Verification of BSF Parallel Computational Model

Nadezhda Ezhova*

South Ural State University, Chelyabinsk, Russia
ezhovana@susu.ru

Abstract. The article is devoted to the verification of the BSF parallel computing model. The BSF-model is an evolution of the "master-slave" model and SPMD-model. The BSF-model is oriented to iterative algorithms that are implemented in cluster computing systems. The article briefly describes the basics of the BSF-model and its cost metrics. The structure of the BSF program is shown in the form of a UML activity diagram. The simulator of BSF-programs, implemented in C++ language using the MPI-library, is described. The results of computational experiments confirming the adequacy of the cost metrics of the BSF-model are presented.

Keywords: parallel computational model · BSF · Bulk Synchronous Farm · scalability · parallel efficiency · distributed memory multiprocessors · simulation · validation

1 Introduction

The Sunway TaihuLight is the fastest supercomputer in the world according to the TOP500 List (edition of June 2017) [1]. It uses 40,960 processors, each of which contains 260 processing cores. The total system memory is 1.3 Petabytes; the peak performance exceeds 120 petaflops. Analysis of the growth dynamics of the performance of supercomputers (see Fig. 1) shows that in 8-9 years the most powerful supercomputer becomes an ordinary system, and that in 5-6 years we can expect the appearance of exascale computing. The emergence of such powerful multiprocessor computing systems brings to the fore issues related to the development of frameworks (templates) that allow creating highly scalable parallel programs oriented to systems with distributed memory. In this context, the most important problem is the development of parallel computation models allowing to estimate the scalability of the algorithm at an early stage of implementing. In paper [2], a new parallel computation model called BSF (Bulk-Synchronous Farm) was proposed. The BSF-model is an evolution of the "master-slave" model and focused on iterative algorithms executing on the

* The work was partially supported by the Government of the Russian Federation according to Act 211 (contract № 02.A03.21.0011.) and the Ministry of Education and Science of Russian Federation (government order 2.7905.2017/8.9).

cluster computing systems. In this paper, the issue concerning to a verification of the BSF model is discussed. The paper has the following structure. Section 2 is the description of the BSF-model. In Section 3, a cost metric of BSF-model is given. In Section 4, the simulator of BSF-programs is described and computational experiments confirming the adequacy of the cost metrics of the BSF-model are presented. In Section 5, the results are summarized and directions for further research are outlined.

Fig. 1. Performance dynamics of supercomputers in TOP500

2 BSF-Model

Parallel computation model BSF (Bulk Synchronous Farm) was proposed in [2]. It extends the "Master-slave" paradigm [3–5] and the BSP computational model [6]. The BSF-model is oriented on multiprocessor systems. BSF-computer is a set of homogeneous processor nodes with private memory connected by the network that provides data transfer from one node to another. Among the processor nodes, there is one called a *master* node. The remaining K nodes are called

slaves. *BSF-computer* must have at least one master node and one slave node ($K \geq 1$). The architecture of the BSF-computer is shown in Fig. 2.

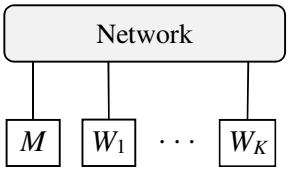


Fig. 2. BSF-computer. M – master; $W_1, \ldots, W_K$ – slaves

BSF-computer works according to the *SPMD* programming model [4, 7]. BSF-program consists of a sequence of macro-steps and global barrier synchronizations. Each macro-step is divided into sections of two types: *master sections* and *slave sections*. The order of such sections within the macro-step is not significant. The data processed by the particular slave is determined by the number of its node. BSF-program includes the following sequential sections (see Fig. 3):

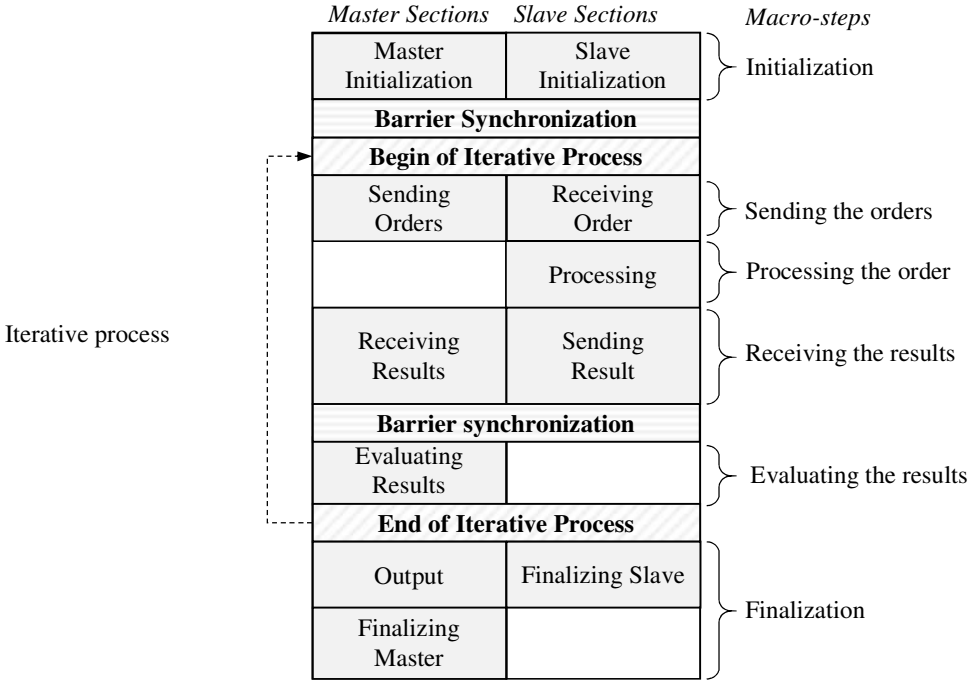


Fig. 3. BSF-program structure

– initialization;
– iterative process;
– finalization.

Initialization is a macro-step, during which the master and slaves read or generate input data. The initialization is completed by a 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. *Body of the iterative process* includes the following macro-steps:

1. sending the order (from master to slaves);
2. processing the order (slaves);
3. receiving the results (from slaves to master);
4. evaluating the results (master).

In the first macro-step, the master sends the same orders to all the slaves. Then, the slaves execute the received orders (the master is idle at that time). All the slaves execute the same program code but act on the different data with the base address depending on the slave-node number.

It means that all slaves spend the same time for calculating. During processing the order, there are no data transfers between nodes. The last is an important property of the BSF-model. In the third macro-step, all slaves send the results to the master. After that, global barrier synchronization is performed. During the last macro-step of iterative process, the master evaluates received results. The slaves are idle at that time. After result evaluations, the master checks the exit condition. If the exit condition is true then iterative process is finished, otherwise the iterative process is continued. The outline of the BSF-program in form of UML activity diagram is shown in Fig. 4.

The *BSF-model* was designed for the scalable iterative numerical methods that have a high computational complexity of iteration with relatively low cost of communications. A scalable iterative method is a method that allows iteration to be splitted into subtasks that do not require data exchanges. An example of such a method can be found in [8].

3 Cost Metric of BSF-Model

The BSF-model provides an analytical estimation of the scalability of a BSF-program. The main parameters of the model are [2]:

K: the number of slave-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: time that the master-node is engaged in sending one order to one slave-node excluding the latency;

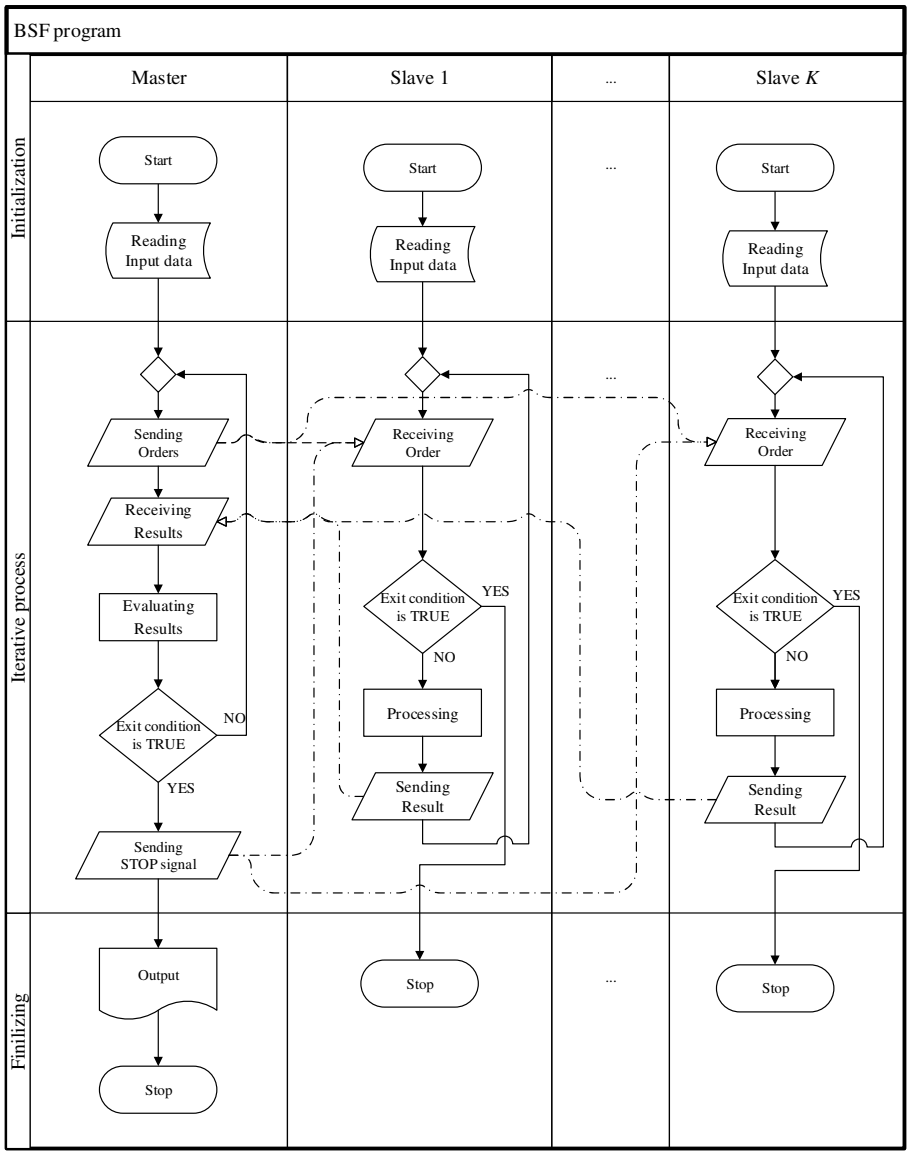


Fig. 4. BSF-program structure

t_v: time that a slave-node is engaged in execution an order within one iteration (BSF-model assumes that this time is the same for all the slave-nodes and it is a constant within the iterative process;
t_r: total time that the master-node is engaged in receiving the results from all the slave-nodes excluding the latency;
t_p: total time that the master-node is engaged in evaluating the results received from all the slave-nodes.

Let's denote $t_w = K \cdot t_v$ - summarized time which is spent by slave-nodes for order executions (without taking into account the effect of parallelization). Then, the upper bound of a BSF-program scalability can be estimated by the following inequality [2]:

$$K \leq \sqrt{\frac{t_w}{2L + t_s}} \tag{1}$$

The speedup of BSF-program can be calculated by the following equation [2]:

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

One more important property of a parallel program is the parallel efficiency. The parallel efficiency of a BSF-program can be calculated by the following approximate equation [2]:

$$e \approx \frac{1}{1 + (K^2(2L + t_s) + K(t_r + t_p))/t_w}. \tag{3}$$

4 Verification of BSF-Model

To verify the cost metrics of the BSF-model, a simulator of BSF-programs was designed and implemented in C++ with MPI-library. The source code of the simulator is freely available on Github, at https://github.com/nadezhda-ezhova/BSF-simulator. The simulator uses SPMD-model and includes both the master and slave sections. The master section is performed if `MPI_Comm_rank` returns 0. The slave section is performed if `MPI_Comm_rank returns` a value greater than 0. The order is an arbitrary string of specified length transferring from the master to slave by the `MPI_Isend` command. The slave reads the order by the `MPI_Irecv` command. The order processing is simulated by calling the *usleep*(t_v) function which causes the slave MPI-process to be suspended from execution until the specified number of real-time microseconds has elapsed. After processing, the master and all the slaves perform global barrier synchronization by using `MPI_Barrier` command. As a result, the slave sends to the master an arbitrary string of specified length using `MPI_Send` command. The master reads the results of all the slaves by `MPI_Recv` command. The synchronization is performed by `MPI_Waitall` command. Then, the master simulates evaluation

of the results by calling the $usleep(t_p)$ function which causes the master MPI-process to be suspended from execution until the specified number of real-time microseconds has elapsed.

To verify the equation (2) determining the speedup, the parameter $v = \lg(t_w/t_s)$ connecting the values of t_w and t_s was introduced. The following values of the parameter were studied: 4, 4.5 and 6. The speedup curves plotted using the equation (2), were compared with the curves obtained as a result of numerical simulations performed via the BSF-simulator. Parameters of the experiments are given in Table (1).

Table 1. BSF parameters

Parameter	Semantics	Value (seconds)
t_r	time that the master-node is engaged in sending one order to one slave-node excluding the latency	0.01
t_p	total time that the master-node is engaged in evaluating the results received from all the slave-nodes	4.99
t_w	total time that slave-nodes are engaged in execution of an orders	500
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	$2 \cdot 10^{-5}$

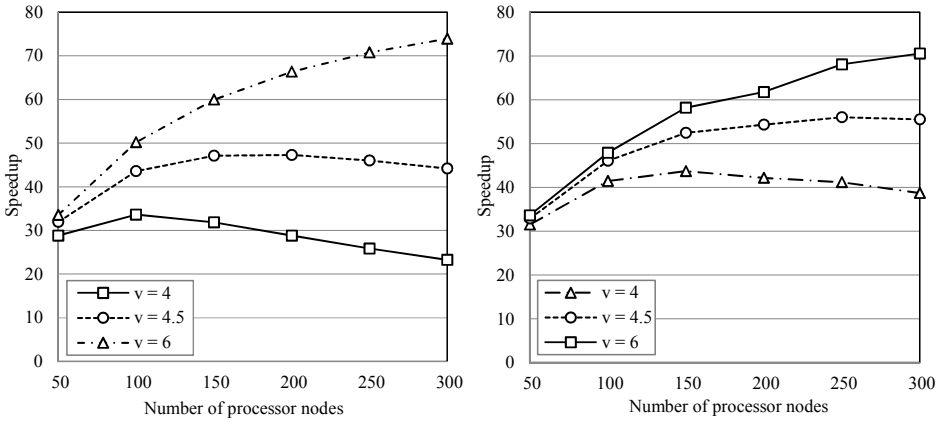
The latency value L was obtained experimentally as the transfer time of the message in 1 byte using the functions `MPI_Send` and `MPI_Recv`. The value of t_r corresponds to sending a message with a length of 14 MB. The values of the parameter t_s depends on v and t_w. This dependence is determined by the equation $t_s = 10^{-v} t_w$. The corresponding values of the parameter were obtained by varying the length of the order (see Table (2)).

Table 2. Adjustment of parameter v

v	t_s (seconds)	Order length
4	0.0206978	60 MB
4.5	0.0158160	6 MB
6	0.00048	200 KB

All the numerical experiments were conducted on the supercomputer "Tornado SUSU" [9]. The verification results of equation (2), shown in Fig. 5, show that the cost metrics of the BSF-model quite well predict the results produced by the BSF-simulator. In this case, the accuracy of the analytical estimates of the speedup increases with increasing of the parameter v value.

To verify the equation (3) determining the parallel efficiency, the parameter $q = t_p + t_r$ connecting the values of t_p and t_r was introduced. The following parameter q values were studied: 0.02, 2 and 20. The parameter t_r value was a constant equal to 0.01, and the parameter t_p took values 0.01, 1.99 and 19.99. The parameter t_s was also a constant equal to 0.005. This corresponds to a value of v equal to 5. The verification results of equation (3), shown in Fig. 6, show that the cost metrics of the BSF-model quite well predict the results produced by the BSF-simulator. At the same time, the accuracy of analytical estimates of parallel efficiency increases with increasing parameter q value.

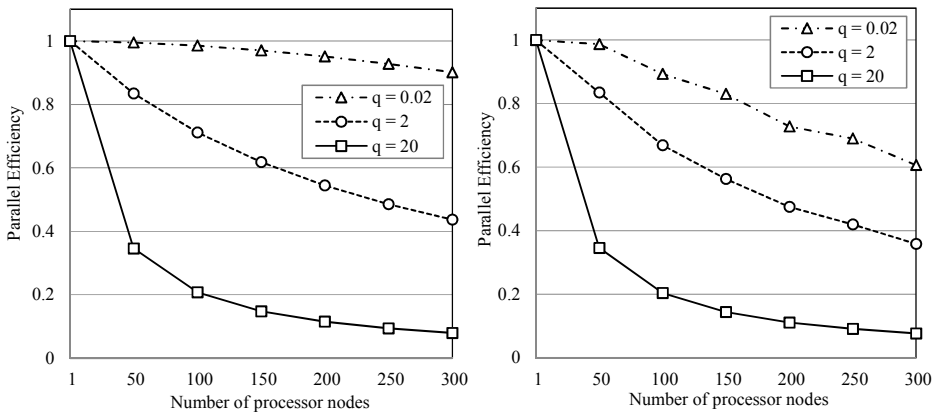


a) Analytical b) Experimental

Fig. 5. Dependence of the speedup on the number of processor nodes

5 Conclusion

In the paper, the issue of the adequacy of the BSF parallel computing model was studied. To verify the model, a simulator of BSF-programs in C++ language was developed using the MPI library. The emulator is implemented on the basis of the "master-slave" model. As orders, the master sends slaves messages of a certain length with arbitrary symbols. As a result, slaves are send to the master message of a certain length with arbitrary symbols. Calculations are simulated using the `usleep()` function. Via the simulator, computational experiments were carried out for various parameters of the BSF-model. The experimental and analytical



a) Analytical b) Experimental

Fig. 6. Dependence of the parallel efficiency on the number of processor nodes (v=5)

speedup curves are compared. The same comparison is made for curves of parallel efficiency. The obtained results confirm the adequacy of the BSF-model.

As future directions of research the following work is planned:

1. design a skeleton for the rapid creation of BSF-applications and implement this framework in C++ using the MPI library;
2. construct a (construct) dialog editor for fast creation of BSF-applications based on BSF-skeleton;
3. implement some numerical methods using BSF-skeleton.

References

1. TOP500 Supercomputer Sites. https://www.top500.org/
2. Sokolinsky L.B.: Analytical study of the "master-worker" framework scalability on multiprocessors with distributed memory. arXiv:1704.05816 [cs.DC]. 2017. P. 15. http://arxiv.org/abs/1704.05816
3. Sahni S.: Vairaktarakis G. The master-slave paradigm in parallel computer and industrial settings. J. Glob. Optim. 1996. Vol. 9, 3–4. P. 357–377.
4. Silva L.M., Buyya R.: Parallel programming models and paradigms. High Performance Cluster Computing: Architectures and Systems. Vol. 2. 1999. P. 4–27.
5. Leung J.Y.-T., Zhao H.: Scheduling problems in master-slave model. Ann. Oper. Res. 2008. Vol. 159, 1. P. 215–231.
6. Valiant L.G.: A bridging model for parallel computation. Commun. ACM. 1990. Vol. 33, 8. P. 103–111.
7. Darema F. et al.: A single-program-multiple-data computational model for EPEX/FORTRAN. Parallel Comput. 1988. Vol. 7, 1. P. 11–24.
8. Sokolinskaya I., Sokolinsky L.B.: On the Solution of Linear Programming Problems in the Age of Big Data. Parallel Computational Technologies - 11th International Conference, PCT 2017, Kazan, Russia, April 3-7, 2017, Proceedings (to

be published in Communications in Computer and Information Science, vol. 753). arXiv:1706.10030 [cs.DS]. 2017. P. 15.
9. Kostenetskiy P.S., Safonov A.Y.: SUSU Supercomputer Resources. Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies (PCT 2016). CEUR Workshop Proceedings. Vol. 1576. 2016. P. 561–573.