Agent-based computational system

Lopatkin R.Yu. [1], Petrov S.A. [2], Ivashchenko V.A. [1]

[1] *Institute of applied physics NASU, Sumy, Ukraine*
[2] *Sumy state university, Sumy, Ukraine*

lopatkin@iap.sumy.org, sergpet@gmail.com, va.ivashchenko@gmail.com

Abstract. *For development of agent-based computational system which allows to manage resources of personal computers was conducted discrete-event simulation. It allowed to determine necessary lifecycle algorithms of agents presented by finite state machines. Showed that agent migration between computers can be used for load balancing and improvement of resources usage. Proved the consistency of agent-based approach for building of compting networks and developed a working prototype on the base of jade framework.*

Keywords

Agent-based system, simulation, parallel computing, resources management.

1 Introduction

Modern tasks of high energy physics, genetic engineering, cosmology and other sciences demand a lot of computing resources. Last time grid technology grows rapidly to satisfy this needs of scientists [1]. But this technology requires a lot of funds and human resources.

On the other hand, each scientist has PC on their workplace (often with good characteristics). The question is: how efficiently resources PCs are used? Low load can be considered as a loss of valuable resources-hours.We believe that mainkind has accumulated a huge amount of computing resources as PCs resources and the question of their use in computing remains still open. According to Forrester Research analytical company forecast, in 2008 amount of PCs in the world exceeded 1 billion. Analysts expect 2 billions of PCs in 2015 [6].

We did monitoring of one LAN segment of Institute of applied physics NASU. This segment includes eleven PCs. On each PC was automatically launched monitoring agent, which translated each 10 seconds data about CPU and RAM load to monitoring service. We overwatched under investigation LAN for one week to get enough data for statistics. Obviously, that at off hours and at days off machines are turned off and monitoring shows zero load.

To solve computational problems in almost every research institute built computing cluster, which ultimately integrated into a grid association. Expensive server equipment, qualified personnel, with special server room air conditioning systems, power supplies, warning of harmful events (for example, a sharp rise in temperature due to the failure of the air conditioning system) is required to create a cluster. Thus, for the construction and maintenance of the cluster a lot of money and human resources is required and on the other hand PCs average loaded is about 1%. And, most importantly, personal computers of employees do not need specialized care, they are not concentrated in one particular place and do not require heat removal, replacement and repairs are cheaper, etc. For example, the Institute of Applied Physics of the National Academy of Sciences of Ukraine has a cluster of 10 dual-processor quad-core nodes that can simultaneously solve 80 independent tasks. But on the balance sheet of the institution is located about 150 personal computers with a total capacity exceeding the capabilities of the computing cluster.

Based on the monitoring data it is safe to say that on the basis of personal computers is possible to build an entire infrastructure solutions for sufficiently large scientific or production computational problems. But such a system, on the other hand have additional requirements. These include adapting and maintaining performance under dynamic changes in available resources: First, the system should minimally affect the usability of the owner of the machine, and secondly, any PC on the network can be turned off at any time, and it should not lead to system crash.

Thus, the problem of utilization of computing power of personal computers is extremely important. To solve it should be developed a flexible, service-oriented, self-adapting, fault-tolerant software system that would managed the resources of personal computers in order to use them for large and complex problems solutions.

Such attempts have been made long enough, however, they are primarily aimed at solving of a particular problem [2, 3]. Also example of such project has been successfully launched in Ukraine [4]. But singularity of these systems is that the participants, who passed into general use their personal computer, do not solve their problems. They download

and install on their computers the program module (same for all), which receives from the main server a portion of the data for the calculation and after returns the result of the calculations. Thus, such a system can't be called multi-user and is not suitable for the described task.

2 Main sections

2.1 System description

The object of study is multiagent agent-based computing system (futher ABCS). This system can be deployed on heterogenous local network or global network areas. Heterogenous in current context means network of arbitrary topology with a variety of communication channels, consisting of computers of different configurations and capacities that runs different operating systems. Such computational network has service-oriented architecture and intended to provide data processing and data storing services of many users.

We used agent-based approach to develop and simulate ABCS. Proposed system contains three main types of agents:

– «Delegates» – this agents provide interface to users for interaction with the system;
– «Calculators » – this agents perform calculations;
– «Containers» – this agents hold and provide neccssary info about their PCs resources to Calculators and Delegates.

Each machine mandatory has one instance of Container. Container can't migrate, but Delegates and Calculators can do that according to their internal logic to the most appropriate machine (from their "point of view").

For simulation of ABCS above agents were represented as finite state machines. The logic of transitions between states of Delegate is shown in Tab. 1.

Tab. 1. The logic of transitions between states of the Delegate.

Input char	**State**	**Transition**
GetTask	Free	Geting_task
CreateTaskAgent	Geting_task	Creating_task_agent
LookAround	Creating_task_agent	Looking_around
Migrate	Looking_around	Migrating
Free	Looking_around	Free
Free	Migrating	Free
*	Free	Free

The logic of transitions between states of the Calculator is more complicated and shown in Tab. 2.

Tab. 2. The logic of transitions between states of the Calculator.

Input char	**State**	**Transition**
Solve	Free	Solving
Wait	Free	Waiting
LookAround	Free	Looking_around
Solve	Looking_around	Solving
Migrate	Looking_around	Migrating
Wait	Looking_around	Waiting
Solve	Waiting	Solving
*	Waiting	Waiting
Wait	Migrating	Waiting
Solve	Migrating	Solving
TakeSolution	Solving	Taking_solution
Destroy	Taking_solution	Destroying

The logic of transitions between states of Container is shown in Tab. 3.

Tab. 3. The logic of transitions between states of the Container.

Input char	State	Transition
Solve	Free	Busy
TakeSolution	Busy	Free

The relationship of finite automata is accomplished by passing the input symbol whenever the state is changed. For example, the Calculator in the transition to a state Solving (solution) transfers incoming symbol Solve to Container state machine container, which leads to a change in it condition.

In general, the system interacts with the flow of resources and tasks. If the incoming flow of resources P_{in} и and outgoing resources P_{out} equals to zero, it's possible to say of a fixed problem where the resources of the whole system unchanged. Clearly, if $P_{in} > P_{out}$, then system «grows» and if $P_{in} < P_{out}$, then system «narrows». The ratio between the incoming flow of tasks from users F_{in} and outgoing flow of solved problems F_{out} shows how efficiently the system deals with calculations. If $F_{in} > F_{out}$, then the system the system cannot cope with the tasks and internal tasks queue grows. This situation in the real world cannot continue for a long time and will inevitably lead to a system crash or failure to provide services. From a practical point of view it is interesting to know the point where for certain parameters of the system $F_{in} = F_{out}$, corresponding to the maximum load of ABCS.

2.2 Simulation results

We have modeled the stationary and non-stationary problem in order to determine the dependence of the efficiency of ABCS of various parameters (parameters and topology of the network, the distribution of tasks complexity, flux density, distribution of the power of computers, agents algorithms given in the form of finite state machines).
The most basic goal was to identify efficient behaviors of agents for which the system will be able consistently and efficiently manage the flow of tasks from the user, as well as to identify the range of applicability of such a system. It was found that the capacity for migration of Delegates and the Calculators though leads to significant overhead to transferred data between the machines, but leads to the uniformity of the system load.
On the Fig. 1 shown one of results – dependency of increasing system resources efficiency with a constant flow of tasks and certain number of users (in this case there was 10 users) for four different situations:
1. Agents don't migrate
2. Only Delegates are able to migrate
3. Only Calculators are able to migrate
4. Both Delegates and Calculators can migrate.

As shown on Fig. 1, at the initial stage, when the system consists of very small amount of computers, obviously, it can't cope with the flow of tasks from the user and the task queue is growing. Then there is a positive flow of resources - the number of computers is growing. If there are no agents migrate the system efficiency drops sharply since Delegates that generate the Calculators remain on the first computer, and the rest are idle.
This task queue grows as the system can't cope with them. The red line in the figure corresponds to the situation when the Delegates are able to migrate. In this case, the efficiency initially rises to 0.9 , but then again falling rapidly. This is due to the fact that all Delegates migrate to other containers and uniformly distribute throughout the system. Further, they create in their containers calculators that can't migrate. It turns out that into the data flow tasks involved as many Containers as Delegates.
Charts 3 and 4 are very similar, as meet the situation when Calculators can migrate . This shows that migration of Delegates not so much affects the load of system. Sharp break can be observed at a time when the system has about 150 computers, because earlier the system wasn't able to cope with the incoming stream of tasks, but then comes a turning point when the number of computers in the system becomes more than count of tasks and all Calculators thanks to the ability to migrate evenly distributed throughout the system. Just at the moment when the 150^{th} computer is connected to the system, the accumulated task queue overs and further resources are not fully utilized.

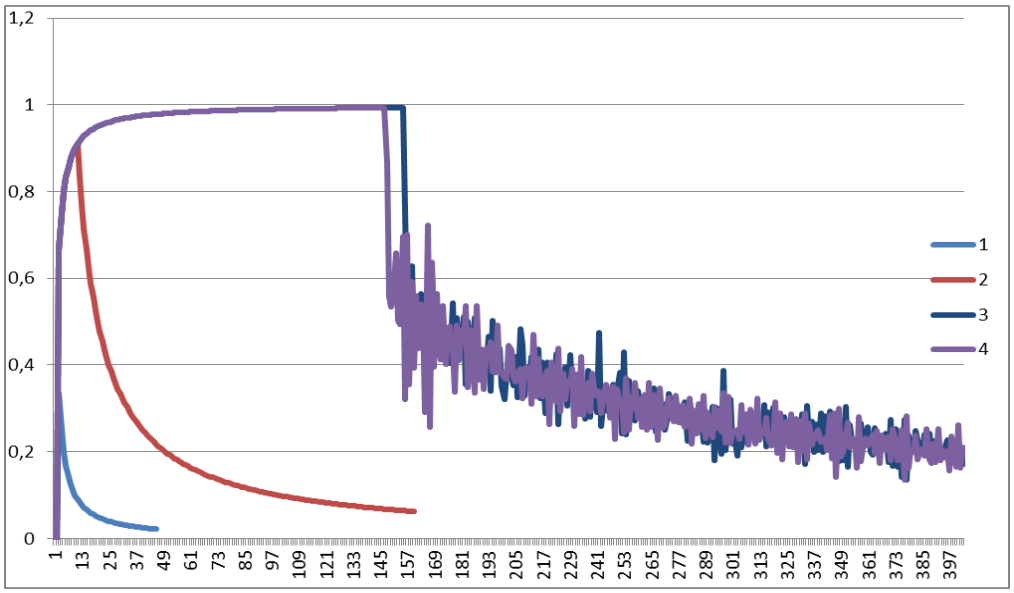

Fig. 1. Dependency of increasing system resources efficiency with a constant flow of tasks and certain number of users

The simulation results show that the above ABCS:

- can be deployed on a network of personal computers;
- sufficiently stable and may be protected by special failure-tolerant algorithms;
- can operate at a sufficiently wide range of internal parameters;
- has a high efficiency of the order of 95% for different conditions.

We develop system prototype based on agents lifecycle algorithms used in ABCS simulation.

2.3 Prototype of agent-based computing system

System architecture is hierarchical and contains five levels: 1. Hardware level – it's a base level, consists of computers and networking. 2. Data transmission protocols – provides communication between computers. 3. The third level is the base for ABCS. It can be any framework, which can work with entities like agents and provides communication between them. Nice example of such framework is Jade [5] (we use this framework in development). Jade implementations are available not only for Jave SE, but also for Java ME and Android platforms. It increases range of system use. 4. The fourth level contains logic of agents, provides main services such as computations, data storing, user authorization e.t.c. Development and optimization with simulation results of main agents algorithms is the main goal of our research. 5. The last, fifth level, provides user interfaces.

Agents classes hierarchy was developed in order to implement logical and applied levels. See Fig. 2.

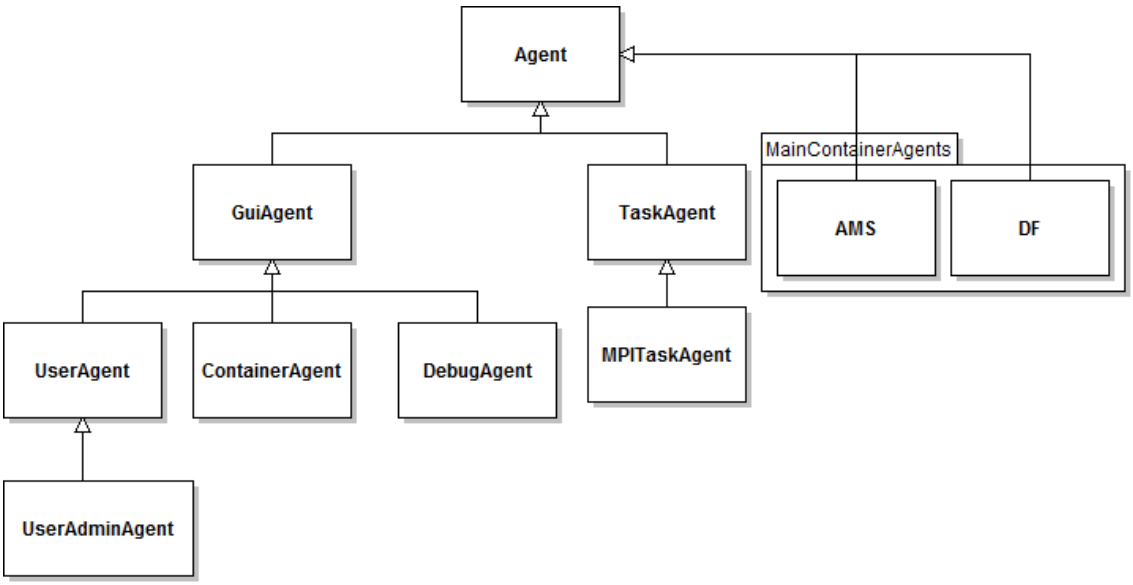

Fig. 2. Agents hierarchy

Agent – ancestor of all agents in the system, located in *jade.core.agent* package. Contains ready functionality for creating, deleting and migration of agents. **UIAgent** – agent with graphical user interface. **UserAgent** – agent of common system user. It is a virtual representative of the user in the system and can migrate to another machine when machine on which it located will be turned off. **UserAdminAgent** – agent of system administrator. This agent allows to perform system configuration, to admin system users, and also provides authorization service for common users. **DebugAgent** – agent for system life debug. **ContainerAgent** – one instance of this agent located on each machine. It responsible for gathering and transmitting of statistics about available hardware and load of it. Also provides information about amount of **TaskAgents** in it PC. **TaskAgent** – this agent will be created for each new task. Throughout it life cycle, this agent migrates to container suitable for the calculation of the problem, and then calculates the task and then waits a certain period of time to give results of calculations to user, and then deletes itself. This agent tied to UserAgent of a particular user. **AMS** and **DF** - standard system agents that are automatically started with the main container launch. **AMS** (Agent Management System) provides the name service (for example, ensures that each agent in the platform has a unique name) and the authority in the platform (for example, you can create and kill agents on remote containers by requests to **AMS**). **DF** - (directory facilitator) provides a "yellow pages" service, by which the agents can find each other by description of the provided services.

In the prototype implemented all user interfaces through which they can send their tasks to the system and receive the results of calculations from it. User does their work in several stages. First, the user writes the code for the solution of their problem according to the provided template. The user then sends the code through Delegate and loads the necessary data. The instance of the Calculator generates a given task, and acts on its own, looks for a suitable container, and expects to challenge for computing resources. At the end of calculation it goes into standby mode when the user requests the results of calculations. After a certain amount of time (adjustable by the administrator of ABCS) Calculator self-destructs. All stages of problem solving you can track through the task manager.

Tests have shown that the system is sufficiently robust to changes in the external environment and can be successfully applied in practice.

3 Conclusion

Thus, proved the consistency of proposed agent-based approach for building of a multiuser computational networks and identified effective agents algorithms. It allows to develop working prototype on the base of simulation results. Prototype based of Jade framework, because at the same time it provides features for multiagent systems components and communication of agents within service-oriented approach. At the moment ends implementation and testing of the prototype.

References

[1] Uwe Schwiegelshohna, Rosa M. Badiab, Marian Bubak Perspectives on grid computing // Future Generation Computer Systems -- 2010. -- Volume 26, Issue 8. -- P. 1104–1115.

[2] SETI@home - Search for ExtraTerrestrial Intelligence at home [Electronic resource]. Available on: http://setiathome.berkeley.edu/. - оглавление с экрана.

[3] Rosetta@home - Protein Folding [Electronic resource]. Available on: http://boinc.bakerlab.org/rosetta/. - оглавление с экрана.

[4] Distributed Computing team of Ukraine [Electronic resource]. Available on: http://distributed.org.ua/

[5] Jade - Java Agent DEvelopment Framework [Electronic resource]. Available on: http://jade.tilab.com/. - оглавление с экрана.

[6] 2 billions PCs forecasted for 2015 [Electronic resource]. Available on: http://www.cybersecurity.ru/hard/25810.html