Description and development of the means of a model experiment for load balancing in distributed computing systems

A E Nagiyev[1], A I Sherstnyova[1], I A Botygin[1], N Y Galanova[2]

[1] Tomsk Polytechnic University, Tomsk, Russia
[2] Tomsk State University, Tomsk, Russia

E-mail: andrew_nagiev09@mail.ru, sherstneva@tpu.ru ,bia@tpu.ru

Abstract. The results of the statistical model experiments research of various load balancing algorithms in distributed computing systems are presented. Software tools were developed. These tools, which allow to create a virtual infrastructure of distributed computing system in accordance with the intended objective of the research focused on multi-agent and multi-threaded data processing were developed. A diagram of the control processing of requests from the terminal devices, providing an effective dynamic horizontal scaling of computing power at peak loads, is proposed.

1. Introduction

Modern trends of technology tendencies in the IT industry (the concept of cloud computing, big data concept, the concept of Internet-things, the concept of fog computing, the concept of mobilization of calculations, etc.) not only exponentially increase the volume of transmitted data, but also uniquely determine the need to use in computing systems distributed multiprocessor and multicomputer systems (MCS). As it is noted by many experts, if the physical infrastructure of MCS complexity is reduced (an era of universal MCS architectures begins), the logical complexity of such software systems only increases [1-3].

Note, that even the necessity of increasing of efficient use of existing computing power and communication resources leads to system solutions for load balancing, optimization of network traffic. This requires the development of special software, capable quickly and flexibly to optimize the management of the information network and computing infrastructure. Also it is necessary to take into account that theoretically possible execution parallelism is not always obvious for many tasks. Because of this imbalance (communication imbalance, platform imbalance, etc.) of load computing nodes of distributed MCS is inevitable.

Thus, one approach to ensure the required efficiency of functioning of distributed IDC is clearly a load balancing components [4-6]. That load balancing (adjustment) optimizes the calculations in a distributed architecture, information and computing resources of complex system and increases their failover. The problem of distribution perform of dynamic requests (load balancing problem) between the components of MCS, which dynamically change topology, relates to the field of basic research in the field of computer technology and belongs to the section of Artificial Intelligence. Therefore, universal solution for load balancing problem does not exist. It should be taken into account in each case of development: character of input data, intensity of their revenue streams, characteristics of processing algorithms, and much more but from the field of human intellectual activity. It can not only eliminate the load imbalance, but also can formulate new goals and objectives of really intelligent distributed MCS.

2. Program experiments

In the present work program experiments have been carried out to test and refine the technological scheme of such a method of load balancing, as a method of dynamic connection of resources (DCR).

During the program experiment first assessment of effectiveness of the classic server solution was carried out, when load is determined randomly by formed applications Poisson flow (flow homogeneous and independent from each other's events). Processing applications obeyed to an exponential law (the absolute continuity of the distribution). After that the effectiveness of the distributed computing system (DCS) was evaluated. The interaction of the components of DCS was performed by the method of the DCR. The obtained results were compared with the results of the classic server solutions.

During the experiment classic software server solution consisting of the management server and dynamically connecting to it terminals of requests was created.

Software experiment showed the ineffectiveness of the server solution. Increasing of the number of requests, which received from the generator, leads to the failure of the entire system. In particular, the simulation server by the law of uniform distribution with the interval 10 ms failure occurred at 850 processing requests. Based on this fact, conclusion can be made, that an increase in the frequency of generating query system will run much slower, because the server needs to receive the request, process and send it back to the terminal. An increasing of the number of received requests in different streams did not increased the server data processing speed. Flows began to compete with each other that led to failure of the system.

According to the authors, a method of dynamic connection of resources was used to improve system performance, which leads to acceleration of processing requests. In the proposed functional structure multicomputer complex (figure 1) the main changes have affected the processing server. Its structure has been improved and represents the main control center (MCC). Also special load controllers have been added, with its help there was a dynamic connection of additional control centers (ACC) and buffer storage of incoming requests. Generator of requests is unchanged.

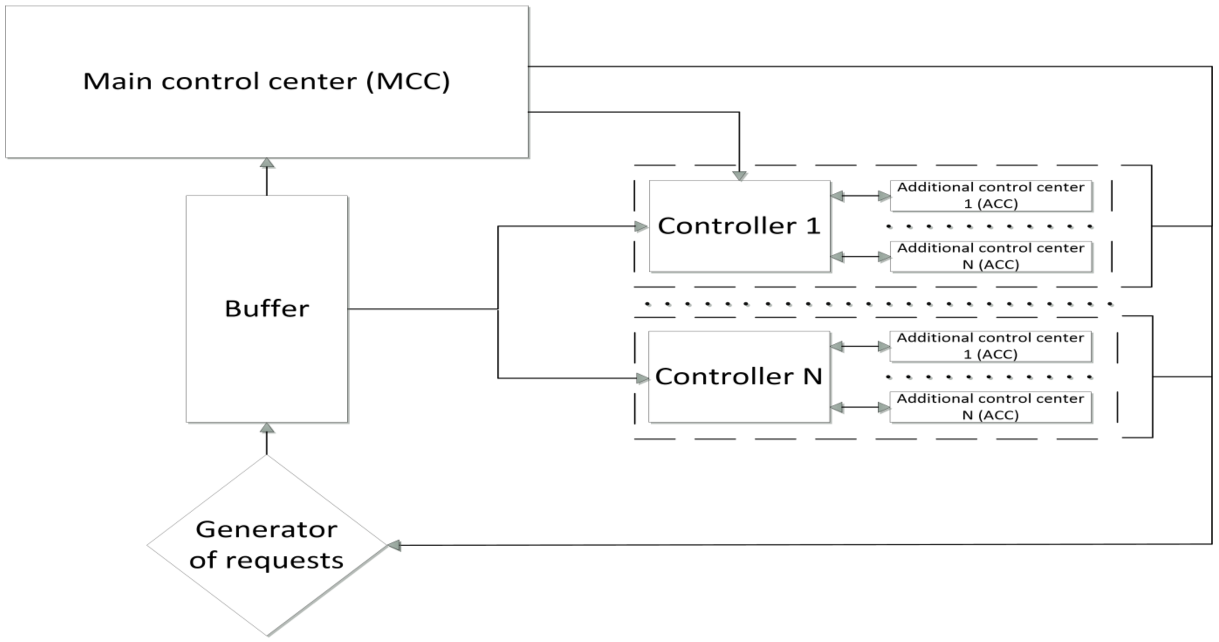

Main control center (MCC)
Buffer
Controller 1
Additional control center 1 (ACC)
Additional control center N (ACC)
Controller N
Additional control center 1 (ACC)
Additional control center N (ACC)
Generator of requests

Figure 1. The functional structure of the simulated MCS

Buffer of requests of MCC intends for storage of incoming requests to the processing of the generator of requests (figure 2). After processing the request for calculation, removal from the buffer occurred.

With the increasing the amount of requests incoming from the generator, the main control center, like a server in a classic server solution, does not have time to process all the requests. However, unlike the classical server solution where there was failure of the system, in MCS using DCR, incoming requests accumulated in the buffer. Load controllers were connected when a certain number of requests for processing in a buffer was reached by the software (figure 3). Controllers in separate streams started to run the ACC. The number of connected ACC is limited only by speed of queries and receipt of the physical features of the system (presence of additional computing power).

Each controller starts running with the buffer in a separate thread. Also, each additional control center runs in a separate thread. ACC closed after processing all the requests submitted via the load controller

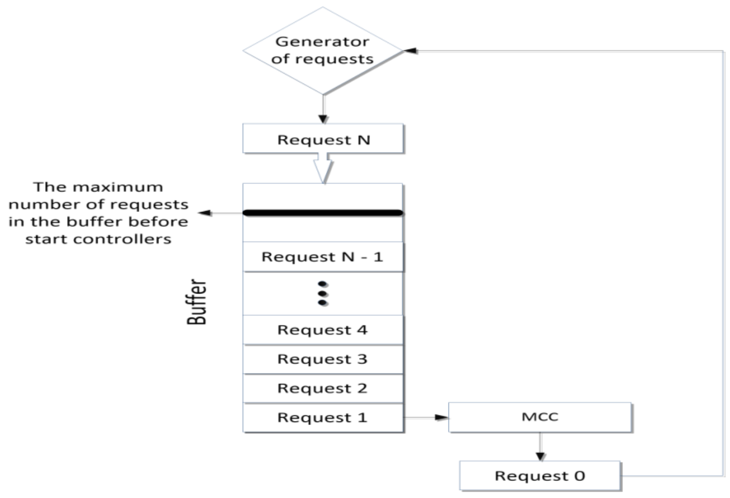


Figure 2. The scheme of processing of incoming requests in MCC

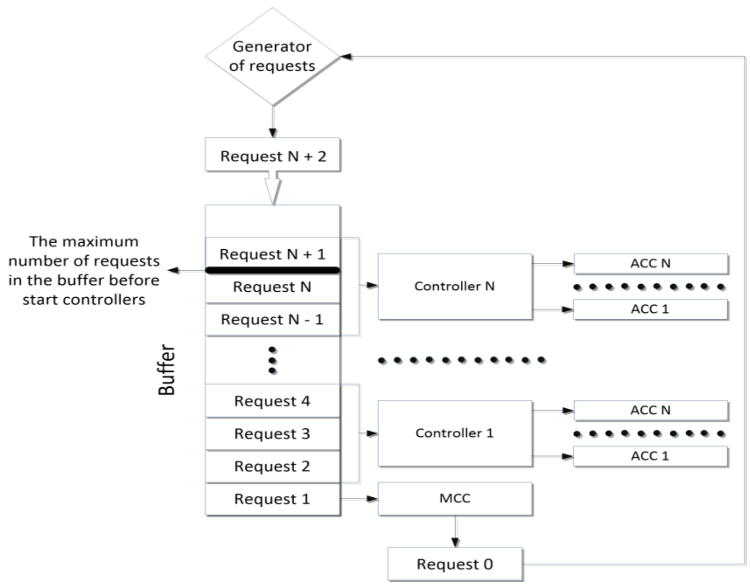
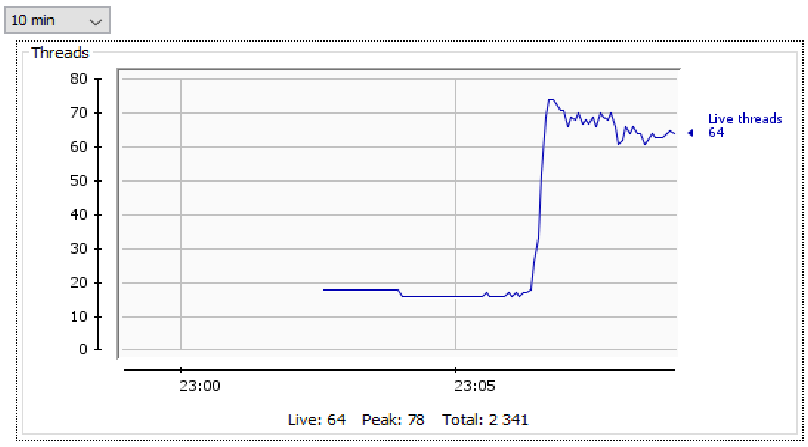


Figure 3. The block diagram of sampling requests of handles controllers

3. Software modelling

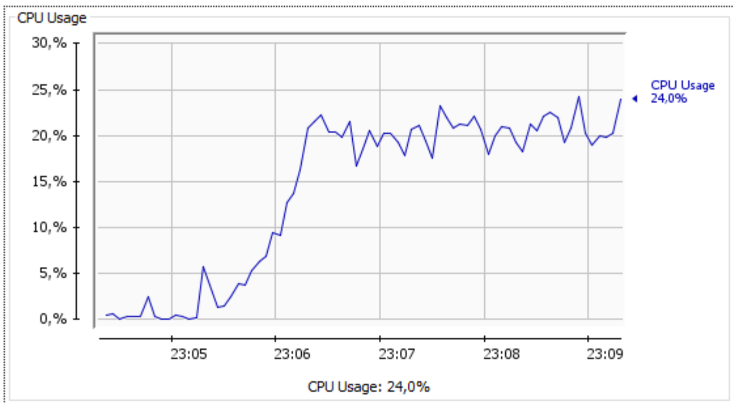
Software modeling of the experiment illustrated in figures 4, 5. In figure 4 there is a curve showing the number of threads running currently in the generator of requests. The main control center loads central processing unit (CPU) (figure 5). As you can see, ACC has time to receive, to process and to send out different threads back into the generator of requests at a certain low frequency of incoming requests which are set by software. However, the number of processing threads is increased with speed of incoming requests. This is due to the fact that the MCC had not got a time to process all the incoming requests, and they accumulated in the buffer of requests. At that moment the classic server solution refused because of lack of buffer, so threads began to compete for server resources. In the proposed structure of MCC load controller connected when a certain number of threads specified by the software. This controller connected the ACC and it increased the computing power system. This led to the fact that the increase of the number of requests in the buffer was stopped. There was a slow decrease of the number of requests in view of their distribution between the additional control centers.



Live: 64 Peak: 78 Total: 2 341

Figure 4. The graph of number of active flows in a distributed computer system.

It can be seen in figure 5 the difference between the CPU load before sending the requests from the generator and during processing of requests. The increase of frequency of receiving requests increases CPU usage by the main control center, which should be faster to handle incoming requests. Then, load growth is ensured by connecting the ACC. This was accompanied by a uniform distribution computing between the CPU cores based on pseudorandom numbers.



CPU Usage: 24,0%

Figure 5. The graph of increasing the CPU load.

There is a processing of the remaining requests in a buffer and switching off the dispatchers and ACC after sending requests from the generator. At the same time a decrease in CPU usage to its original condition, as shown in figure 6, 7.

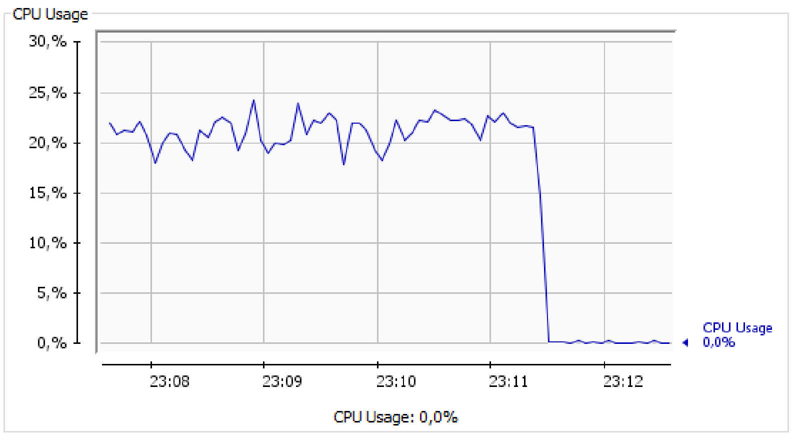


Figure 6. The graph of decreasing the CPU load.

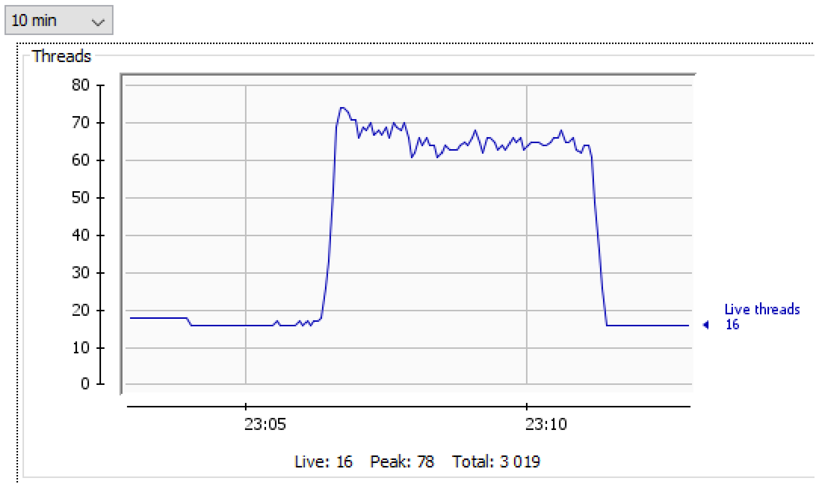


Figure 7. The graph of decreasing threads of the CPU.

4. Conclusion

As a result of experiments, it can be said that the time of connection of terminals and processing requests to MCC was reduced by 57%, compared to the classical version. And in the same time to the main control center is connected and handled on 58.5% more terminals than at the MCC without connecting of ACC. When the second ACC is connected, processing speed increased by 13% compared to the system with a single ACC. When the third ACC is connected, processing speed is also increased by 24% compared to the system with two ACC.

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

Despite the fact that the experiment took place only on a single computer with a quad-core processor, a simple generator of requests with a uniform distribution was enough to experimentally show the effectiveness, flexibility and speed of the proposed method dynamic connection of resources. This provides horizontal scalability of processing power, limited only by the physical ability to connect additional nodes.

On the basis of a model experiment can be said that the considered method of dynamically connect resources will be also effective in a significant increase in the number of connected nodes and zoom in architecture.

References

[1] I. A. Botygin, V. N. Popov, V. A. Tartakovsky, V. S. Sherstnev Architecture of scalability file system for meteorological observation data storing // Proc. of SPIE, 21st International Symposium Atmospheric and Ocean Optics: Atmospheric Physics. – 2015. – vol. 9680. – pp. 96800J-1– 96800J-4. – doi: 10.1117/12.2205749.

[2] Sherstnyov V. S., Sherstnyova A. I., Botygin I. A., Kustov D. A. Distributed Information System for Processing and Storage of Meteorological Data // High Technology: Research and Applications 2015. Key Engineering Materials. – Trans Tech Publications, Switzerland, 2016. – Vol. 685. – pp. 867-871.

[3] Popov V.N., Botygin I.A., Koshelev N.V. Uniform Model of Representation Heterogeneous Data Hydrometeorological Observations // High Technology: Research and Applications 2015. Key Engineering Materials. – Trans Tech Publications, Switzerland, 2016. – Vol. 685. – pp. 925-929.

[4] A.V. Boukhanovsky, S.V. Ivanov, Features of optimization of distribution computational load in tasks of parallel processing of information and simulation modelling, IMMOD-2003, Saint-Petersburg., 2003, pp. 69-75.

[5] Intuit. Threads and multithreading. URL: http://www.intuit.ru/studies/courses/641/497/lecture/11284?page=2 (the date of circulation 06. 10. 2015).

[6] Load balansing: basic algorithms and methods // Habrahabr. URL: http://habrahabr.ru/company/selectel/blog/250201/ (дата обращения 10. 11. 2015)