ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2018, Vol. 39, No. 4, pp. 570–574. © Pleiades Publishing, Ltd., 2018.

Analytical Estimation of the Scalability of Iterative Numerical Algorithms on Distributed Memory Multiprocessors

L. B. Sokolinsky*

(Submitted by E. E. Tyrtyshnikov)

South Ural State University (National Research University), Lenin prospekt, 76, Chelyabinsk, 454080 Russia

Received November 15, 2017

Abstract—This article presents a new high-level parallel computational model named BSF "— Bulk Synchronous Farm. The BSF model extends the BSP model to deal with the compute-intensive iterative numerical methods executed on distributed-memory multiprocessor systems. The BSF model is based on the master-worker paradigm and the SPMD programming model. The BSF model makes it possible to predict the upper scalability bound of a BSF-program with great accuracy. The BSF model also provides equations for estimating the speedup and parallel efficiency of a BSF-program.

DOI: 10.1134/S1995080218040121

Keywords and phrases: *Parallel computation model, bulk synchronous farm, BSF model, iterative algorithms, distributed memory, scalability bound.*

1. INTRODUCTION

One of the most important properties of a numerical algorithm designed for large-scale cluster systems is scalability. *Scalability* can be defined as a measure of a parallel system's capacity to decrease computation time in proportion to the number of processors. The upper bound of scalability is an integral characteristic of a parallel algorithm/program. The *upper bound of scalability* is the least number of processor nodes for which the speedup takes the maximal value. It is valuable to be able to estimate the upper bound of scalability in early phases of program development; the parallel computation model is a tool providing this possibility. A *model of computation* is a framework for specifying and analyzing algorithms or programs [1]. Many parallel computation models have been proposed for distributed-memory multiprocessors. The most famous of these models are the *BSP model family* (see [2–7]) and the *LogP model family* (see [8–14]). Most of these models are low-level models and require detailed description of the structure of the algorithm to the level of code in a programming language or pseudocode. This article extends the basic BSP (Bulk Synchronous Parallelism) model [15] to deal with the compute-intensive iterative numerical methods executed on distributed-memory multiprocessor systems. Iterative methods are an important class of numerical methods. An overview of various iterative methods can be found in [16–19]. The new parallel computation model proposed in this article was named *BSF "— Bulk Synchronous Farm*. The BSF model is a high-level parallel computation model based on the master-worker (master-slave) framework [20] and the SPMD (Single-Program-Multiple-Data) programming model [21, 22]. A distinctive feature of the BSF model is the ability to estimate the upper bound of scalability in the early stages of the algorithm design.

The rest of the article is organized as follows. In Section 2, the BSF parallel computation model presented in this paper is described. Section 3 introduces a cost metric for BSF-programs and provides equations for estimating the speedup and parallel efficiency of an algorithm before its implementation in a programming language. Moreover, a simple inequality to estimate the upper scalability bound of a BSF-program is deduced. Section 4 summarises the results and outlines some directions for future research.

*E-mail: leonid.sokolinsky@susu.ru

2. BSF COMPUTATIONAL MODEL

The *BSF* (*Bulk Synchronous Farm*) model is intended for multiprocessor systems with distributed memory. A *BSF-computer* consists of a collection of homogeneous computing nodes with private memory connected by a communication network delivering messages among the nodes. There is just one node called the *master-node* in a BSF-computer. The rest of the nodes are the *worker-nodes*. A BSF-computer must include at least one master-node and one worker-node.

A BSF-computer utilizes the *SPMD* programming model according to which all the worker-nodes executes the same program but process different data. A BSF-program consists of sequences of macro-steps and global barrier synchronizations performed by the master and all the workers. Each macro-step is divided into two sections: the master section and the worker section. The master section includes instructions performed by only the master. A worker section includes instructions performed by only the workers. The sequential order of the master section and the worker section within the macro-step is not important. All the worker nodes operate on the same data array, but the base address of the data assigned to the worker-node for processing is determined by the logical number of this node. A BSF-program includes the following sequential sections: initialization; iterative process; finalization.

Initialization is a macro-step in which the master and workers read or generate input data. Initialization is followed by barrier synchronization. The *iterative process* repeatedly performs its body until the exit condition checked by the master becomes true. In the *finalization macro-step*, the master outputs the results and ends the program.

The *body of the iterative process* includes the following macro-steps: 1) sending orders (from master to workers); 2) processing orders (by workers); 3) receiving results (from workers to master); 4) evaluating the results (by master). In the first macro-step, the master sends the same orders to all workers. Then, the workers execute the received orders (the master is idle at that time). All the workers execute the same program code but operate on different data with a base address which depends on the worker-node number. Therefore, all workers spend the same amount of time on calculation. There are no data transfers between nodes during order processing. In the third step, all workers send the results to the master. Next, global barrier synchronization is performed. During the fourth step, the master evaluates the results it has received. The workers are idle at this time. After evaluation of the results, the master checks the exit condition. If the exit condition is true, then the iterative process is finished, otherwise the iterative process is continued.

3. EVALUATION OF BSF-PROGRAM SCALABILITY

The main characteristic of scalability is the speedup. For a parallel program, *speedup* $a(K)$ can be defined as a ratio of execution time T_1 on one computing node to execution time T_K on K computing nodes:

$$a(K) = T_1/T_K. \tag{1}$$

Parallel efficiency is another important characteristic of scalability. *Parallel efficiency* $e(K)$ can be defined as a ratio of speedup $a(K)$ to the number K of processors:

$$e(K) = a(K)/K. \tag{2}$$

This section offers a cost metric which can be used to estimate the scalability of a BSF-program. We assume that time spent on initialization and finalization of a BSF-program is negligible compared to the cost of iterative process execution. The cost of an iterative process is equal to the sum of the costs of separate iterations. Therefore, to estimate the execution time of a BSF program, it is sufficient to obtain an estimation of the execution time of a single iteration. For this purpose, the following main parameters of the BSF model are introduced:

K: the number of worker-nodes;

L: an upper bound on the latency, or delay, incurred in communicating a message containing one byte from its source node to its target node;

t_s: the time that the master-node is engaged in sending one order to one worker-node, excluding latency;

t_w: the time a BSF-computer with one worker-node needs to perform one order;

t_r: the total time that the master-node is engaged in receiving the results from all worker-nodes, excluding latency;

t_p: the total time that the master-node is engaged in evaluating the results received from all worker-nodes.

The global barrier synchronization performed in iterative process is implemented by the master waiting for completion of reading all messages from workers, and therefore, it does not require an additional cost.

The time T_1 needed for the execution of a single iteration by a BSF-computer with one master-node and one worker-node can be calculated as follows: $T_1 = t_s + t_w + L + t_p + t_r + L$, which is equivalent to

$$T_1 = 2L + t_s + t_w + t_p + t_r. \tag{3}$$

Now, let us calculate the time T_K a BSF-computer with one master-node and K worker-nodes needs to execute a single iteration. All of the workers receive the same message from the master, so the total time for sending messages from the master to the workers is equal to $K(L + t_s)$. All of the workers perform the same program code on their own data segment, so the time of order execution by a group with K workers is equal to t_w/K. The resulting data volume produced by the workers is a parameter of the task and does not depend on K, so the total time needed for sending messages from the workers to the master is equal to $K \cdot L + t_r$. The time needed for the master to process the results received from the workers is also a task parameter and does not depend on the number of workers. Thus, the total execution time of one iteration in a BSF-computer with one master and K workers can be calculated as follows: $T_K = K(L + t_s) + t_w/K + K \cdot L + t_r + t_p$, which is equivalent to $T_K = 2L \cdot K + t_s \cdot K + t_r + t_p + t_w/K$. By reducing the right-hand side of the equation to the common denominator, we obtain

$$T_K = \frac{K^2(2L + t_s) + K(t_r + t_p) + t_w}{K}. \tag{4}$$

Using equations (1), (3) and (4), we obtain the following equation for the speedup of BSF-program:

$$a(K) = \frac{K(2L + t_s + t_r + t_p + t_w)}{K^2(2L + t_s) + K(t_r + t_p) + t_w}. \tag{5}$$

Let us analyze $a(K)$ as a function depending on $K \geq 1$. The function $a(K)$ takes the value 1 at $K = 1$ which is concordant with the definition of the speedup and equation (1). The function $a(K)$ is a continuous and positive definite function on the interval $[1; +\infty)$. Let us find the derivative of the function $a(K)$:

$$a'(K) = \frac{(2L + t_s + t_r + t_p + t_w)(t_w/K^2 - 2L - t_s)}{(K(2L + t_s) + t_r + t_p + t_w/K)^2}. \tag{6}$$

It follows from (6) that the derivative takes the value 0 at the point $K_0 = \sqrt{t_w/(2L + t_s)}$. Moreover, the derivative takes positive values for $K < K_0$ and negative values for $K > K_0$. This indicates that the point $K = K_0$ is the point at which the BSF-program speedup takes the maximum value. Thus, we may make a conclusion that the value K_0 is the *upper bound of the BSF-program scalability*: $K \leq K_0$. Note that the upper bound of BSF-program scalability does not depend on the amount of time that the master is engaged in receiving and evaluating worker results.

One more important characteristic of a parallel program is parallel efficiency, calculated by equation (2). Let us estimate the efficiency of a BSF-program. Using equations (2) and (5) we obtain

$$e = \frac{2L + t_s + t_r + t_p + t_w}{K^2(2L + t_s) + K(t_r + t_p) + t_w}.$$

Assuming $K \gg 1$, we have

$$\frac{2L + t_s}{K^2(2L + t_s) + K(t_r + t_p) + t_w} \approx 0 \quad \text{and} \quad \frac{t_r + t_p}{K^2(2L + t_s) + K(t_r + t_p) + t_w} \approx 0.$$

Hence,

$$e(K) \approx e_0 = \frac{t_w}{K^2(2L + t_s) + K(t_r + t_p) + t_w}$$

for $K \gg 1$, and we receive the following approximate equation to estimate the parallel efficiency of a BSF-program: $e(K) = e_0$.

4. CONCLUSION

In this article, the new BSF (Bulk Synchronous Farm) model of parallel computations was introduced. The BSF model is intended for evaluating iterative numerical algorithms designed for distributed memory multiprocessors. One distinctive feature of the BSF model is the ability to evaluate the scalability of an algorithm in the early phases of its development. The architecture of a BSF-computer was described. A BSF-computer includes one master-node and several worker-nodes connected by a communication network. The structure of a BSF-program was specified. A BSF-program uses the SPMD (Single-Program-Many-Data) model according to which all the worker-nodes execute the same program but process different data. The execution of a BSF-program is divided into iterations. In each iteration, the master sends the orders to the workers; the workers execute the orders and send the results to the master; the master processes the results and checks the exit condition; if the condition is not satisfied, then the master sends new orders to the workers, beginning the next iteration, otherwise, the calculations are stopped. A cost metric was constructed for BSF-programs. This metric offers the following simple estimation for the upper bound of scalability: $K \leq K_0$, where K is the number of worker-nodes, L is the latency, t_w is the time a BSF-computer with one worker-node needs to execute the order, and t_s is the time needed to send an order to one worker-node, excluding latency.

A BSF-implementation of the *NSLP algorithm* [23] was performed to validate the theoretical studies presented in this article. The NSLP algorithm is used to solve large-scale non-stationary linear programming problems. A BSF-implementation of the NSLP algorithm is described in article [24]. The source code of this implementation is freely available on Github, at `https://github.com/leonid-sokolinsky/BSF-NSLP`. The results of the computational experiments presented in [24] show that the BSF model accurately predicts the upper bound of scalability for the NSLP algorithm implemented as a BSF-program.

Future work concerning the BSF model includes the following directions. First, develop a formalism to describe BSF-programs through higher-order functions. Next, design and implement a BSF skeleton for the rapid development of BSF-programs in C++ using the MPI-library. Finally, validate the BSF model with different well-known iterative numerical methods.

5. ACKNOWLEDGMENTS

This research was partially supported by the Russian Foundation for Basic Research (project No. 17-07-00352a), by the Ministry of Education and Science of Russian Federation (gov. order No. 2.7905.2017/8.9) and by the Government of the Russian Federation according to Act 211 (contract No. 02.A03.21.0011).

REFERENCES

1. G. Bilardi and A. Pietracaprina, "Models of computation, Theoretical," in *Encyclopedia of Parallel Computing* (Springer US, Boston, MA, 2011), pp. 1150–1158. doi 10.1007/978-0-387-09766-4_218
2. L. G. Valiant, "A bridging model for parallel computation," Commun. ACM **33** (8), 103–111 (1990). doi 10.1145/79173.79181
3. F. M. Auf der Heide and R. Wanka, "Parallel bridging models and their impact on algorithm design," in *Proceedings of the International Conference on Computational Science "— ICCS'01, Part II,* Lect. Notes Comput. Sci. **2074**, 628–637 (2001). doi 10.1007/3-540-45718-6_68
4. L. G. Valiant, "A bridging model for multi-core computing," J.Comput. Syst. Sci. **77**, 154–166 (2011). doi 10.1016/j.jcss.2010.06.012
5. V. Blanco, J. A. Gonzalez, C. Leon, C. Rodriguez, G. Rodriguez, and M. Printista, "Predicting the performance of parallel programs," Parallel Comput. **30**, 337–356 (2004). doi 10.1016/j.parco.2003.11.004

6. A. V. Gerbessiotis, "Extending the BSP model for multi-core and out-of-core computing: MBSP," Parallel Comput. **41**, 90–102 (2015). doi 10.1016/j.parco.2014.12.002
7. H. Cha and D. Lee, "H-BSP: a hierarchical BSP computation model," J. Supercomput. **18**, 179–200 (2001). doi 10.1023/A:1008113017444
8. D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: towards a realistic model of parallel computation," in *Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming "— PPOPP'93* (ACM Press, New York, 1993), pp. 1–12. doi 10.1145/155332.155333
9. A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: incorporating long messages into the LogP model for parallel computation," J. Parallel Distrib. Comput. **44**, 71–79 (1997). doi 10.1006/jpdc.1997.1346
10. G. Liu, Y. Wang, T. Zhao, J. Gu, and D. Li, "mHLogGP: a parallel computation model for CPU/GPU," in *Network and Parallel Computing, Proceedings of the 9th IFIP International Conference NPC 2012, Gwangju, Korea, Sept. 6–8, 2012,* Ed. by J. J. Park, A. Zomaya, and S. S. Yeo (Springer, Berlin, Heidelberg, 2012), pp. 217–224. doi 10.1007/978-3-642-35606-3_25
11. F. Lu, J. Song, and Y. Pang, "HLognGP: a parallel computation model for GPU clusters," Concurr. Comput.: Practice Experience **27**, 4880–4896 (2015). doi 10.1002/cpe.3475
12. F. Ino, N. Fujimoto, and K. Hagihara, "LogGPS: a parallel computational model for synchronization analysis," ACM SIGPLAN Not. **36** (7), 133–142 (2001). doi 10.1145/568014.379592
13. K. W. Cameron, R. Ge, and X. Sun, "logNP and log3P: accurate analytical models of point-to-point communication in distributed systems," IEEE Trans. Comput. **56**, 314–327 (2007). doi 10.1109/TC.2007.38
14. L. Yuan, Y. Zhang, Y. Tang, L. Rao, and X. Sun, "LogGPH: a parallel computational model with hierarchical communication awareness," in *Proceedings of the 2010 13th IEEE International Conference on Computational Science and Engineering CSE'10* (IEEE Comput. Soc., Washington, DC, 2010), pp. 268–274. doi 10.1109/CSE.2010.40
15. A. Tiskin, "BSP (Bulk Synchronous Parallelism)," in *Encyclopedia of Parallel Computing* (Springer US, Boston, MA, 2011), pp. 192–199. doi 10.1007/978-0-387-09766-4_311
16. L. A. Hageman and D. M. Young, *Applied Iterative Methods* (Academic, New York, London, Toronto, Sydney, San Francisco, 1981).
17. C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations* (Soc. Ind. Appl. Math., Philadelphia, 1995). doi 10.1137/1.9781611970944
18. A. Hadjidimos, "A survey of the iterative methods for the solution of linear systems by extrapolation, relaxation and other techniques," J. Comput. Appl. Math. **20**, 37–51 (1987). doi 10.1016/0377-0427(87)90124-5
19. S. Ma and A. Chronopoulos, "Implementation of iterative methods for large sparse nonsymmetric linear systems on a parallel vector machine," Int. J. High Perform. Comput. Appl. **4** (4), 9–24 (1990). doi 10.1177/109434209000400402
20. S. Sahni and G. Vairaktarakis, "The master-slave paradigm in parallel computer and industrial settings," J. Global Optimiz. **9**, 357–377 (1996). doi 10.1007/BF00121679
21. F. Darema, D. A. George, V. A. Norton, and G. F. Pfister, "A single-program-multiple-data computational model for EPEX/FORTRAN," Parallel Comput. **7**, 11–24 (1988). doi 10.1016/0167-8191(88)90094-4
22. F. Darema, "SPMD computational model," in *Encyclopedia of Parallel Computing* (Springer US, Boston, MA, 2011), pp. 1933–1943. doi 10.1007/978-0-387-09766-4_26
23. I. Sokolinskaya and L. B. Sokolinsky, "On the solution of linear programming problems in the age of big data," in *Parallel Computational Technologies. PCT 2017,* Commun. Comput. Inform. Sci. **753**, 86–100 (2017). doi 10.1007/978-3-319-67035-5_7
24. I. Sokolinskaya and L. B. Sokolinsky, "Scalability evaluation of NSLP algorithm for solving non-stationary linear programming problems on cluster computing systems," in *Supercomputing, RuSCDays 2017,* Commun. Comput. Inform. Sci. **793**, 40–53 (2017). doi 10.1007/978-3-319-71255-0_4