Using the bag-of-tasks model with centralized storage for distributed sorting of large data array

S V Vostokin[1], I V Bobyleva[1,2]

[1]Samara National Research University, Moskovskoe Shosse, 34A, Samara, Russia, 443086
[2]Joint Stock Company Space Rocket Center Progress, Zemetsa str., 18, Samara, Russia, 443009

e-mail: easts@mail.ru, ikazakova90@gmail.com

Abstract. The article discusses the application of the bag of tasks programming model for the problem of sorting a large data array. The choice is determined by the generality of its algorithmic structure with various problems from the field of data analysis including correlation analysis, frequency analysis, and data indexation. The sorting algorithm is a block-by-block sorting, followed by the pairwise merging of the blocks. At the end of the sorting, the data in the blocks form an ordered sequence. The order of sorting and merging tasks is set by a static directed acyclic graph. The sorting algorithm is implemented using MPI library in C ++ language with centralized storing of data blocks on the manager process. A feature of the implementation is the transfer of blocks between the master and the worker MPI processes for each task. Experimental study confirmed the hypothesis that the intensive data exchange resulting from the centralized nature of the bag of task model does not lead to a loss of performance. The data processing model makes it possible to weaken the technical requirements for the software and hardware.

1. Introduction

The article studies the possibility of using a model of distributed computing, called the bag of tasks [1], for solving problems of analysis and processing a large data array.

To solve the problems involving the processing and analysis of data, expensive specialized hardware and software are commonly used. However, in companies or institutions where such processing is carried out, there is already a large fleet of desktop computers, workstations, laptops of employees, and even cluster systems, whose resources are not fully used in normal operating conditions. Therefore, from the point of possible reduction in equipment costs, the use of the entire available hardware infrastructure not originally dedicated for data processing is promising in solving data intensive problems [2].

For making calculations on non-dedicated equipment, special programming models and software platforms supporting them are used [3]. One of these programming models is the bag of tasks model. This model has a star-like communication topology of processes. It consists of one master process and a group of worker processes. The worker processes execute tasks that come from the master process and when return the result back to the master process. Worker processes do not have a state. The state

of the calculations, the so-termed the bag of tasks, is monitored by the master process. This state can be updated after the current tasks in worker process is completed and returned to the master process.

The definition of parallel computations in the terms of the bag of tasks model has a number of advantages when distributed application is running on unreliable heterogeneous hardware and software environment.

First, the application has a single point of failure, namely the master process. Worker process failure is not critical.

Second, only the master process is required to accept connections, worker process can only support outgoing connections.

Third, to define a new algorithm in the bag of tasks model, it is sufficient to define 4 sequential procedures. These procedures are (1) checking whether the task to the worker process is available in the current state of the bag; (2) creation of a new task and the bag state update; (3) processing of task in the worker process; (4) receiving the task result and the bag state update. The other parts of the code are reused in every bag-of-tasks applications.

Fourth, the control scheme in the bag of tasks model in most cases does not require special methods of load balancing. This is convenient for programmers because the performance of computers on the network can vary significantly.

While having the above mentioned advantages, the bag of tasks model has one potential drawback. The master process may be a bottleneck in the case of intensive data exchange. Therefore, the applicability of the model in a particular case is determined by the joint characteristics of the problem and the equipment that is used to solve it. That is why a study is required before applying the bag of tasks model to data processing problems.

The goal of our research was to test experimentally the applicability of the bag of tasks model for making calculations on cluster systems without optimization of traffic between the master and the worker processes in solving the block sorting problem.

2. Related work

Our research relates to the field of algorithmic skeletons. This technique of building parallel algorithms allows us to separate the algorithmic (sequential) part of the code from the parallel part. Details of the algorithmic skeletons are described in the review [4].

Algorithmic skeletons are widely used in data processing. An example of such a skeleton is the MapReduce [5]. We focused on another well-known skeleton called Bag-of-tasks [1]. The reason is the “star” communication graph, which is typical both for the Bag-of-tasks skeleton and desktop grid computing in general [6]. The MapReduce skeleton does not apply to desktop grids, as it requires a network with a one-to-one communication graph.

We use a cluster system as a model of desktop grid systems. Computer clusters were successfully used as components of the desktop grids in many studies. A good example is parallel image processing, which is discussed in detail in the article [7]. The the technique of combining BOINC grid [8] and clusters with Torque and Slurm batch systems was shown in [9].

Parallel paired data processing is a frequently used algorithmic technique. It was automated in the algorithmic skeleton called AllPair [10]. Our implementation of pairwise processing is different in that we have limitations on parallel-running pairs and the order in which pairs are processed. The introduction of these restrictions allows us to obtain the sorting algorithm. The sorting algorithm is a block version of the bubble sort. The efficiency of the algorithm compared to the quick sorting we studied earlier in [11]. Block algorithms allow to optimize performance in different areas of computing then heterogeneous and/or distributed hardware is in use [12].

The applied objective of the study is to extend the set of algorithms for managing tasks in the Everest platform [3]. For the platform, the authors had previously developed a variant of the Map skeleton [13].

3. Method of the experimental study

In the study, we use the block sorting problem as a reference problem from the data processing domain. The choice of the block sorting problem is determined by the commonality of its algorithmic

structure with various problems of data analysis, including correlation analysis, frequency analysis, and data indexing.

The block sorting algorithm sorts each data block and then makes the pairwise merging of sorted blocks in the round-robin tournament manner. At the end of the block sorting algorithm, the data in all the blocks form an ordered sequence. Two types of tasks are placed in the bag of tasks: sorting of one block and merging of two blocks. Both types of tasks modify the content of blocks by rearranging elements, while the total number of blocks and their size does not change. The order of merging tasks is defined by the DAG (directed acyclic graph in Fig.1), which is interpreted by the master process in the course of calculations.

The peculiarity of the algorithm implementation is the transferring large blocks of data from the master process to the worker processes and back to the master process for each task. In the study, we deliberately did not use assistive technologies that optimize traffic, such as block caching on worker processes or direct data exchange between worker processes.

As a model of the computing equipment in the experiments we used "Sergey Korolev" cluster system of collective use that installed at Supercomputer center of Samara University (hpc.ssau.ru). The choice is due to the simplicity of programming of the distributed sorting in the bag of tasks model using MPI technology and C++ language. Also the deployment of sorting application components to compute nodes using the Torque batch system is simple too.

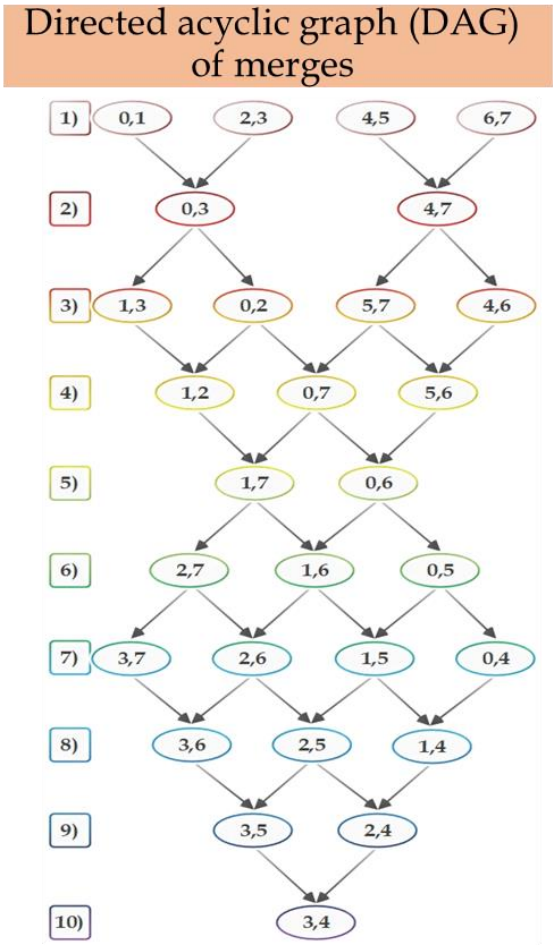

Figure 1. The DAG of merging tasks.

A cluster system has the greatest capacities in the computing infrastructure of any company or institution. A positive test result of block sorting using the bag of tasks model (the speedup of calculations) on the "Sergey Korolev" cluster opens up the possibility of further research on another type of non-dedicated computing equipment available on enterprise network.

4. Results and discussion of the block sorting experiment

The experimental study of the sorting algorithm confirmed the hypothesis that intensive data exchange in cluster system (which is a consequence of the implementation of block sorting algorithm according to the bag of tasks model) nether the less gives the possibility to speedup the calculations.

In the experiment, an array consisted of 8 blocks of 50 million 8-byte integers in each block. The 8 "sorting one block" tasks were performed with standard C ++ std::sort library algorithm and then 28 "merging two blocks" tasks were performed with standard C ++ std::merge algorithm.

The multilevel structure of the DAG had 10 tiers (Fig.1), a maximum of 4 parallel calls to std::merge in one tier. The programming and control of computational experiments on the cluster was performed on the TempletWeb system [14]. The results of experimental study are presented in Table 1 and Fig. 2.

Sequential sorting time of the entire array of 8 blocks on one cluster node (http://hpc.ssau.ru/node/6) using std::sort standard C++ library algorithm was about 110.8 seconds. The sequential sorting time on one cluster node for the block sorting program adapted to the bag of tasks model was about 118.9 seconds. In the parallel version of the program, when the master process and 8 worker processes were deployed on separate nodes of the cluster, the time of execution became 23.6412 seconds (the best result). This result corresponds to nearly five times speedup.

Table 1. Times in seconds of sequential and parallel execution of the block sorting test depending on the number of nodes.

№	Nodes	SEQ	PAR
1	2	118,986	126,947
2	3	118,988	64,4385
3	4	118,996	48,493
4	5	118,99	35,7863
5	6	125,063	39,0551
6	7	124,8	38,1016
7	8	128,148	37,1677
8	9	118,987	23,6412

A series of experiments was carried out in which different variants of the program deployment on the nodes of the cluster system were investigated. It was discovered that despite the fact that the nodes of the cluster support minimum 8 hardware threads, the deployment of all processes on a single node gives the worst result (~35,6 seconds) compared with the deployment of processes under the scheme 1 process on 1 node. The overhead of competing between processes for a shared data bus on a node was even greater than the overhead of transferring data between nodes.

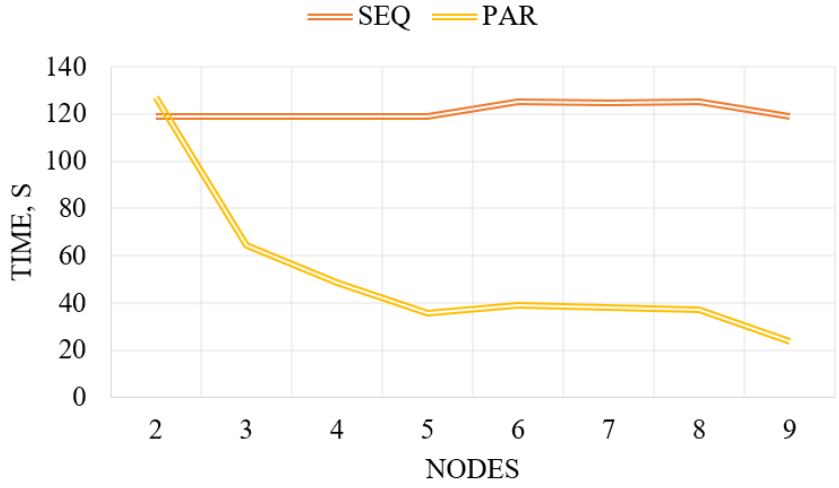

Figure 2. Comparison of sequential and parallel execution depending on the number of nodes.

5. Conclusion

The experiments confirmed the suitability of the bag of tasks model for computing on cluster systems even without using algorithmic techniques to optimize the traffic between the master process and the worker processes in solving the block sorting problem of large data array.

Thus, the bag of tasks model allows to reduce technical requirements for hardware and can be used to solve applied problems of analysis and processing of large information arrays on existing computing networks of companies or institutions.

6. References

[1] Senger H and da Silva F 2012 Bounds on the Scalability of Bag-of-Tasks Applications Running on Master-Slave Platforms *Parallel Processing Letters* **22**
[2] Ivashko E E 2015 Enterprise Desktop Grids *CEUR Workshop Proceedings* **1502** 16-21
[3] Sukhoroslov O and Volkov S 2015 Web-Based Platform for Publication and Distributed Execution of Computing Applications *14th International Symposium on Parallel and Distributed Computing (ISPDC)* 175-184
[4] González-Vélez H and Leyton M 2010 A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers *Software: Practice and Experience* **40(12)** 1135-1160
[5] Dean J and Ghemawat S 2008 MapReduce: simplified data processing on large clusters *Communications of the ACM* **51(1)** 107-113
[6] Cérin C and Fedak G 2012 *Desktop grid computing* (Paris: CRC Press)
[7] Volotovsky S G, Kazanskiy N L, Popov S B and Serafimovich P G 2010 Performance analysis of image parallel processing applications *Computer Optics* **34(4)** 567-572
[8] Anderson D P 2004 BOINC: A system for public-resource computing and storage *Proceedings - IEEE/ACM International Workshop on Grid Computing* 4-10
[9] Afanasyev A P and Lovas R 2011 Increasing the computing power of distributed systems with the help of grid systems from personal computers *Proceedings of the conference "Parallel Computational Technologies* 6-14
[10] Moretti C and Bulosan J 2008 All-pairs: An abstraction for data-intensive cloud computing *IEEE International Symposium on Parallel and Distributed Processing* 1-11
[11] Vostokin S V and Kazakova I V 2018 Implementation of stream processing using the actor formalism for simulation of distributed insertion sort *Journal of Physics: Conference Series* **1096(1)**
[12] Yablokova L V and Golovashkin D L 2018 Block algorithms of a simultaneous difference solution of d'Alembert's and Maxwell's equations *Computer Optics* **42(2)** 320-327 DOI: 10.18287/2412-6179-2018-42-2-320-327
[13] Volkov S and Sukhoroslov O 2015 Running Parameter Sweep Applications on Everest Cloud Plat-form *Computer Research and Modeling* **7(3)** 601-606
[14] Vostokin S V and Artamonov Y S 2018 Templet Web: the use of volunteer computing approach in PaaS-style cloud *Open Engineering* **8** 50-56

Acknowledgments

This work is partially supported by the Ministry of Education and Science of the Russian Federation in the framework of the State Assignments program (#9.1616.2017/4.6).