A MODEL REDUCTION WEB ENVIRONMENT FOR VERY LARGE LINEAR DYNAMICAL SYSTEMS*

PETER BENNER†, RAFAEL MAYO‡, ENRIQUE S. QUINTANA-ORTí‡ , AND GREGORIO QUINTANA-ORTí‡

Abstract. We describe a prototype web service for model reduction of very large-scale linear systems, with dimension in the order of millions of states, that includes a user-friendly interface designed so that the computation can be easily performed via the HTTP protocol. Access via a web browser isolates the user of the service from the complexities of installing and using the parallel model reduction codes and the maintenance of the hardware. In case the routines are found to be appropriate for the problem the user needs to solve, the library can be then downloaded and installed on the user's own computing resources.

This paper illustrates the major issues of the access procedure by means of graphical examples, and describes the structure and implementation of the remote model reduction service. The service is offered in a cluster of Linux machines.

Key words. Linear-time invariant systems, model reduction, web execution environment, web-computing, clusters.

1. Introduction. The development of the Internet and its popularization via the world wide web (www) is producing a tremendous impact in our society. The www is nowadays employed to consult databases, retrieve documents, submit jobs to remote systems, and access a large number of services which is continuously growing. The extension of these services has recently reached the area of scientific computing in projects like NetSolve [4], LAPK (http://lapk3.hsc.usc.edu/lapk), the SLICOT Web Environment [16], NPACI (https://hotpage.npaci.edu), CACTUS (http://www.cactuscode.org), and EOT-PACI (http://www.eot.org), among others. Many of these efforts can be generically classified as problem-solving environments or computational science portals [18]. A related, extensive survey of applications of web-computing and grid-computing can be consulted, e.g., in [10].

In this paper we further explore the www and the HTTP protocol as a means of offering an enhanced service for scientific computing to a certain cluster of the scientific and engineering community. Specifically, we offer a computational service for model reduction of large-scale linear systems, with potentially even millions of states, on a parallel cluster. The idea here is to provide an easy way for remote users to interact with the model reduction codes via the www so that the service becomes a demonstrator of the capabilities of the parallel model reduction library. Note that a usual case is that of a user with a problem that needs to be solved, who has to search for a likely solver (in the form of a code or a library), read the manuals, install the application (after having solved compatibility problems with the compiler, operating system, or even architecture), write the interface programs, and run it just to then discover that the code is not the appropriate one to solve his/her problem.

Thus, the main advantage of providing this service via the www is that the end-user remains completely isolated from the complexities of the software and the hardware system, and the maintenance of both. Then, in case the routines are evaluated as satisfactory by the user, the library itself can be downloaded and installed on any (parallel) architecture

*Peter Benner was supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin. Rafael Mayo, Enrique S. Quintana-Ortí, and Gregorio Quintana-Ortí were supported by the project CTDIA/2002/122 of the *Generalitat Valenciana* and the project PI-1B2001-14 of the *Fundació Caixa-Castelló/Bancaixa*.

†Institut für Mathematik, MA 4-5, Technische Universität Berlin, 10623 Berlin, Germany, benner@math.tu-berlin.de.

‡Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I, 12.071 Castellón Spain, {mayo,quintana,gquintan}@icc.uji.es.

with MPI and the appropriate parallel numerical libraries (basically, BLAS, LAPACK, and ScaLAPACK [2, 9]).

The problem considered here, model reduction, is of fundamental importance in many modeling and control applications involving continuous linear time-invariant (LTI) dynamical systems in state-space form:

$$\begin{array}{rcll} \dot{x}(t) & = & Ax(t) + Bu(t), & t > 0, \qquad x(0) = x^0, \\ y(t) & = & Cx(t) + Du(t), & t \geq 0. \end{array} \tag{1.1}$$

Here, $A \in \mathbb{R}^{n \times n}$ is the state matrix, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, $D \in \mathbb{R}^{p \times m}$, $x^0 \in \mathbb{R}^n$ is the initial state of the system, and the number of states, n, is known as the state-space dimension (or the order) of the system. The transfer function matrix (TFM) associated with the system is defined by $G(s) = C(sI - A)^{-1}B + D$. Hereafter, we assume that A is stable, i.e., the spectrum of A is contained in the open left half plane. This implies that the system is stable, i.e., has no poles in the closed right half plane.

The model reduction problem consists in finding a reduced-order LTI system,

$$\begin{array}{rcll} \dot{\hat{x}}(t) & = & \hat{A}\hat{x}(t) + \hat{B}\hat{u}(t), & t > 0, \qquad \hat{x}(0) = \hat{x}^0, \\ \hat{y}(t) & = & \hat{C}\hat{x}(t) + \hat{D}\hat{u}(t), & t \geq 0, \end{array} \tag{1.2}$$

of order r, $r \ll n$, and associated TFM $\hat{G}(s) = \hat{C}(sI - \hat{A})^{-1}\hat{B} + \hat{D}$ which approximates $G(s)$. The reduced-order model can then replace the original high-order system in subsequent calculations including, e.g., the design of the controller [22], thus saving valuable hardware resources.

Model reduction of large-scale stable linear systems (n in the thousands) arises, among others, in control of large flexible mechanical structures or large power systems, and weather simulation (storm surge forecast and perturbation evolution); see, e.g., [11, 12, 13, 17, 23]. Very large-scale linear systems, with state-space dimension n as large as 10^6, are common in circuit simulation and VLSI design, air quality simulation, etc. [3]. These systems often present a sparse (unstructured or banded) state matrix A.

In general, model reduction methods for LTI systems with dense state matrices have a computational cost of $\mathcal{O}(n^3)$ floating-point arithmetic operations (flops) and require storage for $\mathcal{O}(n^2)$ numbers. While current desktop computers provide enough computational power to reduce models of order n in the hundreds using libraries like SLICOT[1] or the MATLAB control-related toolboxes, large-scale applications clearly require the use of advanced computing techniques. The approach proposed in PLiCMR[2] [7, 8] is to employ dense model reduction methods on parallel computers, thus allowing for larger problems to be reduced. Nevertheless, none of these approaches exploit nor preserve the sparsity of the state matrix and therefore the dimensions of the largest systems they can deal with is quite limited.

The model reduction approach considered in this paper is based on the so-called state-space truncation and requires, at a first stage, the solution of two large Lyapunov equations whose coefficient matrix is the state matrix of the system. These equations are solved via a low-rank iteration [20, 25] which only involves numerical computations such as the solution of linear systems and matrix-vector products. The success of these Lyapunov solvers relies in the use of efficient routines that exploit the sparsity of the coefficient matrix. Once the equations are solved, the reduced-order system is obtained using a slightly modified version

[1] Available from `http://www.win.tue.nl/niconet/NIC2/slicot.html`.
[2] Available from `http://spine.act.uji.es/~plicmr.html`.

of the balanced truncation (BT) method [7, 24], which only requires dense linear algebra operations on much smaller data matrices.

The paper is structured as follows. In Section 2 we briefly revisit the methods for model reduction of very large linear systems, including the square-root BT method and the low-rank solvers for the Lyapunov equations arising in state-space truncation. The usage, structure, and implementation details of the web service are introduced in Section 3. There graphical examples are employed to illustrate the procedure to access the service. Concluding remarks and possible future extensions of the service follow in the final Section.

2. Model Reduction of Very Large Linear Systems.

2.1. The square-root BT Method. Our routines are based on BT model reduction [21, 26, 27, 28]. BT belongs to the family of absolute error methods, which try to minimize $\|\Delta_a\|_\infty = \|G - \hat{G}\|_\infty$. Here, $\|G\|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-norm of a stable, rational matrix function defined as

$$\|G\|_\infty = \operatorname*{ess\,sup}_{\omega \in \mathbb{R}} \sigma_{\max}(G(\jmath\omega)), \tag{2.1}$$

where $\jmath := \sqrt{-1}$ and $\sigma_{\max}(M)$ denotes the largest singular value of the matrix M. In the continuous-time case, a reduced-order model computed by BT approximates the original one well at high frequencies ($\omega \to \infty$), with a perfect match at $\omega = \infty$.

BT methods are strongly related to the controllability Gramian W_c and the observability Gramian W_o of the system (1.1), given by the solutions of two "coupled" Lyapunov matrix equations:

$$AW_c + W_c A^T + BB^T = 0, \tag{2.2}$$

$$A^T W_o + W_o A + C^T C = 0. \tag{2.3}$$

As the coefficient matrix in the equations, A, is assumed to be stable, W_c and W_o are positive semidefinite and therefore can be factored as $W_c = S^T S$ and $W_o = R^T R$. The factors S and R can be the *Cholesky factors* (square and triangular) of the Gramians or *full-rank factors* (possibly rectangular and full).

Efficient Lyapunov solvers that exploit the sparsity of the coefficient matrix A are described in the next subsection.

Consider now the singular value decomposition (SVD) of the product

$$SR^T = U\Sigma V^T = [\,U_1\ U_2\,] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} [\,V_1\ V_2\,]^T, \tag{2.4}$$

where U and V are orthogonal matrices, and $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_\ell)$ is a diagonal matrix containing the singular values of SR^T. Here, $\ell = \max\{s, r\}$, where $S \in \mathbb{R}^{s\times n}$, $R \in \mathbb{R}^{r\times n}$. The singular values $\sigma_1, \ldots, \sigma_\ell$ are known as the *Hankel singular values* of the system. If $\sigma_r > \sigma_{r+1} = 0$, then r is the state-space dimension of a minimal realization of the system.

Next, a reduced system of order r is determined by partitioning the diagonal matrix Σ into $\Sigma_1 = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ and $\Sigma_2 = \operatorname{diag}(\sigma_{r+1}, \sigma_{r+2}, \ldots, \sigma_\ell)$, so that, hopefully, a large gap exists between σ_r and σ_{r+1}. A conformal partition of U and V is induced by the partitioning of Σ.

The square-root BT method determines then the reduced-order model of order r as

$$\begin{array}{rcl} \hat{A} &=& T_l A T_r, \\ \hat{B} &=& T_l B, \\ \hat{C} &=& C T_r, \quad \text{and} \\ \hat{D} &=& D, \end{array} \tag{2.5}$$

with the projection matrices T_l and T_r given by

$$\begin{aligned} T_l &= \Sigma_1^{-1/2} V_1^T R \quad \text{and} \\ T_r &= S^T U_1 \Sigma_1^{-1/2}. \end{aligned} \tag{2.6}$$

The realization of order r computed by the square-root BT algorithm satisfies the error bound

$$\|\Delta_a\|_\infty := \|G - \hat{G}\|_\infty \leq 2 \sum_{i=r+1}^{n} \sigma_i. \tag{2.7}$$

This allows an adaptive choice of the size of the reduced-order model if a given upper bound for the error is to be satisfied.

2.2. Low Rank Solution of Lyapunov Equations. In this subsection we revisit the Lyapunov solvers introduced in [20, 25]. These iterative algorithms benefit from the usual low-rank property of the right-hand side matrix to provide low-rank approximations to a Cholesky or full-rank factor of the solution matrix. These approximations can reliably substitute S and R in the computation of (2.4) and (2.6).

Specifically, given an "l–cyclic" set of (complex) shift parameters $\{p_1, p_2, \ldots\}$, $p_k = \alpha_k + \beta_k\, \jmath$, such that $p_k = p_{k+l}$, the cyclic *low-rank alternating direction implicit* (LR-ADI) iteration proposed in [25] can be reformulated as follows:

$$\begin{aligned} V_0 &= (A + p_1 I_n)^{-1} B, \\ S_0 &= \sqrt{-2\,\alpha_1}\, V_0, \\ V_{k+1} &= V_k - \delta_k (A + p_{k+1} I_n)^{-1} V_k, \\ S_{k+1} &= [S_k\ ,\ \gamma_k V_{k+1}]\,, \end{aligned} \tag{2.8}$$

where $\gamma_k = \sqrt{\alpha_{k+1}/\alpha_k}$, $\delta_k = p_{k+1} + \overline{p_k}$, $\overline{p_k}$ is the conjugate of p_k, and I_n denotes the identity matrix of order n. On convergence, after $k_{\max}$ iterations, a low-rank matrix $\tilde{S}$ of order $n \times k_{\max} m$ is computed such that $\tilde{S}\tilde{S}^T$ approximates $W_c = S^T S$. An analogous iteration provides a low-rank approximation $\tilde{R}$ of R. The use of direct linear system solvers [15] (based, e.g., on the LU or Cholesky factorization) here is appealing as the same coefficient matrix is involved in iterations k and $k + l$. Therefore, the computed factorization can be reused several times as long as sufficient workspace is available to store l (sparse) factorizations.

The performance of iteration (2.8) strongly depends on the selection of the shift parameters. In practice, the set $\{p_1, p_2, \ldots, p_l\}$ should be chosen so that it is closed under complex conjugation (i.e., if p_k is a selected shift parameter, so is $\overline{p_k}$). In case the eigenvalues of A are real, the optimal parameters can be computed explicitly [30, 31]. Otherwise, an optimal solution is not known. A heuristic procedure is proposed in [25] to compute parameters by approximating the largest eigenvalues of A and A^{-1}. The procedure is based on an Arnoldi iteration [19] involving A and A^{-1}. For further details on the convergence of the LR-ADI iteration and the properties of the heuristic selection procedure, see [25]. Preliminary results of a parallel Lyapunov equation solver based on the LR-ADI method are reported in [5].

It should be emphasized that, though mathematically equivalent, the methods just described for solving (2.2)-(2.3) and (2.4) significantly differ from standard methods used in the MATLAB toolboxes or SLICOT [29]. First, the proposed LR-ADI iteration for the solution of the coupled Lyapunov equation exploits the sparsity in the coefficient matrix. Besides, as we are using low-rank approximations to the full-rank or Cholesky factors, the computation of the SVD in (2.4) is usually much more efficient: instead of a computational cost of $\mathcal{O}(n^3)$ when using the Cholesky factors, this approach leads to an $\mathcal{O}(k_{\max}^2 \cdot m \cdot p \cdot n)$ cost

where, in model reduction, often $m, p \ll n$; see [7]. This latter advantage is shared by the routines in our dense parallel model reduction library PLiCMR [7, 8]. However, PLiCMR is not able to exploit the sparsity of the coefficient matrix.

2.3. Parallelization. The LR-ADI iteration and the computation of the acceleration shifts basically require linear algebra operations like matrix-vector products and the solution of linear systems. Our approach for dealing with these matrix operations is based on the use of parallel linear algebra libraries. Specifically, in case sparse matrices are involved, we propose to employ codes like SuperLU [14] or MUMPS [1] to compute LU-like factorizations of the matrices $A + p_k I_n$ and keep the factors if sufficient storage is available. In case the coefficient matrix is dense or banded we use the LU factorization routines in the parallel dense linear algebra library like ScaLAPACK [9]. Note that for full dense coefficient matrices the methods in PLiCMR (see [8] for a description of the library and [6] for a web environment similar to the one described here) are to be preferred as the Lyapunov equation solvers used there have quadratic convergence rate as opposed to (super-)linear convergence of the ADI method at a comparable cost of the remaining computational steps.

Figure 2.1 shows the relation between our library of methods for model reduction, SpaRed, and the underlying parallel linear algebra libraries. All the model reduction codes in the library are written using the C programming language following an approach that could be roughly classified as object-oriented.

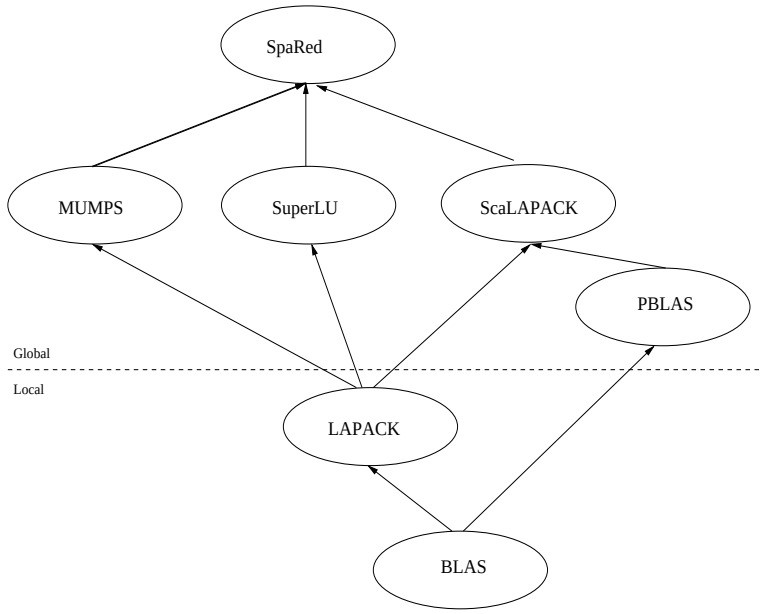

FIG. 2.1. *SpaRed and the underlying libraries.*

3. Web Service. In this section we describe both the access procedure and the structure and implementation of the prototype service for model reduction via the www. This service is provided on a cluster of computers, running the Linux (Suse 7.0) operating system, at the University Jaume I.

3.1. Accessing the Service. Our computational web service for model reduction is provided at http://spine.act.uji.es/~plicmr/SpaRedW3/SpaRedW3.html. On entry (see Figure 3.1), the user is offered the possibility of registering to access the service, perform model

reduction on a given system, check the status of submitted jobs, obtain information on the cluster load, etc.

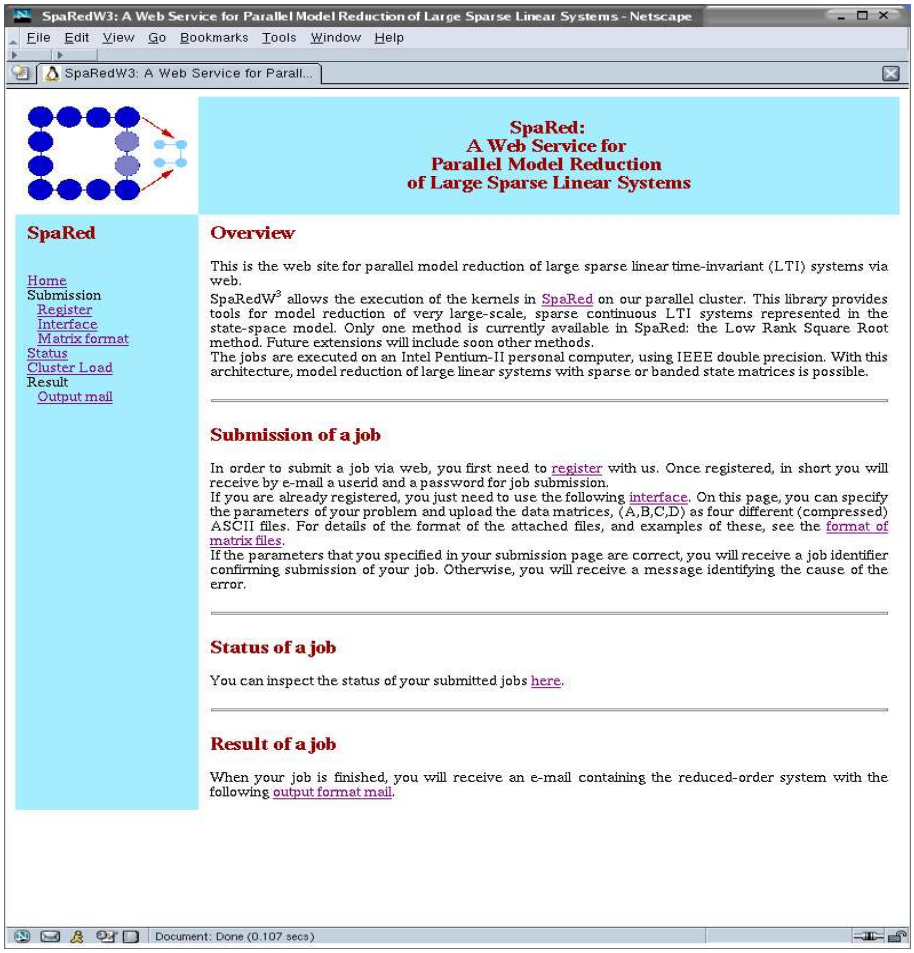

FIG. 3.1. *Access to the parallel model reduction service.*

The actual access to the service is performed through the job submission form (see Figure 3.2). In this form the user is requested to specify the parameters of the job, which can be grouped as follows:

- Validation of access: An identifier (parameter 1 in the job submission form or, for short, P1) and a password (P2), both provided via e-mail on registration, allow identification of the responsible of the requests. This identification is merely for statistical purposes as no real encryption of these data is performed.
- System parameters: The model dimensions, n, m, and p (P6, P7, and P8, respectively), and the type of matrix entries (P15), real or complex; also, the class of state matrix (P14) which, in the current implementation of the service, can be dense, symmetric/unsymmetric banded, or sparse.
- Reduced-order system parameters: Here the user is given the choice (P5) to either select a fixed order r for the reduced system (P9), or allow the order to be automatically determined using tolerance thresholds (P10 and P11).

– Numerical methods: Only the low rank square-root method (P3), a variant of the BT method that solves the Lyapunov equations using the LR-ADI iteration, is currently available. Future extensions will also provide the possibility of selecting the solver for the linear systems of equations, depending on the properties of the state matrix. Notice that this is specially interesting when sparse matrices are involved, as the performance of the model reduction methods strongly depends on the linear system solver that is selected [15].
– Execution environment: The recommended number of nodes of the cluster to participate in the execution of the job (P12), and the e-mail address (P16) where the results should be returned to.
– Data: The files with the system matrices and the compression tool (P13) employed for their preprocessing.

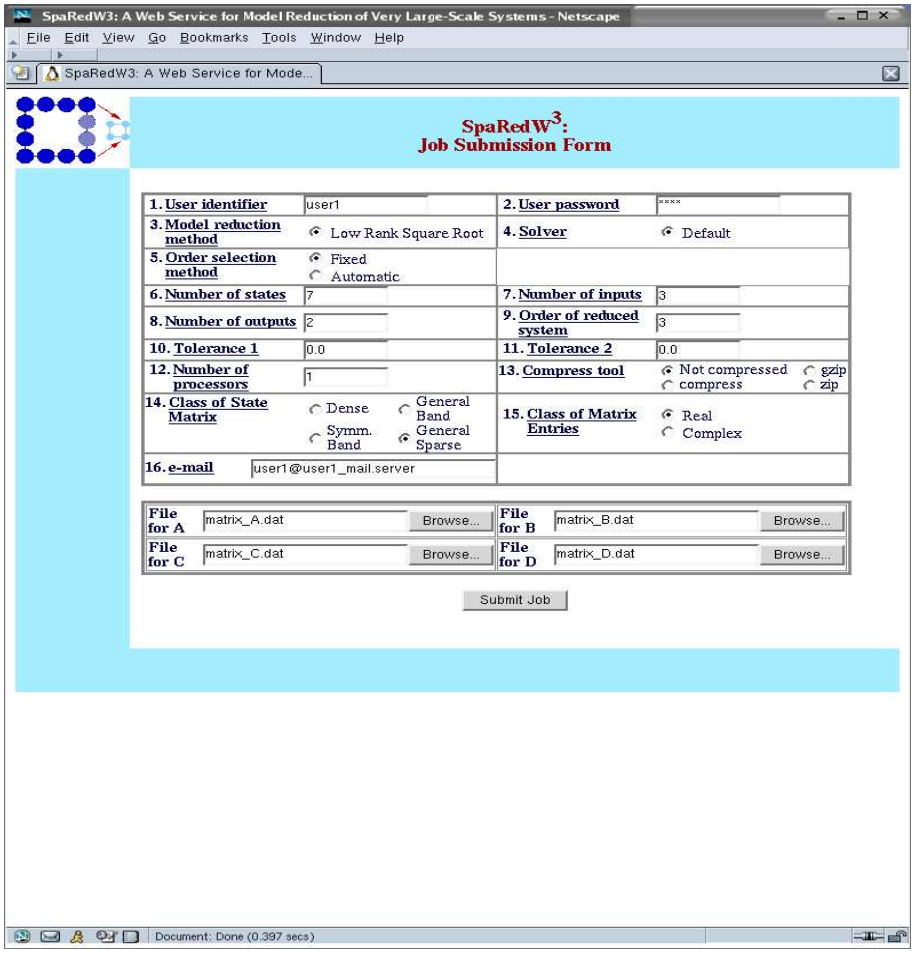

FIG. 3.2. *Job submission form.*

The same form can be used to upload the data matrices from the local file system. The structure of the file containing the data depends on the class of the matrix. Thus, column-major order is compulsory for all but the sparse matrices; files for banded matrices only include the diagonals within the band; files for symmetric matrices only contain the upper

triangular part; and sparse matrices are stored in compressed column (or Harwell-Boeing) format [15]. As an example, in Figure 3.3 we show the contents of a file containing a square matrix of order $n = 6$ with $nnz = 4$ non-zero entries in sparse format:

$$\begin{bmatrix} -0.42 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.36 & -0.58 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1}$$

In particular, the first line of the file specifies the value of *nnz*. This is followed by the matrix stored using the compressed column format: the $n + 1$ entries of vector `colptr`, the *nnz* entries of vector `rowind`, and the *nnz* entries of the matrix. The indices of both vectors start at zero following the convention of the C programming language.

```
   4
   0
   1
   2
   2
   4
   4
   4
   0
   1
   3
   3
 -4.2e-01
  1.0e-01
 -3.6e-01
 -5.8e-01
```

FIG. 3.3. *Contents of a file specifying the matrix in (3.1) in sparse format.*

```
1.  JOB ID. = 00011
    (METHOD='LRSR', SOLVER='default',
     ORDSEL='F', N=7, M=3, P=2, NR=3,
     TOL1=0.0, TOL2=0.0,
     NP=1, COMPRESS='NO')
2.  NR = 3
3.  NHSV = 3
4.  NP = 1
5.  TIME = 0.3507E+1
6.  INFO = 0
```

FIG. 3.4. *Text of an e-mail with the results of a job submission.*

In order to reduce the size of the data files that need to be transfered, compressed data files using several standard compression tools are accepted (e.g., `zip` and `gzip`).

The results of the execution are returned via e-mail, with the matrices of the reduced system as attached files (compressed with the same tool that was used on the data files). The text of the e-mail contains a few lines with the following data:

– The job identifier that was assigned to the user's job after submission. This parameter is followed by a compact description of the parameters of the job.

- The order of the resulting reduced-order system.
- The number of computed Hankel singular values.
- The number of processors that were used to execute the job.
- The wall-time required to execute the job.
- The info result (or status) from the execution of the routine.

Figure 3.4 shows an example of the text of the e-mail that is returned to the user after the job is finished.

Our model reduction service is connectionless in that any data that is submitted to the system is only stored while the reduction of the system is being performed and until the response is returned to the user. A connection-oriented (or session-oriented) service presents some advantages: for instance, the system could keep the user's data, allowing thus to perform interactive model reduction more efficiently (the data only needs to be sent once and remains stored in the system as long as the session is open). This is the approach adopted, e.g., in [16]. However, this translates much of the burden of the client-server (user-web service) interaction to the server side, apart from leading to much more complex failure recovery. Therefore, we prefer the connectionless approach, as the goal of the web service is still to serve as a rapid testbed of our parallel model reduction routines.

3.2. Structure and Implementation. The model reduction web server is composed of several PHP programs and the PBS (portable batch system) job control system. In the previous implementation, the service was accessed via a set of CGI programs, written in a shell script programming language. In the current version the CGI programs have been replaced by PHP codes. The web server includes a management system so that job requests are processed using no external procedures, as was necessary in the CGI-based variant.

Thus, the job submission form includes PHP code that is directly interpreted by the web server. The code is in charge of validating the data of the request and, in case the request is correct, create a pair of files necessary for job submission: a text file with the definition of the job parameters, and a submission shell-script specifying the number of nodes to participate in the execution and the path of executable program and data files. Next, the job is submitted to the job management system that is controlled by PBS.

From then on it is the responsibility of PBS to monitor the execution of the job. This queuing system consists of a server process which is executed in the cluster server and a client process running in each node of the cluster. The scheduler determines when a job begins its execution in the cluster using a FIFO strategy. The number of actual jobs that can be simultaneously in execution in the cluster is limited by the number of available nodes, due to the exclusive use application processes make of the Myrinet network interface cards. The client processes are in charge of receiving and executing job requests and also keep the PBS server informed of the current status of the node (basically, down, idle, or busy),

The submission shell-script maintains information on the actions that are to be performed in response to the user request when the job is finished. This code is implemented in PHP and is responsible for locating the results, composing an e-mail with those data, and sending back the response. Both the parameters of the job submission (except for the data matrices) and the results of the execution are sent as part of this message.

Figure 3.5 summarizes the actors and actions of the web service.

4. Concluding Remarks and Future Work. We have described a web service for parallel model reduction of very large linear systems. By providing a remote access, the end-user is given a computational tool to reduce large-scale systems and/or to test the numerical reliability of the reduced-order models produced via these methods. Moreover, in case the methods are found to be appropriate for the problems to be solved, the parallel library can

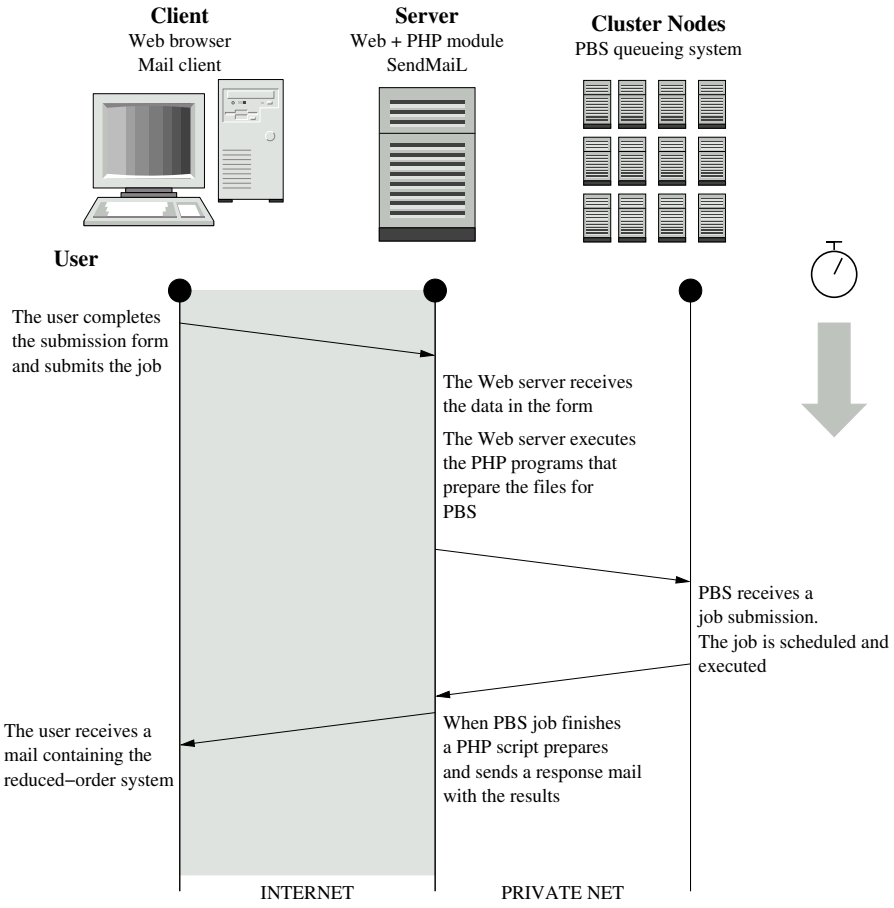

FIG. 3.5. *Actors and actions involved in processing a service request for model reduction.*

be then downloaded and installed on any parallel platform with the appropriate underlying libraries.

Future extensions of this work will include the estimation of the absolute error $\|\Delta_a\|_\infty$ as part of the reduction procedure, the possibility of selecting among different linear system solvers and/or new state-space truncation methods for model reduction, the availability of the model reduction routines via an RPC interface (like, e.g, in NetSolve), and the development of a connection-oriented service, capable of storing the user data for efficient interactive model reduction.

REFERENCES

[1] P. AMESTOY, I. DUFF, J. KOSTER, AND J.-Y. L'EXCELLENT, *MUMPS: a general purpose distributed memory sparse solver*, in Proc. PARA2000, 5th International Workshop on Applied Parallel Computing, 2000, pp. 122–131.

[2] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, third ed., 1999.

[3] A. ANTOULAS, *Lectures on the Approximation of Large-Scale Dynamical Systems*, SIAM Publications, Philadelphia, PA, to appear.

[4] D. ARNOLD, S. AGRAWAL, L. BLACKFORD, J. DONGARRA, M. MILLER, S. VAHDIYAR, K. SAGI, AND Z. SHI, *Users's guide to NetSolve*, Tech. Report UT-CS-01-467, University of Tennessee at Knoxville, Computer Science Dept., Knoxville, TN, 2001. Available from `http://icl.cs.utk.edu/netsolve`.

[5] R. BADÍA, P. BENNER, R. MAYO, AND E. QUINTANA-ORTÍ, *Solving large sparse Lyapunov equations on parallel computers*, in Euro-Par 2002 – Parallel Processing, B. Monien and R. Feldmann, eds., no. 2400 in Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, New York, 2002, pp. 687–690.

[6] P. BENNER, R. MAYO, E. QUINTANA, AND G. QUINTANA-ORTÍ, *Enhanced services for remote model reduction of large-scale dense linear systems*, in PARA'02 Conference on Applied Parallel Computing, Espoo (Finland), 2002, J. Fagerholm, J. Haataja, J. Järvinen, M. Lyly, P. Raback, and V. Savolainen, eds., no. 2367 in Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, New York, 2002, pp. 329–338.

[7] P. BENNER, E. S. QUINTANA-ORTÍ, AND G. QUINTANA-ORTÍ, *Balanced truncation model reduction of large-scale dense systems on parallel computers*, Math. Comput. Model. Dyn. Syst., 6 (2000), pp. 383–405.

[8] ———, *State-space truncation methods for parallel model reduction of large-scale systems*, Parallel Computing, to appear.

[9] L. BLACKFORD, J. CHOI, A. CLEARY, E. D'AZEVEDO, J. DEMMEL, I. DHILLON, J. DONGARRA, S. HAMMARLING, G. HENRY, A. PETITET, K. STANLEY, D. WALKER, AND R. WHALEY, *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA, 1997.

[10] R. BUYYA, *High performance cluster computing: Programming and applications*, Prentice-Hall, Upper Saddle River, NJ, 1999.

[11] Y. CHAHLAOUI AND P. VAN DOOREN, *A collection of benchmark examples for model reduction of linear time invariant dynamical systems*, SLICOT Working Note 2002–2, Feb. 2002. Available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

[12] J. CHENG, G. IANCULESCU, C. KENNEY, A. LAUB, AND P. M. PAPADOPOULOS, *Control-structure interaction for space station solar dynamic power module*, IEEE Control Systems, (1992), pp. 4–13.

[13] P. CHU, B. WIE, B. GRETZ, AND C. PLESCIA, *Approach to large space structure control system design using traditional tools*, AIAA J. Guidance, Control, and Dynamics, 13 (1990), pp. 874–880.

[14] J. DEMMEL, J. GILBERT, AND X. LI, *SuperLU User's Guide*. Available from `http://www.nersc.gov/~xiaoye/SuperLU.html`.

[15] I. DUFF, A. ERISMAN, AND J. REID, *Direct methods for sparse matrices*, Oxford Science Publications, Oxford, UK, 1993.

[16] E. ELMROTH, P. JOHANSSON, B. KÅGSTRÖM, AND D. KRESSNER, *A Web computing environment for the SLICOT library*, SLICOT Working Note 2001–2, `http://www.win.tue.nl/niconet/`, June 2001.

[17] L. FORTUNA, G. NUMMARI, AND A. GALLO, *Model Order Reduction Techniques with Applications in Electrical Engineering*, Springer-Verlag, 1992.

[18] G. FOX, *Introduction to Web computing*, Computing in Science & Engineering, 3 (2001).

[19] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996.

[20] J.-R. LI AND J. WHITE, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.

[21] B. MOORE, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 17–32.

[22] G. OBINATA AND B. ANDERSON, *Model Reduction for Control System Design*, Communications and Control Engineering Series, Springer-Verlag, London, UK, 2001.

[23] C. PAUL, *Analysis of Multiconductor Transmission Lines*, Wiley–Interscience, Singapur, 1994.

[24] T. PENZL, *Algorithms for model reduction of large dynamical systems*, Tech. Report SFB393/99-40, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, FRG, 1999. Available from `http://www.tu-chemnitz.de/sfb393/sfb99pr.html`.

[25] ———, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.

[26] M. SAFONOV AND R. CHIANG, *A Schur method for balanced-truncation model reduction*, IEEE Trans. Automat. Control, AC–34 (1989), pp. 729–733.

[27] M. TOMBS AND I. POSTLETHWAITE, *Truncated balanced realization of a stable non-minimal state-space system*, Internat. J. Control, 46 (1987), pp. 1319–1330.

[28] A. VARGA, *Efficient minimal realization procedure based on balancing*, in Prepr. of the IMACS Symp. on

Modelling and Control of Technological Systems, vol. 2, 1991, pp. 42–47.

[29] ———, *Model reduction software in the SLICOT library*, in Applied and Computational Control, Signals, and Circuits, B. Datta, ed., vol. 629 of The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Boston, MA, 2001, pp. 239–282.

[30] E. WACHSPRESS, *The ADI model problem*, 1995. Available from the author.

[31] ———, *ADI iteration parameters for the Sylvester equation*, 2000. Available from the author.