Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮
FGCS Future Generation Computer Systems
www.elsevier.com/locate/fgcs

Monte Carlo methods for matrix computations on the grid

S. Branford[a], C. Sahin[a], A. Thandavan[a], C. Weihrauch[a,*], V.N. Alexandrov[a], I.T. Dimov[a,b]

[a] *Centre for Advanced Computing and Emerging Technologies, School of System Engineering, Philip Lyle Building, The University of Reading, Whiteknights, PO Box 68, Reading, RG6 6BX, UK*
[b] *Institute for Parallel Processing, Bulgarian Academy of Sciences, Acad. G. Bonchev 25 A, 1113 Sofia, Bulgaria*

Received 23 April 2007; received in revised form 25 June 2007; accepted 14 July 2007

Abstract

Many scientific and engineering applications involve inverting large matrices or solving systems of linear algebraic equations. Solving these problems with proven algorithms for direct methods can take very long to compute, as they depend on the size of the matrix. The computational complexity of the stochastic Monte Carlo methods depends only on the number of chains and the length of those chains. The computing power needed by inherently parallel Monte Carlo methods can be satisfied very efficiently by distributed computing technologies such as Grid computing. In this paper we show how a load balanced Monte Carlo method for computing the inverse of a dense matrix can be constructed, show how the method can be implemented on the Grid, and demonstrate how efficiently the method scales on multiple processors.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Monte Carlo methods; Matrix computations; Grid computing

1. Introduction

Inverting a real $n \times n$ matrix (MI) or solving a system of linear algebraic equations (SLAE) are problems of unquestionable importance in many scientific and engineering applications: e.g real-time speech coding, digital signal processing, communications, stochastic modelling, and many physical problems involving partial differential equations. The direct methods of solution require $O(n^3)$ sequential steps when using the usual elimination or annihilation schemes (e.g. non-pivoting Gaussian elimination, Gauss–Jordan methods) [10]. Consequently the computation time for very large problems, or for real-time solution problems, can be prohibitive and this prevents the use of many established algorithms.

The *Classical (Deterministic)* Iterative Methods (Jacobi, Gauss–Seidel, Successive Over-relaxation, Successive Under-relaxation) require $O(kn^2)$ operations, where k is the number of iterations. The Conjugate Gradient type methods require $O(n^2)$ operations (for dense matrix format), and they are optimal by rate (the best rate of convergence that can be achieved) of complexity. The Monte Carlo methods require $O(n^2)$ operations for dense matrix format and $O(n)$ operations for sparse matrix format and they are also optimal by rate. Nevertheless, for some specific classes of matrices Monte Carlo could be more or less efficient in terms of the number of operations needed to perform the algorithm for solving the problem with *a priori* given accuracy. Some recent results in this direction are presented in [23].

It is known that Monte Carlo (MC) methods give statistical estimates for elements of the inverse matrix, or for components of the solution vector of SLAE, by performing random sampling of a certain random variable, whose mathematical expectation is the desired solution [6,18,22]. We concentrate on MC methods for MI and/or solving SLAE since only $O(N\tau)$ steps are required to find an element of the inverse matrix or component of the solution vector of SLAE (where N is the number of chains and τ is an estimate of the chain length in the stochastic process, both of which are independent of n, the size of the matrix); and since the sampling process for stochastic methods is inherently parallel.

In terms of the computing requirements, there are three critical issues: throughput, peak performance, and the

* Corresponding author.
E-mail addresses: s.j.branford@reading.ac.uk (S. Branford), c.sahin@reading.ac.uk (C. Sahin), a.thandavan@reading.ac.uk (A. Thandavan), c.weihrauch@reading.ac.uk (C. Weihrauch), v.n.alexandrov@reading.ac.uk (V.N. Alexandrov), i.t.dimov@reading.ac.uk, ivdimov@bas.bg (I.T. Dimov).

0167-739X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2007.07.006

availability of the resource. The complex scientific problems solved by MC methods usually demand a high amount of computing power. In certain applications, like Monte Carlo sensitivity studies, the computation of Sobol's sensitivity indexes needs additional computing resources [5]. In other applications, like radiotherapy dose calculations, the computing power must be available on a timely critical basis, and in some cases, is used on almost real-time optimisations [16]. The requirements of the computing resources may be the limiting factor for some small-scale organisations to exploit the advantages of MC methods in their applications. The MC part of an application could easily be delegated to the available compute resources by using distributed computing technologies. The distributed computing environment, which can sustain such a compute power, can be achieved by use of grid technologies.

Grid technologies address coordinated resource sharing and dynamic problem solving in multi-institutional virtual organisations [9]. The sharing includes all types of resources including compute time and data storage capabilities. The resources can be on the same local network or may be located on geographically dispersed area. Grid computing technologies, by enabling the compute resources regardless of their geographical locations, promote the more frequent use of MC methods on very diverse areas of scientific and industrial problems and for any size of organisation. This could be achieved by fitting the requirements and the characteristics of the relevant MC method, which is the MC MI method in this paper's context, with the underlying grid infrastructure.

Coarse grained MC algorithms for MI and SLAE have been proposed by several authors [2,4,20]. In Section 2 we give an overview of using MC for MI and SLAE; in Section 3 we present a MC algorithm for MI; we give an overview of several methods for running jobs on *the grid* in Section 4; in Section 5 we look at performing MC methods on Grid systems; our implementation of the algorithm is described in Section 6; and we conclude our research in Section 7.

2. Monte Carlo matrix computations

Assume that the SLAE is presented in the form:

$$Bx = b, \qquad (1)$$

where B is a real square $n \times n$ matrix, $x = (x_1, x_2, \dots, x_n)^t$ is a $1 \times n$ solution vector $b = (b_1, b_2, \dots, b_n)^t$ and t means transpose.

Assume the general case $\|B\| > 1$ where $\|B\|$ is the spectral norm.[1] We consider the splitting

$$B = \hat{B} - C,$$

where off-diagonal elements of $\hat{B}$ are the same as those of B, and the diagonal elements of $\hat{B}$ are defined as $\hat{b}_{ii} = b_{ii} + \gamma_i \|B\|$, choosing in most cases $\gamma_i > 1$ for $i = 1, 2, \dots, n$. We further consider $\hat{B} = B_1 - B_2$, where B_1 is the diagonal matrix of $\hat{B}$, e.g. $(b_1)_{ii} = \hat{b}_{ii}$ for $i = 1, 2, \dots, n$. As shown in [21] we could transform the system (1) to

$$x = Tx + f, \qquad (2)$$

where $T = \hat{B}^{-1}C$ and $f = \hat{B}^{-1}b$. The multipliers γ_i are chosen so that, if it is possible, they reduce the norm of T to be less than 1 and thus reducing the number of Markov chains required to reach a given precision. We consider two possibilities, first, finding the solution of $x = Tx + f$ using MC method if $\|T\| < 1$ or finding $\hat{B}^{-1}$ using MC and after that finding B^{-1}. Then, if required, obtaining the solution vector by $x = B^{-1}b$.

Consider first the stochastic approach. Assume that $\|T\| < 1$ and that the system is transformed to its iterative form (2). Consider the Markov chain given by:

$$s_0 \to s_1 \to \cdots \to s_k,$$

where the s_i, $i = 1, 2, \dots, k$, belongs to the state space $S = \{1, 2, \dots, n\}$. Then for $\alpha, \beta \in S$, $p_0(\alpha) = p(s_0 = \alpha)$ is the probability that the Markov chain starts at state α and $p(s_{j+1} = \beta | s_j = \alpha) = p_{\alpha\beta}$ is the transition probability from state α to state β. The set of all probabilities $p_{\alpha\beta}$ defines a transition probability matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^n$ [1,2].

We say that the distribution $(p_1, \dots, p_n)^t$ is acceptable for a given vector g, and that the distribution $p_{\alpha\beta}$ is acceptable for matrix T, if $p_\alpha > 0$ when $g_\alpha \neq 0$, and $p_\alpha \geq 0$, when $g_\alpha = 0$, and $p_{\alpha\beta} > 0$ when $T_{\alpha\beta} \neq 0$, and $p_{\alpha\beta} \geq 0$ when $T_{\alpha\beta} = 0$ respectively. We assume $\sum_{\beta=1}^n p_{\alpha\beta} = 1$, for all $\alpha = 1, 2, \dots, n$. Generally, we define

$$W_0 = 1, \qquad W_j = W_{j-1} \frac{T_{s_{j-1}s_j}}{p_{s_{j-1}s_j}},$$

for $j = 1, 2, \dots, n$.

Consider now the random variable $\theta[g] = \frac{g_{s_0}}{p_{s_0}} \sum_{i=1}^\infty W_i f_{s_i}$. We use the following notation for the partial sum:

$$\theta_i[g] = \frac{g_{s_0}}{p_{s_0}} \sum_{j=0}^{i} W_j f_{s_j}.$$

Under condition $\|T\| < 1$, the corresponding Neumann series converges for any given f, and $E\theta_i[g]$ tends to (g, x) as $i \to \infty$. Thus, $\theta_i[g]$ can be considered as an estimate of (g, x) for i sufficiently large. To find an arbitrary component of the solution, for example, the rth component of x, we should choose, $g = e(r) = (\underbrace{0, \dots, 0, 1}_{r}, 0, \dots, 0)$ such that

$$e(r)_\alpha = \delta_{r\alpha} = \begin{cases} 1 & \text{if } r = \alpha \\ 0 & \text{otherwise.} \end{cases}$$

It follows that

$$(g, x) = \sum_{\alpha=1}^{n} e(r)_\alpha x_\alpha = x_r.$$

[1] As discussed later, the choice of matrix norm does not affect significantly the convergence of the algorithm.

The corresponding MC method is given by:

$$x_r = \hat{\Theta} = \frac{1}{N}\sum_{s=1}^{N}\theta_i[e(r)]_s,$$

where N is the number of chains and $\theta_i[e(r)]_s$ is the approximate value of x_r in the sth chain. It means that using MC method, we can estimate only one, few or all elements of the solution vector.

MC MI is obtained in a similar way [1]. To find the inverse $M^{-1} = \{m^{(-1)}_{rr'}\}^{n}_{r,r'=1}$ of some matrix M, we must first compute the elements of matrix $A = I - M$, where I is the identity matrix. Clearly, the inverse matrix is given by

$$M^{-1} = \sum_{i=0}^{\infty} A^i,$$

which converges if $\|A\| < 1$.

To estimate the element $m^{(-1)}_{rr'}$ of the inverse matrix M^{-1}, we let the vector f be the following unit vector

$$f_{r'} = e(r').$$

We then can use the following MC method for calculating elements of the inverse matrix M^{-1}:

$$m^{(-1)}_{rr'} \approx \frac{1}{N}\sum_{s=1}^{N}\left[\sum_{(j|s_j=r')} W_j\right], \tag{3}$$

where $(j|s_j = r')$ means that only

$$W_j = \frac{A_{rs_1}A_{s_1s_2}\dots A_{s_{j-1}s_j}}{p_{rs_1}p_{s_1s_2}\dots p_{s_{j-1}p_j}},$$

for which $s_j = r'$ are included in the sum (3).

Since W_j is included only into the corresponding sum for $r' = 1, 2, \dots, n$, then the same set of N chains can be used to compute a single row of the inverse matrix, which is one of the inherent properties of MC making them suitable for parallelisation.

The *probable error* of the method, is defined as $r_N = 0.6745\sqrt{\frac{D\theta}{N}}$, where $P\{|\bar{\theta} - E(\theta)| < r_N\} \approx \frac{1}{2} \approx P\{|\bar{\theta} - E(\theta)| > r_N\}$, if we have N independent realisations of random variable (r.v.) θ with mathematical expectation $E\theta$ and average $\bar{\theta}$ [18].

In the general case, $\|B\| > 1$, we make the initial split $B = \hat{B} - C$. From this we compute $A = B_1^{-1}B_2$, which satisfies $\|A\| < 1$ (by careful choice, of $\hat{B}$, we make $\|A\| < \frac{1}{2}$, which gives faster convergence). Then we generate the inverse of $\hat{B}$ by (3). From this we wish to recover B^{-1}, which uses an iterative process ($k = n-1, n-2, \dots, 0$) on $\hat{B}^{-1}$

$$B_k^{-1} = B_{k+1}^{-1} + \frac{B_{k+1}^{-1}S_{k+1}B_{k+1}^{-1}}{1 - \operatorname{trace}\left(B_{k+1}^{-1}S_{k+1}\right)}, \tag{4}$$

where $B_n^{-1} = \hat{B}^{-1}$ and $B_0^{-1} = B^{-1}$.

In this work we consider the almost optimal (MAO) transition probability

$$p_{\alpha\beta} = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n}|a_{\alpha\beta}|},$$

for $\alpha, \beta = 1, 2, \dots, n$.

3. Monte Carlo algorithm for matrix inversion

In this paper we consider only present results for diagonally dominant matrices.[2] This leads us to present the following algorithm for a MC method for inverting a diagonally dominant matrix.

```
Step 1. Read in matrix $B$, the matrix to be inverted
        1: Input matrix $B$, parameters $\varepsilon$ and $\delta$
Step 2. Calculate intermediate matrices ($B_1$, $B_2$)
        1: Split $B = B_1 - B_2$, where $B_1 = diag(B)$ and $B_2 = B_1 - B$
Step 3. Calculate matrix $A$ and $\|A\|$
        1: Compute matrix $A = B_1^{-1}B_2$
        2: Compute $\|A\|$ and the number of Markov Chains $N = \left(\frac{0.6745}{\varepsilon(1-\|A\|)}\right)^2$
Step 4. Calculate matrix $P$
        1: Compute the probability matrix, $P$, where $p_{ij} = \frac{|a_{ij}|}{\sum_{k=1}^{n}|a_{ik}|}$
Step 5. Calculate matrix $M$, by MC on $A$ and $P$
        1: For $i = 1$ to $n$
           1.1: For $j = 1$ to $N$
                Markov Chain MC Computation
                1.1.1: Set $W_0 = 1$, $point = i$ and $SUM[k] = \begin{cases}1 \text{ if } i = k\\ 0 \text{ if } i \neq k\end{cases}$
                1.1.2: Select a nextpoint, based on the transition probabilities in $P$, such that $A[point][nextpoint] \neq 0$
                1.1.3: Compute $W_j = W_{j-1}\frac{A[point][nextpoint]}{P[point][nextpoint]}$
                1.1.4: Set $SUM[nextpoint] = SUM[nextpoint] + W_j$
                1.1.5: If $|W_j| > \delta$ set $point = nextpoint$ and goto 1.1.2
           1.2: Then $m_{ik} = \frac{SUM[k]}{N}$ for $k = 1, 2, \dots, n$
Setp 6. Calculate $B^{-1}$
        1: Compute the MC inverse $B^{-1} = MB_1^{-1}$.
```

In Step 3, the chosen norm $\|A\|$ is essential for estimating the number of Markov Chains to ensure a well-balanced algorithm (i.e. an algorithm for which the stochastic error is approximately equal to the systematic error) [4]. Nevertheless, the particular choice of the norm does not reflect too much on the efficiency of the algorithm since all matrix norms are equivalent. While the implementation presented in this paper is for dense matrices the algorithm also applies equally to sparse and banded matrices.

[2] If we, instead, had a non-diagonally dominant matrix then we would have to extend the algorithm to do the initial split, $B = \hat{B} - C$, and recovery, (4), as explained in the previous section.

4. Grid approach for Monte Carlo methods

When making an application available on the Grid, a strategy has to be developed to gain maximum performance. The strategy used, which helps to maximise the advantages of the Grid, depends largely on the application itself.

There are some design issues to consider [15] when using the grid for MC methods:

- Investigation of cross-cluster parallelisation.
- Necessity of a checkpointing mechanism.
- Trustworthiness of the remote resource.
- Determination of the best approach to offer the MC method on the Grid.

The most important feature of MC methods for grid computing is their stochastic nature, which provides a natural parallelisation with very little inter-process communication. This is a promising application area to achieve cross-cluster parallelisation with the help of libraries like MPICH-G [8]. From the point of the MC MI method, cross-cluster parallelisation can be quite expensive to employ. Although relatively little inter-process communication is required for MC MI methods as well, parallel processes of MC MI method need to work with the full matrix — the matrix needs to be transferred to every node of every cluster which makes it communicationally expensive and hence less feasible for the Grid.

An application specific checkpointing mechanism is thought to be essential for many MC methods over the grid to increase the overall performance of the system. MC methods are CPU intensive and may take quite long to complete. Any interruption would result in having to restart the same computation from scratch. On the other hand, they are also easy to reconstruct and resume once the time step of the computation and the value are known. This can easily be achieved by keeping an intermediate state of the computation in a database to recover and resume the computation from the point where it was checkpointed. This saves a substantial amount of CPU time once the task is rescheduled. However, any checkpointing mechanism for the MC MI method is an expensive process, since each checkpoint requires recording the current state of the matrix.

When working on MC computations, the accuracy of the result is influenced by the hardware and the compiler. This can make the application sensitive to computational errors. [15] suggests the use of cross-checking of the data for the trustworthiness of the remote resource. For time critical applications, this is not a viable option. Our suggestion is to use certain remote resources where trustworthiness is guaranteed. This could easily be achieved by using custom MC grid services attached to trusted remote resources. It is also related to the fourth item on the above list — how best to offer the MC MI method over the Grid. Using Grids promises a lot for MC MI method as listed below:

- Grid technologies provide access to compute resources in a virtual organisation, enabling the supply of necessary compute power whenever needed.
- Scheduling and control of many subtasks to different compute resources.
- Grid software provides a complete framework with established protocols to manage a set of distributed tasks.
- Well-established and mature security solutions ensure the protection of users' software and data.
- Grid software offer service-based workflow management where the whole task could be carried out without any human intervention, hence providing a significant speed-up.
- Grid software could be used to efficiently hide the complexities that arise from using heterogeneous hardware resources.

In this study, the Globus Toolkit was used, which offers a framework using well-established protocols. Reasons for this choice included prior experience with the software and the availability of a testbed with the Globus software already deployed. Similar results can be expected to be achieved using other software packages such as UNICORE, Condor-G and g-Lite that also enable grid computing through well-defined frameworks.

The Globus Toolkit provides a set of libraries and programs, which address the challenges presented by the distributed grid computing environment, including remote job submission and execution, data transfer, resource discovery and security. The latest version of the Globus Toolkit, version 4 (GT4) [7], extensively uses Web services mechanisms to define its interfaces. Taking advantage of well-defined Web service protocols and standards, XML-based mechanisms are used to describe and discover the network services which facilitate the development of service-oriented architectures (SOA) [7]. The GT4 framework provides all necessary components and associated services to develop a complete distributed computing application.

Security in a grid environment is a very important issue. Grid security is provided by the Grid Security Infrastructure (GSI) in GT4 to guarantee secure grid-based operations. GSI is based on public key cryptography. It provides two levels of security — message-level security and transport-level security. For MC MI methods we are mainly interested in transport-level security, which uses proxy certificates to ensure the privacy of the code and the data sets on remote resources.

5. Grid implementation

In this section we consider various grid implementations for the MC MI method. Single cluster and workflow implementations are analysed and an MC Linear Algebra (LA) grid service implementation is suggested. Running the MC MI method in a multicluster environment is not investigated as cross parallelisation of the MC MI method is theoretically too expensive to employ. When using the WS-GRAM job submission service directly the code and the data set is assumed not to be available on the remote resource.

For the grid-enabled MC MI method GT4's WS-GRAM service plays an important role. WS-GRAM provides the submission of the code on remote resources [5] as well as file stage-in and stage-out facilities for job submission using

```
<job>
 <executable>matrix</executable>
 <directory>//home/bio1/test1</directory>
 <argument>A-10</argument>
 <argument>0.2</argument>
 <argument>0.2</argument>
 <environment>
     <name>LD_LIBRARY_PATH</name>
     <value>$LD_LIBRARY_PATH:/opt/intel/fc/9.0/lib</value>
 </environment>
 <stdout>out.1</stdout>
 <stderr>err.1</stderr>
 <jobType>single</jobType>
 <fileStageIn>
     <transfer>
       <sourceUrl>gsiftp://machine:2811/home/bio1/A-10</sourceUrl>
       <destinationUrl>file:////home/bio1/test1/A-10</destinationUrl>
     </transfer>
 </fileStageIn>
 <fileStageOut>
     <transfer>
       <sourceUrl>gsiftp://machine:2811/home/bio1/test1/out.1</sourceUrl>
       <destinationUrl>gsiftp://machine:2811/home/bio1/out.machine1</destinationUrl>
     </transfer>
 </fileStageOut>
</job>
```

Fig. 1. An example job description file.

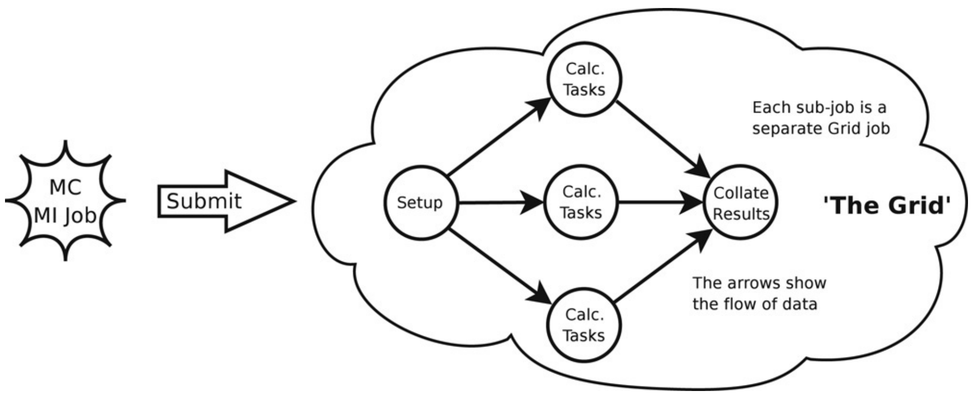

Fig. 2. Running the job using a workflow.

GT4's Reliable File Transfer service. WS-GRAM also provides stateful monitoring of the jobs on the remote resources, with a control over the jobs (cancel, hold, resume, etc.) and comes with credential management as the security in a grid environment is vital. The ManagedJobFactoryService that ships with GT4, provides the scheduling of the application on different types of schedulers like fork, PBS, Condor and LSF.

In the simplest case, the MC MI executable is submitted for execution on a remote cluster using the ManagedJobFactory-Service

WS-GRAM job submission uses a job description file (JDF) which defines the requirements of the job to run. An example JDF is shown in Fig. 1.

The example JDF, Fig. 1, defines a single job because the executable for the platform is assumed to be ready prior to submission. JDFs also facilitate the preparation of multijob definition files in case a compilation prior to execution is needed on the remote platform. Note also that the `jobType` element is 'single' which indicates a sequential job. The JDF, which contains XML-based job specifications (including input and output files, execution information like the name of the scheduler and the number of processors needed) has all the information for the job to run on the remote compute resource.

5.1. *Workflow method*

Another possibility is to run the MC MI code as part of a workflow system, as demonstrated in Fig. 2. This is a more realistic approach, as the MC MI method is mostly used as part of a bigger application or several MC MI instances are run simultaneously to achieve a more accurate result. Such systems need to exchange data, hence the need to behave as part of a bigger system. This could be easily achieved by using grid workflow systems [24].

A possible workflow scenario is the simultaneous runs of the MC MI simulations on different resources to achieve a more accurate result:

(1) The client defines the workflow including how many MC MI simulations need to be scheduled.
(2) GT4 resource discovery service (MDS) provides the available resources in the virtual organisation. The subtasks are scheduled to the remote resources by the availability of their computing power, using GT4 GRAM service.

(3) GRAM also stages the input data into the resource by using GT4's RFT service.
(4) The JDF has all the information on how to run the code on the remote resource. Following the execution of the code, output data is staged to the local resource or to a data server as stated in the workflow description, to generate the final result.

5.2. *Monte Carlo linear algebra grid service*

So far, the WS-GRAM service ManagedJobFactoryService is used to submit and monitor the job on remote clusters. Another option is to implement the MC LA methods as a dedicated grid service. The MC LA grid service ties the service onto the resource where it is deployed, providing certain advantages:

- Trustworthiness is guaranteed by deploying the service only on trusted resources.
- The grid service, which should be easy to use, provides a black box for users where they simply provide their matrices as input and get its inverse as output.
- Code complexities are hidden with the code and libraries already available on the deployed resource.
- Better optimised runs can be expected as the code would be tuned to the underlying hardware and with the use of optimised compilers and random number generators.
- Additional MC LA methods could be added to the service where users could query MC LA specific information through the information services.

The MC LA grid service could be plugged into bigger applications as part of a workflow management system that would return the result in an efficient and trusted manner.

A certain disadvantage is that the MC LA grid service limits the use of the available resources in the virtual organisation. It also requires extra work, including the installation, maintenance and upgrade of the MC LA grid service in each resource it is deployed on.

The client in Fig. 3 represents a human or software interaction. The MC LA grid service will be implemented in the future.

6. Implementation

6.1. *Load balancing parallel algorithm*

Working from the algorithm presented in Section 3 we arrive at the following load-balancing parallel algorithm for MC MI:

1. The master process carries out Steps 1–4.
2. The master process broadcasts n, A, P, $Psum$, and δ.
3. Each process carries out a part of the Markov chains.
4. Each process returns the time taken for the first part of chains to the master process.
5. The master process distributes the remaining chains between the processes, balancing the number of chains to be done by each process based on the time taken for the first part of chains.
6. Each process completes a proportion of the remaining Markov chains.
7. The master process collects the MC calculations.
8. The master process carries out Step 6.

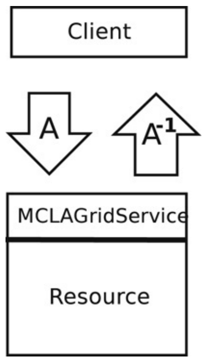

Fig. 3. MC LA grid service used for matrix inversion.

6.2. *Fault tolerance*

MC algorithms, by their very nature, make it easy to implement fault tolerance (FT) [15]. One method is to compute 5% or 10% more chains than required, and just use the results from the successful nodes if one or more nodes fail. The easiest implementation would be to ignore failed nodes and just carry on with the results returned from the working nodes [12,25].

The MPI 1.x standard [17] describes how the MPI implementation could handle node failures by overriding the error handler with `MPI_Comm_set_errhandler()`. The error handler `MPI_ERRORS_ARE_FATAL` is used by default, and this handler causes all nodes to stop if just one node fails. Changing it to `MPI_ERRORS_RETURN` would allow MPI to ignore the failed nodes and the Monte Carlo algorithm could finish the calculation. However, this feature is not implemented in the most popular MPI 1.x implementations. Most grid systems use, or are based on, MPICH [13,11] or LAM-MPI [3,19]. However, to the best of our knowledge these two currently do not support FT. We hope that in the future production grids will move to implementations which support FT. This would allow us to better exploit the natural features of MC algorithms.

6.3. *Binary search*

A large part of the time the MC method needs to compute the solution is spent on finding and jumping to the next element in the matrix. The next element is chosen based on a probability matrix. To improve the implementation of this section of the algorithm a binary search algorithm was implemented [14]. The binary search only needs $\log n$ steps to find the next element compared to a standard linear search algorithm which would need on average $n/2$ steps.

6.4. *Results*

The results were obtained using an SGI Prism system equipped with eight Intel Itanium II 1.5 GHz processors and 16 GB main memory. The code was compiled using the Intel Fortran Compiler 9, and the variables were stored at double precision.

The input matrices were randomly generated diagonally dominant matrices of size 500–6000. As the MC algorithm requires a copy of the matrix on each node, the available memory on the test machine limits us to dense problems of at

Table 1
Timing for the Monte Carlo algorithm

Proc	500	1000	2000	3000	4000	5000	6000
1	80.26	283.51	814.08	1421.55	2098.96	3016.95	3782.21
2	41.94	157.60	751.16	813.52	1202.88	1679.44	1982.10
3	33.78	103.45	341.37	528.24	775.68	1081.43	1680.76
4	23.39	91.05	254.94	388.20	563.18	945.49	998.69
5	18.27	60.60	179.47	310.91	488.13	635.08	869.96
6	17.20	52.36	155.19	272.15	405.34	613.82	842.11
7	15.56	44.61	133.44	250.00	354.41	475.57	642.57
8	12.20	39.16	114.92	205.04	304.57	420.04	570.17

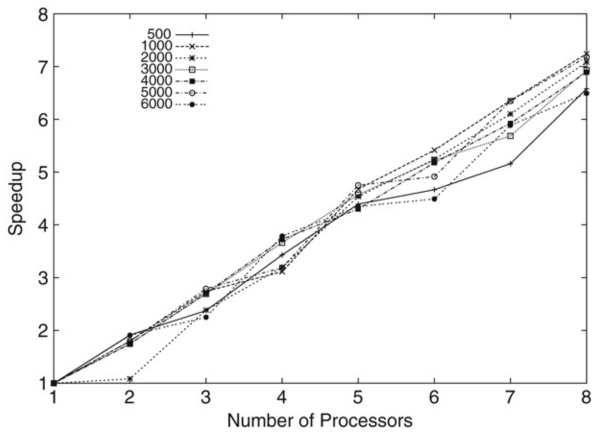

Fig. 4. Speedup of the Monte Carlo algorithm.

most size 6000. Our approach works for some non-diagonally dominant matrices [4] as well.

The results of the experiments are shown in Table 1, and Fig. 4 shows the speedup achieved by the algorithm. As it can be seen the MC MI method scales well and the speedup is near linear. There are a few extraneous results, but these are to be expected from a MC method, since the Markov Chains are random processes; from computer systems, where the nature of the processes running sometimes causes nodes to take longer to compute the test chains, which leads to a wrong balancing result.

When the MC MI method is used with Grid systems, various overheads are added. One overhead is caused by the GT4 WS-GRAM service, while another is introduced by the local scheduler on the remote resource. Another source of overhead is the use of GridFTP to transfer data between resources on the grid. In our case, the file transfer overhead was negligible as the largest matrix transferred was reasonably small in size. Also, the enabling of grid security further increases the overhead.

7. Conclusion and future work

We have shown that MC methods can be used to solve SLAE and do MI. Further to this we have presented an MC algorithm for MI, and then extended this to be a load balanced parallel algorithm. We have then discussed the use of grid technologies and how the MC MI method can be deployed on the grid. The load balancing algorithm has been deployed and we present results showing that the method is efficient at scaling to multiple processors.

In the future we are looking at expanding on this grid MC MI method by looking at other grid MC methods. Possibilities include MC MI methods for different types of matrices, MC methods to solve SLAE, and MC methods for finding eigenvalues. The Monte Carlo algorithm presented here is relevant for sparse and banded matrices — it is only necessary to adjust the implementation to optimise the code for such problems. A combination of these methods could then be offered as a grid MC service, available for scientists to use in their applications.

Acknowledgement

This work was partially supported by NATO under project *Monte Carlo Sensitivity Studies of Environmental Security* PDD(TC)-(ESP.EAP.CLG 982641).

References

[1] V.N. Alexandrov, Efficient parallel monte carlo methods for matrix computation, Mathematics and Computers in Simulation 47 (1998) 113–122.
[2] V.N. Alexandrov, A. Rau-Chaplin, F. Dehne, K. Taft, Efficient coarse grain Monte Carlo algorithms for matrix computation using PVM, Lecture Notes in Computational Science (1998) 323–330.
[3] G. Burns, R. Daoud, J. Vaigl, LAM: An open cluster environment for MPI, in: Proceedings of Supercomputing Symposium, 1994, pp. 379–386.
[4] I.T. Dimov, T. Dimov, T. Gurov, A new iterative Monte Carlo approach for inverse matrix problem, Journal of Computational and Applied Mathematics 92 (1997) 15–35.
[5] I. Dimov, Z. Zlatev, Testing the sensitivity of air pollution levels to variations of some chemical rate constants, in: Large-Scale Computations in Engineering and Environmental Sciences, in: M. Griebel, O. Iliev, S. Margenov, P.S. Vassilevski (Eds.), Notes on Numerical Fluid Mechanics, vol. 62, 1997, pp. 167–175.
[6] G.E. Forsythe, R.A. Leibler, Matrix inversion by a Monte Carlo method, Mathematical Tables and Other Aids to Computation 4 (31) (1950) 127–129.
[7] I. Foster, Globus toolkit version 4: Software for service-oriented systems, in: Lecture Noes in Computer Science, vol. 3779, 2005, pp. 2–13.
[8] I. Foster, N. Karonis, A grid-enabled MPI: Message passing in heterogeneous distributed computing systems, in: Proceedings of 1998 SuperComputing Conference, November, 1998.
[9] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, International Journal of Supercomputer Applications 15 (3) (2001).
[10] G.H. Golub, C.F. van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, London, 1996.
[11] W. Gropp, E. Lusk, User's Guide for mpich, a Portable Implementation of MPI, ©1996 Mathematics and Computer Science Division, Argonne National Laboratory.
[12] W. Gropp, E. Lusk, Fault tolerance in MPI programs, in: Proceedings of Cluster Computing and Grid Systems Conference 2002.
[13] W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing 22 (6) (1996) 789–828.
[14] D. Knuth, Sorting and Searching, third ed., in: The Art of Computer Programming, vol. 3, Addison-Wesley, 1997.
[15] Y. Li, M. Mascagni, Grid-based Monte Carlo application, in: Proceedings of Grid Computing-GRID 2002, in: Manish Parashar (Ed.), Lecture Notes in Computer Science, vol. 2536, 2002, pp. 13–24.
[16] J.T. Moscicki, S. Guatelli, M.G. Pia, M. Piergentili, Monte Carlo Simulation for Radiotherapy in a Distributed Computing Environment, American Nuclear Society, 2005. On CD-ROM.
[17] MPI: A Message-Passing Interface Standard (version 1.1), ©1993, 1994, 1995 University of Tennessee, Knoxville, Tennessee.
[18] I.M. Sobol, Monte Carlo Numerical Methods, Nauka, Moscow, 1973.

[19] J.M. Squyres, A. Lumsdaine, A component architecture for LAM/MPI, in: Proceedings, 10th European PVM/MPI Users' Group Meeting, in: Lecture Notes in Computer Science, vol. 2840, 2003, pp. 379–387.

[20] B. Fathi Vajargah, V.N. Alexandrov, Coarse grained parallel Monte Carlo algorithms for solving systems of linear equations with minimum communication, in: Proceedings of PDPTA, Las Vegas, 2001, pp. 2240–2245.

[21] B. Fathi Vajargah, B. Liu, V.N. Alexandrov, Mixed Monte Carlo Parallel Algorithms for Matrix Computation, in: Lecture Notes in Computational Science, vol. 2330, Springer Verlag, 2002, pp. 609–618.

[22] J.R. Westlake, A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, John Wiley & Sons, Inc., New York, London, Sydney, 1968.

[23] C. Weihrauch, I. Dimov, S. Branford, V. Alexandrov, Comparison of the computational cost of a Monte Carlo and deterministic algorithm for computing bilinear forms of matrix powers, in: Computational Science — ICCS 2006, in: Lecture Notes in Computer Science, vol. 3993, Springer-Verlag, 2006, pp. 640–647.

[24] F. Neubauera, A. Hoheiselb, J. Geiler, Workflow-based grid applications, Future Generation Computer Systems 22 (1–2) (2006) 6–15.

[25] A. Luckow, B. Schnor, Migol: A fault-tolerant service framework for MPI applications in the grid, Future Generation Computer Systems 2007 (in press).

S. Branford is a researcher at the Centre for Advanced Computing and Emerging Technologies, University of Reading. He received his M.Math., from the University of Oxford, in 2001 and M.Sc., from the University of Reading, in 2003. His research is in the field of mathematical computations in science; and in particular hybrid and parallel Monte Carlo methods for sparse matrix problems.

C. Sahin is a researcher and a Ph.D. student at the ACET Centre, University of Reading. He received his M.Sc. in Network Centred Computing from University of Reading and a B.Sc. degree in Physics. His area of research includes computational science, parallel, distributed and grid computing.

A. Thandavan is a researcher at the Centre for Advanced Computing and Emerging Technologies, University of Reading, UK. He received his M.Sc. in Network Centred Computing from the University of Reading in 2002 and B.Sc. in Physics from the University of Madras in 2000. His research interests include Parallel and High Performance Computing, Grid Computing and Workflow systems. He is currently working on the Workflow Builder component of g-Eclipse, a project funded by the European Commission's FP6 programme.

C. Weihrauch is a researcher at the Centre for Advanced Computing and Emerging Technologies, University of Reading. He holds an M.Sc. degree in Network Centred Computing from the University of Reading and a Diploma (FH) in Technische Informatik from FHTW Berlin. His area of research is computational science specialising in parallel computing and Monte Carlo methods.

V.N. Alexandrov is a professor in Computational Sciences at the School of Systems Engineering, Director of the Centre for Advanced Computing and Emerging Technologies and Head of Research of PEDAL Laboratory at the University of Reading, UK. He has obtained his M.Sc. in Applied Mathematics from Moscow State University in 1984 and his Ph.D. in parallel computing from the Institute for Parallel Processing at the Bulgarian Academy of Sciences in 1995. His main interests are in the areas of simulation and modelling of complex systems, parallel scalable algorithms, collaborative, cluster and grid computing.

I.T. Dimov is a professor at the University of Reading, Centre for Advanced Computing and Emerging Technologies as well as IPP, Bulgarian Academy of Sciences. He received his M.S. in 1977, Ph.D. in 1980 and M.Sc. in 1984 in Moscow. His research interests are computational sciences, in particular Monte Carlo numerical analysis, parallel, distributed and grid computing.