3C3-IOS-2-5

# The Development of the Resource Broker of Desktop Grid Federation for Tree Search Applications

Lung-Ping Chen [*1]

I-Chen Wu, Chih-Wei Hsieh,

Der-Johng Sun, and Hung-Hsuan Lin [*2]

[*1] Tunghai University [*2] National Chiao-Tung University

As the desktop grid federation has facilitated many large scale e-Science projects for the resource restricted organizations, people start thinking about a wide variety of applications. This paper studies the desktop grids of artificial intelligence computation applications that require dynamic and adaptive task control. This paper discusses the development of the resource broker that assigns tasks based on user credits as well as resource utilization for the users in and across the organizations. The result demonstrates that, in a desktop grid federation using the well-designed broker, a number of organizations can perform large scale applications efficiently via resource sharing.

## 1. Introduction

Desktop grid is a network computing model that can harvest unused computational power from desktop level computers [1]. The execution of desktop grid applications is coordinated by a central server node, which distributes task units over widely worker nodes, awaits the execution results, and eventually consolidates the result data. Considering the fact that today's personal computers are more powerful than workstations or even mainframes in twenty years ago, this model can offer low-cost and readily available resources by employing a large enough number of workers. Today, many research organizations have built desktop grids as a solution for large scale e-science projects.

Desktop grids have remarkable resilience in host connections since the worker nodes can be of different operating systems, and are not necessarily connection-oriented. For volunteer computing, a server partitions and assigns tasks to the public anonymous participants, called volunteers. Since volunteers are autonomous and can connect or disconnect from time to time, the single-worker response time is usually not a major concern. Statistically, the resource availability can be maintained in a certain level with an enough number of volunteers. Another model of desktop grid is to have dedicated workers which are maintained and directly controlled by the organization. A dedicated worker node provides computing resources of guaranteed quality and quantity.

Most existing desktop grid platform are developed intended to host Bag-of-Task (BoT) applications which contain a large set of task units without explicit precedence relations. Compared to other network computing models, the nature of loosely couple communication of worker nodes of desktop grids makes resource sharing much easier. The notion of reciprocal resource sharing has enabled many large scale projects for the resource-restricted organizations. Several infrastructures for resource sharing are discussed as follows:

Contact: Lung-Pin Chen, Dept. Computer Science and Information Science, Tunghai University, No.181, Sec. 3, Taichung Port Rd., Situn District, Taichung City 407, Taiwan, Phone:(04)+882-4-23590415, Fax:(04)+882-4-23591567, lbchen@thu.edu.tw

- Volunteer desktop grid: All the worker nodes of different organizations are directly connected to a central server [1,2]. The resource allocation policy is implemented by using different credit calculation formulas
- Grid computing community: A desktop grid can use the nodes in a grid computing community [3,4] as its workers.
- Peer to peer platform: This platform is based on peer to peer protocol which can easily achieve fairness. The key is to transfer data via the peer to peer network and to evenly distribute communication and computation load over the entire network.

The dynamic computation applications, requiring dynamic and adaptive task control, are considered beyond the scope of above platforms. The game tree search applications need to generate and prune tasks over loosely coupled worker nodes in a timely manner [5,6]. To address these issues, we develop a resource broker for dynamic computation tasks. The resource broker uses two-stage scheduling to ensure fairness resource sharing for the workers in and across the desktop grids. Also, the proposed broker supports push-mode communication that can generate and prune tasks in a timely manner. So, prompt interaction and dynamic job scheduling can be achieved. For example, in case that one move of a board game is found to be winning, the push-back winning message promptly hints the clients to stop jobs under other sibling.

The remainder of this paper is organized as follows. Section 2 discusses the requirement of game tree search applications. Section 3 discusses the system architecture of the desktop grid system. Section 4 discusses the development of resource broker. Finally, concluding remarks are made in Section 5.

## 2. Artificial Intelligence Game-Tree Search Applications

### 2.1 Computer Board Games

A typical board game contains two players, Black and White, which alternately place black and white stones on empty

intersections of a Go board (a 19x19 board) in each turn. The common computer board games include Connect6, Chess, Chinese Chess and Go, Shogi, etc.

A game tree is a directed graph whose nodes represent states of the game board and whose edges represent moves. The computer board games heavily reply on tree search algorithms in several ways. Starting from a state, the game tree search algorithm is used to evaluate all possible moves and select a move based on certain policy. The challenge of computer board game comes from the fact that the size of state space of the game trees is usually exponential to the input size. Thus, an efficient tree search algorithm seeks to prune useless paths and go deep to the possible best moves. A typical strategy for evaluating best moves is to run a Monte Carlo tree search (MCTS) simulation of the game playing processes.

### 2.2 Application Components

The game tree search application contains two major modules: a game record editor and a job-level module. The game record editor is the interface of the computer board games which displays game status and waits for the player commands. A *job-level* (or *JL*) module is the component that executes jobs such as searching best game moves or end game matching.

The job-level module is a broker that accepts the game tree search jobs from the game record editor and dispatch them to the workers for running. Typical jobs are, by giving a start position, finding the best move, expanding all possible moves of a board state, or running a MCTS simulation. The execution result can be the best move, all the moves, or the simulation result for updating the tree. The job-level model defines the life cycle of a game tree search task into four phases: selection, pre-update, execution, and update phases.

In the board game application, both of the editor and job-level module take huge amount of times or uncertain times for executing tasks, making it difficult to be integrated. Thus, it becomes significant to offload the game tree search jobs to other workers.

## 3. System Architecture

This section describes the architecture of our computer game desktop grid (CGDG) system.

### 3.1 User and Worker Organizations

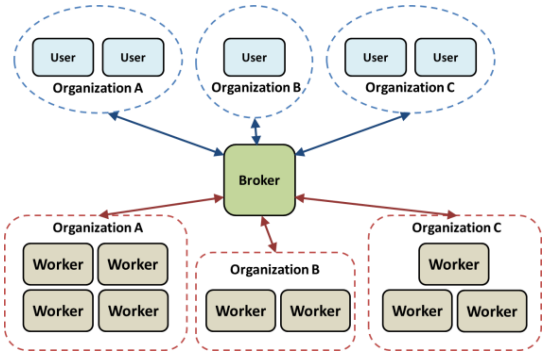


**Fig. 1.** Organization of users and workers.

The CGDG system consists of users, workers, and a broker. A user is usually the game record editor mentioned in Section 2, which accepts game player's instructions and initial the game computation tasks. The tasks are queued in the broker and are dispatched to some workers. The job-level module is a worker component that executes a tree search task.

The CGDG maintains four types of users, each with a different permission level, as described as follows.

- System administrator: The administrator with full access to every aspect of system data, including user profiles, organization profiles, broker policies, and job priorities, etc.
- Organization administrator: Similar to the system administrator but restricted to data of an organization.
- Standard user: A registered user that can submit normal tasks via the game record editor.
- Advanced user: A user that is authenticated to submit tasks with high priority.

Fig. 1 illustrates several user and workers in three organizations.

### 3.2 Resource Broker

A CGDG system contains clients, workers, and VC server. The client is a user application that generates jobs from time to time, while a worker is a computer that requests and solves tasks from VC server. The resource broker in the VC server is designed to coordinate clients and workers, while the web management system is used to manage accounts and system settings, etc. The system is called a push-based VC (PVC) system, since all connections among broker, clients and workers are all dedicated and are allowed to push jobs or messages immediately. For example, the clients push jobs to the broker that in turn pushes to workers, and the workers push or stream the results back to the broker which in turn pushes or streams them back to the clients immediately. So, prompt interaction and highly dynamic job scheduling can be achieved.

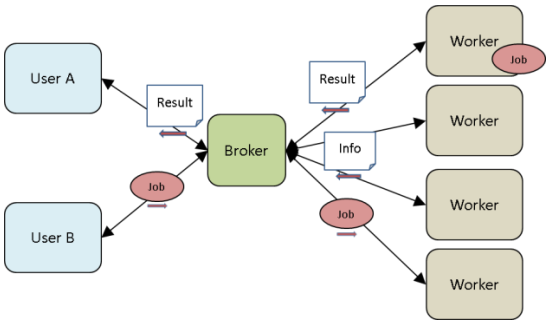


**Fig. 2.** Push-based communication of broker and workers.

Figure 2 shows the whole architecture of our PVC system. The bolded lines indicate dedicated connections, while the dashed lines indicate non-dedicated connections, e.g., HTTP requests.

As illustrated in Figure 3. The resource broker reserves a buffer for each client. Whenever an available worker sends a task request, the broker assigns a job choosing from some buffer. If more than one client has one job in their corresponding buffers, one of them is chosen and sent to the worker according to a designated resource management policy.

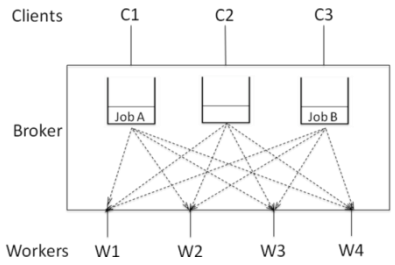


Figure 3: Job allocation in the broker.

## 4. Broker Resource Allocation

This subsection discusses the design and development of the resource allocation policies in the desktop grid federation. In additional to the resource utilization and system performance, the resource broker also concerns the following properties:

- Fairness: for each participant of the federation, the resource utilization is proportional to the user credits.
- Starvation-free: To avoid the situation that some organization cannot get minimal resource for a long time, the broker increases task priority as time goes by.

In each organization, all the tasks are clustered into three classes, A, B, and C. The class A and B, represent the priority of tasks that are submitted by the owner of the desktop grid, while class C represents the priority of the tasks that are generated by some other organizations. The class A, with highest priority, is intended for those interactive or responsive tasks. The tasks of class B are normal tasks and only gain the resource while class A tasks are idle.

The resource allocation algorithm is described follows:

1. A user submits a task to the broker, with an annotation describing the task properties, such as interactive or a normal long-term task.

2. When a broker receives the task, it first tries to assign the task the workers in the organization where the user belongs to. That is, an organization donates its resources only when its tasks completion rate exceeds that of generation. If the organization cannot host the new task, the broker checks the status of the desktop grids in other organizations.

3. If the broker decides to assign organization X's task to some other organization Y, the task is classified as class C in organization Y. The class C task in organization Y can be executed only when Y's tasks are idle.

To manage the game tree search applications of which tree nodes are generated and pruned dynamically, the broker calculates the credits of organizations based on the amount of donated resources. The amount of resources is calculated in terms of CPU cycles, storage space, and network bandwidth etc. Across different organizations, tasks are allocated based on fairness and starvation-free principles. When there are two or more organizations try to assign tasks via the federation broker, the priority is basically proportional to the credits. Also, in order to prevent starvation, the credits decline over time in a certain ratio.

## 4. Conclusions

We develop the desktop grids with the push-mode streaming infrastructure in order to support tightly coupled task control. The push-mode streaming communication can significantly reduce the redundant computations as tree nodes can be generated and pruned in a timely manner. This paper also depicts the characteristics of dynamic game tree search applications and discusses the development of the resource broker for the desktop grid federation of such applications.

## Acknowledgement

This work was supported in part by the National Science Council of the Republic of China (Taiwan) under Contracts NSC 97-2221-E-009-126-MY3, NSC 99-2221-E-009-102-MY3, NSC 99-2221-E-009-104-MY3, and NSC 101-2221-E-029-04.

## References

[1] Anderson, D. P., "Boinc: A system for public-resource computing and storage", 5th IEEE/ACM International Workshop on Grid Computing, November 2004.

[2] Fedak, G., Germain, C., Neri, V., and Cappello, F. Xtremweb: A generic global computing system. In *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2001): Workshop on Global Computing on Personal Devices*, IEEE CS Press, Brisbane, Australia, 582-587, 2001.

[3] Foster, I., Kesselman, C. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, Inc., 1999.

[4] Taiwan UniGrid website, available at http://www.unigrid.org.tw/info.html

[5] Wu, I.C., Huang, D.Y., and Chang, H.C., "Connect6", ICGA Journal, Vol. 28, No. 4, pp. 234-241, December 2005.

[6] Wu, I.C., Chen, C.P., "Desktop Grid Computing System for Connect6 Application", Institute of Computer Science and Engineering College of Computer Science NCTU, August 2009.SW

[7] Wu, I.C., Jou, C.Y., "The Study and Design of the Generic Application Framework and Resource Allocation Management for the Desktop Grid CGDG", Institute of Computer Science and Engineering College of Computer Science NCTU, 2010.

[8] Wu, I.C., Han, S.Y., "The Study of the Worker in a Volunteer Computing System for Computer Games", Institute of Computer Science and Engineering College of Computer Science NCTU, 2011.