Scheduling based on a dynamic resource connection

A E Nagiyev[1], I A Botygin[1], A I Shersntneva[1] and P A Konyaev[2]

[1] Tomsk Polytechnic University, 30, Lenina Ave., Tomsk, 634050, Russia
[2] V E Zuev Institute of Atmospheric Optics Siberian Branch of Russian Academy of Sciences, 1, Academician Zuev Sq., Tomsk, 634055, Russia

E-mail: andrew_nagiev09@mail.ru

Abstract. The practical using of distributed computing systems associated with many problems, including troubles with the organization of an effective interaction between the agents located at the nodes of the system, with the specific configuration of each node of the system to perform a certain task, with the effective distribution of the available information and computational resources of the system, with the control of multithreading which implements the logic of solving research problems and so on. The article describes the method of computing load balancing in distributed automatic systems, focused on the multi-agency and multi-threaded data processing. The scheme of the control of processing requests from the terminal devices, providing the effective dynamic scaling of computing power under peak load is offered. The results of the model experiments research of the developed load scheduling algorithm are set out. These results show the effectiveness of the algorithm even with a significant expansion in the number of connected nodes and zoom in the architecture distributed computing system.

1. Introduction

At present, distributed computing systems (DCS) are increasingly intensely [1]. Parallel processing using a large number of computing nodes is a distinctive feature of the distributed computing systems. The advantage of this approach is the possibility of a rapid increase in productivity by the horizontal scaling of computing nodes and resources.

At the moment, there are three types of distributed systems [2]:

- A cluster —a certain number of computing nodes, combined with the LAN. Resources are used by only one working group [3]. In the DCS, clustering is performed only at the software level.
- An enterprise-class computing system is a computer system. The resources of this system are used to fulfil the tasks of several working groups.
- A Grid System — a system that combines a large number of geographically dispersed computing nodes. This system is intended for simultaneous processing of a large number of working groups.

2. DCS and Load Balancing

2.1. Problems of the DCS

There are some problems, which are associated with the development of the DCS:

- The inability to define the global time for the entire distributed computing system. Each node of

the system has its own astronomical time. Sometimes it creates difficulties connected with synchronization of the tasks on all of the nodes.

- Communication between nodes is carried out not instantaneously and with significant delays. This requires additional steps for the development of the software and complicates the development of distributed computing systems.
- Communications are insecure that can lead to a loss of data during transmission between the nodes of the system.
- The complexity of the software configuration to complete the tasks on the system due to the heterogeneity of platforms on which the nodes of the distributed computing system are deployed.
- The complexity of ensuring an effective load distribution on the nodes of the DCS, etc.

Thus, the system development is hampered by the fact that computing nodes can have different productivity; communication lines have different bandwidths; tasks may require different processing power and others.

2.2. Description of the Load Balancing

According to the authors, the main attention should be paid to the distribution of load between the nodes (load balancing) in the development of distributed computing systems. Load Balancing (LB) is used to optimize the fulfilment of the distributed computing. Load Balancing allows one to distribute the computational load on the existing nodes in the system effectively. Software, using a predetermined algorithm of the LB, fulfils decomposition of the task, splitting it into modules after transferring the tasks to fulfil in the DCS. These modules have transmitted to the nodes in the system for further processing.

Load balancing is divided into static, dynamic and semi-dynamic [4-7]. The static balancing is fulfilled before the beginning of a task. Semi-dynamic LB assumes that the distribution of tasks is carried out at the stage of initialization. The dynamic load balancing task allocation takes place during the computation. If we imply the way LB interacts in a computer system, the balancing is divided into centralized and distributed. In the centralized LB, a special computing node (the so-called controller) distributes modules tasks between calculators. In the distributed LB using a special algorithm, the data of the status are exchanged between all nodes in the system.

Also, balancing is divided into universal and specialized (designed to fit the specified algorithm in a computing task). [8] At the moment, there are many techniques and load balancing algorithms. Among them are: Round Robin, Weighted Round Robin, Least connections, Weighted least connections [9-11].

3. The Method of a Dynamic Connection of Resources

In this paper, another method of load balancing in distributed computing systems was developed - the method of a dynamic connection of resources (DCR) as a result of the program experiments. Special program experiments were conducted to assess the degree of efficiency of the developed method for horizontal scaling calculators and increasing load.

During the program experiment, at first, the effectiveness of the classic server solution without load balancing was evaluated. Then carrying capacity of this system was compared with the carrying capacity of another system, which interaction of the elements was performed by a method of a dynamic connection of resources.

As it was expected during the program experiment, the classical server solution, which consists of a control server and terminals requests, proved to be ineffective at high loads. Namely, the program experiment conducted by the computer-based system, which consists of a management server and dynamically connected terminals requests from simulators, showed that increasing the number of incoming requests for processing it was failure of the entire system.

In the experiment, on the basis of the computer system using a quad-core processor and simulation of requests in separate threads by the law of uniform distribution at an interval of 10 ms, failure

occurred in the processing of 850 requests. From this it can be concluded that with the increase in frequency, the request generation system productivity decreases. The server does not have time to accept, process and send back the results. Therefore, to increase the effectiveness of tasks (performance, etc.), the method of dynamic connection of resources is used. This approach allows one to dynamically connect additional compute nodes, through which parallel computation of various tasks of the management server is provided. This provides server unloading.

The program experiment was conducted for a practical demonstration of the proposed method. This experiment is described below. The functional structure of a distributed system which is shown in figure 1 is used for the experiment. As can be seen, the main changes have happened to the management server, which was replaced by the main control center (MCC). The system infrastructure has changed. The main control center and the dynamic link load controllers have been added; they will be connected to the additional control centers (ACC). The generator of requests remained unchanged.

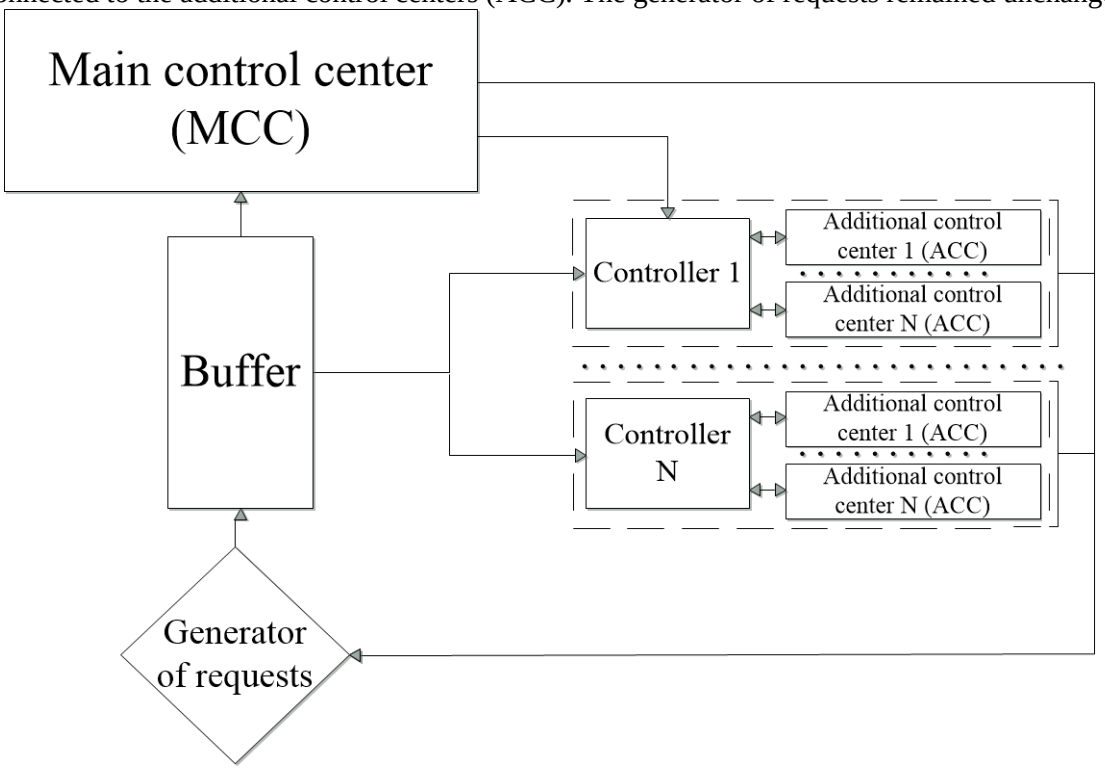

Figure 1. The block diagram of the distributed computing system in the program experiment.

Unlike the previous program experiment, the connection address for each request terminal is recorded in a special buffer of the MCC, which removes this connection after the request is processed and the task which has been set before is solved (Figure 2).

With the increase in the number of incoming requests, MCC does not have time to promptly process all incoming requests. It becomes a ‘bottleneck’ of the system, which causes the accumulation requests in the buffer. The controller of load balancing starts operating when a certain number of requests which are waiting to be processed in the MCC are reached (Figure 3). The controller starts to run additional control centers. Their number is limited by the rate of incoming requests and the presence of additional computing powers in the system.

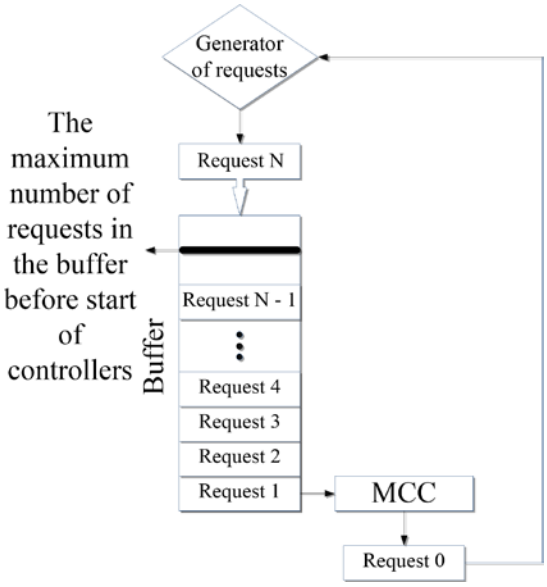

Figure 2. Processing of incoming requests by MCC without additional control centers.

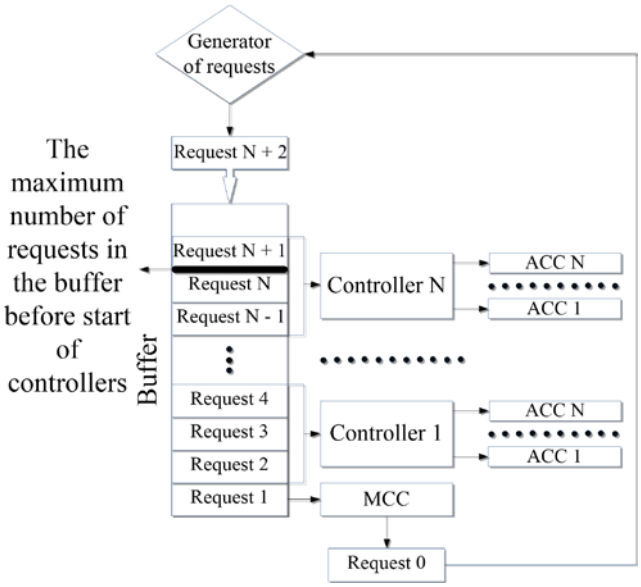

Figure 3. The block diagram of the sample requests which controllers process from the buffer and send them to ACC.

Each of the controllers runs in its own separate thread. The number of controllers is limited only by processing power of the distributed system infrastructure. The controller selects a certain number of requests from the buffer and sends them to the ACC, which it launched in separate threads. ACC is switched off only after processing requests, which were received from the controller.

4. Results of Experiment

As a result of experiments, we can say that the time of terminals' connection and the processing time of requests to MCC is reduced by 57% as compared to the classic version. During the same time, 58.5% more requests from the terminals connect to the MCC than those during working of the main control center without ACC.

During the connection of the second ACC, processing speed has increased by 13% relative to a system with single ACC. During the connection of the third ACC, processing speed has also increased by 24% relative to the system, which employs two additional control centers. The experimental results are shown in graph figure 4.

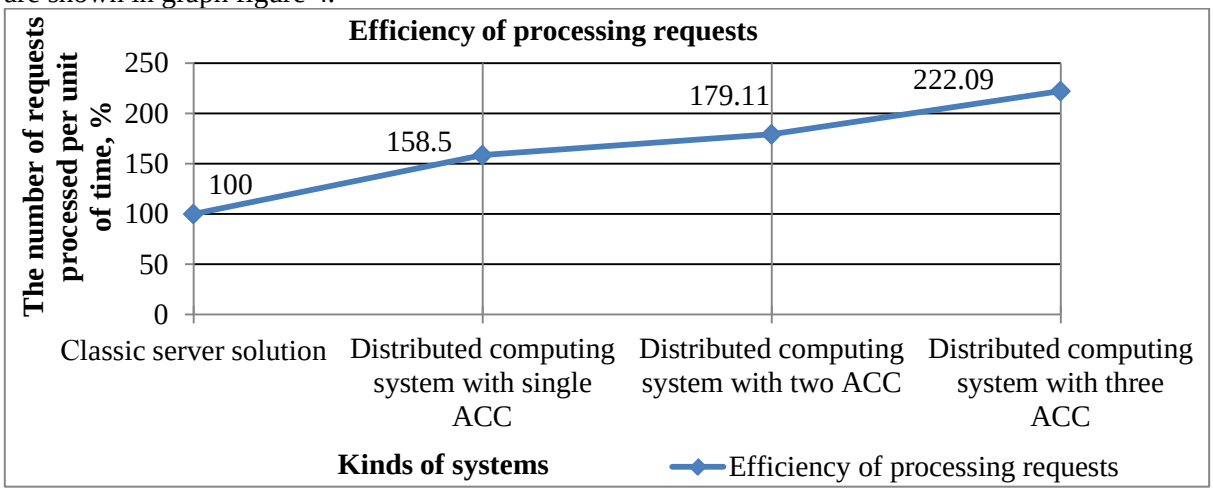

Figure 4. The block diagram of the sample requests controllers processing from the buffer and sending them to ACC.

Experiments were carried out by connecting different numbers of requests terminals (from 10 to 10 000) with uniform distribution in the interval from 0 to 100 arbitrary time units.

If considering CPU usages used in the program experiment qualitatively, the connection of even 250 terminals in substantially separate threads charge CPU. During the connection of 10000 applications for processing, we can clearly see that each of the 4-CPU cores is loaded evenly. In figures 5 and 6, we can see the difference between the load of CPU cores without processing requests from the simulator and with processing the large number of requests according to the method, which was described above.

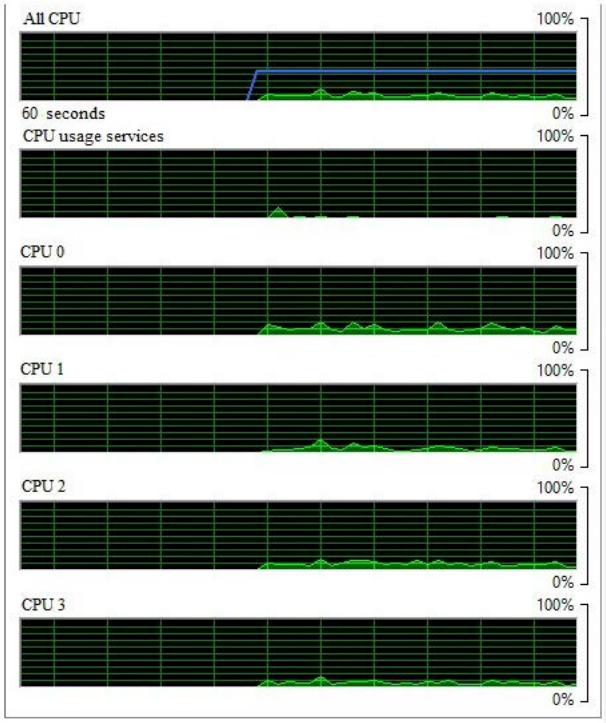

Figure 5. The block diagram of the sample requests controllers processing from the buffer and sending them to ACC.

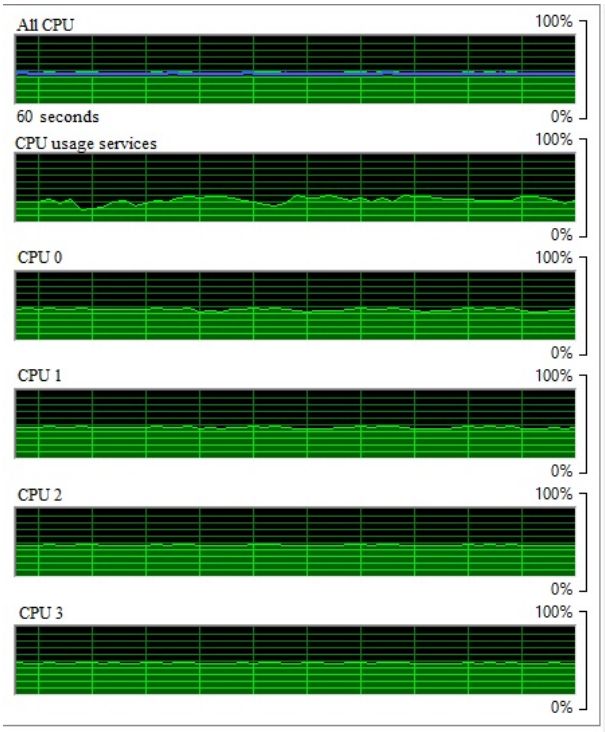

Figure 6. The block diagram of the sample requests controllers processing from the buffer and sending them to ACC.

As is seen from figures 5 and 6, there is a uniform distribution on the computation of the CPU cores based on pseudorandom numbers.

5. Conclusion

The generator of requests, which operates according to the law of uniform distribution and which plays a role of a terminal simulator, experimentally demonstrated performance, efficiency and flexibility of the proposed system, despite the fact that the experiment took place only on one computer with the quad-core processor. Due to this, the ensuring scale of the computing power occurs, which is limited only by the physical abilities of connected additional nodes.

The experiment, which is described above, has demonstrated the effectiveness of the method of dynamic connection of resources. Through modeling of this system, we can state that the considered method for dynamic connection of resources remains effective with a significant increase of the connection of computational nodes.

References

[1] Kshemkalyani A D and Singhal M 2008 *Distributed Computing: Principles, Algorithms, and Systems* (Cambridge, UK: Cambridge University Press)

[2] Tanenbaum A S and Van Steen 2007 *Distributed Systems: Principles and Paradigms* (*2nd*) (Upper Saddle River, NJ, USA: Pearson Education)
[3] Carlos A Varela and Gul Agha 2013 *Programming Distributed Computing Systems: A Foundational Approach Hardcover* (Cambridge, USA: MIT Press)
[4] Load balancing in distributed systems *Intuit* URL: http://www.intuit.ru/studies/courses/1146/238/lecture/6153
[5] Bershadskiy A M, Kurilov L S, Finogeev A G 2009 *Proc. of the Univ. Volga Region* vol 4 (Penza, Russia: PSU Publishing house) pp 38–48
[6] Qiu Y W and Hwang J I G 2016 *IEEE 14th Intl Conf. on Dependable, Autonomic and Secure Computing* (Piscataway, NJ, USA: IEEE Press) pp 565-571
[7] Sommer M, Klink M, Tomforde S and Hähner J 2016 *IEEE Int. Conf. on Autonomic Computing* (Piscataway, NJ, USA: IEEE Press) pp 300-307
[8] Sherstnyov V S, Sherstnyova A I, Botygin I A, Kustov D A 2016 *Key Eng. Mat.* **685** 867-871
[9] Sherstnev V S 2005 *The 9th Russian-Korean Int. Symp. on Science and Technology* vol 1 (Piscataway, NJ, USA: IEEE Press) pp 696-700
[10] Botygin I A, Nagiyev A E 2015 *Int. Conf. on Mechanical Engineering, Automation and Control Systems* (Piscataway, NJ, USA: IEEE Press) pp 1-4
[11] Bal H E, Kaashoek M F and Tanenbaum A S 1992 *IEEE Transactions on Software Engineering* vol 18 (Piscataway, NJ, USA: IEEE Press) pp 190-205