УДК 004.9

USAGE THE OPERATING SYSTEMS MULTITASKING FOR ORGANIZATION MULTI-ACCESS TO COMPLEX SYSTEMS MODELING COMPUTING SYSTEM

Viacheslav Zosimov

Mykolayiv V.O. Sukhomlynsky National University, 54030,
Ukraine, Mykolayiv, Nikolska str., 24

zosimovvv@bk.ru

Описана організація мультидоступу до програмного комплексу моделювання складних систем. Використання багатозадачних операційних систем дозволяє запускати на багатопроцесорних (багатоядерних) комп'ютерах паралельне обчислення декількох задач, шляхом виділення для обчислення кожної задачі окремого ядра процесора..
Ключові слова: Індуктивне моделювання, МГУА, мульти-доступ, багатозадачність, моделювання складних систем

The paper describes multi-access method organization to computing system of complex systems modeling. Operating system multitasking allows you to run on multiprocessor (multi-core) computer multiple tasks in parallel by assignment a separate processor core for each task
Keywords: Inductive modeling, GMDH, multi-access, multitasking, complex systems modeling.

Описана организация мультидоступа к программному комплексу моделирования сложных систем. Использование многозадачных операционных систем позволяет запускать на многопроцессорных (многоядерных) компьютерах параллельное вычисление нескольких задач, путем выделения для вычисления каждой задачи отдельного ядра процессора.
Ключевые слова: Индуктивное моделирование, МГУА, мульти-доступ, многозадачность, моделирования сложных систем.

Introduction

A growing amount of research are carried out in the area of distributed large-scale (global) calculations. Are developed middleware, libraries, and tools that allows to share geographically distributed, but combined resources as a single powerful platform for parallel and distributed applications. Such an approach to computing was formerly known under several names, such as metacomputing, scalable and global computations.

Talking about parallel computing in the article, there are two main options for parallelization:

1) by tasks - flows performs different tasks;

2) by data - flows accomplish the same task, but each with its own part of shared data;

This article describes the first option - parallelization by tasks.

1. Parallelization by tasks

Consideration of this issue is easier to start with the basic set of tasks to be performed by the program [1, 2]:

1) Operations with the interface (rendering widgets, to change size and window transfer responses, buttons pressing responses, changing tabs, opening and choice in the drop-down lists, etc.).

2) Operations with the repository (create, delete, modify, retrieve a list of the various project files, reading and decoding information from the calculations and results files);

3) schedule construction;

4) direct calculations.

It is easy to note that these tasks are quite different by the execution time. Calculations - the most time-consuming task and can be performed by several orders longer than the other actions.

However, during a sequential execution the "fast" tasks will have to wait for the end of the "slow". As a result, with calculating start user will lose the opportunity to see other results, compare their schedules, and generally interface will hang. The problem is compounded in multi-user environment, if the waiting time due to its actions still possible to bear, the constant fading due the actions of other people is not acceptable. To solve this problem it is necessary to provide the ability to perform all actions in different threads.

The most convenient in this case is the usage of client-server model.

2. Client-server model

In this model (Fig. 1), the interface is engaged in client side, which sends requests to the server for receiving data from it. The server receives the request and every request starts a separate thread that stops the existence after receiving the results and response client

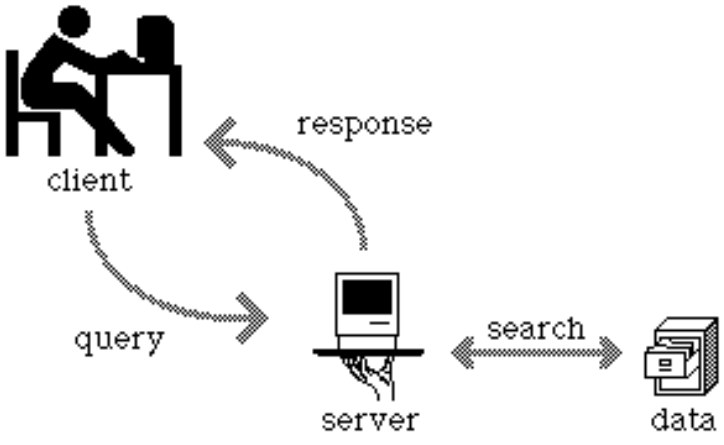

Fig. 1**.** Client-server model [3].

There are many options for implementing client-server model. For a software implementation, was chosen the most universal and popular option - a web-based application. Main advantages of this option are:

1) on the client side there is no need to install any software, all you need - a web browser;

2) the almost total independence from the operating system and customer equipment and a weak dependence on the server side;

3) on the server side there is no need for self-scheduling requests and generation flow is dealt with by a web server;

4) natural multi-user environment;

5) the scaling and balancing is done by means of administration without changes in the source code.

Let us stop in detail on the last point. In conditions of low hardware resources the server and client can be run on the same computer, including the case of only one processor core. The operating system will switch tasks on its own and, though it is not received advantage by the time of execution, but from the user interface point of view it will be multi-tasking. During extending hardware base server can run on a separate computer, and if it has multiple cores there will be obtained acceleration, as several calculations can be performed simultaneously, each on its core.

At this stage is necessary to consider the number of cores and to maximize the efficiency not to run at the same time more calculations than the cores in the system, and even better to leave one core to perform short-term demands and operating system tasks. If there is the ability to dedicate for the server, several computers, this is making under the scheme backend-frontend. (Fig. 2)

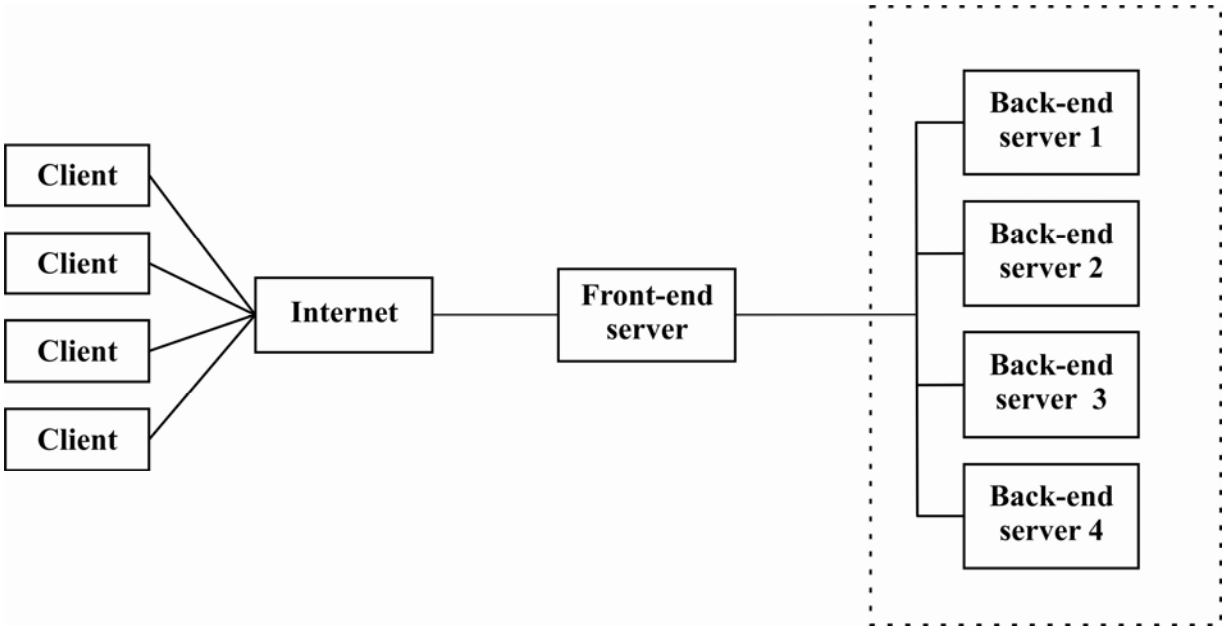

Fig. 2. Backend-frontend scheme [4].

Most of the computers is backend and does most of the work on each of them is running a standard web server and the program. A smaller part of the computers are frontend. It runs the web server in the mode of reverse-proxy. Frontend accepts user requests, distributes them to the production web server, receives and caches the response from them, sends results to users.

In order to ensure transparency at the file level is using a network storage that is connected to all backend servers. This scheme allows almost linearly increase the computing power of the cluster, as well provides continuous work in the case of failure of some backend servers. Importantly, to achieve scalability there is no need to make any changes to the source code.

The use of this type of parallelization allows to organize multi-access to the program. This is very useful for teaching students. With multi-access to the program, they can simultaneously solve the same problem, each with its own settings of input data, or different tasks.

As can be seen, parallelization by tasks has many advantages, but there is one serious drawback. Despite the ability to significantly accelerate the execution of multiple calculations at the same time, it is completely absent the opportunity to speed up a single calculation. No matter how many processor cores are available, a separate calculation will be able to use only one core. To solve this problem, is needed a different version of parallelization - parallelization by data [5]. This will significantly accelerate the solution of a complex task by dividing the computing process into multiple threads, each of which will run in parallel on different processor cores.

Once it is necessary to note that this type of parallelization is not available for all tasks. Moreover, we can not parallelize the entire task.

Each complex task will be split into three sections:

1) initial section in which there is a data separation and splitting by groups;

2) parallel execution section, where computing threads work independently on separate processor cores;

3) the final section, where the results of the individual streams are collected together, analyzed, followed either by the task completion, or return to the first section.

It is necessary to consider that the computing threads creation and data exchange with them, requires CPU time too, and if these costs are comparable with running time of computing the flow, then it is possible to obtain significant slowdown instead of acceleration.

Acceleration is possible only in case of separate cores, and if possible the separate memory subsystem. If the number of computing threads exceeds the number of cores, then again we obtain slowing rather than accelerating.

3. Conclusion

The method of parallelization computing process by tasks can significantly enhance the software package ability, that implements the generalized iterative algorithm GMDH work. This is possible through the implementation of multiaccess to the program, allocating for each calculating task the separate processor core.

Application of the parallelization by tasks along with parallelization by data allows to achieve maximum efficiency of software complex, depending on the available computing resources and tasks.

References

1. Volodymyr Stepashko, Oleksandra Bulgakova, Viacheslav Zosimov. Modified multilayered GMDH algorithm with combinatorial optimization of partial descriptions complexity. – *Proceedings of the International Workshop on Inductive Modelling IWIM-2010*, Ukraine. – Yevpatoria, 2010.
2. Zosimov V., Stepashko V., Bulgakova O. Enhanced technology of efficient Internet retrieval for relevant information using inductive processing of search results. / Artificial Intelligence Methods and Techniques for Business and Engineering Applications / G.Setlak, M.Alexandrov (Eds.). – Rzeszow, Poland; Sofia, Bulgaria: ITHEA, 2012. – 345 p. / – P. 99-112.
3. Astratyan R.A. Internet service providing information interaction in modern distributed heterogeneous systems / R. A. Astratyan, V.N. Lebedev. – Moscow : Lenand, 2009. – 130 p.
4. General shema «backend-frontend» - http://www.danshin.ms
5. Zosimov V., Bulgakova O. Usage of grid systems for the distribution of the computing process by data in the inductive modeling algorithms. - Intellectual systems for decision making and problems of computation intelligence ISDMCI-2013, Ukraine. – Yevpatoria, 2013.
6. Stepashko V., Bulgakova O., Zosimov V. Performance of Hybrid Multilayered GMDH Algorithm // Proceedings of IWIM 2011. 4nd International Workshop on Inductive Modeling, July 4-10, 2011, Kyiv. – P. 109-113.
7. Bulgakova O. Comparison of prediction accuracy using different forecasting models / Bulgakova O., Zosimov V. // Materials XIV International Conference on Automatic Control - Sevastopol, 2007. – P. 126-128.