Performance Analysis of Distributed Iterative Linear Solvers

7th WSEAS Int. Conf. on MATHEMATICAL METHODS and COMPUTATIONAL TECHNIQUES IN ELECTRICAL ENGINEERING, Sofia, 27-29/10/05 (pp194-199)

Performance Analysis of Distributed Iterative Linear Solvers

W.M. ZUBEREK and T.D.P. PERERA
Department of Computer Science
Memorial University
St.John's, Canada A1B 3X5

Abstract: The solution of large, sparse systems of linear equations is an inherent part of many computational methods in science and engineering. For such systems, iterative methods are often more attractive than direct methods because of their small (and constant) memory requirements. Also, the performance of iterative solvers can easily be improved by using distributed systems. A common performance characteristic of distributed applications is their speedup which is usually defined as the ratio of the execution time of an application on a single processor to the execution time of the same workload on a P–processor system. The paper estimates the speedup of distributed linear iterative solvers, analyzes the influence of different communication schemes on the speedup, and compares the estimates with the measurements of real distributed programs.

Key-Words: Distributed computing, speedup, computation-to-communication ratio, sparse systems of linear equations, iterative methods.

1 Introduction

Advances in many areas of science and engineering, business and even medicine, depend on efficient solution of increasingly complex problems and this, in turn, depends on availability of high-performance computer systems. There are two basic ways of increasing the performance of computer systems; one approach increases the performance of a uniprocessor system, while the other improves system's performance by increasing the number of processors. The current computer technology favors multiprocessor systems because they are more economical [5].

Distributed systems are often considered as less expensive and more easily available alternatives to parallel systems [14]. The Beowulf cluster [17] is probably the most popular example of a system composed of (many) standard (or "off-the-shelf") components, connected by a communication medium that exchanges messages among the components of the system. Different forms of communication medium are used in distributed systems, from high–speed specialized interconnects to the internet.

Due to steadily increasing performance of microprocessors, which, for several decades, has been doubling every 18 months (the so called Moore's Law [10]) and to improving communication bandwidth, distributed computing is becoming an attractive platform for high-performance computing. Indeed, a recent survey of the most powerful supercomputing systems shows that seven out of 10 most powerful systems are clusters [11], and the list of 500 most powerful systems includes 208 clusters. The trend towards cluster computing is expected to continue.

Although the number of practical applications of distributed computing is still somewhat limited [4] and the challenges – in particular, the standardization – are still significant, there are some spectacularly successful examples of large–scale distributed computing, such as SETI@home [19] or ClimatePrediction [18] projects, in which hundreds of thousands and even millions of processors are performing coordinated computations within the same project.

In distributed applications, the total workload is divided among the processors of the system with an expectation of concurrent execution of the distributed tasks. One of the main performance characteristics of a distributed application is its speedup [16], which is usually defined as the ratio of the application's execution time on a single processor, $T(1)$, to the execution time of the same workload on a system composed of P processors, $T(P)$:

$$S(P) = \frac{T(1)}{T(P)}.$$

The speedup depends upon a number of factors which include the number of processors and their performances, the connections between the processors, the algorithm used for the distribution of the workload, etc. Some of these factors may be difficult to take into account when estimating the speedup of a distributed application. Therefore, in many cases, a simplified analysis is used to characterize the steady–state behavior of an application. This simplified analysis is based on a number of assumptions, such as a uniform distribution of workload among the processors, constant communication times, and so on.

This paper estimates the speedup of iterative solvers of (large) sparse systems of linear equations. It shows the speedup of distributed solvers as a function of the number of processors used and the "computation–to–communication ratio," that relates the performance of processors to the bandwidth of the communication medium used. The speedup estimates are compared with measurements of implemented distributed linear solvers. Effects of different communication schemes on the performance of speedup of distributed iterative solvers are also discussed.

2 Distributed Iterative Solvers

The solution of large, sparse systems of linear equations is an inherent part of many scientific applications. Large systems of sparse linear equations arise in applications of finite element and finite difference approximations, they are used in analysis of electronic circuits, in the steady–state solutions of discrete systems whose behavior is represented by Markov chains, and many other cases. For large sparse systems of equations, iterative methods [1], [7], [9] are more attractive than direct methods because they are less demanding with respect to memory and can require significantly less computational power. The standard Gaussian elimination applied to a sparse system typically leads to fill–ins, so that the total number of operations required for a solution of a system of N equations is estimated as $N^{7/3}$ [15]; for a system of 1000 equations with the density of 0.01 (i.e., with the average of 10 nonzero elements in each equation), the computational requirement of Gaussian elimination is equivalent to an iterative 1000-step solution of the same system of equations. For 10,000 equations (with the same number of nonzero elements per equation), iterative solution with 1000 steps is computationally more than 10 times less demanding than the direct Gaussian elimination. Distributed computing can further reduce the solution time of the iterative approach. Iterative approach can also be used to improve the accuracy of a direct solution of large systems of equations, when cumulative effects of rounding errors can distort the results [6]. Iterative approach can be very attractive for applications in which a fast, approximate solution is needed, that can be obtained in just a few iteration steps. On the other hand, the convergence of the iterative approach can be a troublesome issue, and some additional techniques may be needed to improve the convergence of the iterative process [3].

For distributed processing, the (large number of) equations is divided into approximately equal sections allocated to different processors assuming that all equations have a similar number of nonzero elements and that the processors have similar performance characteristics (if these assumptions are not valid, a different scheme of load distribution is needed). After distributing the system of equations among the processors, the iterative process repeatedly executes the following three consecutive steps:

1. the current approximation to the solution is distributed to all processors,
2. sections of the next approximation to the solution are calculated (concurrently) by the processors,
3. the new approximation is collected from all processors and the convergence is checked.

Although there are several techniques that can be used for iterative solution of (sparse) linear systems, the Gauss-Seidel method [6] has been chosen because it is simple to implement and for some applications it converges to the solution regardless of the initial approximation. The discussion and all examples used in this paper are based on the Gauss-Seidel method, however, only minor changes are needed to adjust the presentation to a different iterative method.

In each iteration, the first step typically uses a broadcast (or multicast) operation, and it can be assumed that its execution time, T_b, does not depend upon the number of processors (in fact, it usually depends on this number, but this dependence is ignored here as it is rather inessential). The last step requires a transfer from each processor to a single processor which performs the convergence check, so the transfers are performed sequentially. It is assumed that the total execution time of this step is equal to $P * T_c$, where P is the number of processors and T_c is the communication time of a single processor. Although T_c depends upon P, the dependence is not ver strong [13], and is neglected here.

Let T_s denote the total (sequential) computation time of one iteration. The (approximate) time of a distributed execution of a single iteration is then:

$$T(P) \;=\; T_b + T_s/P + P * T_c.$$

For simplicity, it can also be assumed that $T_b = T_c$, which is an oversimplification, but not very significant, as it appears. With this additional assumption, the speedup is:

$$S(P) \;=\; \frac{T_s}{T_s/P + (P+1) * T_c}.$$

Let $r_{comp/comm}$ denote the ratio T_s/T_c, i.e., the ratio of total (sequential) computation time (per iteration) to the communication associated with a single processor (such a ratio makes sense only for programs with cyclic behavior). Then the speedup becomes a function of two variables, P and $r_{comp/comm}$:

$$S(P) \;=\; \frac{r_{comp/comm}}{r_{comp/comm}/P + P + 1}.$$

Fig.1 shows the values of $S(P)$ for $P = 2, ..., 50$ and for $r_{comp/comm} = 10, ..., 100$.

Reasonable speedups (around 5) can be obtained only when the computation–to–communication ratio is sufficiently high. The best speedup is obtained for a rather small number of processors (5 to 10); for a larger number of processors, the execution time actually increases as it is dominated by the communication time.

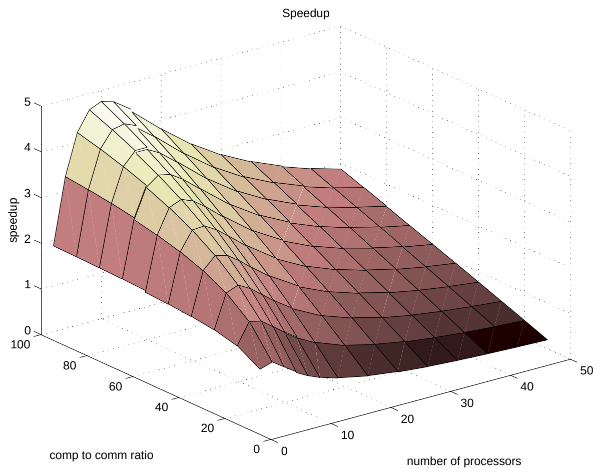

Fig.1. Speedup of distributed iterative solvers.

It should be observed that the simplifying assumptions are not really important because they affect terms which do not have significant influence on the speedup, especially for large values of P.

For larger values of P, the communication time becomes the dominating term in speedup estimation of linear solvers. Therefore the collection of results (step (3)) can be organized in a hierarchical, 2–level manner, in which first the results of computations are collected in groups of, say, K processors, and then the results of groups are combined together. It can be shown that the number of groups that minimizes the total communication time is equal to $\sqrt{P}$, and then the speedup becomes:

$$S(P) \;=\; \frac{r_{comp/comm}}{r_{comp/comm}/P + 2 * \sqrt{P} + 1}.$$

Fig.2 shows the values of $S(P)$ for $P = 5, ..., 50$ and for $r_{comp/comm} = 10, ..., 100$ for this modified approach.

In Fig.2, the relation between the speedup and the number of processors is quite different than in Fig.1; better speedup values can be obtained and the speedup is much less sensitive to the number of processors. For large values of P (i.e., P greater than 100), further improvement can be obtained by additional levels of the hierarchical collection of results.

If the broadcast operation of step (1) is replaced by a sequence of P unicast send operations, the

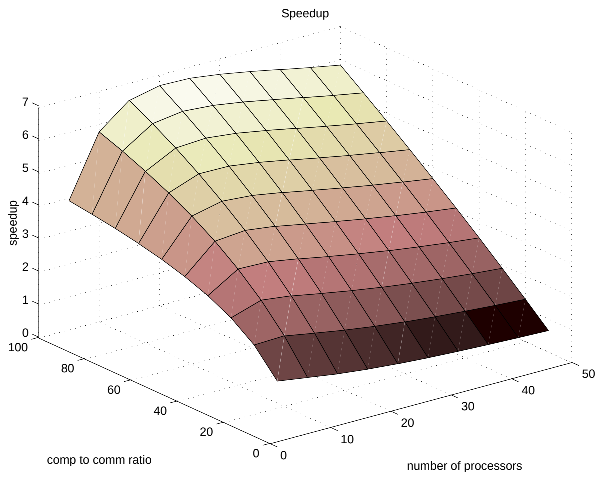

Fig.2. Speedup of modified iterative solvers.

speedup is not affected in a significant way. This is due to a staggered execution phase in which the consecutive processors start their computation (step (2)), one after another, right after receiving the current approximation to the solution. The return of results is staggered in a similar way. Consequently, if the time of sending the approximation to the solution is denoted by T_a, the execution time of one distributed iteration is:

$$T(P) \;=\; P * \max(T_a, T_c) + T_s/P + \min(T_a, T_c)$$

which, for $T_a \approx T_c$, simplifies to:

$$T(P) \;=\; T_s/P + (P+1) * T_c,$$

the same formula as at the beginning of this section (with $T_b = T_c$).

3 Experimental Results

Practical speedups, obtained for three systems of sparse linear equations with different sparsity patterns, are shown in Fig.3.

In Fig.3, the density, i.e., the ratio of the number of nonzero elements to the total number of elements, of systems (1) and (2) is of the order of 0.015, and that of the system (3) is 0.025).

All three systems of equations have banded structure, with (almost) uniform sparsity of the equations:

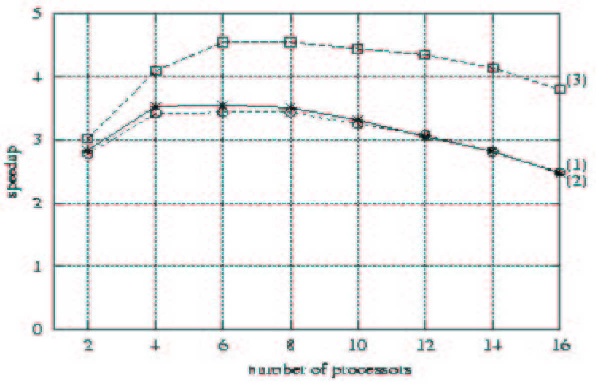

Fig.3. Speedups of distributed iterative solvers.

$$\left[\begin{array}{cccccccccc}
\ast & \ast & & & & & & & & \\
\ast & \ast & \ast & & & & & & & \\
 & \ast & \ast & \ast & & & & & & \\
 & & \ast & \ast & \ast & & & & & \\
 & & & \ast & \ast & \ast & & & & \\
 & & & & \ast & \ast & \ast & & & \\
 & & & & & \ast & \ast & \ast & & \\
 & & & & & & \ast & \ast & \ast & \\
 & & & & & & & \ast & \ast & \ast \\
 & & & & & & & & \ast & \ast
\end{array}\right]
\qquad
\left[\begin{array}{cccccccccc}
\ast & & \ast & & & & & & & \\
 & \ast & & \ast & & & & & & \\
\ast & & \ast & & \ast & & & & & \\
 & \ast & & \ast & & \ast & & & & \\
 & & \ast & & \ast & & \ast & & & \\
 & & & \ast & & \ast & & \ast & & \\
 & & & & \ast & & \ast & & \ast & \\
 & & & & & \ast & & \ast & & \ast \\
 & & & & & & \ast & & \ast & \\
 & & & & & & & \ast & & \ast
\end{array}\right]$$

$$(\mathbf{1}) \qquad\qquad (\mathbf{2})$$

$$\left[\begin{array}{cccccccccc}
\ast & \ast & & & \ast & & & & & \\
\ast & \ast & \ast & & & \ast & & & & \\
 & \ast & \ast & \ast & & & \ast & & & \\
 & & \ast & \ast & \ast & & & \ast & & \\
\ast & & & \ast & \ast & \ast & & & \ast & \\
 & \ast & & & \ast & \ast & \ast & & & \ast \\
 & & \ast & & & \ast & \ast & \ast & & \\
 & & & \ast & & & \ast & \ast & \ast & \\
 & & & & \ast & & & \ast & \ast & \ast \\
 & & & & & \ast & & & \ast & \ast
\end{array}\right]$$

$$(\mathbf{3})$$

The distribution of the workload among the processors assigns the same number of equations to each processor (except of the “main processor”).

The results were obtained on a cluster of PCs connected by Ethernet to a central switch, so the communication times for all pairs of processors were practically the same. The MPI communication library [8] was used for setting up the distributed system and exchanging the messages.

It can be observed that the results in Fig.3 follow the general outline shown in Fig.1. All three systems have the same size (so their communication times are also similar), but the computation times for system (3) are approximately two times greater than that of systems (1) and (2) (so $r_{comp/comm}$ for

(3) is approximately two times greater than that for (1) and (2)), which explains the increased values of the speedup obtained for system (3).

4 Distributed Convergence

Although it might be expected that the number of processors, P, affects the convergence of the iterative process, apparently this is not the case if the system is sufficiently large.

Fig.4 shows the (total) number of iterations needed for distributed iterative solution of systems of 500 (1), 1500 (2) and 2500 (3) equations with sparsity pattern (3).

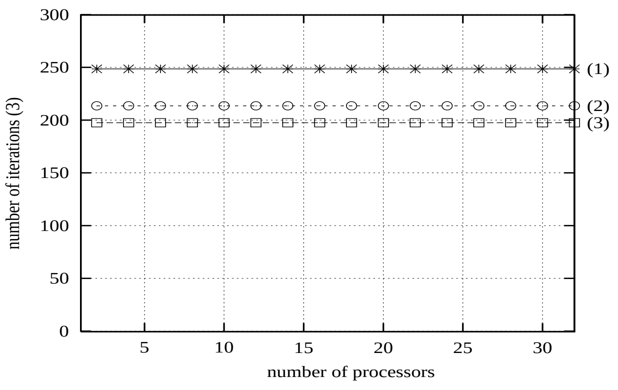

Fig.4. Numbers of iterations of distributed iterative solvers for systems of 500 (1), 1500 (2) and 2500 (3) 5–diagonal equations (sparsity pattern (3)).

In Fig.4, the number of required iterations is practically independent of the number of processors. Moreover, this number of iterations actually decreases with increased size of the system of equations; this can be due to improved overall convergence properties of the iterative process when the number of equations assigned to each processor increases. Fig.5 shows the dependence between the size of the system of equations and the number of iterations needed for its solution for systems with sparsity pattern (3).

A similar approach can be used to study the effects of relaxation, preconditioners, and other techniques used for improving the convergence of the iterative process.

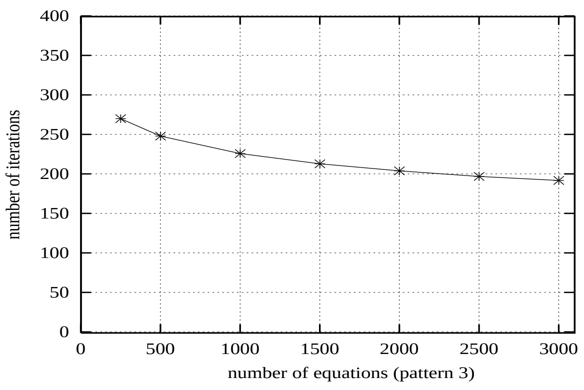

Fig.5. The number of iterations of distributed iterative solvers as a function of the number of equations (sparsity pattern (3)).

5 Concluding Remarks

This paper analyzes, in very general terms, the speedup that can be obtained in distributed iterative solvers of large sparse systems of linear equations. The paper shows that the straightforward distribution of equations among the processors of a distributed system introduces a limitation of the speedup, and that using more processors may actually reduce the speedup and increase the solution time.

The very simple workload distribution (based on the same numbers of equations assigned to different processors) is satisfactory only when the sparsity structure is uniform over the set of equations (as shown in Section 3). For other systems of linear equations, for example, for systems of equations with a “margin” (which is characteristic for the solution of Markov chains) or block-diagonal structures:

```
┌ **          ┐   ┌ ***         ┐
│ ***         │   │ ****        │
│  ***        │   │ ********    │
│   ***       │   │  *******    │
│    ***      │   │   ******    │
│     ***     │   │   ******    │
│      ***    │   │   *******   │
│       ***   │   │   ********  │
│        ***  │   │       ****  │
└ **********  ┘   └        ***  ┘
```

a different workload distribution is needed, in which approximately equal numbers of nonzero elements are assigned to processors rather then the same number of equations, otherwise a significant imbalance can occur and reduce the performance of the solver.

The solution of systems of linear equations is just one example of applications which may benefit from distributed environments. Other applications which are well suited for distributed execution include:

- Complex modeling and simulation techniques that increase the accuracy of results by increasing the number of random trials; the trials can be run concurrently on many processors, and the results combined to achieve greater statistical significance.
- Applications that require exhaustive search through a huge number of results that can be distributed over the many processors, such as drug screening [4].
- Simulations of complex systems (such as VLSI designs) in which the design is partitioned into a number of smaller parts, and these parts are simulated concurrently on different processors [12].

It is expected that in future applications, traditional distributed systems will continue to be used in specialized domains, such as transaction processing for banking applications, in mainstream applications, however, grid computing is emerging as the next evolutionary platform for large–scale high–performance computing [2].

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through grant RGPIN-8222.

References

[1] O. Axelsson, *Iterative solution methods*; Cambridge University Press 1994.

[2] F. Berman, G. Fox, T. Hey, *Grid computing: making the global infrastructure a reality*; J. Wiley 2003.

[3] J.J. Dongarra, I.S. Duff, D.C. Sorenson, H.A. van der Horst, *Numerical Linear Algebra for High-Performance Comp-uters*; SIAM 1998.

[4] L. Erlander, "Distributed computing: an introduction"; *Extreme Tech*, April 4, 2002.

[5] V.K. Garg, *Principles of distributed systems*; Kluwer Academic Publ. 1998.

[6] G.H. Golub, C.F. van Loan, *Matrix computations*; The Johns Hopkins Univ. Press 1983.

[7] A. Greenbaum, *Iterative methods for solving linear systems* (Frontiers in Applied Mathematics 17); SIAM 1997.

[8] W. Gropp, E. Lusk, A. Skjellum, *Using MPI: portable parallel programming with the message–passing interface* (2-nd ed.); MIT Press 1999.

[9] W. Hackbusch, *Iterative solution of large sparse systems of equations* (Applied Mathematical Sciences 95); Springer-Verlag 1995.

[10] S. Hamilton, "Taking Moore's law into the next century"; *IEEE Computer Magazine*, vol.32, no.1, 1999, pp.43-48.

[11] R. Merritt, "Intel, clusters on the rise in 'Top 500 Supercomputer' list"; *EE Times Online*, November 18, 2003.

[12] R.A. Saleh, K.A. Gallivan, M-C. Chang, I.N. Hajj, D. Smart, T.N. Trick, Parallel circuit simulation on supercomputers; *Proceedings of the IEEE* vol.77, no.12, 1989, pp.1915-1931.

[13] M.R. Steed, M.J. Clement, "Performance prediction of PVM programs"; Proc. 10-th Int. Parallel Processing Symposium (IPPS), 1996, pp.803-807.

[14] J.A. Stankovic, "Distributed computing"; in *Distributed computing systems*, IEEE CS Press 1994.

[15] H. van der Vorst, "Iterative methods for linear systems and implementation on parallel computers"; in *Iterative methods in scientific computing*; R.H. Chan, T.F. Chan, and G.H. Golub (eds.), Springer-Verlag 19999, pp.1-44.

[16] B. Wilkinson, *Computer Architecture – Design and Performance* (2-nd ed.); Prentice Hall 1996.

[17] The home of page of Beowulf clusters is "`www.beowulf.org`",

[18] The home page of ClimatePrediction is "`www.climateprediction.net`".

[19] The home page of the SETI@home project is "`setiathome.ssl.berkeley.edu`".