Available online at www.sciencedirect.com

ScienceDirect

AASRI Procedia 9 (2014) 131 – 137

AASRI Procedia

www.elsevier.com/locate/procedia

2014 AASRI Conference on Circuit and Signal Processing (CSP 2014)

New Method to Use Idle Personal Computers for Solving Coherent Tasks

Anatoly Kalyaev[a]*, Iakov Korovin[a]

[a]*Scientific Research Institute of Multiprocessor Computing Systems of Southern Federal University, Chekov st., 2, Taganrog 347922, Russia*

Abstract

In this paper authors offer the new method for solving coherent tasks in distributed computing system, based on resources of personal computers. Parameters of such resources are dynamically varying and that makes it hard for their application in distributed computations. To achieve ability of effective usage of personal computers, proposed method uses multiagent approach: proactive agent controls every personal computer in distributed computing system, and process of task solving is dispatched decentralized by interactions of agents. To solve every incoming coherent task agents of the system unite into community and that makes it easier to dispatch and perform computations. The main benefit of proposed method is decreasing the price for creating and maintenance of distributed computing system.

Peer-review under responsibility of Scientific Committee of American Applied Science Research Institute

Keywords: distributed computing system; multiagent system; coherent tasks; communities; proactive agents; decentralized system.

1. Introduction

The idea of using distributed personal computing resources for solving complex tasks appeared more than thirty years ago, but only about ten years ago due to progress in global and local networking different

* Corresponding author. Tel.:+7-952-5666699; fax.:+7-952-5666699.
*E-mail address:*anatoly@kalyaev.net.

Peer-review under responsibility of Scientific Committee of American Applied Science Research Institute
doi:10.1016/j.aasri.2014.09.021

organizations begin to use distributed computing systems (CS) based on such resources. At first it were GRIDs proposed in early 2000s [1]. Today, the relevance of such systems grows, powered by a progress of both networking and personal computers.

While analyzing some widely spread distributed CS [2][3] we noticed that majority of them have limitations: part of such systems can solve only easily decomposing non-coherent tasks and other part can be based only on a set of similar dedicated computing nodes. However, today networks connect many personal computers and almost all such computers are not fully loaded all the time.

The main problem of using such personal computers (PCs) in distributed CS is varying of their parameters (such as performance and so on) any moment due to their owners actions. Another problem is probability of coexistence of many different PCs in one distributed CS. All that makes the process of effective distributing and solving of tasks very complicated.

Solving coherent tasks in such distributed CSs based on private PCs collaboration is considerably more difficult task. Today effective loading of a distributed CS while solving coherent tasks is a challenge [5]. However, solving such tasks in case of varying of parameters of PCs dynamically changes is even harder, that is why creation of new method of organizing of distributed computation of coherent tasks is very important task.

2. New method of solving tasks in distributed computing system based on personal computers

This new method is based on multiagent approach [6] and collective decision-making principles [7]. In the proposed method of distributed CS organization, we try to avoid dedicated servers and to make a decentralized organization of the distributed CS [8]. Every PC of distributed CS has proactive agent software installed. Interaction between program agents of the system realizes process of dispatching of task solving.

The first problem we faced appeared because distributed CS can consist of different set of PCs any moment. That is why interaction between user and distributed CS is complicated because user does not know where to send his task. To solve this problem, we decided to use passive service nodes that serve as "bulletin boards" (BB) [9]. A structure chart of proposed decentralized distributed CS is shown on figure 1.

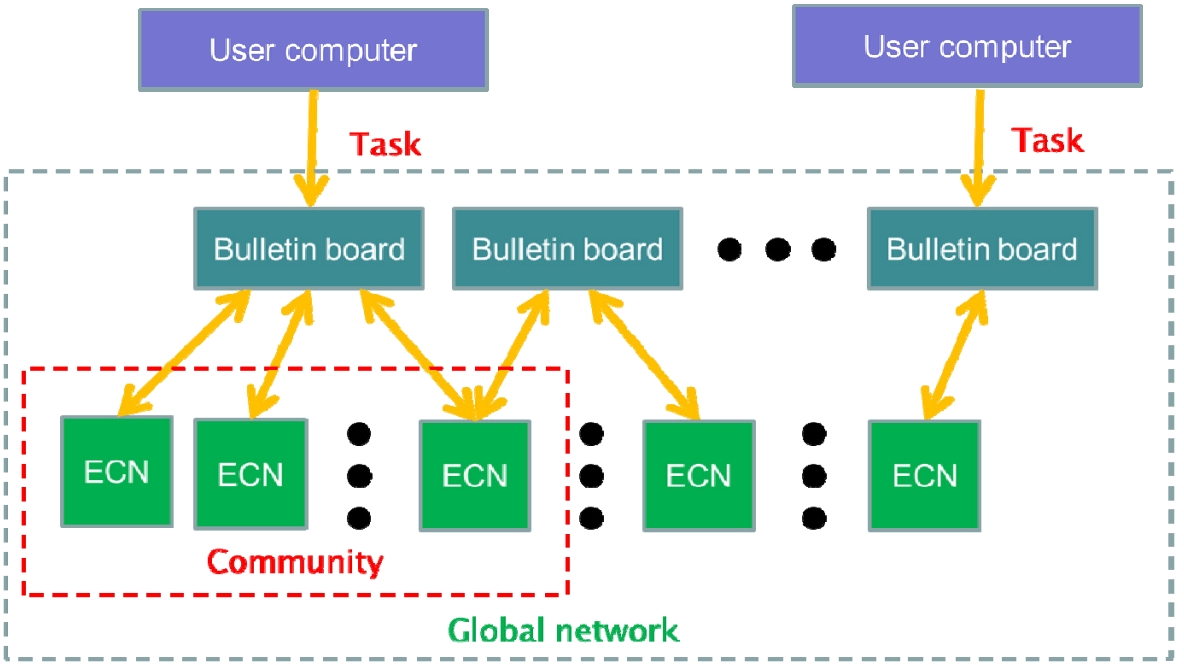

Fig.1. a structure chart of distributed CS

As it can be seen, distributed CS is divided into three layers. Executive computing nodes (ECNs on the figure) perform almost all service tasks in such distributed CS. Agents dispatch the process of solving user tasks in distributed CS. In order to solve coherent tasks they create dedicated communities. The system may experience failures of PCs, variation of PCs computing power, reducing of network bandwidth, and all that can cause increasing of the task solution time. In case of all this situations community of agents have to change its composition in order to solve the task in time, specified by user.

In general, method of solving of incoming task in decentralized distributed CS can be described as follows. User of the system forms task needed to be solved, sets requirement for the time to solve the task and determines a virtual payment for solving, which is represented by some virtual points. After that user sends all the task data to a bulletin board of distributed CS. Proactive agents of the distributed CS are monitoring BBs and look for incoming tasks. If some agent finds new tasks, it evaluates profit of solving of each task and then chooses the most beneficial task. After that agent joins the community to solve chosen task and takes some subtasks to solve. Every time new agent joins community, it evaluated time of task solving. If the performance of the community becomes enough to solve the task in time the community stops its expansion. When new agent joins the community, it takes some subtask and begins to solve it. While solving the subtasks agents are monitoring their PCs' parameters. If variations of parameters cause exceeding of user-specified time of task solving, the community begins accepting new agents. When community solves the task, the agent that produce the result send it to the user and then make the community to dissolve itself [10].

To make it easier to understand proposed method we decided to divide it into 3 stages:

- • The formal representation of the task;
- • Creating community;
- • Distribution of subtasks.

Let us describe stages in more details.

2.1. Formal representation of task

At first, user has to present the task in right form to let the distributed CS perform the solution. We decided to represent task in the system in form of task information graph (TIG). TIG is a graph where vertices represent subtasks of the user's task and arcs represent data transfers between subtasks. In the TIG G (Q, X) every vertex qi represents i-th subtask. For very subtask user specifies the estimated computational complexity Yi. Every arc x(qi, qj) is specifies to the amount of data Wij in transfer from subtasks qi to subtask qj.

For convenience, we decided to present the TIG in the multi-layer parallel form (MPF) to simplify the further parallelization of the task graph. To do that all the vertices of the graph should be divided into subsets Vi according to the rule: if the arc x(qm, qn) connects the layer j to the layer k that means that $j < k$.

When TIG is represented in the MPF, we can easily evaluate the number of parallel computations of the task and the maximum number of agents in community for solving this task as the maximum quantity of vertices in layer of the MPF TIG.

At the next step the user have to specify maximum time to solving his task and priority.

After all that steps user sends the task to any bulletin board of distributed CS.

2.2. Creating community

When the user send the task to the bulletin board of the distributed CS, nodes of the distributed CS start to collaborate in a community, which will allow solving the task in user specified time [11]. While creating community of agents decentralized distributed CS follow steps below:

1. BB receives data of new incoming task and marks the incoming task as "incomplete".
2. Agent Aj looks for tasks that are incomplete at bulletin boards of distributed CS. For each found task, agent receives:
   - a A TIG G of the found task;
   - b The priority P of the task;
   - c Subtasks qi (i = 1, K) statuses ("ready" / "working" / "new");
   - d Maximum time to solve task Tmax;
   - e Maximum number of agents afforded by the task M;
3. Agent Aj evaluates the profit of solving all found incomplete tasks:
   - a Agent Aj ranks tasks their profitability value (P / Y) starting with the most profitable;
   - b The agent Aj looks for the next task from the list. If the list is empty, then go to step 4;
   - c Agent Aj receives parameters of agents in community and number of agents N;
   - d If N = M, the agent go to step 3b;
   - e Agent figures out if community have enough PCs to solve task in user-specified time, if it is true then the agent removes a this task from a list of incomplete tasks and proceed to step 3b;
   - f Agent Aj joins the community;
   - g If the performance of agents (found by formula (1)) is enough to perform users task in specified time, then agent marks the task as complete on the bulletin board;
4. If the agent Aj have successfully joined the community, then proceed to step 5, else agent goes into standby mode for time interval and go to step 2;
5. If task is marked as complete, the community is successfully created and agents start solving the task.

Steps above let the agents of distributed CS form a community to solve incoming coherent task in time. The set of agents in the community depends on their actual performance. After that agents of community need to effectively distribute subtasks make task be solved in time specified by the user.

2.3. Distribution of subtasks

Once agents of distributed CS successfully create the community, they have to perform the distribution of subtasks between their PCs:

1. When some agent in community becomes free (does not have subtask to solve), then this agent sends messages to all agents in community to make them start of free subtasks allocation procedure.
2. To participate in this procedure that each agent of community look for next unsolved subtask qR in TIG and calculates the time of its solving on PC of this agent;
3. Agent Aj sends the resulting value to all agents of the community;
4. When all agents in community receive all the values, they choose the lowest execution time. The agent As, who can afford this time take subtask qR, mark it as "working" and start its solution;
5. Agent As, requests the data required for the solution of task qR from all agents, which are owners of subset of vertices connected with vertex qR by incoming arc;
6. If one of agents finishes decision of some subtask, it marks this subtask as "solved";
7. After solving every subtask agent use its computational complexity YR and time of decision Tsol to determine the current performance of its PCs Sj, as Sj = YR / Tsol and sends the result values to all other agents and to bulletin board;
8. Agent estimate characteristics of the community to find out if the task can be solved in specified time. If community cannot solve the task, then agent marks it as "incomplete" and the creation of community restarts;
9. If there are any unsolved subtasks then go to step 1;
10. If the agent finishes solving of the last subtask in the user task it send the resulting data to the bulletin board and then to the user.

The main benefit of proposed algorithm is ability to adjust computing process to actual parameters of personal computers of distributed computing system. Two proposed algorithms allow solving coherent tasks in distributed CS based on personal computers with varying parameters. The next step is exploring effectiveness of proposed method and algorithms.

3. The experiments and the results

Verifying the efficiency of proposed algorithms for solving coherent task in distributed CS based on private PCs resources is very hard task because there is great amount of parameters in such system and it is almost impossible to carry out experiments with real hardware. That is why we had to create the program, which allows modeling the distributed CS containing of up to 1000 PCs solving up to 250 tasks. [12]. The form in the right shows a set of tasks of the distributed CS, their statuses ("new" / "solving" / "solved") and its parameters (the maximum time for solution, complexity, number of virtual points for solution, etc.). Each task in this table has different color. A visual representation of distributed CS PCs can be found in the center. Each circle represent modeled PC and its color matches the color of the task that is solving at PC.

The main target of experimental research is the practical proof of efficiency of proposed algorithms, but also we wanted to evaluate of the relative loss of processor time on PCs due to dispatching of the decentralized computing. That is why we had to carry out several series of experiments with different parameters of PCs and computer networks. While researching on the proposed algorithms, we used some types of real tasks we work with in Southern Federal University [13]. We decided to use percent of actual load at PCs, which can show loss of processor time due to organizational purposes and percent of tasks, solved in time, which show efficiency of system for end-user [14]. There are many important parameters in distributed CS that is why we decided to use planning of experiments: we divided parameters to three groups: Primary (system parameters) (P), Secondary (parameters of specific tasks and PCs) (S), and Tertiary (user-conditional

parameters) (T). In the end, we decided to explore influence of following parameters of distributed CS: number of PCs in distributed CS (P1); frequency of incoming tasks (P2); computing power of PC (S1); bandwidth of network (S2); computational complexity of task (S3); volume of data to send between subtasks of the task (S4); priority (T1); number of subtasks (T2); time for solving the task (T3).

For primary parameters, we decided to use absolute values, for secondary parameters relative values and for tertiary parameters limited random values. To make result more precise due to random values we carried out every experiment three times.

The results show that the percent of actual load at PCs was good: from 72% to 92%. That means that time loss due to organization ranged from 8% to 28% of the total time. Modern researches in the same directions [5] show that average effectiveness of solving coherent tasks in distributed CS is about 40%, which means that proposed methods are effective.

Conclusion

In this paper, we present the new method for solving coherent tasks in distributed computing system based on resources of personal computers, which are already in use.

To achieve ability of effective usage of personal computers proposed method uses multiagent approach. The main benefit of presented method is decreasing the price for creating and maintenance of distributed computing system. This approach was already successfully used in decision support, forecasting and diagnostic systems for oilfield equipment [15-18].

The next step of research planned by author is creation of functional prototype of distributed CS, which will allow creating real distributed computing system on base of computers of science laboratory.

Acknowledgements

The paper is published due the financial support of Russian Fund of fundamental research, projects 13-08-01172, 14-08-00776 and 14-08-00800.

References

[1] Foster, I., Kesselman, C. and Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.
[2] Thain, D., Tannenbaum, T., Livny, M. (2005). Distributed Computing in Practice: The Condor Experience Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, 323-356, 2005.
[3] Berman, F. Wolski, R. (2003). Adaptive computing on the Grid using AppLeS, IEEE Transactions on Parallel and Distributed Systems, Vol. 14, 369 – 382, 2003.
[4] Poleshaev P.N. (2012). Experimental research of algorithms of task planning for GRID-systems with simulator, Material of international conference “High Performance Computations” HPC-UA’2012 (Ukraine, Kiev, 8-10 October 2012).
[5] Ivashko E.E., Golovin A.S. (2012). Computation effectiveness of BOINC-GRID / Material of international conference “High Performance Computations” HPC-UA’2012 (Ukraine, Kiev, 8-10 October 2012) P187-193.
[6] Wooldridge M. (2002). An Introduction to MultiAgent Systems, John Wiley & Sons Ltd, 2002, 366 pages.
[7] Tarasov V.B. (2002). From multiagent systems to intelligent organizations, 2002.
[8]Kalyaev A.I. (2011). Decentralized organization manager for GRID community-based agents, Proceedings

of the SFU. Technical sciences, № 8, 2011, 230-238.
[9] Kalyaev A.I , Melnik E.V. (2010). About one way of the decentralized organization of calculations in grid, Extreme robototechnics - materials of conference with elements of young scientists school –2010. – P. 73-75.
[10] Kalyaev A.I. (2013). Multiagent Approach for Building Distributed Adaptive Computing System, Procedia Computer Science, Volume 18, 2013, Pages 2193-2202.
[11] Kalyaev A.I. (2012). Method and algorithms of the adaptive organization of the distributed computations in a decentralized GRID, Herald of computer and information tecnoligies №4, 2012г.– Moscow– P. 28-33.
[12] Melnik E.V., Kalyaev A.I. (2010). One way of decentralized organization of GRID-computations, Infocommunications and computational tecnologies – Ulan-Ude, 2010. - P.160-162.
[13] Hisamutdinov M.V., Korovin I.S., Kalyaev A.I. (2013). The Application of Evolutionary Algorithms in the Artificial Neural Network Training Process for the Oilfield Equipment Malfunctions' Forecasting, 2013 2nd International Symposium on Computer, Communication, Control and Automation (3CA 2013) Proceedings. – Atlantis press. Pages 253-257.
[14] Gorelova G.V., Radchenko S.A., Melnik E.V., Kalyaev A.I. (2007). Planning of experiment while researching new methods and algorithms of organization of distributed computations, Herald of computer and information tecnologies №10, 2007.– Moscow– P. 49-56.
[15] Korovin, Ya.S., Kalyaev. (2013). A.I.Methods and algorithms of improving the efficiency of data transmission systems in oil corporations' enterprise networks. Neftyanoe Khozyaistvo - Oil Industry, Moscow, №9, pp. 96-100.
[16] Korovin, Ya.S. (2007). Decision support system for electrical submersible pumps control on the neural network basis. Neftyanoe Khozyaistvo - Oil Industry, Moscow, №1, pp. 80-83.
[17] Korovin, Ya.S., Khisamutdinov, M.V.,Tkachenko, M.G. (2013). Forecasting of oilfield equipment work conditions with the application of evolutionary algorithms and artificial neural networks. Neftyanoe Khozyaistvo - Oil Industry, №12, pp. 128-132.
[18] Korovin, Ya.S., Tkachenko, M.G.,Kononov, S.V. (2012). Oilfield equipment's state diagnostics on the basis of data mining technologies. Neftyanoe Khozyaistvo - Oil Industry, №9, pp. 116-118.