutlich entnommen werden kann, haben die Vordergrundprozesse keine absolute Priorität.[1] Stattdessen wird die insgesamt verfügbare Prozessorzeit auf die einzelnen Prozesse aufgeteilt, wobei selbst die Prozesse mit niedrigster Priorität nicht verhungern. Dies heißt aber auch, dass eine Grid-Anwendung auch dann noch störend sein kann, wenn sie mit niedrigster Priorität läuft. Der Grund liegt darin, dass die Prozesse auf dem untersuchten System standardmäßig in die Klasse „Time-Sharing“ eingeordnet wurden.

Geeigneter für die Ausführung von Grid-Jobs wäre die Klasse „Fixed-Priority“. Allerdings kann man nicht erwarten, dass auf einem System wegen der Ausführung von Grid-Jobs die Prozesse standardmäßig in diesem Modus laufen müssen, da es natürlich gute Gründe für die Wahl der Klasse „Time-Shared“ gibt. Zum Vergleich wurde deshalb mit dem Tool `priocntl` nur der Hintergrundprozess auf die niedrigste Priorität der Klasse „Fixed-Priority“ gesetzt, während der Vordergrundprozess in der Klasse „Time-Sharing“ belassen wurde. Die Ergebnisse sind in Tabelle A-3 dargestellt:

	unbelastet (Dauer)	Hintergrundprozess	
		Dauer	Abweichung
Minimum	19.376 ms	19.439 ms	0,3 %
Maximum	19.999 ms	21.749 ms	8,8 %
Durchschnitt	19.530 ms	20.691 ms	5,9 %

Tabelle A-3: Effekt von Hintergrundprozessen der Klasse „Fixed Priority“ unter Solaris 5.9

Offensichtlich führte diese Änderung dazu, dass sich der Hintergrundprozess weniger stark bemerkbar machte. Allerdings ist anzumerken, dass auch hier der Hintergrundprozess noch immer nicht aus der Zuteilung der Prozessorzyklen herausgehalten wird. Darüber hinaus ist das Verhalten für diesen Fall (der Mischung von Prozessen der beiden Klassen „Time-Sharing“ und „Fixed-Priority“) zumindest in der Hilfe zu `priocntl` nicht definiert.

A.3. Linux

Des weiteren wurden Messungen unter Linux (SuSE, Kernel-Version 2.6.8) durchgeführt. In dem PC arbeiteten zwei Xeon-Prozessoren von Intel mit 2,66GHz, der RAM hatte eine Größe von 2 GB. Da der Rechner mit zwei Prozessoren bestückt war, wurden hier umfangreichere Tests durchgeführt. Zunächst wurde untersucht, inwieweit ein einzelner Hintergrundprozess Auswirkungen auf das normale Arbeiten hat, die entsprechenden Ergebnisse sind in Tabelle A-4 dargestellt. Wie kaum anders zu erwarten, hat ein rechenintensiver Hintergrundprozess auf einen anderen rechenintensiven Prozess keine messbaren Auswirkungen, da beide Prozesse unterschiedlichen Prozessoren zugewiesen werden können.

[1] Der Maximalwert im unbelasteten Zustand scheint dabei ein Ausreißer zu sein.

	unbelastet (Dauer)	Mit Hintergrundprozess	
		Dauer	Abweichung
Minimum	4645 ms	4632 ms	-0,3 %
Maximum	4655 ms	4690 ms	0,8 %
Durchschnitt	4647 ms	4639 ms	-0,2 %

Tabelle A-4: Effekt eines Hintergrundprozesses auf einer 2-Prozessor-Maschine unter Linux

Folglich wurde außerdem geprüft, wie sich zwei Hintergrundprozesse auf die Performance auswirken (Tabelle A-5). In diesem Fall ist festzuhalten, dass wiederum eine spürbare Beeinflussung der Vordergrundprozesse nicht ausgeschlossen ist, wenngleich die Beeinflussung geringer ausfällt als auf dem getesteten Unix-System. Das auf dem Unix-System genutzte Tool `priocntl` war zumindest auf dem verwendeten Linux-System nicht verfügbar.

	unbelastet (Dauer)	2 Hintergrundprozesse (nice)		2 Hintergrundprozesse (nice 19)	
		Dauer	Abweichung	Dauer	Abweichung
Minimum	4645 ms	8114 ms	74,7 %	4854 ms	4,5 %
Maximum	4655 ms	8593 ms	84,6 %	4891 ms	5,1 %
Durchschnitt	4647 ms	8215 ms	76,8 %	4862 ms	4,6 %

Tabelle A-5: Effekt von zwei Hintergrundprozessen auf einer 2-Prozessor-Maschine unter Linux

Anhang B – Implementierungsdetails

In diesem Anhangskapitel werden einige Details der für die Messungen genutzten Implementierungen näher erläutert.

B.1. Zeitraum für die dynamische Bestimmung des Wertebereichs

Bei der Beschreibung der Transformationsvariante T_{dynRange} in Abschnitt 4.2.2.3.3 war darauf hingewiesen worden, dass sich die dynamische Bestimmung des Wertebereiches grundsätzlich zwar an einem Zeitraum von 10 Wochen orientierte, dieser Zeitraum aus Effizienzgründen jedoch nicht exakt eingehalten wurde. Das genaue Verfahren zur Bestimmung der oberen Grenze ist in Abbildung B-1 als Pseudocode dargestellt, in dem `value` für den neuesten Messwert steht und `maxValue` für die dynamisch bestimmte obere Grenze des Wertebereichs. Demnach wird die obere Grenze durch einen variablen Zeitraum von 10 bis 20 Wochen bestimmt.

```
if (value≥maxValue) {
  maxValue=value;
  newMaxValue=-1;
}
else {
  if (maxValue ist älter als 10 Wochen) {
    if (value>=newMaxValue) newMaxValue=value;
    if (maxValue ist älter als 20 Wochen) {
      maxValue=newMaxValue;
      newMaxValue=-1;
    }
  }
}
```

Abbildung B-1: Pseudocode für dynamische Bestimmung der oberen Grenze des Wertebereichs beim Transformationsverfahren T_{dynRange}

B.2. Generierung von Auslastungswerten für die Daten aus [Long95]

Da die Daten aus [Long95] nur die Zeitspannen des Angeschaltetseins der einzelnen Rechner, nicht aber die eigentlichen Auslastungswerte liefern, mussten letztere für die Zeiten des Angeschaltetseins generiert werden. Das in den Messungen angewendete

Zufallsverfahren zur Bestimmung der Auslastungswerte im Abstand von 10 Sekunden ist in Abbildung B-2 dargestellt. Die genauen Parameter im angegebenen Algorithmus sind in gewissen Grenzen willkürlich, wurden jedoch so gewählt, damit die stündlichen Mittelwerte der Auslastung sich häufig im Bereich zwischen 5 und 20 Prozent bewegen. Ferner wird mit diesen Parametern und der im Algorithmus realisierten Verkettung aufeinander folgender Zufallswerte auch eine relevante Schwankung der Größe K_{free} erreicht. Aus den in Abschnitt 4.3.4 genannten Gründen werden durch den Algorithmus keine regelmäßigen Schwankungen erzeugt.

```
if (Zufallsereignis mit 0,04% Wahrscheinlichkeit) {
  auslastung=(zufälliger Wert zwischen 0 und 20%)
  while (auslastung<5%) & (Zufallsereignis mit 50% Wahrscheinlichkeit) {
    auslastung=(zufälliger Wert zwischen 0 und 20%)
  }
}
else {
  factor=1.0+(Zufallszahl zwischen 0 und 0,02);
  if (Zufallsereignis mit 50% Wahrscheinlichkeit) factor=1/factor;
  auslastung=(alter Auslastungswert)*factor
  if (auslastung≥100%) auslastung=(alter Auslastungswert);
}
```

Abbildung B-2: Pseudocode für zufällige Generierung von Auslastungswerten

Anhang C – Verworfene Ansätze

C.1. Clusterbildung zur variablen Unterscheidung der Häufigkeitsverteilungen

In Abschnitt 4.2.4.1 wurde die Grundidee für eine variable Unterscheidung von Häufigkeitsverteilungen unter Nutzung von Cluster-Verfahren kurz skizziert. Die genaue Vorgehensweise bei den Experimenten ist im Folgenden dargestellt.

C.1.1. Grundsätzliches Vorgehen zur Bestimmung der Cluster

Für die Gruppierung von Objekten in verschiedene Cluster gibt es eine Reihe empfehlenswerter Vorgehensweisen. Im Rahmen dieser Arbeit wurde auf die hierarchisch-agglomerativen Verfahren zurückgegriffen, wie sie in [Ecke80] und [Kauf84] beschrieben sind. In diesen Verfahren wird zunächst von dem Zustand ausgegangen, dass jedes Objekt einen eigenen Cluster bildet, wobei unter diesen Objekten hier die empirischen Häufigkeitsverteilungen zu verstehen sind. Schritt für Schritt werden dann jeweils die beiden Cluster vereinigt, welche die geringste Distanz zueinander haben. Beendet wird das Verfahren, wenn die gewünschte Zahl von Clustern erreicht ist oder aber kein Paar von Clustern mehr gefunden werden kann, die zueinander eine Distanz unterhalb einer bestimmten Grenze d_{max} aufweisen. Dargestellt ist eine solche Clusterbildung in der folgenden Abbildung C-1 für acht empirischen Häufigkeitsverteilungen V_1 bis V_8, aus denen schließlich die drei Cluster C_1 bis C_3 gebildet werden.

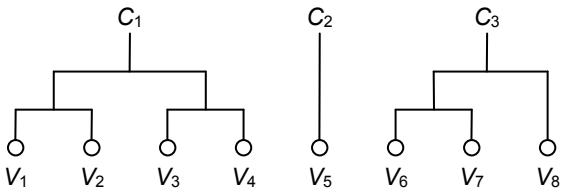

Abbildung C-1: Beispiel einer hierarchisch-agglomerativen Clusterbildung für acht empirische Verteilungen

Da bei dem skizzierten Vorgehen Cluster nur vereinigt, nicht aber aufgeteilt werden können, ist von solchen Häufigkeitsverteilungen auszugehen, die bereits eine maximale Spezialisierung aufweisen. Für die bessere Vorstellung kann dabei zunächst davon ausgegangen werden, dass es sich um die 50 Häufigkeitsverteilungen entsprechend der Klassifikation C_{uptime} handelt; auf alternative Ausgangslagen wird weiter unten noch kurz eingegangen.

C.1.2. Beurteilung der Distanz zwischen Häufigkeitsverteilungen

Um bei dem beschriebenen Verfahren sukzessive die zueinander nächsten Cluster vereinigen zu können, ist es erforderlich, Merkmale zur Beurteilung der Distanz zwischen verschiedenen Häufigkeitsverteilungen zu haben. Dafür erscheint es sinnvoll auf solche Merkmale zurückzugreifen, die Verteilungen recht gut charakterisieren. Die diesbezüglich wichtigsten und am häufigsten eingesetzten Merkmale sind der Mittelwert und die Varianz bzw. die Streuung. Alternative Kriterien sind bspw. der Median und bestimmte Perzentile. Im Rahmen der vorliegenden Arbeit wurde vor allem mit den folgenden beiden Kombinationen experimentiert:

- Mittelwert + Streuung
- Median + Streuung

Für die erste Variante spricht, dass der Mittelwert für die Ermittelung der Streuung ohnehin notwendig ist. Für die zweite Variante hingegen spricht, dass der Median tendenziell robuster gegenüber Ausreißern ist. Diese Robustheit erscheint insofern vorteilhaft, da die Grundlage für die Einteilung in Cluster nicht die tatsächlichen Verteilungen sein können, sondern nur empirische Häufigkeitsverteilungen, die von den realen Verteilungen mehr oder weniger stark abweichen.

Neben der Auswahl der zu nutzenden Merkmale stellt sich die Frage, wie aus diesen die Ähnlichkeit bzw. Distanz zwischen zwei Häufigkeitsverteilungen abgeleitet wird. Die wohl am häufigsten eingesetzte Methode zur Bestimmung der Distanz ist das Euklidsche Distanzmaß d_{Euklid}. Ist die Differenz zwischen den jeweiligen n Merkmalen zweier Verteilungen gerade $\Delta(m_i)$, dann berechnet sich d_{Euklid} wie folgt:

$$d_{\text{Euklid}} = \sqrt{\sum_{i=1}^{n} (f_i \cdot \Delta(m_i))^2}$$

Die in der Formel vorkommenden Faktoren f_i stellen Gewichtungen für die unterschiedlichen Merkmale dar. Aufgrund des geringen Erfolgs bei der Nutzung des Euklidschen Distanzmaßes wurde ferner mit anderen Varianten der Zusammenfassung der Merkmale experimentiert. Insbesondere wurde mit der Variante d_{add} gearbeitet, die wie folgt definiert ist:

$$d_{\text{add}} = \sum_{i=1}^{n} f_i \cdot \left|\Delta(m_i)\right|$$

Ein Vorteil von d_{add} gegenüber d_{Euklid} besteht darin, bei den f_i auch mit negativen Werten experimentieren zu können. Solche negativen Werte können bspw. sinnvoll sein, falls Verteilungen mit ähnlichem Erwartungswert gerade dann vereinigt werden sollten, wenn sich die Streuungen besonders stark unterscheiden, da dies evt. darauf hindeutet, dass im einen Fall die Streuung unterschätzt und im anderen Fall überschätzt wurde. Anzumerken ist, dass das Maß d_{add} mit negativen f_i-Werten nicht mehr die

Bedingungen eines Distanzmaßes erfüllt, da d_{add} dann auch negative Werte annehmen kann. Für die hier diskutierte Anwendung zur Bestimmung zu vereinigender Cluster ist jedoch kein Grund ersichtlich, aus dem solche negativen Werte problematisch wären.

Schließlich wurde speziell für den Vergleich von Häufigkeitsverteilungen das Maß d_{overlap} entwickelt, das wie folgt definiert ist:

$$d_{\text{overlap}} = f_1 \cdot |\mu_1 - \mu_2| + f_2 \cdot (s_1 + s_2)$$

Die in dieser Formel verwendeten Variablen s_1 und s_2 stehen für die Streuungen der beiden Verteilungen, während μ_1 und μ_2 für die Mittelwerte stehen, alternativ aber auch für die Mediane stehen könnten. Auch dieses Maß ist i. Allg. kein Distanzmaß. Die Intention bei diesem Maß liegt darin, dass zwei eigentlich sehr ähnliche Wahrscheinlichkeitsverteilungen zu Häufigkeitsverteilungen führen können, deren Mittelwerte sich stark voneinander unterscheiden, wenn die Streuungen besonders groß sind. Das Maß zielt also darauf ab einzuschätzen, ob die den empirischen Verteilungen zugrunde liegenden realen Wahrscheinlichkeitsverteilungen zueinander ähnlich sein könnten.

Weiterhin wurde versucht, bei der Auswahl der zu vereinigenden Cluster die Zahl der Werte in den jeweiligen empirischen Verteilungen zu berücksichtigen. Tendenziell sollten bevorzugt Cluster mit wenigen Werten vereinigt werden, da in diesen Clustern einzelne Ausreißer besonders große Fehler produzieren. In den Experimenten konnte jedoch keine Variante gefunden werden, in der die Berücksichtigung der Zahl der Werte pro Cluster von Vorteil war.

C.1.3. Beurteilung der Distanz zwischen Clustern

Zur Bestimmung der Distanzen zwischen den einzelnen Clustern sind neben den gerade diskutierten Merkmalen und Maßen weitere Festlegungen notwendig, da ein Cluster mehrere Häufigkeitsverteilungen enthalten kann und die Diskussion in Abschnitt C.1.2 sich auf die Ähnlichkeit zweier einzelner Häufigkeitsverteilungen bezog.

Die hier naheliegendste Lösung ist durch die Bedeutung der Cluster bestimmt. So sind diese dafür da, empirische Häufigkeitsverteilungen mit mehr Werten zu erzeugen. Dementsprechend können alle Einzelwerte eines Clusters der gleichen Häufigkeitsverteilung zugeordnet werden, für die sich folglich auch Mittelwert, Median und Streuung berechnen lassen. Auf diese Weise ließe sich die Distanz zwischen beliebigen Clustern ermitteln. Gegen diese Vorgehensweise spricht jedoch die relativ aufwendige Berechnung. Jedes mal, wenn zwei Cluster vereinigt werden, müssen zunächst der Mittelwert bzw. Median und die Streuung für diesen Cluster berechnet werden. Im Anschluss daran muss eine Neukalkulation der Distanzen zu diesem neuen Cluster stattfinden.

Effizienter sind Varianten, bei denen die Aktualisierung der Distanz-Matrix rekursiv nach folgender Formel erfolgen kann ([Kauf84]):

$$d(C_{\text{new}}, C_k) = \alpha_i \cdot d(C_i, C_k) + \alpha_j \cdot d(C_j, C_k) + \beta \cdot d(C_i, C_j) + \gamma \cdot |d(C_i, C_k) - d(C_j, C_k)|$$

In dieser Formel ist d das Maß zur Bestimmung der Distanz zwischen zwei Clustern. Die C_x sind die Cluster, wobei C_{new} der Vereinigung von C_i und C_j entspricht und die Indizes i, j und k paarweise verschieden sein sollen (vgl. Abbildung C-2).

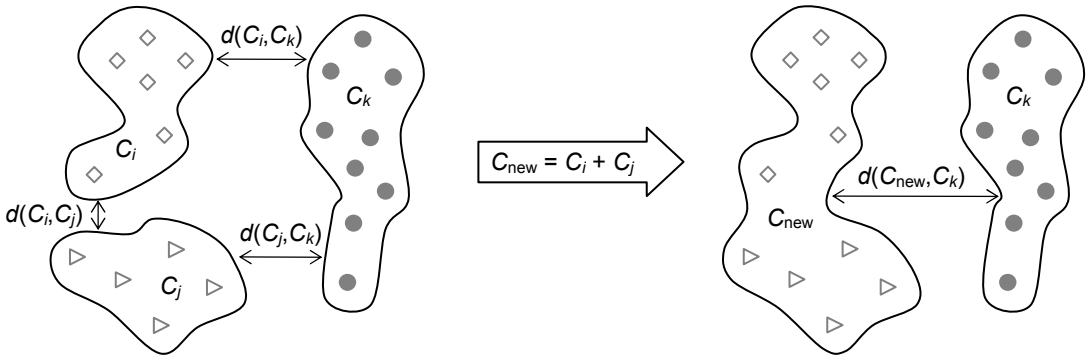

Abbildung C-2: Veranschaulichung der Cluster C_i, C_j, C_k und C_{new}

Durch die Parameter α_i, α_j, β und γ lassen sich unterschiedliche, anschaulich vorstellbare Verfahren wählen, u. a. die in Tabelle C-1 aufgeführten. Die beim Average-Linkage-Verfahren aufgeführten Variablen n_i und n_j stehen für die Anzahl der Objekte (hier: Anzahl der Häufigkeitsverteilungen) in den Clustern C_i und C_j.

Verfahren	α_i	α_j	β	γ
Single Linkage	½	½	0	-½
Complete Linkage	½	½	0	½
Average Linkage	$\frac{n_i}{n_i + n_j}$	$\frac{n_i}{n_i + n_j}$	0	0

Tabelle C-1: Verfahren zur rekursiven Berechnung der Cluster-Abstände

Die ersten beiden Verfahren in Tabelle C-1 definieren den Abstand zwischen zwei Clustern als den Abstand der einander nächsten bzw. voneinander am weitesten entfernten Objekte. Eine anschauliche Gegenüberstellung dieser beiden Varianten befindet sich in Abbildung C-3. Das dritte Verfahren (Average Linkage) versucht, für den Abstand einen Mittelwert zwischen den beiden genannten Extremen anzusetzen.

Wird das Single-Linkage-Verfahren als Auswahlkriterium für die zu vereinigenden Cluster verwendet, so ergeben sich tendenziell klar voneinander unterscheidbare Cluster, die jedoch keine große innere Homogenität aufweisen müssen. Bei Verwendung des Complete-Linkage-Verfahrens hingegen ergeben sich tendenziell recht homogene Cluster, die dafür aber u. U. nur geringe Abstände untereinander haben. Für das hier verfolgte Ziel erscheint damit das Single-Linkage-Verfahren als eher ungeeignet, da es für die gewünschten Vorhersagen kaum nachteilig ist, wenn es zwei Cluster mit ähnlichen Verteilungen gibt. Gravierendere Auswirkungen sind zu erwarten, wenn die Verteilungen innerhalb eines Clusters recht unterschiedlich sind. In den empirischen

Untersuchungen wurde dennoch mit allen drei in Tabelle C-1 aufgeführten Varianten experimentiert, wobei die erfolgreichste gefundene Variante das Complete-Linkage-Verfahren verwendete.

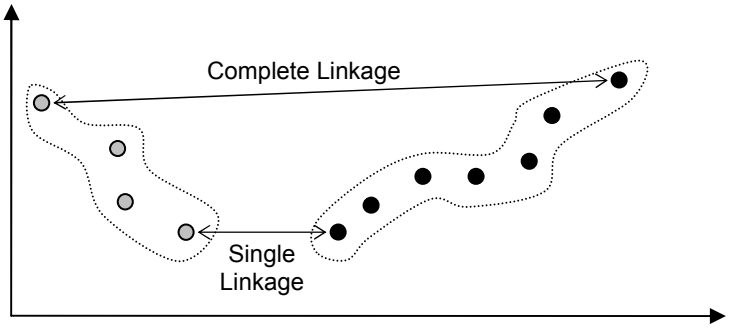

Abbildung C-3: Single Linkage vs. Complete Linkage (Abbildung angelehnt an [Ecke80])

C.1.4. Modifikation des Verfahrens

In Abwandlung zum gerade beschriebenen, allgemeinen hierarchisch-agglomerativen Verfahren wurde mit einer Einschränkung der für Vereinigungen in Frage kommenden Cluster experimentiert. So stellt sich bei der hier betrachteten Klassifikation C_{uptime} vor allem die Frage, wie klein bzw. groß die aneinander grenzenden Intervalle der Laufzeiten sein sollen, für die jeweils eine Häufigkeitsverteilung gebildet wird. Zwei verschiedene Varianten sind in Abbildung C-4 angedeutet: Links ist eine Aufteilung in fünf Intervalle dargestellt, während rechts eine Aufteilung in drei größere Intervalle zu sehen ist.

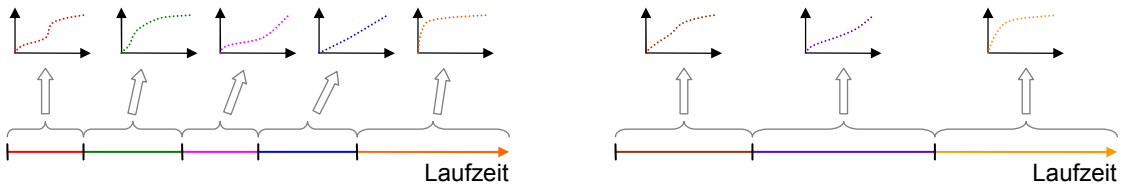

Abbildung C-4: Mögliche Aufteilungen des Bereichs möglicher Laufzeiten in zusammenhängende Intervalle

Aus dem bisher beschriebenen Vorgehen zur Bildung der Cluster könnten jedoch auch Aufteilungen resultieren, in denen eine Häufigkeitsverteilung mehrere, nicht zusammenhängende Intervalle repräsentiert, so wie es in Abbildung C-5 dargestellt ist. In solchen Fällen besteht aber die erhöhte Gefahr, dass die Clusterbildung nur aufgrund einer Ähnlichkeit der empirischen Häufigkeitsverteilungen durchgeführt wird, ohne dass auch die realen Wahrscheinlichkeitsverteilungen zueinander ähnlich sind.

Vermeiden lassen sich solche Aufteilungen wie die in Abbildung C-5 dargestellte dadurch, dass nur Cluster vereinigt werden, die benachbarte Intervalle repräsentieren.

Diese Vorgehensweise stellte sich in den Tests schließlich auch als vorteilhaft heraus. Formal definieren lässt sich die geforderte Nachbarschaftsbeziehung für die hier diskutierte Klassifikation C_{uptime} wie folgt:

$$C_i \text{ ist Nachbar von } C_j \Leftrightarrow \exists k : \left(V_k \in C_i \wedge V_{k+1} \in C_j\right) \vee \left(V_k \in C_j \wedge V_{k+1} \in C_i\right)$$

Die Variablen V_k und V_{k+1} stehen für die empirischen Häufigkeitsverteilungen entsprechend der Klassifikation C_{uptime}, so wie sie auch in Abschnitt 4.2.1.2.1 definiert wurden.

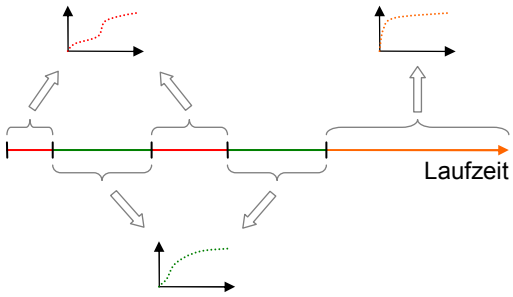

Abbildung C-5: Bildung von Häufigkeitsverteilungen aus mehreren, nicht zusammenhängenden Intervallen

Zur Verdeutlichung des Verfahrens sei abschließend noch einmal auf die bereits in Abbildung C-1 dargestellte Zuordnung von acht Verteilungen zu drei Clustern eingegangen. Ein möglicher Entstehungsweg ist in Abbildung C-6 dargestellt. Es könnte bspw. so gewesen sein, dass unter den ursprünglichen acht Verteilungen V_6 und V_8 zueinander die geringste Distanz aufgewiesen haben. Da diese beiden jedoch nicht als benachbart gelten, konnten sie nicht vereinigt werden; stattdessen wurde V_3 mit V_4 vereinigt, da diese die nächstgrößere Distanz hatten. Auch in den weiteren Schritten wurden nur gemäß obiger Definition benachbarte Cluster vereinigt, wobei im Schritt 3 letztlich auch die Verteilungen V_6 und V_8 dem gleichen Cluster zugeordnet werden konnten, da der bereits aus V_6 und V_7 entstandene Cluster mit V_8 benachbart ist.

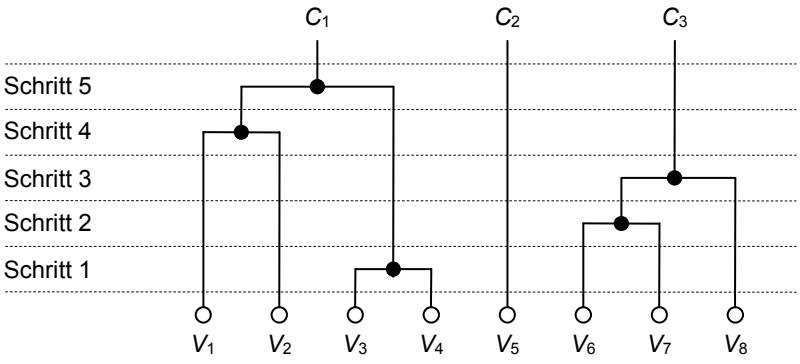

Abbildung C-6: Schrittweise Darstellung einer hierarchisch-agglomerativen Clusterbildung für acht empirische Verteilungen

C.1.5. Verwendetes Abbruchkriterium

Ein wesentliches Problem bei allen Verfahren zur Clusteranalyse ist die Bestimmung der Clusterzahl. Für das hier verfolgte Ziel ist dabei die Vorgabe einer festen Zahl weniger sinnvoll, da gerade erreicht werden soll, dass bei manchen Rechnern mehr und bei anderen weniger Cluster existieren. Aus diesem Grunde wurde bei den Verfahren mit Grenzwerten für die verwendeten (Distanz-)Maße gearbeitet. Gab es bei den betrachteten Cluster-Paaren keines mehr, für welches das Maß einen Wert unterhalb dieses Grenzwertes ergab, dann wurde die Vereinigung von Clustern beendet. In den Tests wurde mit verschiedenen Grenzwerten experimentiert.

C.1.6. Empirische Ergebnisse

In den empirischen Tests wurde mit zahlreichen Kombinationen der angesprochenen Parameter experimentiert. Dabei wurde auch mit anderen anfänglichen Klassifizierungen als C_{uptime} gearbeitet. Insbesondere wurde versucht, aufbauend auf der Einteilung C_{phase} Cluster zu bilden, was allerdings nicht zu besseren Ergebnisse führte. Weil außerdem die Definition der Nachbarschaftsbeziehung bei C_{phase} wesentlich schwieriger bzw. weniger eindeutig ist, sind im Folgenden nur die Ergebnisse ausgehend von der Klassifizierung C_{uptime} dargestellt.

Als vergleichsweise günstig hat sich dabei die Variante mit d_{add} unter Nutzung von Mittelwert und Streuung und den beiden Faktoren 1 und -1 herausgestellt. Für die rekursive Berechnung der Distanzen zwischen Clustern wurde das Complete-Linkage-Verfahren angewendet, als Transformationsverfahren kam $T_{\text{fullRange}}$ in Verbindung mit der adaptiven Korrektur A_{full} zum Einsatz. Als Grenzwert für die Vereinigung von Clustern wurde 0 verwendet. Um den Aufwand für die Cluster-Bildung nicht ausufern zu lassen, wurden die Cluster im Wochenabstand neu berechnet. Die Ergebnisse für die vier Testrechner sind in Abbildung C-7 und Abbildung C-8 unter dem Namen $C_{\text{uptime,clustered}}$ den bisher betrachteten Klassifizierungskriterien gegenübergestellt worden. Die genauen Zahlenwerte sind außerdem in Tabelle D-9 dokumentiert.

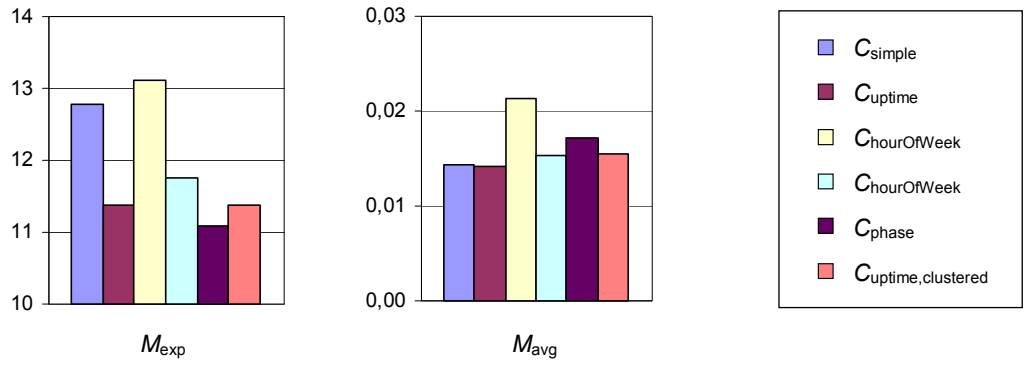

Abbildung C-7: Ergebnisse des getesteten Cluster-Verfahrens bzgl. M_{exp} und M_{avg} unter Verwendung der Transformationsvariante $T_{\text{fullRange}}$ und der adaptiven Korrektur A_{full}

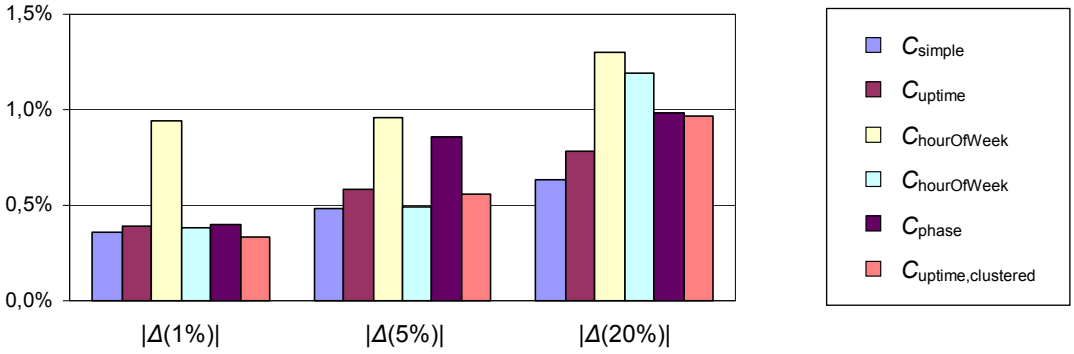

Abbildung C-8: Ergebnisse des getesteten Cluster-Verfahrens bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$ unter Verwendung der Transformationsvariante $T_{\text{fullRange}}$ und der adaptiven Korrektur A_{full}

Offenbar ergibt sich für die vier Testrechner keine echte Verbesserung gegenüber den statischen Klassifizierungskriterien C_{uptime} und C_{phase}. Da bereits mit den besten gefundenen Parametern für die Cluster-Methode gearbeitet wurde, spricht dies gegen die Wirksamkeit solcher Cluster-Verfahren. Das gilt um so mehr, als dass die beschriebenen Cluster-Verfahren relativ komplex sind und daher nur bei relativ deutlichen Vorteilen eingesetzt werden sollten. Auch die in Abschnitt 4.3 dokumentierten Messungen bestätigen die geringe Eignung der beschriebenen Cluster-Verfahren; eine Diskussion der Ursachen befindet sich in Abschnitt 4.2.4.1.

C.2. Mischverfahren

Entsprechend der in Abschnitt 4.2.4.2 beschriebenen Idee für Verfahren mit wechselndem Klassifikationskriterium wurden unterschiedliche Varianten untersucht. Von den getesteten Kombinationen verschiedener Klassifizierungskriterien lieferten die beiden Varianten $C_{\text{uptime,simple,2}}$ und $C_{\text{phase,simple,2}}$ die besten Ergebnisse und werden daher hier näher betrachtet. Die Variante $C_{\text{uptime,simple,2}}$ kombiniert die beiden Unterscheidungskriterien C_{uptime} und C_{simple}. Für die Entscheidung, ob die Prognose mit der Klassifizierung C_{uptime} oder die mit der Klassifizierung C_{simple} zu wählen ist, wird die Bewertung anhand des Maßes M_{exp} für den Zeitraum der letzten zwei Wochen genutzt. Die Variante $C_{\text{phase,simple,2}}$ funktioniert sehr ähnlich zu $C_{\text{uptime,simple,2}}$. Der einzige Unterschied ist der, dass statt des Unterscheidungskriteriums C_{uptime} das Kriterium C_{phase} als Alternative zu C_{simple} in Betracht gezogen wird.

In den Tests wurde auch mit zahlreichen Alternativen zu den beiden genannten Varianten experimentiert. Variiert wurde bspw. der Zeitraum, über den alternative Prognoseverfahren bewertet wurden; der hier genutzte Zeitraum von zwei Wochen stellte sich jedoch als vergleichsweise günstig heraus. Ferner wurde mit Varianten experimentiert, die Bewertung der zur Auswahl stehenden Klassifizierungsvarianten differenzierter

vorzunehmen: Statt wie bei den beiden genannten Varianten die Prognosegüte zu sämtlichen Zeitpunkten der letzten zwei Wochen zu betrachten, ist es ebenso denkbar, nur Messwerte von solchen Zeitpunkten zu berücksichtigen, die einem bestimmten Klassifizierungskriterium genügen. Beispielsweise wurde versucht, getrennte Bewertungen für die unterschiedlichen Klassen des Kriteriums C_{uptime} vorzunehmen. Hintergedanke dabei war, dass es Laufzeiten der Rechner geben könnte, für die eine Prognose gemäß C_{simple} sinnvoller sein könnte, während bei anderen Laufzeiten bspw. C_{phase} oder C_{uptime} günstiger sein könnten. Trotz zahlreicher Experimente mit unterschiedlichen Kriterien zur Differenzierung der Bewertung der möglichen Prognoseverfahren konnte jedoch gegenüber den beiden oben vorgestellten Verfahren keine Verbesserung erzielt werden. Für diese beiden Varianten $C_{\text{uptime,simple,2}}$ und $C_{\text{phase,simple,2}}$ sind die mit den vier Testrechnern erzielten Ergebnisse in Abbildung C-9 und Abbildung C-10 den Ergebnissen der weiter oben vorgestellten Klassifizierungskriterien gegenübergestellt. Als Transformationsvariante kam $T_{\text{fullRange}}$ in Verbindung mit der adaptiven Korrektur A_{full} zum Einsatz. Die genauen Zahlenwerte der Messungen sind in Tabelle D-10 dokumentiert.

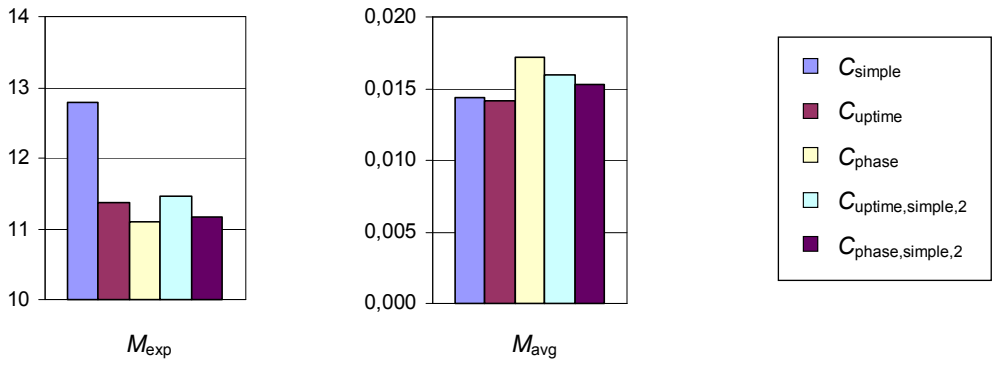

Abbildung C-9: Ergebnisse der Klassifizierungskriterien bzgl. M_{exp} und M_{avg}

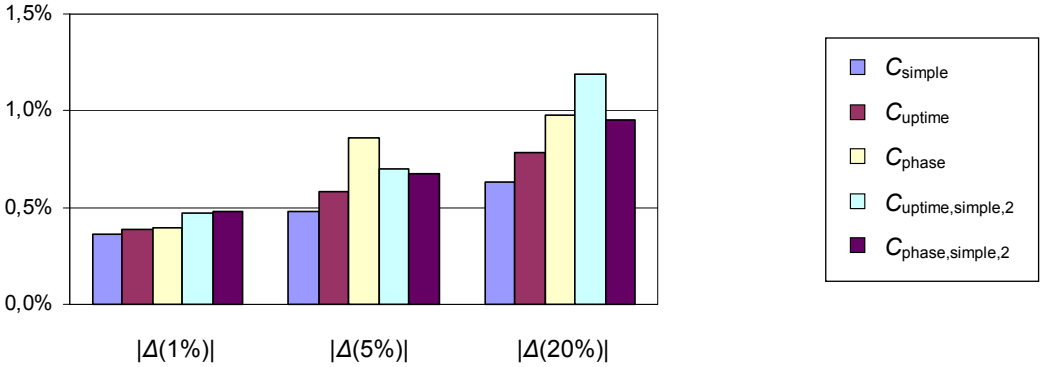

Abbildung C-10: Ergebnisse der Klassifizierungskriterien bzgl. $|\Delta(1\%)|$, $|\Delta(5\%)|$ und $|\Delta(20\%)|$

Die Ergebnisse der beiden Mischverfahren führten für die Gruppe mit den vier Rechner gegenüber den entsprechenden reinen Klassifizierungskriterien C_{uptime} und C_{phase} offensichtlich nicht zu den erhofften Verbesserungen. Berücksichtigt man den erhöhten Aufwand zur Erstellung mehrerer möglicher Prognosen, sprechen diese Ergebnisse gegen die Verwendung der angesprochenen Mischverfahren. Die in Abschnitt 4.3 aufgeführten Ergebnisse bestätigen die hier gezogen Schlussfolgerung.

Anhang D – Messergebnisse

In diesem Anhangskapitel sind die Messergebnisse entsrechend der in Abschnitt 4.1.5 erläuterten Struktur dokumentiert.

D.1. Messergebnisse aus der Anpassungsphase

In diesem Abschnitt finden sich die zum Abschnitt 4.2 gehörenden Messergebnisse für die vier in der Anpassungsphase genutzten Rechner.

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	16,66	0,0860	1,37 %	2,13 %	6,05 %
C_{uptime}	15,38	**0,0843**	**1,07 %**	1,20 %	**5,34 %**
$C_{hourOfWeek}$	16,91	0,0906	1,24 %	1,28 %	5,93 %
$C_{hourOfDay}$	15,87	0,0893	1,17 %	1,27 %	5,82 %
C_{phase}	**15,16**	0,0889	1,08 %	**1,18 %**	5,98 %

Tabelle D-1: Vergleich möglicher Klassifizierungskritierien unter Nutzung der Transformationsvariante $T_{hyperexponential}$

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	13,79	**0,1396**	**1,59%**	**3,49%**	**6,84%**
C_{uptime}	**12,70**	0,1583	2,42%	5,04%	9,19%
$C_{hourOfWeek}$	16,11	0,2043	6,00%	8,95%	15,20%
$C_{hourOfDay}$	13,81	0,1742	3,41%	6,23%	11,11%
C_{phase}	12,96	0,1744	3,77%	6,28%	10,82%

Tabelle D-2: Vergleich möglicher Klassifizierungskritierien unter Verwendung der Transformation T_{histo}

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	13,30	**0,0236**	**0,58 %**	**1,28 %**	**2,02 %**
C_{uptime}	**11,87**	0,0341	3,59 %	3,46 %	3,56 %
$C_{hourOfWeek}$	15,27	0,0735	17,46 %	13,66 %	9,64 %
$C_{hourOfDay}$	12,65	0,0376	6,40 %	5,67 %	4,81 %
C_{phase}	11,99	0,0474	8,40 %	7,29 %	5,16 %

Tabelle D-3: Vergleich möglicher Klassifizierungskritierien unter Verwendung der Transformation $T_{1:1}$

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	13,29	**0,0234**	**0,50 %**	1,28 %	**2,01 %**
C_{uptime}	11,64	0,0240	0,67 %	**1,12 %**	2,08 %
$C_{hourOfWeek}$	13,92	0,0381	1,32 %	2,52 %	3,84 %
$C_{hourOfDay}$	12,37	0,0277	0,68 %	1,48 %	2,87 %
C_{phase}	**11,41**	0,0274	0,79 %	1,63 %	2,35 %

Tabelle D-4: Vergleich möglicher Klassifizierungskritierien unter Verwendung der Transformation $T_{fullRange}$

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	13,29	**0,0234**	**0,50 %**	1,28 %	**2,01 %**
C_{uptime}	11,63	0,0252	0,67 %	**1,11 %**	2,08 %
$C_{hourOfWeek}$	13,79	0,0375	1,31 %	2,49 %	3,80 %
$C_{hourOfDay}$	12,34	0,0276	0,67 %	1,48 %	2,86 %
C_{phase}	**11,37**	0,0286	0,81 %	1,61 %	2,35 %

Tabelle D-5: Vergleich möglicher Klassifizierungskriterien unter Nutzung der Transformationsvariante $T_{dynRange}$

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	12,78	0,0144	**0,36 %**	**0,48 %**	**0,63 %**
C_{uptime}	11,38	**0,0142**	0,39 %	0,58 %	0,78 %
$C_{hourOfWeek}$	13,12	0,0213	0,94 %	0,96 %	1,30 %
$C_{hourOfDay}$	11,75	0,0153	0,38 %	0,49 %	1,19 %
C_{phase}	**11,09**	0,0172	0,40 %	0,86 %	0,98 %

Tabelle D-6: Vergleich möglicher Prognoseverfahren unter Verwendung der Transformationsvariante $T_{fullRange}$ und der adaptiven Korrektur A_{voll}

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	12,92	**0,0162**	**0,43 %**	**0,52 %**	**0,85 %**
C_{uptime}	11,46	0,0171	0,46 %	0,59 %	1,21 %
$C_{hourOfWeek}$	13,38	0,0262	0,95 %	1,44 %	2,19 %
$C_{hourOfDay}$	11,96	0,0192	0,48 %	0,67 %	1,37 %
C_{phase}	**11,20**	0,0206	0,49 %	0,86 %	1,45 %

Tabelle D-7: Vergleich möglicher Klassifizierungskriterien unter Verwendung der Transformationsvariante $T_{fullRange}$ und der adaptiven Korrektur $A_{partiell}$ mit einem Korrekturgewicht von $f_{korr} = 0,5$

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	12,92	**0,0162**	**0,43 %**	**0,52 %**	**0,83 %**
C_{uptime}	11,42	0,0168	0,46 %	0,62 %	1,10 %
$C_{hourOfWeek}$	13,23	0,0250	0,93 %	1,31 %	2,07 %
$C_{hourOfDay}$	11,91	0,0191	0,47 %	0,62 %	1,41 %
C_{phase}	**11,13**	0,0200	0,51 %	0,84 %	1,38 %

Tabelle D-8: Vergleich möglicher Klassifizierungskriterien unter Verwendung der Transformationsvariante $T_{fullRange}$ und der adaptiven Korrektur $A_{variable}$

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	12,78	0,0144	0,36 %	**0,48 %**	**0,63 %**
C_{uptime}	11,38	**0,0142**	0,39 %	0,58 %	0,78 %
$C_{hourOfWeek}$	13,12	0,0213	0,94 %	0,96 %	1,30 %
$C_{hourOfDay}$	11,75	0,0153	0,38 %	0,49 %	1,19 %
C_{phase}	**11,09**	0,0172	0,40 %	0,86 %	0,98 %
$C_{uptime,clustered}$	11,37	0,0155	**0,33 %**	0,56 %	0,97 %

Tabelle D-9: Ergebnis des getesteten Cluster-Verfahrens unter Verwendung der Transformationsvariante $T_{fullRange}$ und der adaptiven Korrektur A_{full}

Klassifizierungs-kriterium	Durchschnitt von				
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$
C_{simple}	12,78	0,0144	**0,36 %**	**0,48 %**	**0,63 %**
C_{uptime}	11,38	**0,0142**	0,39 %	0,58 %	0,78 %
C_{phase}	**11,09**	0,0172	0,40 %	0,86 %	0,98 %
$C_{uptime,simple,2}$	11,45	0,0160	0,47 %	0,70 %	1,19 %
$C_{phase,simple,2}$	11,16	0,0153	0,48 %	0,67 %	0,95 %

Tabelle D-10: Ergebnisse mit den getesteten Mischverfahren unter Verwendung der Transformationsvariante $T_{fullRange}$ und der adaptiven Korrektur A_{full}

D.2. Messergebnisse für die Pool-Rechner

Klassifizierungs-kriterium	Durchschnitt von					Trans-formation
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$	
C_{simple}	23,94	0,1040	1,20 %	2,90 %	4,59 %	$T_{hyperexponential}$
C_{uptime}	20,92	0,0864	0,83 %	2,26 %	4,61 %	
$C_{hourOfWeek}$	23,43	0,1245	2,16 %	5,96 %	8,95 %	
$C_{hourOfDay}$	21,97	0,0907	1,29 %	3,01 %	4,86 %	
C_{phase}	21,27	0,1170	1,29 %	3,33 %	5,79 %	
$C_{uptime,clustered}$	21,10	0,0738	0,85 %	2,18 %	4,43 %	
$C_{uptime,simple,2}$	22,43	0,0953	0,90 %	2,42 %	5,05 %	
$C_{phase,simple,2}$	22,92	0,1158	1,17 %	2,84 %	5,41 %	
C_{simple}	23,47	0,1239	2,14 %	5,21 %	8,79 %	T_{histo}
C_{uptime}	21,01	0,1551	5,29 %	9,06 %	12,74 %	
$C_{hourOfWeek}$	23,74	0,1827	4,59 %	12,56 %	19,00 %	
$C_{hourOfDay}$	22,11	0,1664	5,81 %	11,81 %	15,95 %	
C_{phase}	21,57	0,1675	4,23 %	9,55 %	14,98 %	
$C_{uptime,clustered}$	21,12	0,1484	5,83 %	9,14 %	12,83 %	
$C_{uptime,simple,2}$	21,83	0,1389	3,58 %	7,19 %	10,64 %	
$C_{phase,simple,2}$	22,38	0,1409	2,92 %	7,25 %	11,29 %	
C_{simple}	23,98	0,0860	2,79 %	2,95 %	5,76 %	$T_{1:1}$
C_{uptime}	23,92	0,1211	22,09 %	19,23 %	11,70 %	
$C_{hourOfWeek}$	26,95	0,1850	24,60 %	25,35 %	19,02 %	
$C_{hourOfDay}$	24,69	0,1317	24,67 %	22,02 %	11,96 %	
C_{phase}	23,83	0,1494	19,62 %	18,10 %	10,67 %	
$C_{uptime,clustered}$	23,98	0,1110	21,17 %	18,83 %	13,67 %	
$C_{uptime,simple,2}$	25,28	0,1124	13,71 %	12,67 %	10,55 %	
$C_{phase,simple,2}$	24,39	0,1181	11,11 %	10,66 %	8,55 %	
C_{simple}	23,18	0,0725	0,77 %	1,76 %	5,33 %	$T_{fullRange}$
C_{uptime}	19,98	0,0637	0,86 %	2,04 %	4,00 %	
$C_{hourOfWeek}$	22,35	0,1103	2,39 %	7,30 %	9,55 %	
$C_{hourOfDay}$	20,40	0,0596	1,46 %	3,85 %	5,10 %	
C_{phase}	20,34	0,0903	1,40 %	3,43 %	5,27 %	
$C_{uptime,clustered}$	20,43	0,0562	1,27 %	3,88 %	5,74 %	
$C_{uptime,simple,2}$	21,92	0,0721	0,81 %	1,98 %	5,22 %	
$C_{phase,simple,2}$	21,82	0,0831	1,03 %	2,67 %	4,93 %	
C_{simple}	23,68	0,0786	0,77 %	1,76 %	5,33 %	$T_{dynRange}$
C_{uptime}	21,42	0,0821	0,81 %	1,64 %	3,18 %	
$C_{hourOfWeek}$	23,44	0,1135	1,79 %	5,12 %	6,38 %	
$C_{hourOfDay}$	21,72	0,0744	1,29 %	2,91 %	3,95 %	
C_{phase}	21,80	0,1062	1,16 %	2,61 %	3,92 %	
$C_{uptime,clustered}$	21,47	0,0710	1,22 %	3,51 %	5,30 %	
$C_{uptime,simple,2}$	22,15	0,0801	0,80 %	1,94 %	4,70 %	
$C_{phase,simple,2}$	22,10	0,0889	1,03 %	2,55 %	4,41 %	
C_{simple}	23,16	0,0573	**0,68 %**	1,65 %	4,34 %	$T_{fullRange}$ + A_{voll}
C_{uptime}	**19,91**	**0,0371**	0,81 %	1,64 %	3,05 %	
$C_{hourOfWeek}$	21,65	0,0563	0,99 %	2,63 %	4,47 %	
$C_{hourOfDay}$	20,70	0,0390	0,78 %	1,87 %	3,17 %	
C_{phase}	20,08	0,0477	0,82 %	1,66 %	**2,56 %**	
$C_{uptime,clustered}$	20,61	0,0399	1,26 %	3,11 %	3,67 %	
$C_{uptime,simple,2}$	21,34	0,0454	0,77 %	1,58 %	3,49 %	
$C_{phase,simple,2}$	21,41	0,0505	0,77 %	**1,53 %**	2,99 %	

Tabelle D-11: Ergebnisse für die Pool-Rechner unter Windows

Klassifizierungs-kriterium	Durchschnitt von					Trans-formation
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$	
C_{simple}	15,17	0,1409	16,33 %	16,96 %	17,47 %	$T_{hyperexponential}$
C_{uptime}	14,69	0,1238	15,80 %	16,74 %	15,74 %	
$C_{hourOfWeek}$	19,14	0,1862	18,98 %	25,56 %	30,29 %	
$C_{hourOfDay}$	16,27	0,1303	15,48 %	18,17 %	18,85 %	
C_{phase}	15,10	0,1161	15,06 %	17,28 %	15,76 %	
$C_{uptime,clustered}$	14,79	0,1226	16,18 %	16,97 %	16,07 %	
$C_{uptime,simple,2}$	15,01	0,1336	15,60 %	16,30 %	16,53 %	
$C_{phase,simple,2}$	14,72	0,1257	14,57 %	15,41 %	15,49 %	
C_{simple}	11,41	0,1490	14,95 %	18,37 %	22,24 %	T_{histo}
C_{uptime}	13,32	0,2054	18,00 %	23,75 %	31,22 %	
$C_{hourOfWeek}$	21,84	0,3015	21,12 %	33,09 %	47,75 %	
$C_{hourOfDay}$	16,89	0,2503	19,20 %	27,42 %	39,41 %	
C_{phase}	15,45	0,2357	19,39 %	27,21 %	36,41 %	
$C_{uptime,clustered}$	12,42	0,1878	17,79 %	22,26 %	28,26 %	
$C_{uptime,simple,2}$	11,72	0,1620	15,46 %	19,47 %	24,34 %	
$C_{phase,simple,2}$	12,00	0,1643	15,34 %	19,98 %	24,89 %	
C_{simple}	10,82	0,0699	4,16 %	3,42 %	5,38 %	$T_{1:1}$
C_{uptime}	12,03	0,1034	18,04 %	15,68 %	11,13 %	
$C_{hourOfWeek}$	19,18	0,2110	33,29 %	36,75 %	34,45 %	
$C_{hourOfDay}$	14,90	0,1384	25,39 %	24,43 %	19,38 %	
C_{phase}	13,09	0,1316	22,58 %	21,36 %	16,33 %	
$C_{uptime,clustered}$	12,00	0,0905	14,55 %	12,80 %	10,00 %	
$C_{uptime,simple,2}$	11,60	0,0778	8,42 %	7,00 %	6,51 %	
$C_{phase,simple,2}$	11,47	0,0848	8,62 %	7,49 %	7,19 %	
C_{simple}	**10,58**	0,0549	2,00 %	2,17 %	5,22 %	$T_{fullRange}$
C_{uptime}	11,05	0,0626	3,29 %	4,30 %	6,10 %	
$C_{hourOfWeek}$	17,64	0,1779	11,69 %	18,50 %	24,62 %	
$C_{hourOfDay}$	13,47	0,0906	5,05 %	7,97 %	11,40 %	
C_{phase}	12,18	0,0834	4,38 %	6,88 %	9,25 %	
$C_{uptime,clustered}$	10,96	0,0563	3,05 %	4,01 %	6,29 %	
$C_{uptime,simple,2}$	10,79	0,0557	2,38 %	2,71 %	5,45 %	
$C_{phase,simple,2}$	10,84	0,0571	2,70 %	3,12 %	5,87 %	
C_{simple}	10,71	0,0618	2,00 %	2,17 %	5,22 %	$T_{dynRange}$
C_{uptime}	10,86	0,0644	3,15 %	3,45 %	5,37 %	
$C_{hourOfWeek}$	13,48	0,1167	10,41 %	13,21 %	15,55 %	
$C_{hourOfDay}$	11,86	0,0712	4,58 %	5,62 %	7,17 %	
C_{phase}	11,10	0,0697	4,01 %	4,92 %	6,64 %	
$C_{uptime,clustered}$	10,88	0,0626	2,89 %	3,47 %	5,84 %	
$C_{uptime,simple,2}$	10,82	0,0633	2,28 %	2,95 %	5,17 %	
$C_{phase,simple,2}$	10,81	0,0639	2,51 %	3,10 %	5,76 %	
C_{simple}	11,11	0,0327	**1,85 %**	**1,30 %**	**1,82 %**	$T_{fullRange}$ + A_{voll}
C_{uptime}	11,16	0,0336	2,24 %	1,71 %	2,63 %	
$C_{hourOfWeek}$	13,43	0,0836	3,16 %	8,20 %	13,34 %	
$C_{hourOfDay}$	12,53	0,0370	2,28 %	3,10 %	4,06 %	
C_{phase}	11,70	0,0384	2,22 %	2,49 %	3,47 %	
$C_{uptime,clustered}$	11,15	**0,0324**	2,18 %	1,89 %	2,51 %	
$C_{uptime,simple,2}$	**11,05**	0,0347	2,06 %	1,55 %	2,50 %	
$C_{phase,simple,2}$	11,22	0,0351	2,16 %	1,67 %	2,64 %	

Tabelle D-12: Ergebnisse für die Pool-Rechner unter Linux

Klassifizierungs-kriterium	Durchschnitt von					Trans-formation
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$	
C_{simple}	20,39	0,0919	0,91 %	2,48 %	6,90 %	$T_{hyperexponential}$
C_{uptime}	18,62	0,0791	0,58 %	1,64 %	5,66 %	
$C_{hourOfWeek}$	21,13	0,0815	0,92 %	2,68 %	6,86 %	
$C_{hourOfDay}$	19,75	0,0817	0,76 %	2,31 %	6,13 %	
C_{phase}	19,19	0,0823	0,70 %	1,85 %	6,20 %	
$C_{uptime,clustered}$	18,61	0,0797	0,59 %	1,61 %	5,75 %	
$C_{uptime,simple,2}$	18,84	0,0794	0,56 %	1,75 %	5,59 %	
$C_{phase,simple,2}$	19,54	0,0849	0,70 %	1,94 %	6,17 %	
C_{simple}	18,25	0,1012	0,80 %	1,98 %	4,06 %	T_{histo}
C_{uptime}	17,24	0,1275	4,07 %	5,05 %	6,24 %	
$C_{hourOfWeek}$	20,67	0,1589	7,11 %	8,98 %	11,17 %	
$C_{hourOfDay}$	18,41	0,1338	4,03 %	5,49 %	7,73 %	
C_{phase}	18,21	0,1443	4,85 %	6,22 %	7,82 %	
$C_{uptime,clustered}$	16,90	0,1216	2,71 %	4,18 %	6,12 %	
$C_{uptime,simple,2}$	17,22	0,1194	3,35 %	4,43 %	6,48 %	
$C_{phase,simple,2}$	17,80	0,1222	3,42 %	4,76 %	7,09 %	
C_{simple}	18,03	0,0354	0,81 %	1,42 %	2,66 %	$T_{1:1}$
C_{uptime}	16,79	0,0481	8,70 %	7,46 %	5,87 %	
$C_{hourOfWeek}$	22,25	0,0902	23,76 %	19,79 %	11,88 %	
$C_{hourOfDay}$	17,93	0,0496	9,34 %	7,23 %	5,80 %	
C_{phase}	18,60	0,0743	16,39 %	13,72 %	8,67 %	
$C_{uptime,clustered}$	16,62	0,0458	5,79 %	5,46 %	5,77 %	
$C_{uptime,simple,2}$	17,12	0,0472	7,28 %	6,70 %	5,59 %	
$C_{phase,simple,2}$	18,42	0,0534	8,01 %	6,87 %	5,32 %	
C_{simple}	18,02	0,0351	0,65 %	1,30 %	2,58 %	$T_{fullRange}$
C_{uptime}	16,13	0,0323	0,50 %	1,75 %	3,04 %	
$C_{hourOfWeek}$	18,84	0,0441	1,40 %	3,37 %	3,70 %	
$C_{hourOfDay}$	17,29	0,0352	0,80 %	2,02 %	2,61 %	
C_{phase}	16,63	0,0366	0,89 %	2,14 %	3,08 %	
$C_{uptime,clustered}$	16,21	0,0372	0,89 %	2,51 %	4,28 %	
$C_{uptime,simple,2}$	16,41	0,0345	0,52 %	1,85 %	3,24 %	
$C_{phase,simple,2}$	16,76	0,0397	0,79 %	2,16 %	3,36 %	
C_{simple}	18,02	0,0351	0,65 %	1,30 %	2,58 %	$T_{dynRange}$
C_{uptime}	16,12	0,0316	0,50 %	1,75 %	3,04 %	
$C_{hourOfWeek}$	18,83	0,0442	1,40 %	3,37 %	3,70 %	
$C_{hourOfDay}$	17,29	0,0360	0,80 %	2,02 %	2,61 %	
C_{phase}	16,61	0,0391	0,89 %	2,14 %	3,08 %	
$C_{uptime,clustered}$	16,21	0,0374	0,89 %	2,51 %	4,28 %	
$C_{uptime,simple,2}$	16,40	0,0342	0,52 %	1,85 %	3,23 %	
$C_{phase,simple,2}$	16,77	0,0408	0,77 %	2,11 %	3,36 %	
C_{simple}	18,09	0,0211	**0,32 %**	**0,45 %**	1,61 %	$T_{fullRange}$ + A_{voll}
C_{uptime}	**16,00**	0,0199	0,39 %	0,98 %	1,96 %	
$C_{hourOfWeek}$	18,51	0,0224	0,82 %	1,30 %	2,14 %	
$C_{hourOfDay}$	17,12	**0,0186**	0,58 %	0,69 %	**1,16 %**	
C_{phase}	16,38	0,0225	0,54 %	1,05 %	1,88 %	
$C_{uptime,clustered}$	16,07	0,0227	0,59 %	1,28 %	2,48 %	
$C_{uptime,simple,2}$	16,07	0,0201	0,38 %	1,18 %	2,05 %	
$C_{phase,simple,2}$	16,47	0,0233	0,47 %	1,16 %	2,13 %	

Tabelle D-13: Ergebnisse für die Rechner des RNKS-Pools

D.3. Messergebnisse für die PCs

Klassifizierungs-kriterium	Durchschnitt von					Trans-formation
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$	
C_{simple}	15,96	0,1221	1,12 %	2,07 %	8,08 %	$T_{hyperexponential}$
C_{uptime}	15,19	0,1339	1,08 %	2,20 %	10,45 %	
$C_{hourOfWeek}$	14,65	0,1509	0,95 %	2,35 %	12,60 %	
$C_{hourOfDay}$	14,30	0,1528	0,99 %	2,42 %	12,48 %	
C_{phase}	14,89	0,1375	0,99 %	2,29 %	11,64 %	
$C_{uptime,clustered}$	15,12	0,1360	1,06 %	2,24 %	10,64 %	
$C_{uptime,simple,2}$	15,19	0,1351	1,09 %	2,18 %	10,34 %	
$C_{phase,simple,2}$	14,85	0,1397	1,00 %	2,24 %	11,36 %	
C_{simple}	11,97	0,0735	1,08 %	1,53 %	2,03 %	T_{histo}
C_{uptime}	11,02	0,1048	2,62 %	3,20 %	4,23 %	
$C_{hourOfWeek}$	11,47	0,1545	4,03 %	4,05 %	7,72 %	
$C_{hourOfDay}$	9,81	0,1178	2,43 %	2,58 %	5,52 %	
C_{phase}	11,23	0,1215	3,28 %	3,90 %	5,24 %	
$C_{uptime,clustered}$	10,85	0,0956	2,10 %	2,91 %	3,60 %	
$C_{uptime,simple,2}$	10,82	0,0887	1,85 %	2,41 %	3,39 %	
$C_{phase,simple,2}$	10,44	0,0889	1,88 %	2,61 %	3,33 %	
C_{simple}	11,84	0,0237	0,64 %	0,99 %	1,80 %	$T_{1:1}$
C_{uptime}	10,61	0,0350	4,44 %	3,97 %	3,37 %	
$C_{hourOfWeek}$	10,02	0,0843	17,28 %	13,52 %	9,97 %	
$C_{hourOfDay}$	8,95	0,0409	5,82 %	4,09 %	4,42 %	
C_{phase}	10,50	0,0518	9,21 %	8,15 %	5,79 %	
$C_{uptime,clustered}$	10,62	0,0341	2,92 %	3,59 %	2,66 %	
$C_{uptime,simple,2}$	10,54	0,0300	2,65 %	2,82 %	2,53 %	
$C_{phase,simple,2}$	10,11	0,0355	4,47 %	4,19 %	3,14 %	
C_{simple}	11,84	0,0234	0,54 %	0,97 %	1,76 %	$T_{fullRange}$
C_{uptime}	10,44	0,0259	0,88 %	1,41 %	1,80 %	
$C_{hourOfWeek}$	9,43	0,0482	0,95 %	1,56 %	4,68 %	
$C_{hourOfDay}$	8,85	0,0273	0,55 %	0,70 %	2,34 %	
C_{phase}	10,04	0,0284	0,98 %	1,53 %	1,83 %	
$C_{uptime,clustered}$	10,47	0,0280	0,51 %	1,51 %	1,80 %	
$C_{uptime,simple,2}$	10,47	0,0256	0,88 %	1,51 %	1,75 %	
$C_{phase,simple,2}$	9,97	0,0263	0,98 %	1,61 %	1,76 %	
C_{simple}	11,84	0,0234	0,54 %	0,97 %	1,76 %	$T_{dynRange}$
C_{uptime}	10,33	0,0321	0,88 %	1,39 %	1,76 %	
$C_{hourOfWeek}$	9,28	0,0461	1,00 %	1,60 %	4,63 %	
$C_{hourOfDay}$	**8,80**	0,0267	0,55 %	0,70 %	2,36 %	
C_{phase}	9,88	0,0353	0,96 %	1,51 %	1,84 %	
$C_{uptime,clustered}$	10,44	0,0311	0,51 %	1,51 %	1,82 %	
$C_{uptime,simple,2}$	10,36	0,0315	0,88 %	1,50 %	1,71 %	
$C_{phase,simple,2}$	9,82	0,0326	0,96 %	1,63 %	1,76 %	
C_{simple}	11,91	0,0150	**0,33 %**	0,77 %	1,10 %	$T_{fullRange}$ + A_{voll}
C_{uptime}	10,48	0,0150	0,90 %	1,00 %	1,16 %	
$C_{hourOfWeek}$	9,34	0,0340	0,63 %	1,02 %	2,45 %	
$C_{hourOfDay}$	8,87	0,0267	0,47 %	**0,65 %**	1,63 %	
C_{phase}	10,04	0,0174	0,83 %	1,06 %	0,96 %	
$C_{uptime,clustered}$	10,50	**0,0148**	0,48 %	0,84 %	**0,88 %**	
$C_{uptime,simple,2}$	10,56	0,0159	0,91 %	1,12 %	1,01 %	
$C_{phase,simple,2}$	10,08	0,0163	0,87 %	1,10 %	0,99 %	

Tabelle D-14: Ergebnisse für die vier PCs

D.4. Messergebnisse für die Server-Maschine

Klassifizierungs-kriterium	Durchschnitt von					Trans-formation
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$	
C_{simple}	25,01	0,1807	28,97 %	25,26 %	21,04 %	$T_{hyperexponential}$
C_{uptime}	24,77	0,1760	29,01 %	25,22 %	20,83 %	
$C_{hourOfWeek}$	25,12	0,1777	28,99 %	25,19 %	20,73 %	
$C_{hourOfDay}$	24,77	0,1794	28,99 %	25,17 %	20,72 %	
C_{phase}	24,84	0,1750	29,01 %	25,21 %	20,83 %	
$C_{uptime,clustered}$	24,77	0,1762	29,01 %	25,22 %	20,81 %	
$C_{uptime,simple,2}$	24,66	0,1771	29,01 %	25,21 %	20,81 %	
$C_{phase,simple,2}$	24,70	0,1769	29,02 %	25,21 %	20,79 %	
C_{simple}	19,41	0,1554	13,26 %	15,18 %	11,74 %	T_{histo}
C_{uptime}	18,89	0,1694	12,18 %	14,40 %	12,03 %	
$C_{hourOfWeek}$	22,13	0,2079	17,25 %	15,62 %	12,10 %	
$C_{hourOfDay}$	20,23	0,1859	13,63 %	15,49 %	12,58 %	
C_{phase}	19,36	0,1808	12,96 %	14,87 %	11,76 %	
$C_{uptime,clustered}$	18,68	0,1669	12,18 %	14,41 %	11,98 %	
$C_{uptime,simple,2}$	18,28	0,1601	12,46 %	14,26 %	12,08 %	
$C_{phase,simple,2}$	18,41	0,1611	12,94 %	14,73 %	12,00 %	
C_{simple}	19,08	0,0437	1,21 %	3,38 %	6,40 %	$T_{1:1}$
C_{uptime}	18,33	0,0480	2,85 %	3,43 %	4,63 %	
$C_{hourOfWeek}$	20,67	0,0565	7,57 %	6,07 %	6,99 %	
$C_{hourOfDay}$	19,25	0,0462	2,34 %	4,04 %	6,46 %	
C_{phase}	18,56	0,0527	3,25 %	3,92 %	4,53 %	
$C_{uptime,clustered}$	18,25	0,0472	2,22 %	2,92 %	4,30 %	
$C_{uptime,simple,2}$	17,89	0,0442	2,25 %	2,93 %	4,43 %	
$C_{phase,simple,2}$	17,93	0,0454	2,14 %	3,08 %	5,15 %	
C_{simple}	19,08	0,0436	1,17 %	3,38 %	6,40 %	$T_{fullRange}$
C_{uptime}	18,29	0,0452	1,51 %	2,65 %	4,46 %	
$C_{hourOfWeek}$	20,10	0,0456	4,05 %	3,12 %	6,30 %	
$C_{hourOfDay}$	19,13	0,0425	1,12 %	2,94 %	6,21 %	
C_{phase}	18,47	0,0484	1,53 %	2,72 %	4,43 %	
$C_{uptime,clustered}$	18,23	0,0456	1,34 %	2,51 %	4,28 %	
$C_{uptime,simple,2}$	17,83	0,0422	1,38 %	2,52 %	4,43 %	
$C_{phase,simple,2}$	17,90	0,0429	1,34 %	2,58 %	4,81 %	
C_{simple}	19,08	0,0436	1,17 %	3,38 %	6,40 %	$T_{dynRange}$
C_{uptime}	18,29	0,0459	1,51 %	2,65 %	4,46 %	
$C_{hourOfWeek}$	20,10	0,0459	4,05 %	3,12 %	6,30 %	
$C_{hourOfDay}$	19,13	0,0427	1,12 %	2,94 %	6,21 %	
C_{phase}	18,47	0,0493	1,53 %	2,72 %	4,43 %	
$C_{uptime,clustered}$	18,23	0,0460	1,34 %	2,51 %	4,28 %	
$C_{uptime,simple,2}$	17,83	0,0425	1,38 %	2,52 %	4,43 %	
$C_{phase,simple,2}$	17,90	0,0431	1,34 %	2,58 %	4,81 %	
C_{simple}	19,17	0,0407	**0,74 %**	2,65 %	5,35 %	$T_{fullRange}$ + A_{voll}
C_{uptime}	17,65	0,0367	1,10 %	2,47 %	4,28 %	
$C_{hourOfWeek}$	20,21	0,0407	3,77 %	2,89 %	5,51 %	
$C_{hourOfDay}$	19,36	0,0404	0,95 %	2,35 %	5,06 %	
C_{phase}	17,88	0,0381	1,21 %	2,20 %	3,79 %	
$C_{uptime,clustered}$	**17,57**	**0,0349**	0,77 %	**2,08 %**	**3,52 %**	
$C_{uptime,simple,2}$	17,62	0,0372	1,03 %	2,48 %	4,17 %	
$C_{phase,simple,2}$	17,75	0,0380	0,99 %	2,24 %	4,13 %	

Tabelle D-15: Ergebnisse für die untersuchte Server-Maschine

D.5. Messergebnisse für die Rechner aus [Long95]

Klassifizierungs-kriterium	Durchschnitt von					Trans-formation
	M_{exp}	M_{avg}	$\lvert\Delta(1\%)\rvert$	$\lvert\Delta(5\%)\rvert$	$\lvert\Delta(20\%)\rvert$	
C_{simple}	16,78	0,2381	0,94 %	4,78 %	19,09 %	$T_{hyperexponential}$
C_{uptime}	16,89	0,2349	0,94 %	4,78 %	19,12 %	
$C_{hourOfWeek}$	16,80	0,2337	0,94 %	4,77 %	19,11 %	
$C_{hourOfDay}$	16,77	0,2368	0,95 %	4,78 %	19,09 %	
C_{phase}	17,09	0,2295	0,94 %	4,78 %	19,12 %	
$C_{uptime,clustered}$	16,87	0,2367	0,94 %	4,78 %	19,12 %	
$C_{uptime,simple,2}$	16,80	0,2371	0,94 %	4,78 %	19,10 %	
$C_{phase,simple,2}$	16,83	0,2364	0,94 %	4,78 %	19,09 %	
C_{simple}	5,36	0,0367	0,62 %	1,48 %	2,31 %	T_{histo}
C_{uptime}	5,89	0,0471	1,17 %	1,45 %	3,03 %	
$C_{hourOfWeek}$	5,74	0,0807	4,01 %	1,88 %	8,26 %	
$C_{hourOfDay}$	5,49	0,0589	1,24 %	1,52 %	5,10 %	
C_{phase}	6,62	0,0540	1,22 %	1,44 %	5,36 %	
$C_{uptime,clustered}$	5,71	0,0422	1,16 %	1,42 %	2,53 %	
$C_{uptime,simple,2}$	5,50	0,0378	0,70 %	1,40 %	2,26 %	
$C_{phase,simple,2}$	5,46	0,0373	0,62 %	1,38 %	2,13 %	
C_{simple}	**5,32**	0,0219	0,61 %	1,55 %	2,16 %	$T_{1:1}$
C_{uptime}	5,72	0,0308	3,28 %	3,02 %	2,09 %	
$C_{hourOfWeek}$	5,85	0,0395	9,39 %	5,39 %	5,34 %	
$C_{hourOfDay}$	5,41	0,0237	2,18 %	2,13 %	2,98 %	
C_{phase}	6,31	0,0380	4,04 %	3,47 %	2,09 %	
$C_{uptime,clustered}$	5,56	0,0283	2,69 %	2,62 %	2,08 %	
$C_{uptime,simple,2}$	5,45	0,0235	0,95 %	1,66 %	2,10 %	
$C_{phase,simple,2}$	5,44	0,0225	0,90 %	1,56 %	1,98 %	
C_{simple}	**5,32**	0,0219	0,59 %	1,54 %	2,16 %	$T_{fullRange}$
C_{uptime}	5,71	0,0272	0,82 %	1,58 %	2,04 %	
$C_{hourOfWeek}$	5,82	0,0235	0,80 %	3,72 %	2,53 %	
$C_{hourOfDay}$	5,40	0,0210	0,55 %	1,33 %	2,40 %	
C_{phase}	6,25	0,0322	0,87 %	1,59 %	2,54 %	
$C_{uptime,clustered}$	5,55	0,0254	0,89 %	1,59 %	1,84 %	
$C_{uptime,simple,2}$	5,46	0,0226	0,64 %	1,48 %	1,91 %	
$C_{phase,simple,2}$	5,45	0,0224	0,63 %	1,46 %	1,93 %	
C_{simple}	**5,32**	0,0219	0,59 %	1,54 %	2,16 %	$T_{dynRange}$
C_{uptime}	5,71	0,0273	0,82 %	1,58 %	2,04 %	
$C_{hourOfWeek}$	5,82	0,0233	0,80 %	3,72 %	2,53 %	
$C_{hourOfDay}$	5,40	0,0211	0,55 %	1,33 %	2,40 %	
C_{phase}	6,26	0,0326	0,87 %	1,59 %	2,54 %	
$C_{uptime,clustered}$	5,55	0,0254	0,89 %	1,59 %	1,84 %	
$C_{uptime,simple,2}$	5,46	0,0226	0,64 %	1,48 %	1,91 %	
$C_{phase,simple,2}$	5,45	0,0224	0,63 %	1,46 %	1,93 %	
C_{simple}	5,37	0,0180	0,47 %	1,28 %	**1,43 %**	$T_{fullRange}$ + A_{voll}
C_{uptime}	5,73	0,0209	0,57 %	1,29 %	1,66 %	
$C_{hourOfWeek}$	5,80	0,0183	0,54 %	2,37 %	1,79 %	
$C_{hourOfDay}$	5,43	**0,0175**	**0,32 %**	**0,95 %**	1,58 %	
C_{phase}	6,21	0,0222	0,58 %	1,36 %	2,02 %	
$C_{uptime,clustered}$	5,59	0,0203	0,61 %	1,23 %	1,51 %	
$C_{uptime,simple,2}$	5,48	0,0186	0,50 %	1,18 %	1,58 %	
$C_{phase,simple,2}$	5,48	0,0189	0,50 %	1,20 %	1,63 %	

Tabelle D-16: Ergebnisse für 20 Rechner aus [Long95]

Glossar

In diesem Glossar werden in dieser Arbeit verwendete Begriffe kurz erklärt. Die Existenz einer solchen Erklärung wird bei der jeweils ersten Verwendung eines Begriffes innerhalb eines Kapitels durch eine grau gestrichelte Unterstreichung explizit angezeigt (wie z. B. bei Perzentil).

Checkpoint

Ein Checkpoint ist der zu einem bestimmten Zeitpunkt in der Programmausführung erreichte Zwischenstand, der auf geeignete Weise gespeichert wird. Der Zweck eines Checkpoints ist, nach einem unterwarteten Anwendungs- oder Systemausfall die Programmausführung mit dem gespeicherten Zwischenstand fortsetzen zu können, die Programmausführung also nicht wieder ganz von vorn beginnen zu müssen.

Einzelrechner

Unter Einzelrechnern seien hier solche Rechner verstanden, die von ihrer Leistungsfähigkeit und ihrer grundlegenden Architektur her in etwa einem modernen PC entsprechen. Explizit ausgeschlossen sind Cluster und andere Massiv-parallele Rechner. Unter diese Definition des Einzelrechners fallen somit insbesondere Arbeitsplatz- und Privatrechner, Workstations in Computer-Pools und auch viele Server.

Genauigkeit im Durchschnitt

Macht ein Prognoseverfahren in n Fällen die Zusicherungen, dass mit einer Wahrscheinlichkeit p jeweils eine Rechenkapazität von mindestens $x_{p,i}$ zur Verfügung steht, dann sollte auch in etwa $p \cdot n$ Fällen die jeweilige Rechenkapazität $x_{p,i}$ erreicht oder überschritten werden. Gilt dies für jedes p aus dem Intervall [0;1], dann hat das Prognoseverfahren die Eigenschaft, im Durchschnitt genau zu sein.

Grid-Anwendung

Bei einer Grid-Anwendung handelt es sich um eine Anwendung, die auf einem Rechen-Grid ausgeführt wird.

Grid-Computing

Der Begriff des Grid-Computings bezeichnet allgemein das Szenario der Nutzung von Ressourcen durch Anwender, denen diese Ressourcen nicht (notwendigerweise) gehören. Im Rahmen der vorliegenden Arbeit wird der Begriff ausschließlich im Zusammenhang mit Rechen-Grids verwendet; bei den Ressourcen handelt es sich somit um Rechner.

Grid-Job

Ein Grid-Job ist ein Auftrag, der in einem Rechen-Grid ausgeführt werden soll. Faktisch handelt es sich um eine mit bestimmten Parametern versehene Grid-Anwendung.

Häufigkeitsverteilung

Eine Häufigkeitsverteilung gibt für jeden möglichen Wert die Häufigkeit des Auftretens an. In dieser Arbeit sind dabei stets kumulierte relative Häufigkeitsverteilungen gemeint. Relativ bedeutet dabei, dass die Häufigkeitsangabe relativ zur Gesamtzahl der aufgetretenen Werte erfolgt, also als Wert aus dem Intervall [0;1]. Kumuliert bedeutet, dass für jeden möglichen Wert x der Anteil der aufgetretenen Werte angegeben ist, der nicht größer als x war.

Idle-Zeit

Unter einer Idle-Zeit eines Rechners wird hier ein zusammenhängender Zeitraum verstanden, innerhalb dessen ein Rechner nicht aktiv genutzt wird, der Rechner also eigentlich nichts zu tun hat.

Job

Siehe Grid-Job.

Kumulierte Verteilungsfunktion

Wird auch als kumulative Verteilungsfunktion bezeichnet. Siehe Verteilungsfunktion.

Mean Time to Failure (MTTF)

Bei der Mean Time to Failure handelt es sich um die durchschnittliche Zeit von der Inbetriebnahme bis zum Ausfall eines Gerätes oder Bauteils.

Perzentil

Ein Perzentil ist ein Quantil der Ordnung p, für das $(p \cdot 100)$ eine ganze Zahl darstellt.

Punktgenauigkeit

Eine Vorhersagemethode wird hier als punktgenau bezeichnet, wenn die für die verschiedenen Zeitpunkte t_i $(1 \leq i \leq n)$ jeweils vorausgesagten Wahrscheinlichkeitsverteilungen mit den realen Verteilungen der entsprechenden Zeitpunkte übereinstimmen. Mit der realen Verteilung zu einem Zeitpunkt t sei dabei jene gemeint, welche sich bei Kenntnis aller bis zum Zeitpunkt t gewinnbaren Informationen ergibt.

Quantil

Der Begriff des Quantils bezieht sich auf eine Verteilung mit einer kumulierten Verteilungsfunktion F. Ein Quantil der Ordnung p (mit $0 \le p \le 1$) ist dann der kleinste Wert x, für den $F(x) \ge p$ gilt.

(Ein Quantil der Ordnung p liefert also den kleinsten Wert x, der von ($p \cdot 100$) Prozent der Werte nicht überschritten wird.)

Rechen-Grid

Ein Rechen-Grid ist eine Infrastruktur, auf der (insbesondere rechenintensive) Anwendungsprogramme ausgeführt werden können. Wesentliches Charakteristikum ist, dass die Programme dabei auch für Nutzer ausgeführt werden (können), denen die Ressourcen des Rechen-Grids nicht gehören.

Stationarität

Die Eigenschaft der Stationarität bezieht sich auf den zu einer Zeitreihe Z gehörenden stochastischen Prozess (X_t), wobei die in der Zeitreihe zu den einzelnen Zeitpunkten t gehörenden Werte x_t jeweils die aufgetretenen Ereignisse der Zufallsgröße X_t sind. Die Stationarität fordert, dass alle diese Zufallsverteilungen X_t gleich sind. Ferner wird gefordert, dass auch die stochastischen Zusammenhänge zwischen den zu verschiedenen Zeitpunkten gehörenden Zufallsgrößen nur von deren zeitlichem Abstand abhängen, ansonsten aber nicht aber vom konkreten Zeitpunkt. Formal bedeutet dies, dass *„die gemeinsame Verteilungsfunktion jedes endlichen Systems von Zufallsvariablen* $(X_{t_1},X_{t_2},\ldots,X_{t_n})$ *des Prozesses identisch ist mit der gemeinsamen Verteilungsfunktion des um s Zeitpunkte verschobenen Systems* $(X_{t_1+s},X_{t_2+s},\ldots,X_{t_n+s})$*“* ([Schl99]). Da man diese Eigenschaft in der Praxis so nicht überprüfen kann, prüft man üblicherweise nur die Mittelwerte, Varianzen und Kovarianzen, was zum Begriff der schwachen Stationarität führt. Die meisten Modelle der Zeitreihenanalyse setzen in irgendeiner Form eine solche Stationärität voraus bzw. versuchen diese zu erreichen.

Verteilungsfunktion

Eine Wahrscheinlichkeitsverteilung X kann durch ihre Verteilungsfunktion F_X beschrieben werden, die wie folgt definiert ist:

$$F_X(x) = P(x \le X)$$

Die Bezeichnung als kumulierte Verteilungsfunktion kann helfen, Verwechselungen mit der Wahrscheinlichkeitsfunktion oder der Dichtefunktion zu vermeiden. ([Wiki_a])

Zeitreihe

Eine Zeitreihe ist eine zeitlich geordnete Folge von Werten einer bestimmten Größe. Typischerweise ist bei einer Zeitreihe dabei der zeitliche Abstand zwischen aufeinanderfolgenden Werten konstant. Ein Beispiel für eine Zeitreihe sind die im monatlichen Abstand bestimmten Arbeitslosenzahlen in Deutschland.

Literaturverzeichnis

[Abe01] Shigeo Abe:
„Pattern Classification: Neuro-fuzzy Methods and Their Comparison“
Springer-Verlag, London, 2001

[Amaz07] *„Amazon Elastic Compute Cloud (Amazon EC2) – Limited Beta“*
Amazon.com, http://www.amazon.com/gp/browse.html?node=201590011, zitiert am 18.6.2007

[Amaz07a] *„Amazon Elastic Compute Cloud (Amazon EC2) – Limited Beta FAQs“*
Amazon.com, http://www.amazon.com/b/ref=sc_fe_c_0_201590011_4/105-9881562-2665257?ie=UTF8&node=201591011&no=201590011&me=A36L942TSJ2AJA, zitiert am 19.6.2007

[Amaz08] *„Amazon Simple Storage Service (Amazon S3)“*
Amazon.com, http://www.amazon.com/gp/browse.html?node=16427261, zitiert am 22.1.2008

[Ande03] D. Anderson, J. Dykes, E. Riedel:
„More than an interface – SCSI vs. ATA“
Proc. of the 2nd USENIX Conference on File and Storage Technologies, 2003

[Andr02] Artur Andrzejak, Martin Arlitt, Jerry Rolia:
„Bounding the Resource Savings of Utility Computing Models“
Hewlett-Packard Company, 2002, http://www.hpl.hp.com/techreports/2002/HPL-2002-339.html, zitiert am 3.8.2007

[Anse06] E. Ansett:
„Addressing the Power & Cooling Problem Using Pi Performance Indices“
EYP Mission Critical Facilities, 2006,
http://www.fdc2006.com/__data/assets/pdf_file/0009/176094/Ed_Ansett_Presentation.pdf, zitiert am 28.11.2007

[Apfe07] D. Apfelbaum:
„Wer verdient wie viel? Ergebnisse der c't-Gehaltsumfrage 2006“
c't 6/2007, S. 104: Gehaltsumfrage, Heise Verlag, http://www.heise.de/ct/07/06/104/

[ASGC07] GStat, Monitoring-Statistik, ASGC/CERN, http://goc.grid.sinica.edu.tw/gstat/, zitiert im November 2007

[BBC01] David Whitehouse:
„Number takes prime position“
BBC News online, 5. Dezember 2001, http://news.bbc.co.uk/1/hi/sci/tech/1693364.stm

[Behr99] Christian-Uwe Behrens, Matthias Kirspel:
„Grundlagen der Volkswirtschaftslehre: Einführung“
R. Oldenbourg Verlag, München, 1999

[Bers02] Viktors Berstis:
„Fundamentals of Grid Computing“
Redbooks paper, IBM, 2002, http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf

[Bird07] I. Bird:
„The EGEE Production Grid”
EGEE User Forum, 9 – 11 May 2007, Manchester (UK), www.eu-egee.org

[Blum02] Ralf Blumhardt:
„Numerische Optimierung des Crashverhaltens von Fahrzeugstrukturen und -komponenten"
Dissertation, Shaker Verlag, Aachen, 2002

[Boeg03] H. Bögeholz:
„IDE kontra SCSI – Festplatten im Servereinsatz”
c't 1/2003, Heise Verlag

[Bolo00] William J. Bolosky, John R. Douceur, David Ely, Marvin Theimer:
„ Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs“
ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Santa Clara, California, United States, 2000

[Brev04] John Brevik, Daniel Nurmi, Rich Wolski:
„Automatic Methods for Predicting Machine Availability in Desktop Grid and Peer-to-peer Systems“
IEEE International Symposium on Cluster Computing and the Grid, pp.190-199, April 19-22, 2004,
http://pompone.cs.ucsb.edu/~nurmi/mypapers/ccgrid04.pdf

[Bulh04] P. T. Bulhões, C. Byun, R. Castrapel, O. Hassaine:
„Sun N1 Grid Engine 6 Features and Capabilities“
Sun Microsystems, 2004, http://www.sun.com/software/gridware/whitepapers.xml

[Busc00] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal:
„Pattern-orientierte Softwarearchitektur – Ein Pattern-System“
Addison Wesley, 1. korr. Nachdruck, 2000

[Comp06] *„Sun will mit Partnern sein Grid-Angebot ausbauen“*
Computerwoche 38/2006

[Cord07] *„EGEE“*
Project Fact Sheet, Cordis, FP 6, Project Reference 508833, http://cordis.europa.eu/, zitiert im Juni 2007

[Cord07a] *„EGEE II“*
Project Fact Sheet, CORDIS, FP 6, Project Reference 31688, http://cordis.europa.eu/, zitiert im Juni 2007

[DANT06] *„GÉANT2 Brochure. 3rd Edition”*
DANTE, 2006, http://www.geant2.net/upload/pdf/PUB-06-156_GEANT2_Brochure-FINAL.pdf, zitiert am 9.1.2008

[Desp01] F. Desprez, M. Quinson, F. Suter:
„Dynamic Performance Forecasting for Network-Enabled Servers in a Heterogeneous Environment“
International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2001), 25 – 28 June, 2001, http://citeseer.ist.psu.edu/article/quinson02dynamic.html

[DFN07] DFN – Deutsches Forschungsnetz, http://www.dfn.de/content/, zitiert 2007

[Dind98] Peter A. Dinda:
„Load Trace Archive“
Web-Seite mit Lastinformationen zu einer Gruppe von Rechnern,
http://www.cs.northwestern.edu/~pdinda/LoadTraces/

[Dind99] Peter A. Dinda, David R. O'Hallaron:
„An Extensible Toolkit for Resource Prediction in Distributed Systems“
Techical Report CMU-CS-99-138, School of Computer Science, Carnegie Mellon University, July 1999, citeseer.ist.psu.edu/dinda99extensible.html

[Dind00] Peter A. Dinda, David R. O'Hallaron:
„Host Load Prediction Using Linear Models“
Cluster Computing, volume 3, no. 4, pp. 265 - 280, 2000, citeseer.ist.psu.edu/dinda00host.html

[Ecke80] Thomas Eckes:
„Clusteranalysen“
Verlag W. Kohlhammer GmbH, 1980

[Ecke00] H.-F. Eckey, R. Kosfeld, C. Dreger:
„Statistik. Grundlagen – Methoden – Beispiele“
2., überarbeitete Auflage, Betriebswirtschaftlicher Verlag, Wiesbaden, 2000

[EEX05] *„EEX-Spotmarktkonzept“*
European Energy Exchange, Dokumentversion 12A, 1. August 2005, http://www.eex.de/

[EGEE08a] *„EGEE Activities Explained“*
EGEE, www.eu-egee.org, http://public.eu-egee.org/files/EGEE_ALL_ACTIVITIES_EXPLAINED.pdf, zitiert am 9.1.2008

[EGEE08b] *„NA1 – Projekt Management“*
EGEE, www.eu-egee.org, http://public.eu-egee.org/files/NA1AusNov05.pdf, zitiert am 9.1.2008

[EGEE08c] *„NA2 – Informationsvermittlung und Öffentlichkeitsarbeit“*
EGEE, www.eu-egee.org, http://public.eu-egee.org/files/NA2AusNov05.pdf, zitiert am 9.1.2008

[EGEE08d] *„NA3 – Anwenderschulung und -einführung“*
EGEE, www.eu-egee.org, http://public.eu-egee.org/files/NA3AusNov05.pdf, zitiert am 9.1.2008

[EGEE08e] *„NA4 – Suche nach und Unterstützung von Anwendungen“*
EGEE, www.eu-egee.org, http://public.eu-egee.org/files/NA4AusNov05.pdf, zitiert am 9.1.2008

[EGEE08f] *„NA5 – Grundsatzfragen und Internationale Zusammenarbeit“*
EGEE, www.eu-egee.org, http://public.eu-egee.org/files/NA5AusNov05.pdf, zitiert am 9.1.2008

[Euro06] *„Gas and electricity market statistics. Data 1990 – 2006“*
Eurostat, European Communities, 2006, ISBN 92-79-02837-5

[Feit05] Dror Feitelson:
„Parallel Workloads Archive“
Web-Seite mit Last-Informationen für Parallelrechner,
http://www.cs.huji.ac.il/labs/parallel/workload/index.html

[Foga04] K. Fogarty:
„Grid Computing: Too Good to Be True?“
Baseline Magazine, Juni 2004, www.baselinemag.org, zitiert am 24.11.2007

[Fost98] Ian Foster, Carl Kesselman:
„Computational Grids“
aus: “The Grid: Blueprint for a Future Computing Infrastructure”, Ian Foster and Carl Kesselman (Eds.), Morgan Kaufmann Publishers, 1998

[Gagl05] F. Gagliardi, B. Jones, F. Grey, M.-E. Bégin, M. Heikkurinen:
„Building and infrastructure for scientific Grid computing: status and goals of the EGEE project“
Philosophical Transactions of the Royal Society, Volume 363, Number 1833 / 15.8.2005

[Gies98] Jürgen Giesecke, Emil Mosonyi:
„Wasserkraftanlagen. Planung, Bau und Betrieb“
Springer-Verlag, Berlin, 1998

[Good99] Phillip I. Good:
„Resampling Methods. A Practical Guide to Data Analysis“
Birkhäuser Boston, 1999

[Good05] Phillip I. Good:
„Permutation, Parametric and Bootstrap Tests of Hypotheses“
3rd Edition, Springer Series in Statistics, 2005,
http://www.springerlink.com/content/r01546/?p=6e08039e6b284c7e925673ca9e937736&pi=0

[Gray03] J. Gray:
„Distributed Computing Economics“
Technical Report, MSR-TR-2003-24, Microsoft Research, März 2003

[Grid07] GridKa, http://grid.fzk.de/, zitiert im Juni 2007

[Hart99] Joachim Hartung, Bärbel Elpelt, Karl-Heinz Klösener:
„Statistik. Lehr- und Handbuch der angewandten Statistik“
R. Oldenbourg Verlag, München, 1999

[Hetz06] *„Hetzner nimmt neues Rechenzentrum in Nürnberg in Betrieb“*
Presseinformation # 0706, Hetzner, November 2006, http://www.hetzner.de/presse_november06_b.html

[Hips] Ron Hipschman:
„How SETI@home works“
http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_1.html, zitiert am 30.6.2008

[HP04] *„Servicing the Animation Industry. HP's Utility Rendering Service Provides On-Demand Computing Resources“*
Hewlett-Packard, 2004, http://www.hpl.hp.com/SE3D/whitepaper-urs.pdf, zitiert am 5. Juli 2007

[HP07] *„Data center cooling strategies”*
Hewlett-Packard, 2007,
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01153741/c01153741.pdf

[Hübn04] T. Hübner:
„Energieverbrauch aktueller Prozessoren”
ComputerBase, 22.5.2004,
http://www.computerbase.de/artikel/hardware/prozessoren/2004/bericht_energieverbrauch_prozessoren/,
zitiert am 3.1.2008

[Idea06] *„Preisvergleich bei Idealo.de“*
idealo Internet GmbH, http://www.idealo.de/preisvergleich/ProductCategory/5432.html, zitiert am 23.7.2006

[Idea07] *„Preisvergleich bei Idealo.de“*
Idealo Internet GmbH, http://www.idealo.de/preisvergleich/SubProductCategory/1342.html, zitiert am 29.11.2007

[Inte03] *„Distributed Desktop Grid, PC Refresh Help Novartis Enhance Innovation“*
White Paper, Intel, 2003, http://www.intel.com/ca/business/casestudies/pdf/novartis.pdf, zitiert am 21.7.2006

[Jone05] B. Jones:
„Plans for EGEE II“
EGEE, 20.6.2005, http://lcg.web.cern.ch/LCG/peb/documents/pob_20jun05/egeeII_LCG_POB_june20.ppt,
zitiert am 14.6.2007

[Kauf84] Heinz Kaufmann, Heinz Pape:
„Clusteranalyse“
aus: "Multivariate statistische Verfahren", Herausgeber: Ludwig Fahrmeir und Alfred Hamerle, Walter de Gruyter, 1984

[Kalt95] Martin Kaltschmitt, Manfred Fischedick:
„Wind- und Solarstrom im Kraftwerksverbund. Möglichkeiten und Grenzen“
Institu für Energiewirtschaft und Rationelle Energieanwendung (IER) an der Universität Stuttgart, 1. Auflage, Müller Verlag, Heidelberg, 1995

[Knue05] *„Effectively Transfer Heat With Water“*
Knürr, 2005, http://www.water-cooled-cpu.com/pdf-zip/en/Thermalmanagment.pdf, zitiert am 13.6.2007

[Kond05] Derrick Kondo:
„Scheduling Task Parallel Applications For Rapid Turnaround on Desktop Grids“
Dissertation, University of California, San Diego, 2005, http://citeseer.ist.psu.edu/kondo05scheduling.html

[Kozi01] Kozierok, C. M.:
„The PC Guide“
http://www.PCGuide.com, Site Version 2.2.0 (17. April 2001), zitiert im Januar 2007

[Litz88] Michael J. Litzkow, Miron Livny, Matt W. Mutka:
„Condor – A Hunter of Idle Workstations“
8th International Conference of Distributed Computing Systems, 104–111, San Jose, California, Juni 1988, http://www.eas.asu.edu/~cse591os/readings.htm

[Long95] Darrel Long, Andrew Muir, Richard Golding:
„A longitudinal survey of Internet host reliability“
Technical Report UCSC-CRL-95-16, Symposium on Reliable Distributed Systems, 1995, http://citeseer.ist.psu.edu/91773.html

[Loom07] C. Loomis:
„NA4: Application Identification and Support“
EGEE-II 1st EU Review (CERN), Mai 2007, https://edms.cern.ch/file/844260/3/EGEEII_ReviewI_NA4.ppt, zitiert am 20.11.2007

[Löw05] Edgar Löw:
„Rechnungslegung für Banken nach IFRS. Praxisorientierte Einzeldarstellungen“
Gabler Verlag, 2., vollständig überarbeitete und erweiterte Auflage, Juni 2005

[Mabs05] Mabst, P.:
„Cooling Today“
2005, www.datapcsysteme.cc/kuehlersysteme/50223094fe01cf3ae

[Mann01] P. S. Mann:
„Introductory Statistics“
Fourth Edition, John Wiley & Sons, 2001

[Math07] Jeff Mathers:
„How Johnson & Johnson Took Its Grid Global“
GRIDtoday, 19. März 2007, http://www.gridtoday.com/grid/1320910.html, zitiert am 22.6.2007

[Marc04] Adam Marczyk:
„Genetic Algorithms and Evolutionary Computation“
April 2004, http://www.talkorigins.org/faqs/genalg/genalg.html

[MSI06] MSI, 7.7.2004, http://www.msi-technology.de/news/mitteilung.php?Typ=1&News_id=129, zitiert am 23.7.2006

[Mutk92] Matt W. Mutka:
„An Examination of Strategies for Estimating Capacity to Share Among Private Workstations“
ACM SIGSMALL/PC Notes, Volume 18, Issue 1-2, pp. 53 - 61, 1992, http://portal.acm.org/citation.cfm?id=134334&coll=portal&dl=ACM

[NeuZ03] *„Schlafende PC mutieren zum Supercomputer”*
Neue Zürcher Zeitung, 7. November 2003, http://nzz.de/netzstoff/2003/2003.11.07-em-article97JEJ.html

[Nimm99] Srinivas Nimmagadda, Joshua LeVasseur, Rumi Zahir:
„High-End Workstation Compute Farms Using Windows NT“
3rd USENIX Windows NT Symposium, Seattle, Washington, July 12-13, 1999

[Nurm03] Daniel Nurmi, John Brevik, Rich Wolski:
„Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments“
Technical Report CS2003-28, U.C. Santa Barbara, Computer Science Department, Oktober 2003, http://citeseer.ist.psu.edu/nurmi03modeling.html

[Nurm04] Daniel Nurmi, Rich Wolski, John Brevik:
„Model-Based Checkpoint Scheduling for Volatile Resource Environments“
University of California, Santa Barbara, Computer Science, Tech. Rep. TR-2004-25, Nov. 6 2004, http://www.cs.ucsb.edu/research/tech_reports/reports/2004-25.pdf

[Olss98] Marita Olsson:
„The EMpht-programme“
Department of Mathematics, Chalmers University of Technology, Göteborg, June 1998, http://citeseer.ist.psu.edu/olsson98emphtprogramme.html

[Patt07] M. K. Patterson, D. G. Costello, P. F. Grimm, M. Loeffler:
„Data center TCO; a comparison of high-density and low-density spaces”
THERMES 2007, Santa Fe, 2007, http://www.intel.com/technology/eep/datacenter.pdf

[PCWe07] *„Amazon offers online storage in Europe”*
PC-Welt, 6.11.2007, http://www.pcwelt.de/it-profi/englishnews/Internet/99196/index2.html, zitiert am 22.1.2008

[Pech05] Prof. Dr. Hand Pechtl:
„Einführung in die Betriebswirtschaftslehre”
Lehrstuhl Marketing, Ernst-Moritz-Arndt-Universität, Greifswald, 2004/2005

[Pest98] W. R. Pestman:
“Mathematical Statistics: An Introduction”
Walter de Gruyter, Berlin, 1998

[Pixa04] Pixar Animation Studios:
„How We Make A Movie. Pixar's Animation Process“
http://www.pixar.com/howwedoit/index.html, Download: 7.6.2004

[Plas06] Pawel Plaszczak, Richard Wellner, Jr.:
„Grid computing: the savvy manager's guide“
Morgan Kaufmann Publishers, 2006

[Port06] Portwell (UK), http://www.portwell.co.uk/html/motherboards/gcs15.htm, zitiert am 23.7.2006

[Prei06] Preis.de, comparado GmbH, http://www.preis.de/, zitiert am 23.7.2006

[Prei06a] Preissuchmaschine.de, metashopper Europe GmbH, http://www.preissuchmaschine.de/, zitiert am 23.7.2006

[Prei07] *„Computer & Zubehör Preisvergleich, Testberichte und Community“*
Preisvergleich.de GmbH, http://www.preisvergleich.de/category/index/id/1739, zitiert am 29.11.2007

[Puru04] T. Purusothaman, S. Annadurai, H. Vijay Ganesh, C. T. Chockalingam, B. Uthra Kumar:
„Dynamically Scalable, Heterogeneous and Generic Architecture for a Grid of Workstations“
Journal of Grid Computing, S. 239 – 246, Vol. 2, Number 3, September 2004,
www.kluweronline.com/issn/1570-7873/

[Radi02] *„MTBF Calculation for Video Processor“*
Radiant, 3.12.2002, http://www.dvmd.com/downloads/MTBF.pdf, zitiert am 23.7.2006

[Rica05] Aaron Ricadela:
„Slow Going On The Global Grid“
Information Week, 21. Februar 2005,
http://www.informationweek.com/story/showArticle.jhtml?articleID=60402106, zitiert am 4.1.2008

[Rinn03] Horst Rinne:
„Taschenbuch der Statistik”
Verlag Harri Deutsch GmbH, Frankfurt am Main, 2003

[Robi03] Franck Robin, Andrea Orzati, Esteban Moreno, Otte J. Homan, Werner Bächtold:
„Simulation and Evolutionary Optimization of Electron-Beam Lithography With Genetic and Simplex-Downhill Algorithms”
IEEE Transactions on Evolutionary Computation, Vol. 7, No. 1, February 2003,
http://www.uam.es/personal_pdi/ciencias/emoreno/journ_2003-02_ieeetec.pdf

[Röhr04] Armin Röhrl, Stefan Schmiedl:
„Let's Grid”
Linux Enterprise, August 2004, http://www.linuxenterprise.de/itr/online_artikel/psecom,id,576,nodeid,9.html

[RWE91] *„Erzeugung der elektrischen Energie”*
Lehrerfachheft, RWE Energie Aktiengesellschaft, Abt. Öffentlichkeitsarbeit und Information, 3. Auflage 1991

[Sarm99] Luis F. G. Sarmenta, Satoshi Hirano:
„Bayanihan: Building and Studying Web-Based Volunteer Computing Systems Using Java“
Future Generation Computer Systems, 15(5/6), 1999, http://citeseer.ist.psu.edu/330739.html, zitiert am 6.2.2008

[Schä07] Bernd Schäppi, Frank Bellosa, Bernhard Przywara, Thomas Bogner, Silvio Weeren, Alain Anglade:
„Energy efficient servers in Europe. Energy consumption, saving potentials, market barriers and measures. Part II“
Draft-Version, The Efficient Servers Consortium, October 2007,
http://www.efficient-server.eu/fileadmin/docs/reports/E-Server-Report_PartII.pdf

[Sche00] Bruce Schechter:
„Putting a Darwinian Spin on the Diesel Engine“
The New Yourk Times, September 19, 2000,
http://query.nytimes.com/gst/fullpage.html?res=9C01E7DD173BF93AA2575AC0A9669C8B63

[Schl99] Rainer Schlittgen, Bernd H. J. Streitberg:
„Zeitreihenanalyse“
8., überarbeitete Auflage, Oldenbourg Verlag, 1999

[Schn06] Marcel Schneider:
„Grid-Computing - Science oder Fiction?“
Wirtschaftsinformatik 48(2006) 1, 75 - 76, 2006, http://www.springerlink.com/content/85370352k64k8206/

[Schr07] Bianca Schroeder, Garth A. Gibson:
„Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?“
5th Usenix Conference on File and Storage Technologies (FAST 2007), 2007,
http://www.cs.cmu.edu/~bianca/fast07.pdf

[Schw05] Peter Schwarz:
„Organisation in Nonprofit-Organisationen: Grundlagen, Strukturen“
Haupt Verlag, 2005

[Sess03] Werner Sesselmeier:
„Zur Reform des Sozialstaats – Mögliche Entwicklungstrends und Konsequenzen für die sozial Sicherung“
Der Sozialstaat in der Diskussion, Heft 4/2003

[Silv05] J. Silverstein:
„Cure Cancer With Your Computer“
ABC News, 16.12.2005, http://abcnews.go.com/Technology/story?id=1410682, zitiert am 22.7.2006

[Skaw02] K. Z. Skawinski:
„IT & SETI: The Role of Computer Technology in the Search for Extraterrestrial Intelligence”
California Computer News, Juli 2002, http://www.ccnmag.com/story.php?id=146, zitiert am 18.1.2007

[Smit03] Chris Smith:
„Computational Grids Meet The Database”
Technical Whitepaper, Platform Computing, 2003, http://www.platform.com/resources/whitepapers/

[Stew05] *„Informationen zum Strompreis“*
Steweag-Steg GmbH, 2005, http://www.selectstrom.at/content/view/full/232

[Sun02] *„Sun Cluster Grid Architecture“*
Technical Whitepaper, Sun Microsystems, Mai 2002,
http://www.sun.com/software/gridware/whitepapers.xml, zitiert am 4.1.2008

[Sun06] Sun Store U.S., http://store.sun.com/, zitiert am 20.7.2006

[Sun06a] *„Is the $1/CPU-hr Sun Grid Right for You?“*
Sun System News, 30.11.2006, Article #17349, Volume 105, Issue 5,
http://www.sun.com/emrkt/expertexchange/transcript-SEE-20061207.pdf, zitiert am 25.6.2007

[Sun06b] *„Sun Grid Compute Utility Quick Start Guide“*
Sun Microsystems, 2006, http://www.sun.com/service/sungrid/SunGridQuickStart.pdf, zitiert am 10.1.2008

[Ther05] Niklas Therning, Lars Bengtsson:
„Jalapeno – Decentralized Grid Computing using Peer-to-Peer Technology“
2nc conference on Computing frontieres, Ischia, Italy, 2005

[Turn06] W. P. Turner, J. H. Seader:
„Dollars per kW plus Dollars per Square Foot Are a Better Data Center Cost Model than Dollars per Square Foot Alone“
White Paper, The Uptime Institute, 2006,
http://www.datacenterdynamics.com/Media/DocumentLibrary/TUI808DollarsPerkW_WP.pdf

[Turn06a] W. Pitt Turner, John H. Seader, Kenneth G. Brill:
„Tier Classifications Define Site Infrastructure Performance“
Whitepaper, The Uptime Institute, 2006,
http://dev.tierfour.com/colocation/knowledge%20base/TUI705CTierClassification_WP.pdf

[Veld98] David A. Van Veldhuizen, Brian S. Sandlin, Robert E. Marmelstein, Gary B. Lamont, Andrew J. Terzuoli:
„Finding Improved Wire-Antenna Geometries with Genetic Algorithms”
IEEE International Conference on Evolutionary Computation, Piscataway, New Jersey, 1998,
citeseer.ist.psu.edu/vanveldhuizen97finding.html

[Wasz01] B. Waszkowycz, T. D. J. Perkins, R. A. Sykes, J. Li:
„Large-scale virtual screening for discovering leads in the postgenomic era“
IBM Systems Journal, Volume 40, Number 2, 2001, ("Deep computing for the life sciences"),
http://www.research.ibm.com/journal/sj/402/waszkowycz.html

[Weer04] Silvio Weeren:
„Green IT Procurement: Ways to Energy Efficient Computing“
IBM, 2004, http://ec.europa.eu/environment/gpp/pdf/weeren.pdf, zitiert am 28.11.2007

[Wies98] Dirk Wiesmann, Ulrich Hammel, Thomas Bäck:
„Robust Design of Multilayer Optical Coatings by Means of Evolution Strategies“
IEEE Transactions on Evolutionary Computation, Vol. 2, No. 4, 1998,
citeseer.ist.psu.edu/wiesmann98robust.html

[Wiki_a] *“Verteilungsfunktion“*
In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 8. Mai 2008, 20:26 UTC. URL:
http://de.wikipedia.org/w/index.php?title=Verteilungsfunktion&oldid=45814591 (Abgerufen: 10. Juli 2008, 07:09 UTC)

[Wols98] Rich Wolski:
„Dynamically Forecasting Network Performance Using the Network Weather Service“
Cluster Computing, January 1998, Volume 1, Issue 1, http://citeseer.nj.nec.com/wolski98dynamically.html

[Wols99] Rich Wolski, Neil T. Spring, Jim Hayes:
„The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing“
Future Generation Computer Systems, vol. 15, no. 5-6, pp. 757-768, 1999,
citeseer.ist.psu.edu/wolski98network.html

[Wols00] Rich Wolski, Neil Spring, Jim Hayes:
„Predicting the CPU Availability of Time-shared Unix Systems on the Computational Grid“
Cluster Computing, vol. 3, no. 4, pp. 293-301, 2000, citeseer.ist.psu.edu/wolski98predicting.html

[Wyck98] Peter Wyckoff, Theodore Johnson, Karpjoo Jeong:
„Finding Idle Periods on Networks of Workstations“
Technical Report: TR1998-761, New York University New York, NY, USA, 1998,
http://citeseer.ist.psu.edu/wyckoff98finding.html

[Yang03] Lingyun Yang, Ian Foster, Jennifer M. Schopf:
„Homeostatic and Tendency-based CPU Load Predictions“
International Parallel and Distributed Processing Symposium (IPDPS2003), 2003,
citeseer.ist.psu.edu/579551.html

[Zhou05] Dayi Zhou, Virginia Lo:
„Wave Scheduler: Scheduling for Faster Turnaround Time in Peer-based Desktop Grid Systems“
Job Scheduling Strategies for Parallel Processing, 11th International Workshop, JSSPP 2005, Cambridge, MA, USA, June 19, 2005, LNCS 3834, ccof.cs.uoregon.edu/papers/wave.pdf