MULTIAGENT WORKGROUP COMPUTING

Ben Choi
Computer Science, Louisiana Tech University, USA

ABSTRACT

The future of computing is moving from personal computers to communities of self-organizing intelligent agents. Although currently most computers are networked and can communicate with each other, they cannot yet fully work together and help each other. This paper describes a framework for networked computers to work in groups where computers can help each other perform various tasks. In this framework, a computer acts autonomously like a person in a community. Computers, having various abilities and workloads, join together to form workgroups and to benefit from belonging to communities. Future personal computer will no longer be working alone for one person but will work with a large number of other computers helping other people. Any person using a computer will have access to not just the computing power of his/her own PC but vast computing power of a community of computers. This framework combines many key technologies, including intelligent agents, multi-agent system, object space, and parallel and distributed computing, into a new computing platform, which has been successfully implemented and tested.

KEYWORDS

Intelligent agents, cloud computing, multi-agent system, parallel distributed processing, social network.

1. INTRODUCTION

Nowadays most computers are networked and can communicate with each other. Communicate is a key requirement for collaboration. The networked computers have provided a much faster and more effective media for people to communicate and to collaborate. The next stage is to create a platform for computers themselves to collaborate with each other.

Current researches on parallel and distributed computing and grid computing attempt to employ a very large number of computers to solve very large computing problems. These researches focus solely on computing speed. They partition a very large computing problem into small pieces, send each pierce to be computed by a computer, and then wait for all the results. This centralized control method of computing simply ignores the problem of collaboration between computers. On the other hand, current researches on distributed file sharing based on peer-to-peer networks attempt to allow every person to share his/her files and storage spaces through a decentralized network. This distributed file sharing method facilitates sharing of storage spaces but ignores the needs to share computing power.

Our projects attempt to create a platform for computers themselves to collaborate with each other to share computing power. In this platform, computers can help each other both in term of running applications and providing computing power. If a person needs to complete some tasks that are not capable on his own personal computer, his computer will ask other computers for help. His computer makes requests to other helping computers which complete the required computations and returns the results back to his computer. If a person working on certain job needs more computing power, her computer will ask other idle computers for help. Any person using a computer will have access to not just the computing power of his/her own computer but the vast computing power of a community of computers.

Our projects combine many key technologies, including parallel and distributed computing, intelligent agents, multi-agent system, object space, and multicast protocol, to form a unified computing platform. The platform should require minimal user involvement and system administration. To achieve this, our projects extend the notions of intelligent agents (Plekhanova 2002) and multi-agent system (Shamma 2008, Dignum 2009) to conceive of a computer as a whole including its software and hardware as an active agent. A computer acts autonomously like a person in a community. Computers, having various abilities and workloads, join together to form workgroups where they can help each other both in terms of the abilities and the workloads. This in turn requires a share place for the computers to communicate with each other. To achieve this, our projects extend the concept of Object Space to become an Active Space, which can function as a rendezvous, a repository, a cache, a responder, a notifier, and a manager of its own resources. This further requires a computer to be able to broadcast its requests to some or all computers in the workgroup. To achieve this, our projects use multicast network protocols for the communication.

The remaining of this paper is organized as follows. Section 2 outlines the related researches. Section 3 defines the framework of Multiagent Workgroup Computing. Based on the proposed framework, Section 4 describes an implementation of a platform for general computing, while Section 5 describes another implementation of a platform for high performance. And, Section 6 gives the conclusion and outlines the future research.

2. RELATED RESEARCHES

Although currently most computers are networked and can communicate with each other, they cannot yet fully work together and help each other. The ability of a personal computer depends on the installed software and the processing power of its CPU. If a person needs some new applications and more computing power, he/she needs to buy new software and new computer.

Current researches on collaboration focus on allowing people to work together. For instance, Microsoft NetMeeting provides a complete Internet conferencing solution. These researches do not intend to address the problem for computers themselves to collaborate. Current researches on parallel and distributed computing and grid computing attempt to employ a very large number of computers to solve very large computing problems (Berman 2003, Foster 2003, Joseph and Fellenstein 2004). For instance, Folding@home (Pande 2008) uses a very large number of personal computer and PlayStations to tackle previously intractable problems in computational biology. SETI@home (Anderson et al 2002) uses millions of personal computers (Volunteer computing (Miller et al 2009)) worldwide to search for extraterrestrial intelligence. These researches focus solely on computing speed and solving very large problems. Current researches on Cloud computing (Rittinghouse and Ransome 2009, Velte et al 2009) focus on delivering Web services. On the other hand, current researches on distributed file sharing based on peer-to-peer networks (Subramanian and Goodman 2005, Androutsellis-Theotokis and Spinellis 2004, Steinmetz and Wehrle 2005) attempt to allow every person to share his/her files and storage spaces through a decentralized network but ignore the needs to share computing power.

Our computing platform uses a share space for intelligent agents to communicate with each other. This share space is built upon an extended notion of Object Space (Freeman et al 1999). An Object Space is a shared medium that simply acts as a rendezvous for agents to meet there either to serve or be served without the knowledge of each other identity, location, or specialization. Other variations of Object Space are JavaSpace (Freeman et al 1999), IBM's TSpaces (Lehman 2010), TONIC (2008), JINI (2010), and TupleSpace (Carriero 2001). Object Space has also been used for other applications. One of the proposed applications (Engelhardtsen and Gagnes 2002) utilizes an Object Space as a repository of various roles where agents adapt to changing demands placed on the system by dynamically requesting their behavior from the space. A framework for cluster computing using JavaSpace has been described in (Batheja and Parashar 2001), which uses a network management module for monitoring the state of the agents and uses the state information to schedule tasks to the agents. JavaSpace has also been used for scientific computation (Noble and Zlateva 2001). These support the perspective that JavaSpace can be used for high performance computing.

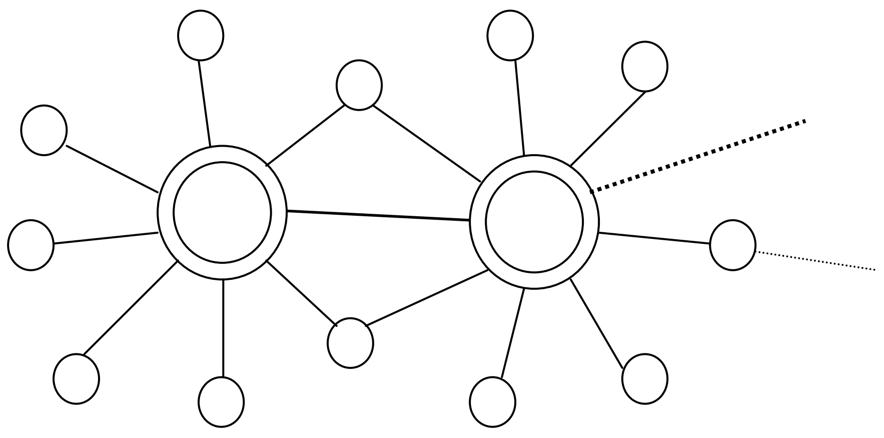
Figure 1. Computers joining to form two workgroups. (A ring depicts a workgroup manager and a circle depicts a computer.)

3. DEFINING MULTIAGENT WORKGROUP COMPUTING

In this section, the framework for Multiagent Workgroup Computing is defined. This computing framework is formulated such that any computer on the Internet can join workgroups. A workgroup can also link to other workgroups forming a whole community of computers. The organization of the computers can mimic the organization of a community. A computer can belongs to several workgroups and benefit from them. Figure 1 depicts a simple organization of twelve computers to form two workgroups. The center of a workgroup is the workgroup manager that is depicted by a ring. The workgroup manager is a computer serving to provide and maintain a share space for communication. Each of the computers is depicted by a circle. Two of the computers join both workgroups and one of the computers joins another workgroup not shown in the figure. The two workgroups are linked together and one of the workgroups is linked to another workgroup not shown in the figure. A computer has the freedom to join or leave any workgroup at any time. It is a free community and the computing community evolves over time like human community.

The ability of a workgroup is more than the sum of ability of its individuals. The function of a workgroup can be considered to be similar to the function of a discussion group. A computer needs a task to be done and posts the request on the workgroup. Another member in the workgroup reads the request, finishes the task, and posts the result on the workgroup. In this analogy, the workgroup serves two purposes: (1) it is a shared place for communication and (2) it is a depository of shared knowledge. Also similar to hosting a discussion group on a server, a computer can be used as a workgroup manager. The workgroup manager helps maintain the shared place for communication and organize the shared knowledge base.

The workgroup provides a shared knowledge base for the computers. This function of the workgroup is also similar to the function of a discussion group. If we have a question, we may find the answer on the prior postings of our discussion group. In this case, we do not need to ask anyone to help. Similarly, if a computer needs a task to be done and can just find the results on the shared knowledge base, then there is no need to repeat the computation.

The workgroup provides a mechanism for parallel and distributed computing. If a computer needs two tasks to be done, it can post both of them on a workgroup. Two other computers can concurrently work on the two independent tasks. This analogy can be extended to multiple tasks and multiple computers working in parallel.

Multiagent Workgroup Computing provides a general framework for multiple computers to work together in groups to share computing power and knowledgebase. To show the capabilities, this computing framework has been implemented and tested by two research projects, which are described in more detail in the following sections.

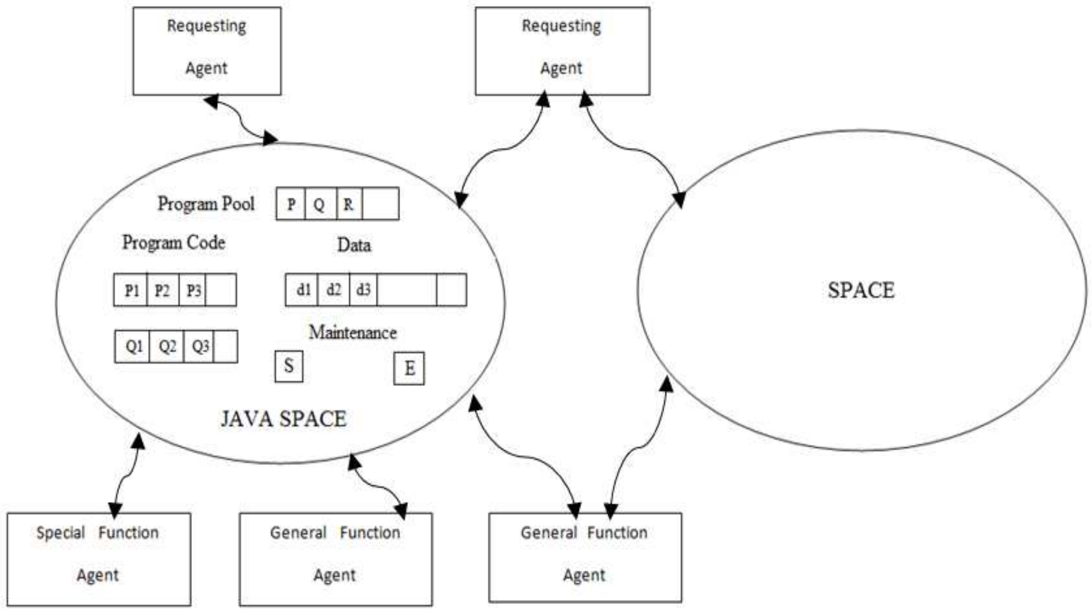

Figure 2. Agents joining space to form workgroups.

4. IMPLEMENTING A PLATFORM FOR GENERAL COMPUTING

Based on the framework for Multiagent Workgroup Computing, a computing platform for general computing has been developed, implemented, and tested. For the implementation of this platform, several key technologies have been used, including multi-agents, Javaspace, code mobility, caching, and multicast network protocol (Williamson 1999, Wen 2001). Figure 2 shows an overview of varies agents joining Space to form workgroups. In general a workgroup may consist of a large number of agents and the agents may join several spaces (only five agents and two spaces are show in the figure). Agents and spaces are implemented using JINI and Javaspace API's developed by Sun Microsystems (JINI 2010).

A computer can run many agents and assume many roles. In this platform, we defined three types of agents (as shown in Figure 2 (Bingi 2010)). A computer requests other computers for help by using Requesting Agents. A computer serves other computers by using Special Function Agents and General Function Agents. A Special Function Agent can only perform a specific predefined task. A General Function Agent can perform any task that is specified by a Requesting Agent.

Code mobility technique is used in this project to enable the platform for general computing. When a computer needs more processing power, it can send both the program code and the data through a Requesting Agent to a Space. The request, in this case, consists of both the program code and the data is stored on the Space. Another computer running a General Function Agent will monitor the Space and retrieve the pending request. The General Function Agent retrieves both the program code and the data. It uses the program code to process the data and generates the required results, which is then send back to the Space. The results are stored in the Space. The requesting computer, through the Requesting Agent, will then retrieve the results from the Space. In general a large number of requests can be send to a Space and a large number of computers will concurrently work on the requests, creating a general purpose, parallel and distributed computing platform.

A Special Function Agent, unlike a General Function Agent, can only perform a specific predefined task. In this case, the Requesting Agent does not need to send the program code, but only the name of the specific function and the data. A Special Function Agent retrieves and processes the request that matches its specialty. This processing method is similar to remote execution. However, the communications, in this case, are all through Space. A Requesting Agent does not need to know the destination IP address of a Special Function Agent.

A Space is used in this project not only as a share place for communications but also as a repository of knowledgebase. A sever computer can run the service of a Space. However, unlike other servers, this server does not process requests. Its main purpose is to serve as a share place for Agents to meet. In this project, we use multicast network protocol for an Agent to discover a Space to join. Thus, an Agent does not need to know the IP address of a Space. Through the multicast network protocol, it broadcasts its wish to join a Space. A Space responds to the request, and then establishes direct communication with the Agent. An Agent communicates with a Space, by placing requests on the Space and by retrieving results from the Space. Both the requests and the results are cached on the Space, which now serves as a repository of knowledgebase. The requests program codes are cashed on the Space, thus that the Requesting Agent does not need to resend the same program codes for used with different sets of data. The computed results are also cashed on the Space, thus that when another Requesting Agent needs the same request, it simply retrieves the results from the Space.

This computing platform has been implemented in our lab using several computers each of which has a network card that supports multicast protocol. Many test cases have been successfully executed to verify the various functionalities of this platform (Bingi 2010). One test case, for example, tests the fault tolerance of this computing platform, in which a General Agent died (maybe due to computer malfunction) before completing a task. In this case, another General Agent was able to pick up and finish the task.

5. IMPLEMENTING A PLATFORM FOR HIGH PERFORMANCE

Based on the framework for Multiagent Workgroup Computing, a computing platform for high performance computation has also been developed, implemented, and tested (Choi and Dhawan 2003, 2004). Although the framework is for general purpose, parallel and distributed computing, a search engine application that serves millions of users was chosen as a test case for implementing and testing the platform. Figure 2 shows the agent and space architecture designed for search engine (Choi 2001, 2006). Without going into too much detail, the search engine architecture consists of agents to handle requests, a space for searching, agents to handle search words, a space for searching words, and agents to retrieve search results.

High performance of the architecture is achieved by simply adding more agents, spaces, or networks. Another feature of the architecture for high performance is the result of using space as cache. When an agent needs to perform certain task and finds that the result of the task is already stored in the space, there is no need to repeat the computations. The agent simply reads the results from the cache. This not only reduces repeated computations when several agents need the same result but also reduces the response time, which is practically beneficial for search engine applications.

The architecture is highly scalable. Any number of agents, spaces, or networks can be added. Adding an Agent is as simple as connecting the agent to a network. The agent will then discover a space in the network and become part of the workgroup. It is a plug and play process. No manual configuration is needed. Similarly, adding a space is simply connecting the space to the network and the space will broadcast its present through the multicast network. Adding a network is as easy as connecting agents and spaces into the network. This is made possible by the fact that agents and spaces can be connected to multiple networks through multiple ports.

High availability and fault tolerance is achieved through multiple agents, spaces, and networks. For instance, having multiple agents performing the same role, the failure of an agent only downgrades the performance and will not affect the overall functionality of the system. Replacing an agent can be as simple as disconnecting the agent from the network and connecting another one. Similarly, having multiple spaces, the failure of one space again will only downgrade the performance. An agent pending for a request to be completed will discover that the space is not available and will then send the request to another space. Similarly, having multiple networks, the failure of one network will only downgrade the performance in a larger extent. Agents and spaces can continue to communicate through their ports that are connected to a live network.

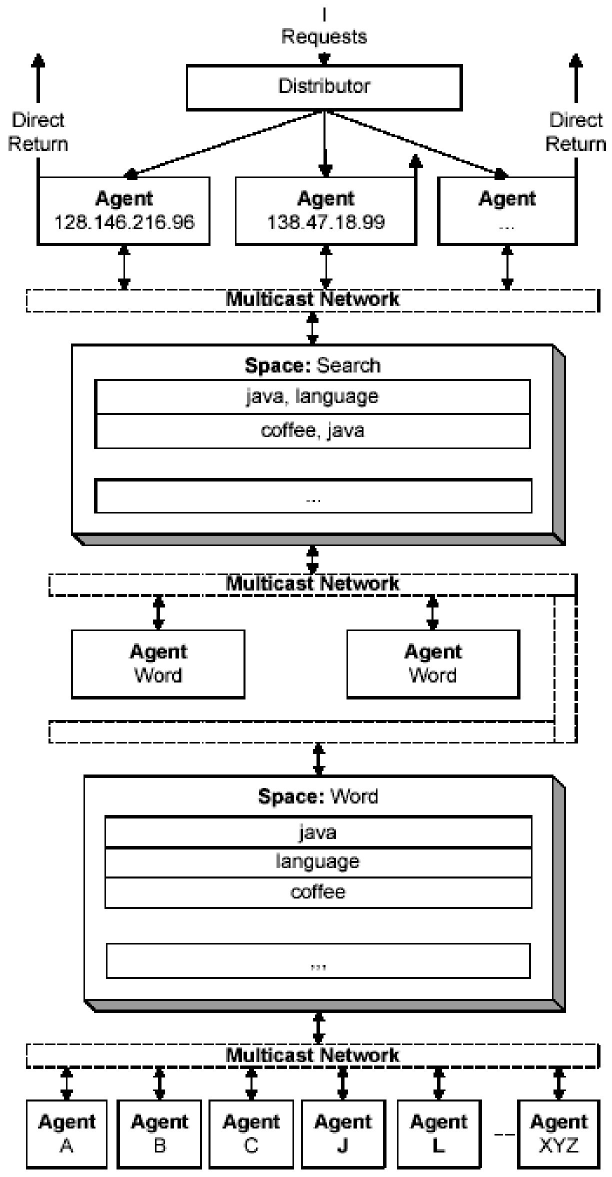

Figure 3. Agent space architecture for search engines

6. CONCLUSION AND FUTURE RESEARCH

This paper describes a framework for networked computers to work in groups where computers can help each other perform various tasks. A computer can run many agents and assume many roles. Agents join a workgroup through a share place of communication called Space. A sever computer can run the service of a Space. A Space is used in the framework not only as a share place for communications but also as a repository of knowledgebase. Based on the framework, two computing platforms have been implemented and tested. One platform is designed for general computing, which uses code mobility to allow a computer to specify a request by sending both the program code and the data. Another platform is designed for high performance and especially for use on search engine applications. Our experiences with these projects indicate that the proposed framework of Multiagent Workgroup Computing has many advantages, including high performance, scalability, and availability, requiring less human intervention, and providing natural fault tolerance.

The proposed framework is also applicable for extending the notion of cloud computing. This framework allows computation to be highly parallel and distributed all over the networked computers, and allows personal computers to join together to form large computing communities. The future of computing is moving from personal computers to communities of self-organizing intelligent agents.

REFERENCES

Anderson, David P.; Cobb, Jeff; Korpela, Eric; Lebofsky, Matt & Werthimer, Dan, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, Volume 45, Issue 11, Pages: 56 - 61, November 2002.

Androutsellis-Theotokis, Stephanos & Spinellis, Diomidis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, Vol. 36, Issue 4, pp. 335-371, December 2004.

Batheja and Parashar, "A Framework for Opportunistic Cluster Computing using JavaSpaces", http: //www. caip. rutgers. edu/ TASSL/ Papers/ jinihpc-hpcn01.pdf , 2001

Berman, Fran (Editor); Fox, Geoffrey (Editor); Hey, & Anthony J.G. (Editor), *Grid Computing: Making The Global Infrastructure a Reality*, ISBN-10: 0470853190 Wiley, 2003.

Bingi, S.C. "*Code Mobility with Cache in Distributed Systems using Javaspaces*", Louisiana Tech University practicum report (supervised by Ben Choi), March 2010.

Carriero, Nicholas and Gelernter, David, *A Computational Model of Everything*, CACM 44(11): 77-81, 2001.

Choi, Ben. "Invention: Method and Apparatus for Individualizing and Updating a Directory of Computer Files", United States Patent # 7,134,082, patent issued on November 7, 2006.

Choi, Ben. "Making Sense of Search Results by Automatic Web-page Classifications," *WebNet 2001 -- World Conference on the WWW and Internet*, pp.184-186, 2001.

Choi, Ben and Dhawan, Rohit, "Agent Space Architecture for Search Engines," *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp. 521-525, 2004.

Choi, Ben and Dhawan, Rohit, "Distributed Object Space Cluster Architecture for Search Engines," *High Availability and Performance Computing Workshop*, 2003.

Dignum, Virginia (Editor), *Handbook of Research on Multi-agent Systems: Semantics and Dynamics of Organizational Models*, ISBN-10: 1605662569, Information Science Reference, 2009.

Engelhardtsen and Gagnes, "Using JavaSpaces to create adaptive distributed systems", http://www.nik.no/ 2002/ Engelhardtsen.pdf , 2002.

Foster, Ian (Editor) & Kesselman, Carl (Editor), *The Grid 2: Blueprint for a New Computing Infrastructure*, ISBN-10: 1558609334, Morgan Kaufmann, 2003.

Freeman, Eric; Hupfer, Susanne; and Arnold, Ken, *JavaSpaces: Principles, Patterns, and Practice*, Addison-Wesley, Reading, Massachusetts, 1999.

JINI, "Jini Specifications and API Archive", http://java.sun.com/products/jini/, 2010

Joseph, Joshy & Fellenstein, Craig, *Grid Computing: On Demand Series*, ISBN 0131456601, Prentice Hall PTR, 2004

Kilduff, Martin and Tsai, Wenpin, Social Networks and Organizations, ISBN-10: 0761969578, Sage Publications Ltd, 2003.

Lehman, T., et al, IBM Almaden research Center, http://www.almaden.ibm.com/cs/TSpaces/, 2010.

Miller, Frederic P.; Vandome, Agnes F.; and McBrewster, John, *Climateprediction.net: Personal computer, Parametrization (climate), Volunteer computing, Berkeley Open Infrastructure for Network Computing, University ... BOINC Credit System, FLOPS, Climate model*, ISBN-10: 6130215304, Alphascript Publishing, 2009.

Noble M. S. and Zlateva S., "Scientific computation with javaspaces," in Proceedings of the 9th International Conference on High Performance Computing and Networking, June 2001.

Pande, Vijay, "Folding@home distributed computing home page," Stanford University, http://folding.stanford.edu/, 2008.

Plekhanova, Valentina, *Intelligent Agent Software Engineering*, ISBN-10: 1591400465, IGI Global, 2002.

Rittinghouse, John; Ransome, James, *Cloud Computing: Implementation, Management, and Security*, ISBN-10: 1439806802, CRC, 2009.

Shamma, Jeff (Editor), *Cooperative Control of Distributed Multi-Agent Systems*, Wiley-Interscience, 2008.

Steinmetz, Ralf (Editor), Wehrle, Klaus (Editor), *Peer-to-Peer Systems and Applications*, ISBN-10: 354029192X, Springer, 2005.

Subramanian, Ramesh (Editor), Goodman, Brian D. (Editor), *Peer to Peer Computing: The Evolution of a Disruptive Technology*, ISBN-10: 1591404304, IGI Global, 2005.

TONIC, "Scientific Computing with JAVA TupleSpaces", http://hea-www.harvard.edu/ ~mnoble/tonic/doc/, 2008.

Velte, Toby; Velte, Anthony; and Elsenpeter, Robert, *Cloud Computing, A Practical Approach*, ISBN-10: 0071626948, McGraw-Hill Osborne Media, 2009.

Wen, Su; Griffioen, James; and Calvert, Kenneth, "Building multicast services from unicast forwarding and ephemeral state," In *OPENARCH 01*, March 2001.

Williamson, Beau, *Developing IP Multicast Networks*, Vol. 1, Cisco Press, 1999.