Peer-to-Peer Scheduling System with Scalable Information Sharing Protocol

Norihiro Umeda
Tokyo Institute of Technology
norihiro.umeda@is.titech.ac.jp

Hidemoto Nakada
National Institute of Advanced
Industrial Science and Technology
hide-nakada@aist.go.jp

Satoshi Matsuoka
Tokyo Institute of Technology & NII
matsu@is.titech.ac.jp

Abstract

In traditional job scheduling systems for the Grid, a single or a few machines handle information of all computing resources and scheduling tasks. This centralized approach is not scalable, since it introduces single point of failure and bottleneck. Some decentralized scheduling systems have been proposed to improve scalability. They avoid concentration of scheduling costs by broadcasting job execution requests in a peer-to-peer style. However, resource utilization tends to be low, because most of them are not aware of the dynamic states of computing resources before throwing execution requests. This paper introduces a decentralized scheduling system that improves resource utilization by using a Gossip-based multicast protocol. With this protocol, peers can gather information of each other efficiently and schedule jobs individually. The simulation shows that our system is scalable and it handles many jobs efficiently in large scale Grid environments.

1 Introduction

In Grid environment, job scheduling systems have been widely used to manage computing resources and tasks. Their function is to accumulate resources information such as architecture, memory, disk storage, network bandwidth, software and its owner's policy and to assign submitted tasks to suitable resources to achieve improvement of utilization with referring these information Traditional scheduling system has introduced centralized architecture in resource allocation, hence there is single point of failure and bottlenecks.

On the other hand some decentralized scheduling systems have been proposed aiming to avoid these concentration of scheduling processes. In these systems job execution requests are announced by peer-to-peer network and peers voluntary accept these request. Accordingly, job allocator can not choose desirable peers to execute tasks efficiently.

Our proposed job scheduling system is that each peer shares resources information with Gossip-based multicast and can individually allocate resources for job execution, thus it accomplishes both decentralized architecture and utilization concurrently.

The rest of this paper is structured as follows. In Section 2 we discuss the related works. In Section 3 we present Gossip-based multicast. Section 4 presents that our decentralized scheduling system. In Section 5 we show the result of evaluation by simulation. Finally, Section 6 concludes this paper and .

2 Related works

2.1 Centralized Job Scheduling System

There are centralized job scheduling systems in practical use: Condor [6], Sun Grid Engine [4] and BOINC [2], etc. In these centralized systems one or a few machines manage a lot of computing resources and management nodes must be configured by system administrator. These system have two defects: single point of failure and concentration of management costs. If management nodes failed, all resources would be unavailable. And the increase of computing resources and tasks will cause the explosion of management cost. We consider these centralized job scheduling systems lack scalability for large-scale Grid environments.

Condor is a job scheduling system developed at the University of Wisconsin and now has been used widely in the Grid. In Condor, a set of computing resources is called Condor Pool. Condor Pool consists of a Central Manager as resource manager and resources providers. Central Manager is only one in a Condor pool and every resource provider periodically sends its status to the Central Manager. These information are expressed by ClassAd [7]. User submits job with ClassAd which express the requirement of execution and a criterion of choosing adequate resource. Central Manager decides resource allocation by using Matchmaking [8] with stored ClassAds. ClassAd and MatchMaking enable to improve efficiency of job execution and resources utilization.

Central Manager can have spare nodes recovering failure of original one. But there is only one Central Manager in Condor Pool at a time, hence concentration of management duty is unavoidable.

2.2 Peer-to-Peer Based Job Scheduling System

To recover defects of centralized systems, some decentralized systems like P3 [9] and Zorilla [3] have been proposed. Peers in these systems can behave resource provider and consumer simultaneously.

When a task is submitted to a peer by user, the peer broadcasts its execution request to other peers by using a overlay network. A request contains resource requirements to execute the job. When a peer receives a request message, it checks that whether it satisfies the requirements of the request. If its resources satisfy all of the requirements, the peer will start to execute the task.

However, job allocation peer is not aware of resource information, so can not choose resources from the viewpoint of efficiency and utilization.

3 Network Multicast Protocol for Large-Scale System

Gossip [1, 5] is a multicast protocol and provides highly reliable and scalable message dissemination.

Each peer selects randomly neighbor peers to send message at regular intervals. When a peer receives unknown message, it also starts Gossip process. Thus the number of peers having message grows exponentially with rounds. If the number of neighbors that have already received message is much enough, the peer will stop to send it.

Since a path of message is changed dynamically, fault of partial peers and network connections is automatically evaded and does not affect the whole of message propagation.

The shortcoming of Gossip-based multicast is that it is not guaranteed that the messages reach all members of the group certainly because its behavior is probabilistic. Nonetheless, the scalability and robustness are desirable properties to large-scale Grid which consists of enormous computing resources.

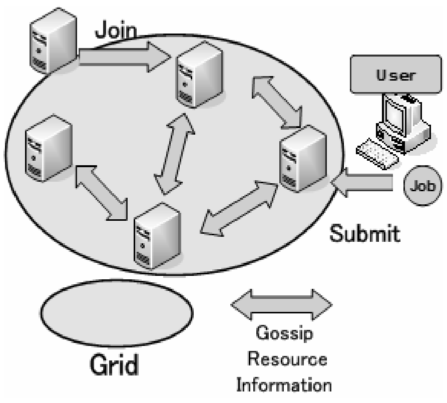

Figure 1. Overview of our system

4 Our Job Scheduling System with Gossip-based Multicast

To avoid concentration and single point of failure, we introduce peer-to-peer approach; each peer in computing resource pool acts as a resource provider and scheduler.

4.1 Overview

We describe how our proposed system works here. Figure 1 shows the overview of our system. Each peer in resource pools works as a scheduler which allocate tasks to other peers as well as worker to provide its computing resources. Resource information is shared among peers in the Grid, therefore there is no centralized element. A peer uses Gossip protocol to broadcast its status and owner's policies and stores other status. When one peer accepts job submission from users, it makes a pair of the task and machine referring to stored resource information like a Condor's MatchMaking.

4.2 Broadcasting Computing Resource Information

When a peer joins the Grid, it sends its resource information to arbitrary neighbor peer and receives resources information of others such as architecture, utilization of CPU, memory and disk, etc. This joining message is broadcasted with Gossip, hence new peer is quickly recognized by others. A peer in the Grid advertises its information with Gossip periodically or when its status change. Every resource information has a timestamp, so peers remove expired information. Consequently, dead peer is automatically excluded in resource lists of other peers.

Table 1. Composition of Peers

Relative Performance	0.6	0.8	1.0	1.2	1.4
Rate	0.2	0.2	0.2	0.2	0.2

Table 2. Parameters of Application

Number of Tasks	$1 * 10^6$
Average Process Time of Task in Standard Machine	10 minutes
Rate Parameter of Exponential Distribution	1
Scheduling Cost Per Task	15 sec * Number of Resources

4.3 Distributed Task Allocation

In our system, a user can submit tasks to arbitrary peers in the Grid. When a peer receives tasks from the user, it allocates a peer as worker referring to accumulated resource information. If allocated peer is not already available state, immediately scheduler chooses another peer again. Each peer which tries to allocate jobs works independently, therefore the larger the number of job submission points are, the more collision of resource allocation will occur. But a reallocation time is enough short, so this overhead will be negligible except when the job execution time is short.

5 Performance Evaluation

In this section, we describe the performance of our decentralized scheduling system. We implemented a simulator as a evaluation environment. It can simulate network communication, task allocation and execution.

5.1 Simulation Setup

We supposed a baseline reference machine whose performance value was 1 and every machine's performance was presented by relative value based on the reference. If a machine has 1.2 performance value, the task that the baseline finishes in 60 minutes will be processed in $60/1.2 = 50$ minutes. Composition of resources in the experiments are described in Table 1.

The sizes of tasks follows an exponential distribution. These parameters about submitted application are described in Table 2. This scheduling cost is based on our measurement result of Condor.

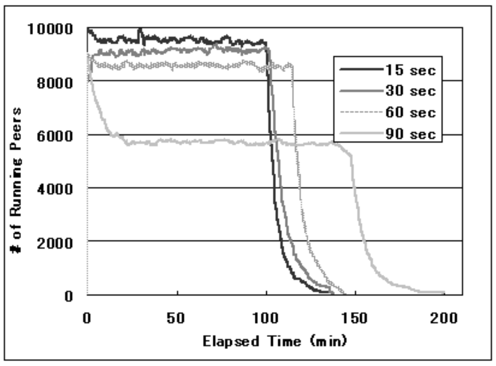

Figure 2. Gossip Interval

5.2 Evaluation Results

5.2.1 Gossip Interval

Firstly, we evaluated that the effect of Gossip interval. We designated gossip process interval to $n(n = 15, 30, 60, 90)$ seconds and submit the tasks to one peer. Figure 2 shows the results. As one can observe, the decrease of interval makes progress of resource utilization. However, excessly frequent gossiping consumes the network bandwidth. Since difference between the utilization of 15 seconds and 30 seconds was enough small, we chose 30 seconds as Gossip interval in the following experiments.

5.2.2 Number of Peers

Now we compare our proposed system and the centralized system similar to Condor. Table 3 shows the summary of results. The centralized system works effectively in small environments because the load of collecting information is modest and the overhead of Gossip-based multicast is relatively large in small group. But our system attains higher utilization than centralized in large system.

Figure 3 shows the detail of utilization in 10000 peers. At the begining, the centrarized system achieves high efficiency, but afterwards concentration of collecting resource information causes the decline of utilization. In contrast, our system constantly accomplishs good utilization.

5.2.3 Multiple submission points

We evaluated that how the numbers of submission peers effect the performance of the our system. We divided the tasks described in Table 2 evenly into $m(m = 1, 4, 16)$ and submitted them to m different peers. Figure 4 presents the time required to process the tasks. The result shows that the larger the number of submission point gets, the more utilization improved. This experiment proves the overhead

Table 3. Job Execution Speed and Utilization

	Centralized	Our System
	Finished Time	Finished Time
100	12941	14109
500	2607.2	2687.0
1000	1342.2	1303.0
2500	450.3	428.8
5000	294.2	268.0
7500	210.3	176.7
10000	170.4	138.9

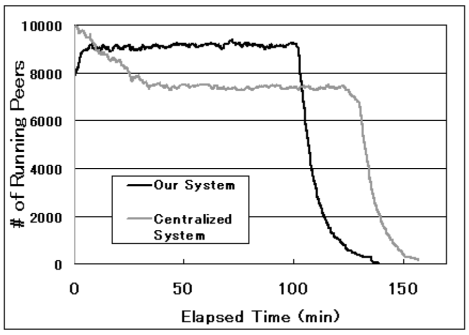

Figure 3. Utilization of 10000 Peers

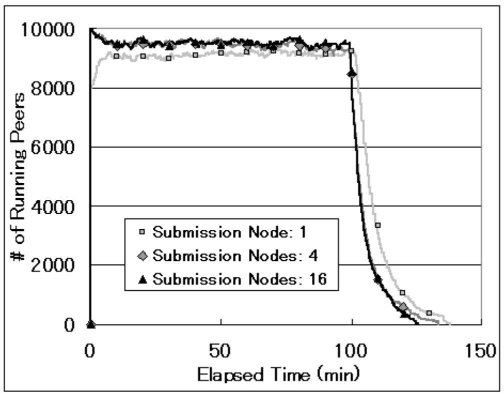

Figure 4. The Effects of Multiple Submission Points

of collision presented in Section 4.3 is enough minimal. Furthermore the increase of submission points improves the utilization because it reduces probabilities that no submission point is received the resources information.

6 Conclusions

We have proposed a peer-to-peer job scheduling system that allows every peer to allocate jobs to other peers as computing resource according to shared resource information using Gossip-based multicast. We evaluated our proposed system and the result showed that our system worked well with reasonable Gossip interval and it was more scalable than the centralized system with 1000 peers or more. And we examined the submission tasks from multiple point, the result showed the penalties of collision presented in Section 4.3 is negligible and the weakness of Gossip-based multicast is reduced.

For future work, we will introduce more parameters such as dependencies of tasks and failures of peers/networks into our simulation. And also we plan to implement job scheduling with our proposal and prove the feasibility in actual environment and applications.

References

[1] D. Agrawal, A. E. Abbadi, and R. C. Steinke. Epidemic algorithms in replicated databases (extended abstract). In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 161–172, New York, NY, USA, 1997. ACM Press.

[2] BOINC. Berkeley open infrastructure for network computing. http://boinc.berkeley.edu/.

[3] N. Drost, R. V. van Nieuwpoort, and H. Bal. Simple locality-aware co-allocation in peer-to-peer supercomputing. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, page 14, Washington, DC, USA, 2006. IEEE Computer Society.

[4] W. Gentzsch. Sun grid engine: Towards creating a compute power grid. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, page 35, Washington, DC, USA, 2001. IEEE Computer Society.

[5] K. Jenkins, K. Hopkinson, and K. Birman. A Gossip Protocol for Subgroup Multicast. In *International Workshop on Applied Reliable Group Communication (WARGC 2001)*, Apr. 2001.

[6] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems (ICDCS)*, pages 104–111, Washington, DC, 1988. IEEE Computer Society.

[7] M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for high throughput computing. *SPEEDUP Journal*, 11(1), June 1997.

[8] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7)*, Chicago, IL, July 1998.

[9] K. Shudo, Y. Tanaka, and S. Sekiguchi. P3: P2p-based middleware enabling transfer and aggregation of computational resources. In *Proceedings of Cluster Computing and Grid 2005 (CCGrid 2005)*, 2005.