A Simultaneous Solution for General Linear Equations on a Ring or Hierarchical Cluster

G. Molnárka, N. Varjasi

"Széchenyi István" University Győr, Hungary, H-9026 Egyetem tér 1.
Phone: +3696503400, fax:
e-mail: varjasin@sze.hu

Abstract: There are several iterative models for solving general, large linear equations. In this paper a parallel algorithm with slow convergence speed for studying the speed-up effect of the parallel algorithms has been presented. The difference between the ring and hierarchical topology considering running speed and efficiency has also been explored. Detailed numerical test results of the algorithm including the speedup of parallel execution are shown.

Keywords: parallel programming, linear equation, cluster computing

1. The minimal residual algorithm

The main goal of this article is to study the speed up effect on a parallel computer using a parallel algorithm for solving a full rank, general, symmetric, positive definite not sparse (but dense) linear equation with high condition number ($n = 1000$, $cond = 2 \cdot 10^{10}$; $n = 5000$, $cond = 4 \cdot 10^{13}$; $n = 10000$, $cond = 7 \cdot 10^{12}$; $n = 15000$, $cond = 5 \cdot 10^{16}$), where $cond(A) = \|A\| \cdot \|A^{-1}\|$ and Euclidean norm was used.

The condition number shows the difficulty of the linear equation. Higher condition number means the complexity of the problem, in other words the equation gets more and more difficult with numerical methods.

The solving algorithms of the general $Ax = b$ equation are well known [1] [2] [3]. In the case of large systems direct algorithms are inefficient. Only iterative methods can be used that can produce results with the desired precision [4] [10], otherwise the floating point arithmetic causes several rounding errors.

The base of the presented numerical algorithm for the solution of linear systems of equations is a generalisation of the classical one-step iterative algorithm (such as the gradient method). Generalisation will not improve the convergence speed of the algorithm but it highly improves parallel execution. In a sequential case the base algorithm [5] [6] has slow convergence speed. The minimal residual algorithm is a widely known algorithm, but the suggested versions of *Algorithm1* and *Algorithm2* have been created by the authors. The results of the parallel realisation of the algorithms and the measured data are the results of the present research.

The suggested algorithms give a good opportunity to study the effects of parallel processing. If a cluster or a multiprocessor computer is used, one can expect considerable speed-up effect.

2. Methods for parallelisation on homogeneous and heterogeneous systems

The aim of parallel processing is to break a large problem down to several smaller components or calculations that can be solved parallelly with different processors at the same time. The most efficient tools for scientific computations would be massively parallel computers, with a large shared memory, but this hardware is expensive and unattainable for the research team. Other solutions can be distributed systems and cluster computing. A small cluster of 16 PCs with normal network connection and an interconnected cluster machine with 88 processors (HP BladeSystem C3000) were used. On the cluster computing model a message passing software (MPI) was used to solve the tasks.

2.1. Ring and hierarchical topologies

In parallel solutions there are two bottlenecks for optimisation: the communication and the calculations. These aspects have been studied with two topologies. For linear equations based on numerical models the ring model is often used [8]. In this case every node has a connection with the two neighbouring nodes, or other nodes. This solution works efficiently on homogeneous systems. On this model the heartbeat algorithm is useful (see *Figure 1 a*). First it starts an initialization procedure, then a loop starts. In the first phase of the loop a data sending and receiving mechanism process is accomplished (synchronization). This is the data exchange period between each computation node. After that, every node runs the computation algorithm. This is the "cpu" period of the work. The loop runs until the stopping criteria. In this model every node has an equivalent role. This model needs a homogeneous network and the same type of processor because each synchronization step made by the slowest node. Fast nodes need to wait for them.

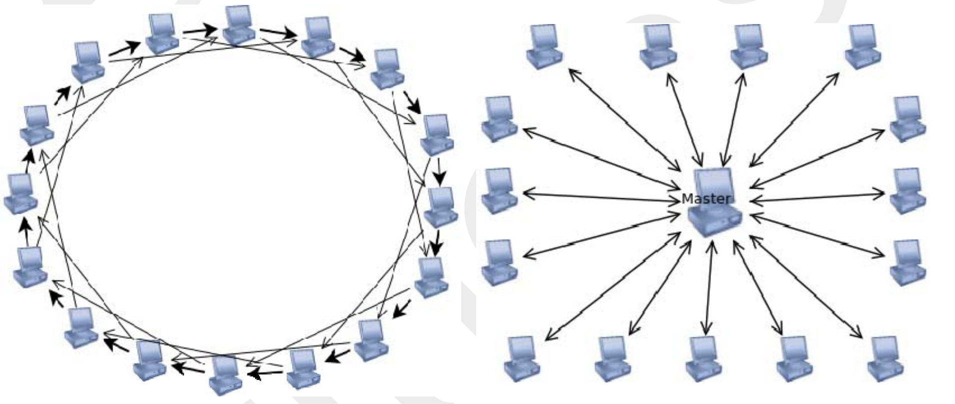

Figure 1. Ring (a) and hierarchical (b) topology

In other cases hierarchical topologies are useful [9]. The master-worker model is based on a distributed and large, heterogeneous cluster (see *Fig 1b.*). The master node controls the running processes, assigns problems to workers and manages the partial solutions. The role of the worker nodes is to solve smaller parts of the problem. This model works well with asynchronous methods, too because worker nodes have not connected to each other. Every worker node can reach the maximum performance of the processors.

2.2. Algorithm 1

For a ring topology the following algorithm has been used:

```
I.   Let $P$ be the number of nodes, let *eps* be the tolerated error value.
II.  Let $A$ be the matrix to be solved and let $b$ be the solution vector.
III. For every node $p \in P$ do in parallel: generate $x_1$ random vector.
IV.  For every node $p \in P$ do in parallel:
         *Operation()* while a result arrives or converges.
V.   The result of solution is $x_1$ on master node.
```

The algorithm uses the *Operation()* function on every node:

```
1. do
2.     let $x_2$ be a new random vector
3.     let $r_1 = Ax_1 - b$ and $r_2 = Ax_2 - b$, where $r_1 - r_2 \neq 0$
4.     let $c_{12} := \frac{(r_2 - r_1, r_2)}{\|r_1 - r_2\|^2}$
5.     let $x_{12} := c_{12}x_1 + (1 - c_{12})x_2$
6.     let $r_{12} := c_{12}r_1 + (1 - c_{12})r_2$
7.     let $x_1 := x_{12}$ , and $r_1 := r_{12}$
8.     if $\|r_1\|^2 < min$ or $iterationnum > n$
           then send $x_1$ to the next node and wait for a new $x_2$ vector
9. while $\|r_1\|^2 < eps$
10.    return $x_1$ , the solution with desired precision.
```

Remarks:

From the vector exchange it is expected that the given result is better, or when the algorithm reaches a local minimum value this x_1 solution is sent to another node, which continues the computation with a new random number coming from another node (see *Figure 2*).

In the implementation of the algorithm the Mersenne-twister pseudorandom number generator has been applied [7] and the independence of iteration sequences is based on the independent clock of the computing nodes.

Two error measure methods have been used in the algorithm. The first was the general residual error: $err_r = \|r_1\|^2 = \|Ax - b\|^2$. When the exact solution of the test case is known (x'), there is a chance to compute the absolute error value: $err_x = \|x - x'\|^2$, where x is the approximate solution vector.

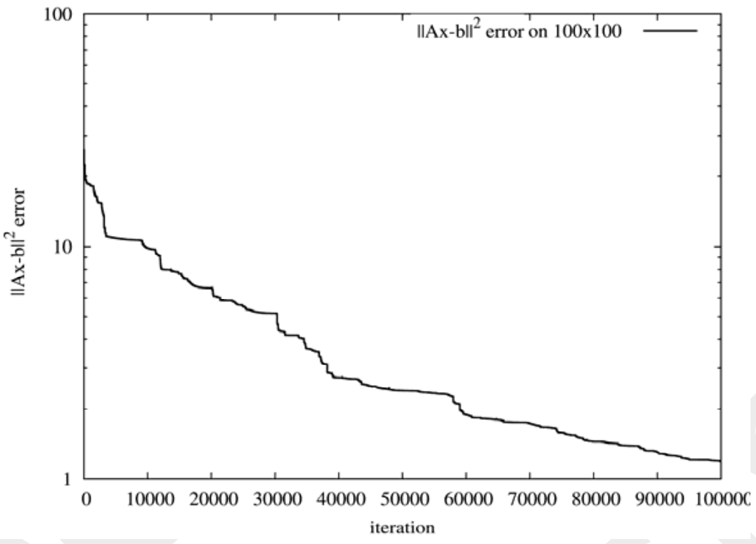

Figure 2. The convergence of Algorithm1 (n = 100)

In the case of large-sized, badly conditioned linear equations these error values are relatively high numbers (with $cond = 10^{16}$, $err_r = 1000$ means a close solution as shown in *Figure 4*, in detail that means $\|x - x'\|^2$ is $\sum_{i=1}^{n} |x_i - x_i'|^2$ and the error for every member of the solution vector is approximately $|x_i - x_i'|^2 \approx 10^{-8}$).

If a problem was solved where the correct solution had already been known, and the residual and absolute error were compared it has to be noted that the absolute error of the solution is always better than the residual.

The efficiency of the algorithm depends on load balancing: the operation can be repeated several times with slow convergence speed or the result vector can be exchanged between nodes to give extra speedup. In this case and referring to Amdahl's law the ring model has a theoretical maximum number of nodes. If more nodes are used and the best solution is sent to the next node, the larger ring will increase running time as the solution waves slowly on to the other nodes.

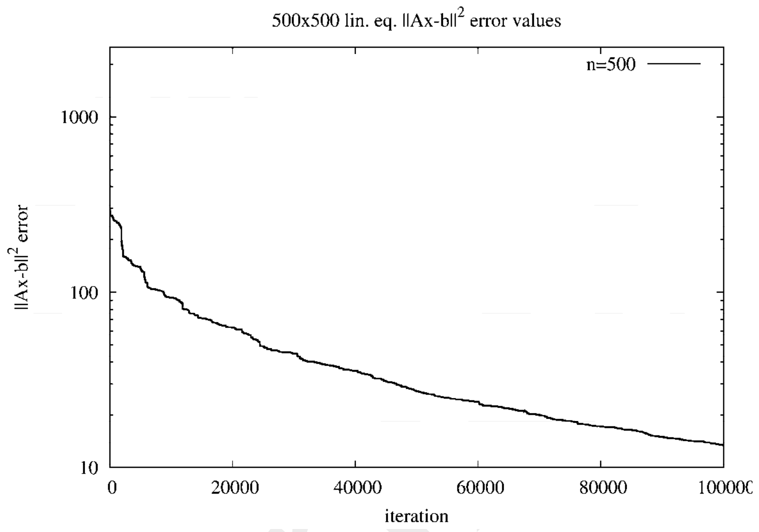

Figure 3. The convergence of Algorithm1 (n = 500)

This *Algorithm1* works well on a homogeneous cluster (see *Figure3*). But if there is a slower or a loaded computer on the ring the send-receive method will be slow and several traffic jams are expected and running times also grow.

Algorithm1 has been revised and a new, hierarchical model has been composed.

2.3. Algorithm 2

I. Let P be the number of nodes, let *eps* be the tolerated error value.

II. Let A be the matrix to be solved and let b be the solution vector.

III. The master node generates a random vector x_2 and sends for every worker node.

IV. For every node $p \in P$ do in parallel: generate a random vector x_1

V. On master node do *Control()* while $\|x_1\|^2 < eps$

VI. For every worker node $p \in P$ do in parallel:

Operation(x1) while a vector arrives

VII. The result of solution is x_1 on master node.

On the worker nodes the *Operation()* function uses a residual approach like *Algorithm1*. The difference is that the operation function sends and receives data from the master node only.

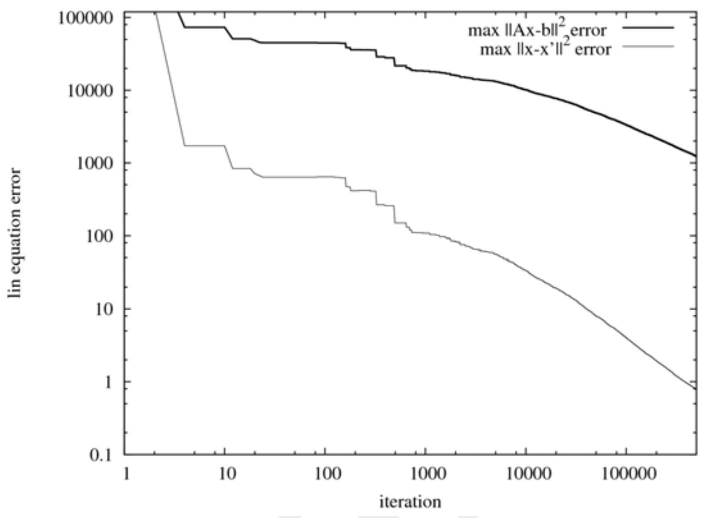

Figure 4. Residual and absolute error of the solution vector (n = 15000, P = 88 cpus, log-log scale)

The *Control* function of the master node distributes and collects data from every worker node. On the master node the problem is not solved, but the result is presented here. The master node controls the data exchanges, and presents the best approximate result vector for every node (see *Figure 4*). This model is flexible because the number of worker nodes number can grow dynamically [8].

This model can be used on heterogeneous clusters, too, because the worker nodes are independent and communicate only with the master node. Every node works on the master's best solution.

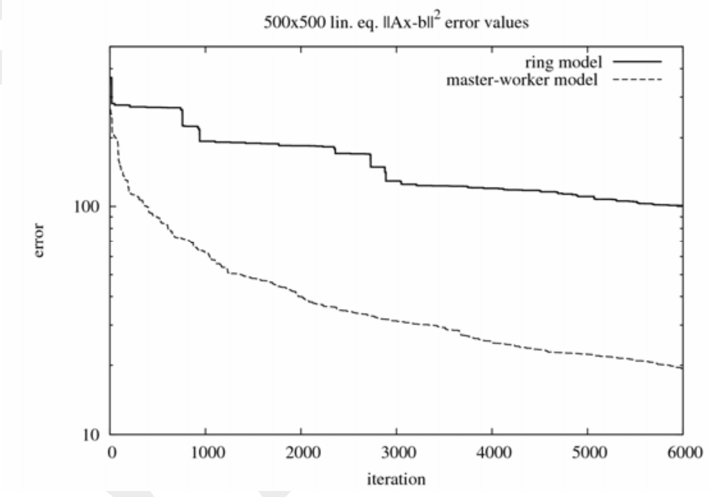

Figure 5. Convergence speed between topologies (n = 500, P = 16 cpus);

The difference between the ring and hierarchical algorithm is that the hierarchical algorithm results in smaller computational time and better convergence. As it can be seen in *Figure 5* the ring solution (solid line) has a minor gradient whereas the hierarchical solution (dotted line) has a steeper gradient. This is because at the ring model the corrective effect of a new solution reaches the previous node in *P–1* steps, while in the hierarchical model the corrective effect is achieved in one step. Moreover, on the ring model we can only take the result of one or two neighbours into consideration.

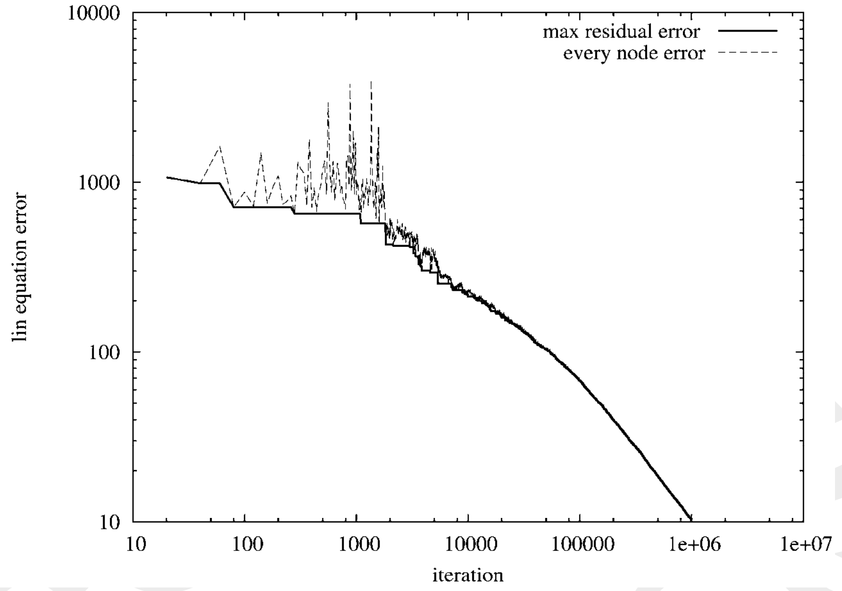

Figure 6. The results of working nodes (dotted) and the minimal residual error (solid line) (n = 10000, P = 88, log-log scale)

The hierarchical model has better convergence features. At every iteration loop the master node sends the best solution vector for the workers. This adds some genetic features to the algorithm [5]. If we examine the details on *Figure 3* and *Figure 4* steps in the curve can be seen. It has to be noted that in a P-processor master-worker model only P-1 processors solve the linear equation.

Let us focus now on this hierarchical solution. If the convergence curves are observed it can be noticed that all of the solutions are similar, and the final solution has always the same order of error in every case. The differences between the exact values of the errors are unfortunately caused by the pseudo-random number generator. In this algorithm only an approximate solution is achieved, not the exact vector.

If the problem is examined from another point of view and the execution time of the algorithm is recorded, the results shown in *Figure 7* are achieved. If more computing nodes are used the running time is expected to decrease.

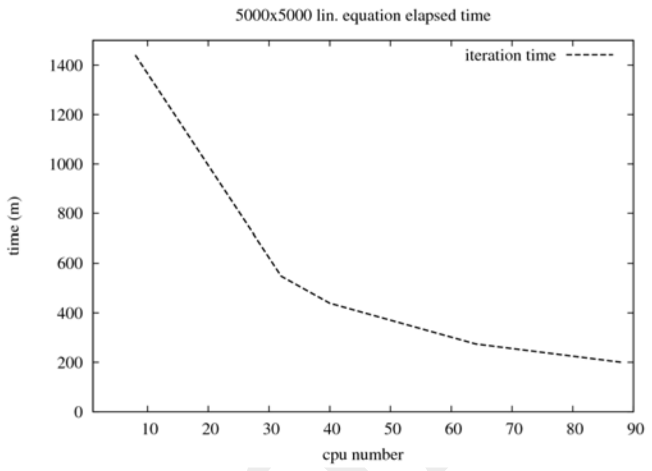

Figure 7. Time of execution and processor numbers (n = 5000);

In these cases only parallel execution provides results in an acceptable time. At some points larger than linear relative speed-up can be achieved, as it is shown in *Fig 8.* But when the number of processors grows, the effective speedup and efficiency decreases as the worker nodes report their own solutions and the load of the master node grows. More precise load balancing can be used, but the size of the problem and the communication delays prevent further advances. For better results another method has to be used.

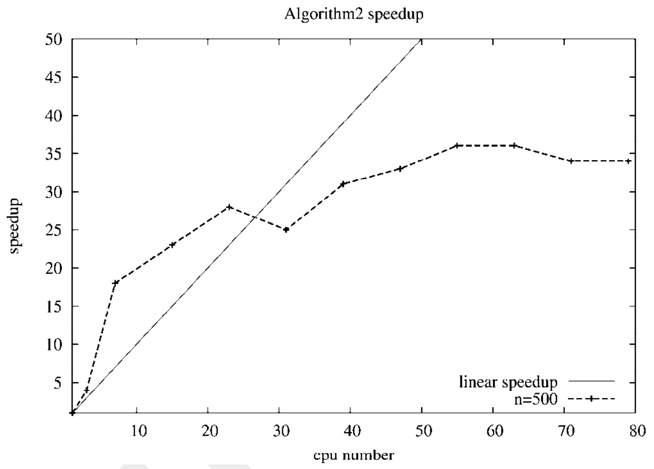

Figure 8. Algortihm2 relative speedup (n = 500)

3. Results

Above a new type of algorithm for the solution of a linear equation system on heterogeneous clusters has been presented. The algorithm is based on a residual minimisation technique with master-worker solutions. The algorithm has some genetic features because the new, better vectors are made from a group of good vectors as seen in *Figure 8* for extra speed-up.

Computer tests have proved the theoretical results; parallel implementation is much better than the sequential one. We get a considerable speed-up effect using a parallel computer.

The goal of creating these algorithms has been basic research but the solution of bad condition linear equations is a useful method for several practical and realistic problems. Optimization, finite element methods, control or simulation problems are often based on large dense linear equations.

We have to note that only the simplest algorithm has been tested. The test with more effective algorithms will be the subject of a forthcoming work.

4. References

[1] Louis A. Hageman, Davis M. Joung: *Applied Iterative Methods*, Computer Science and Applied Mathematics, Academic Press, (1981).

[2] P. G. Ciarlet: *Introduction à l'analyse numérique matricielle et à l'optimisation*, MASSON, Paris, (1982).

[3] G. Golub, A. Greenbaum, M. Luskin, eds., *Recent Advances is Iterative Methods,* The IMA Volumes in Math. and its Applications Vol.60., Springer Verlag, (1994).

[4] G. Molnárka, N. Varjasi: *Parallel algorithm for solution of general linear systems of equations*, Informatika a felsőoktatásban 2005, Debrecen ISBN 963 472 909 6, pp.176.

[5] G. Molnárka: *A scalable parallel algorithm for solving general linear system equations*, 77th GAMM annual meeting 2006, Berlin (2006) pp.441.

[6] N. Varjasi, *Parallel Algorithm for linear equations with different network topologies,* Proceedings of International e-Conference on Computer Science (IeCCS) 2006 in Lecture Series on Computer and Computational Sciences, (2007) pp. 502-505, Brill Academic Publishers, ISBN 978-90-04-15592-3

[7] M. Matsumoto and T. Nishimura, Mersenne Twister: *A 623-dimensionally equidistributed uniform pseudorandom number generator,* ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January (1998) pp. 3-30

[8] A. Basermann, B. Reichel, C. Schelthoff, *Preconditioned CG methods for sparse matrices on massively parallel machines*, in Parallel Computing 23. (1997) pp. 381-398

[9] E. J. H. Yero, M. A. A Henriques, *Speedup and scalability analysis of Master-Slave applications on large heterogeneous clusters,* in Journal of Parallel and Distributed Computing 67. (2007) pp. 1155-1167

[10] I. S. Duff, H. A. van der Vorst, *Developments and trends in the parallel solution of linear systems,* in Parallel Computing 25. (1999) pp. 1931-1970