Performance Evaluation Model for Scheduling in Global Computing Systems

Kento Aida[1] Atsuko Takefusa[2] Hidemoto Nakada[3]

Satoshi Matsuoka[4] Satoshi Sekiguchi[3] Umpei Nagashima[5]

[1]Tokyo Institute of Technology, E-mail: aida@noc.titech.ac.jp
[2]Ochanomizu University, E-mail: takefusa@is.ocha.ac.jp
[3]Electrotechnical Laboratory, E-mail: {nakada,sekiguchi}@etl.go.jp
[4]Tokyo Institute of Technology, E-mail: matsu@is.titech.ac.jp
[5]National Institute of Materials and Chemical Research, E-mail: umpei@nimc.go.jp

Abstract

Striking progress of network technology is enabling high-performance global computing, in which computational and data resources in a wide area network (WAN) are transparently employed to solve large-scale problems. Several high-performance global computing systems, such as Ninf, NetSolve, RCS, Legion and Globus have already been proposed. Each of these systems proposes to effectively achieve high-performance with some efficient scheduling scheme, whereby a scheduler selects a set of appropriate computing resources that solve the client's computational problem. This paper proposes a performance evaluation model for effective scheduling in global computing systems. The proposed model represents a global computing system by a queueing network, in which servers and networks are represented by queueing systems. Verification of the proposed model and evaluation of scheduling schemes on the model showed that the model could simulate behavior of an actual global computing system and scheduling on the system effectively.

1 Introduction

Striking progress of network technology is the enabling technology for high-performance global computing, in which computational and data resources in a wide area network (WAN) are transparently employed to solve large-scale problems. Several high-performance global computing systems, such as Ninf [1, 2], NetSolve [3, 4], Legion [5, 6], Globus [7, 8], and RCS [9] have already been proposed, where clients solve their problems by using remote computing resources on the network. Each of these systems proposes to effectively achieve high-performance with some efficient scheduling technique, whereby a scheduler selects a set of appropriate computing resources that solve the client's given computational problem. These include the Ninf metaserver [1], NetSolve agent [3, 4], AppLeS [10, 11, 12], Nimrod [13], Matchmaking [14, 15], Prophet [16] and Legion scheduling model [6].

Although these scheduling systems offer software mechanisms to achieve resource location and scheduling, scheduling schemes to effectively schedule multiple client tasks within a WAN on these systems and their performance evaluation models have not been discussed well in their literature. Rather, the systems are meant to be instantiated by such algorithm and schemes.

By contrast, job scheduling schemes as well as load sharing among computational servers in a tightly-coupled network have been extensively studied with formal performance evaluation models [17, 18, 19, 20, 21, 22]. However, although the models embody dynamic information of computational servers in the network to achieve efficient scheduling [23, 24], dynamic information of networks themselves is usually ignored. Although this is fine for tightly-coupled networks as network influence to performance is almost negligible, for wide-area networks, we have actually measured in benchmarks that network throughput greatly affects overall performance, even for simple situations such as single-server settings[2].

Thus, although an appropriate performance evaluation model is necessary to accommodate effective scheduling schemes, models for scheduling in global computing systems have not been well established. Models for tightly-coupled systems are not well-applicable. Performance benchmarking experiments us-

ing actual global computing systems, as we have done in [2], could be a partial solution, but experiments alone are not sufficient to evaluate performance of scheduling schemes in a general way for the following reasons: (1) although we can monitor the various fluctuations in the computing server and network load, they are usually quite unpredictable and difficult to replicate: (2) the scale of effectively measurable systems is quite small, as benchmarking actually takes considerable amount of time, (3) furthermore, it is not clear if results obtained on one system is applicable to other systems with different sets of servers and network topology or bandwidth.

We propose a performance evaluation model for effective scheduling in global computing systems. The model represents a global computing system by a queueing network, and can be shown to properly characterize the scheduling in WAN settings. We compare simulation results of the model to the actual benchmark measurements taken in [2], and verify that they closely match. We also show that how an exemplar scheduling situation can be simulated using the model, and show that network throughput must be carefully considered for effective overall performance.

The rest of this paper is organized as follows. Section 2 describes canonical architecture of global computing systems and its scheduling functionalities. Next, Section 3 describes the proposed performance evaluation model. Section 4 shows verification of the proposed model and performance evaluation results of basic scheduling schemes on the model.

2 Job Scheduling in Global Computing Systems — a Canonical Perspective

We first present a canonical architecture of global computing systems and its scheduling functionalities.

A global computing system is a form of distributed computing system, and typically consists of clients, servers and scheduling systems, which are interconnected via WANs such as the Internet, as shown in Figure 1. Clients solve their problems by using remote computing resources, or servers, on the network in a transparent way, typically using remote procedure calls and other forms of remote execution. A scheduling system consists of a directory service, a scheduler and a monitor/predictor. The monitor/predictor periodically watches/predicts loads of servers and congestion of network between clients and servers. The dynamic information watched by the monitor/predictor is stored into the directory service. The scheduler selects the appropriate server to compute tasks requested by clients using the information stored in the directory service.

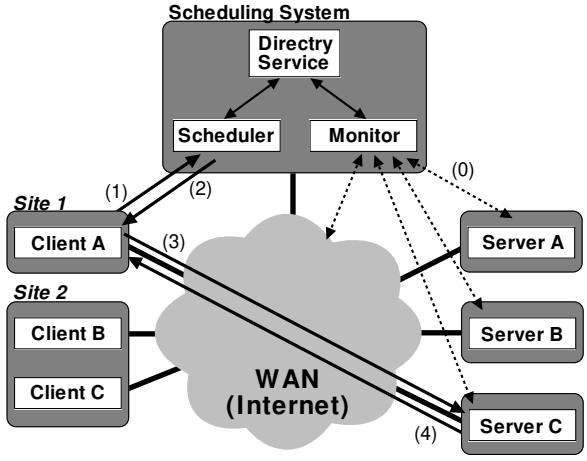

Figure 1: A Canonical Architecture of Global Computing Systems

Processes to schedule tasks invoked by clients are as follows:

1. The monitor/predictor in a scheduling system watches/predicts dynamic information of servers and networks periodically. ((0) in Figure 1)
2. The client queries the scheduler for an appropriate server to compute the client's task. ((1) in Figure 1)
3. The scheduler assigns the appropriate server to the client. ((2) in Figure 1)
4. The client requests the assigned server to compute the client's task and transmits the associated data to the server. ((3) in Figure 1)
5. The server computes the task and returns the computed result to the client. ((4) in Figure 1)

In practice, the client's main program contains a series of remote execution requests. Some systems allow the combination of client tasks to be executed in a data-flow manner.

A scheduling system is usually implemented as a software module executed on a certain node within the distributed system. The Ninf metaserver [1] is a server that allows the clients in WAN to transparently inquire server information on remote calls, and performs dynamic scheduling of dependent tasks. NetSolve agent [3, 4] has a similar feature. Globus-MDS [8] provides the directory service. AppLeS [10, 11, 12] provides scheduling methodology performed in the scheduler. Prophet [16] provides a scheduling mechanism performed by schedulers that are distributed in several sites. Network

Weather Service (NWS) [25, 26] is the system that performs monitoring and prediction of the dynamic information in a global computing system.

3 The Proposed Performance Evaluation Model

Scheduling in a global computing system is required to select appropriate servers under heterogeneous and dynamic environment. Performance of computing servers is heterogeneous and loads on the servers change dynamically. Also, topology of networks and bandwidths of networks are heterogeneous, and congestion on the networks changes dynamically. Scheduling results are affected by these heterogeneous and dynamic factors. Therefore, The requirement of a proper performance evaluation model for scheduling in a global computing system is as follows:

1. Topology
   The model has to be able to easily represent various interconnection topologies constructed by clients, servers and networks,

2. Server
   The model has to be able to effectively represent performance of servers, loads on servers and variance of the loads over time,

3. Network
   The model has to be able to effectively represent bandwidths of networks, throughput and variance of the throughput over time.

In order to satisfy the requirements, the proposed model employs a queueing network in which servers and networks are modeled as queueing systems. Figure 2 shows an example of the model: a server, a network from a client to the server, and a network from the server to the global computing client (hereafter simply referred to as clients) are represented by queues Q_s, Q_{ns}, and Q_{nr}, respectively. In the figure, Clients A and A' actually denote the same client, but they are distinguished for notational convenience. Tasks or data that arrive at Q_s, Q_{ns} and Q_{nr} are not confined to those issued by clients in a given global computing system, but also include those invoked by other, non-related external processes residing at all the nodes and elsewhere.

Arrivals of tasks and data are assumed to be Poisson, but other arrivals can be employed in the model. λ_s denotes the arrival rate of tasks, which have been issued both by target global computing clients and by external processes, to be executed at Q_s. Here, the arrival rate of tasks from external processes indicates the congestion of tasks on the server. Similarly, λ_{ns} and λ_{nr} denote the

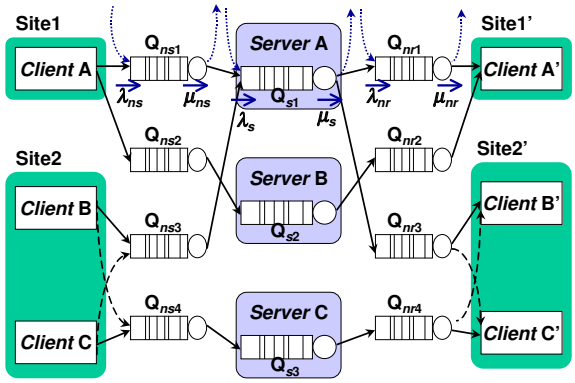

Figure 2: An Example of the Performance Evaluation Model

combined arrival rates of data associated with tasks of both the target global computing clients and external processes, arriving at Q_{ns} and Q_{nr}, respectively. The arrival rate of data transmitted from external processes indicates the congestion of data on the network. μ_s denotes the service rate on Q_s and represents the processing power of the server. μ_{ns} and μ_{nr} denote the service rates on Q_{ns} and Q_{nr} respectively, and represent the bandwidth of each network.

As an example, in Figure 2, clients B and C are assumed to be located within the same local site, with an underlying assumption that both clients share the network to any given server. In order to represent this sharing, data transmitted from both clients B and C to the same server are simply queued into the same queue.

We next describe the behavior of the client, the server, and the network, during their computing:

3.1 Behavior of the Client

A task invoked by a client is represented by following three factors:

1. computation of the task,

2. data transmitted to a server with the task as an input of the computation,

3. data transmitted from the server, or the computed result.

When a client requests to compute the task on a remote server, the client firstly queries a scheduler for an appropriate server to compute the task. After scheduler assigns an appropriate server for the client's task, the client transmits the data, which are associated with the

task, to the assigned server. Here, the data are decomposed into logical packets and each packet is transmitted to Q_{ns} that represents the connection to the destination server. The size of the logical packet, W_{packet}, and the transmission rate of the packet ,λ_{packet}, are derived in the following way:

1. W_{packet} is derived relatively arbitrarily by considering the tradeoff between accuracy of evaluation and time it takes to perform the evaluation. That is to say, although smaller W_{packet} gives more accurate simulation results, simulation time will increase significantly. The tradeoff is evaluated in the next section.

2. λ_{packet} is derived with (1) where T_{net} denotes the bandwidth of the network to the destination.

$$\lambda_{packet} = \frac{T_{net}}{W_{packet}} \tag{1}$$

3.2 Behavior of the Network from Client to Server

Q_{ns} corresponds to a single server queue with finite buffer [27]. Packets transmitted both from clients and from external processes arrive at Q_{ns}. When the buffer is full, packets transmitted from clients are retransmitted to Q_{ns} until they can be queued, while packets transmitted from external processes are simply *thrown away*. This means that the arrival rate of packets transmitted from external processes, namely λ_{ns_others}, determines the actual throughput obtained by client tasks on the network throughout a global computing system. Queued packets from the clients are processed for $\frac{W_{packet}}{T_{net}}$ time in First Come First Served (FCFS) manner. After being processed in Q_{ns}, the retained packets from clients are transmitted to Q_s while packets transmitted from external processes are thrown away.

λ_{ns_others} can be derived by average actual throughput of the target network to be simulated. In order to determine λ_{ns_others}, we employ (2) where T_{act} denotes the average actual throughput of the target network.

$$\lambda_{ns_others} = (\frac{T_{net}}{T_{act}} - 1) \times \lambda_{packet} \tag{2}$$

Also, the buffer size of the Q_{ns}, namely N, which determines network latency, is determined by (3) where $T_{latency}$ denotes the average actual latency of the target network to be simulated.

$$N = \frac{T_{latency} \times T_{net}}{W_{packet}} \tag{3}$$

As an example, suppose that we have a network whose bandwidth is $T_{net} = 1.0[MB/s]$. The average actual throughput and latency measured on the network are $T_{act} = 0.1[MB/s]$ and $T_{latency} = 0.1[sec]$ respectively. We define that average size of logical packets transmitted on the network is $avg.W_{packet} = 0.01[MB]$. Thus, the congestion of this network caused by external processes, or λ_{ns_others}, and the latency, or N, can be derived in the following way:

$$\lambda_{packet} = \frac{1.0}{0.01} = 100$$

$$\lambda_{ns_others} = (\frac{1.0}{0.1} - 1) \times 100 = 900 \tag{4}$$

$$N = \frac{0.1 \times 1.0}{0.01} = 10 \tag{5}$$

3.3 Behavior of the Server

Q_s corresponds to a single server queue [27]. As described earlier, tasks that arrive at Q_s include those from target global computing clients and from external processes. Thus, the arrival rate of tasks from external processes, namely λ_{s_others}, determines response time of client tasks on the server. A client's task is queued into Q_s after all the data packets associated with the task have arrived at Q_s from Q_{ns}. Although we assume that queued tasks are processed in FCFS manner, other scheduling strategies such as round robin can be employed. A queued task waits for its turn and is processed for $\frac{W_c}{T_{ser}}$ time where W_c denotes computation size of the task and T_{ser} denotes the CPU performance of the server. After processing on Q_s, data of the computed result generated for a client's task are again decomposed into logical packets and are transmitted to Q_{nr}. Here, the packet transmission rate is determined by (1). Tasks from external processes are thrown away after processing on Q_s; thus, we assume that they do not generate large amount of result data.

λ_{s_others} can be derived by average actual utilization on the target server to be simulated. In order to determine λ_{s_others}, we employ (6) where U denotes the average actual utilization on the server and W_{s_others} denotes average computation size of tasks invoked from external processes.

$$\lambda_{s_others} = \frac{T_{ser}}{W_{s_others}} \times U \tag{6}$$

As an example, suppose that we have a server whose performance is $T_{ser} = 100[MFlops]$, and the average actual utilization measured on the server is $U = 0.1$. We define that average computation size of tasks invoked from external processes is $W_{s_others} = 0.1[MFlops]$. Thus, the congestion of this server caused by external processes, or λ_{s_others}, can be derived in the following way:

$$\lambda_{s_others} = \frac{100}{0.1} \times 0.1 = 100 \tag{7}$$

Table 1: Requirements to Performance Evaluation Model

Resource	Factor	Representation	Parameter
topology	topology of queues	links among queues	information of links among clients and servers
client	packet transmission rate	λ_{packet}	bandwidth of the network, logical packet size
server	processing power	μ_s	CPU performance of the server
	degree of congestion	λ_{s_others}	CPU performance of the server, avg. actual utilization measured on the server, avg. computation size of tasks invoked from external processes
	change of the degree	distribution of arrival	Poisson etc.
network	bandwidth	μ_{ns},μ_{nr}	bandwidth of the network
	degree of congestion	λ_{ns_others},λ_{nr_others}	bandwidth of the network, avg. actual throughput measured on the network, logical packet size
	latency	N	bandwidth of the network, avg. actual latency measured on the network, logical packet size
	change of the degree	distribution of arrival	Poisson etc.

3.4 Behavior of the Network from Server to Client

The queue Q_{nr} is almost identical to Q_{ns}. Packets transmitted from Q_s and from external processes arrive at Q_{nr}. After processing on Q_{nr}, packets from Q_s, or computed results, are transmitted back to the client. The arrival rate of packets transmitted from external processes, namely λ_{nr_others}, are derived in the same way as λ_{ns_others}.

3.5 Summary of the Proposed Model

Table 1 summarizes requirements of a performance evaluation model for scheduling in global computing systems, and how our proposed model satisfies them. In the table, "Resources" and "Factor" indicate resources and their factors that the performance evaluation model has to represent. "Representation" shows how our proposed model represents the "Factor." "Parameter" describes the parameters required to derive the "Representation" in our model.

4 Evaluation

We now verify the validity of our proposed performance evaluation model by comparison of the result of the simulation on the purposed model with real performance measurements, and also show the utility of the model by illustrating the simulation results of three different basic scheduling schemes on the model.

4.1 Verification of the Proposed Model

In order to verify the validity of our proposed model, we compare the simulation results on the model with experimental results obtained from our high-performance global computing system Ninf that we have presented previously in [2]. The Ninf system is high-performance global computing system based on a client-server model. The client can request remote execution of various computing libraries and queries database resources on the server by invoking `Ninf_Call` on the client.

Figure 3 shows the global computing testbed prepared for the verification experiment using the Ninf system. In this environment, there are four clients located on nodes at different sites dispersed within Japan, namely, those at University of Tokyo (U-Tokyo), Ochanomizu University (Ocha-U), Nagoya Institute of Technology (NITech) and Tokyo Institute of Technology (TITech); the server is a J90 located at the Electrotechnical Laboratory (ETL), Tsukuba, Japan. Parenthesized values in the figure show the actual FTP throughput and Ping latency between the clients and the server, obtained with

Table 2: Simulation Parameters Provided for the Verification

Client	avg. logical packet size	10, 50, 100[KB]
Server	CPU performance of the server	500[MFlops]
	avg. actual utilization on the server	0.04
	avg. task computation size from external processes	10[MFlops] (exponential distribution)
	task arrival from external processes	Poisson
Network	bandwidth of the network	1.5[MB/s]
	avg. actual throughput of the network	see Figure 3
	avg. actual latency of the network	see Figure 3
	avg. logical packet size	10, 50, 100[KB] (exponential distribution)
	packet arrival from external processes	Poisson

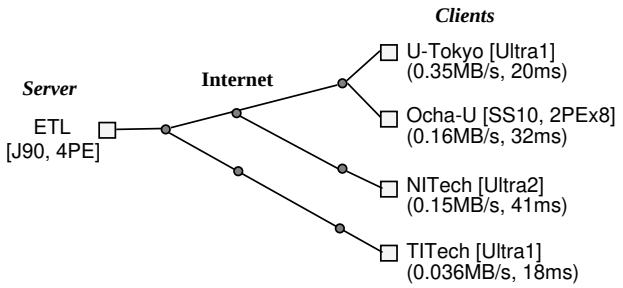

Figure 3: Global Computing Testbed for the Verification Experiment

actual benchmarking measurements.

The experiment was performed on the testbed in an actual internet environment. It means that there are tasks and data from other users on the networks and the server respectively, during the experiment. In both the experiment and the simulation, a client at each site invokes `Ninf_Call` with proper request parameters so that a Linpack routine is executed remotely on the server, and receives the result[2]. The Linpack solves a system of linear equation by Gaussian elimination, with computational complexity of $O(\frac{2}{3}n^3 + 2n^2)$ and communication complexity of $8n^2 + 20n + O(1)$ bytes for an n by n matrix. `Ninf_Call`s are invoked repeatedly in a non-overlapping manner, with sufficient repetitions so that the system stabilizes and allows us to properly measure the average elapsed time of each call.

Table 2 summarizes the parameters employed for the simulation using our proposed model. CPU performance and average actual utilization on the server, and average actual throughput and average actual latency on the network are obtained by measurement on the given testbed environment. Logical packet size in the simulation can affect accuracy of simulation results and the time spent in the simulation. Smaller packet size improves the accuracy of simulation results, while it takes long time for the simulation. Meanwhile, larger packet size can degrade the accuracy of simulation results, while it reduces simulation time. We practiced different simulations in which we define distinct logical packet size, which equals 10[KB], 50[KB] or 100[KB], for each to investigate the effect by the logical packet size.

We practiced 30 replications in one simulation and calculated average values as a result. All simulation results presented in this paper have confidence intervals of less than $\pm 10\%$ at a 90% confidence level.

Figure 4 shows the average overall performance obtained for each `Ninf_Call` invoked by the single client at Ocha-U computing Linpack. The overall performance is calculated from its average elapsed response time. In the figure, "problem size" indicates the size of matrices solved by Linpack, and "packet size" indicates the logical packet size employed in the simulation. Also, Figure 5 shows the average communication throughput obtained for data transfer of `Ninf_Call` invoked by the client at Ocha-U.

These results show that the average communication throughput and the average overall performance of `Ninf_Call` in the simulation closely match the experimental results on the testbed. Also, we see that the effect of the difference in logical packet size employed in the simulation is almost negligible; this means that, at least for Linpack and likely for more general settings, larger packet sizes can be employed for simulations, greatly speeding up the time required for running the simulation.

Figure 6 and Figure 7 shows the average overall performance and the average communication throughput obtained for `Ninf_Call`s invoked by clients at Ocha-U, U-Tokyo, NITech and TITech. Here, the logical packet size employed in the simulation is 100[KB]. Again, these results show that simulation results are almost identical to the experimental results for these multiple sites, whose actual network throughput is different as shown

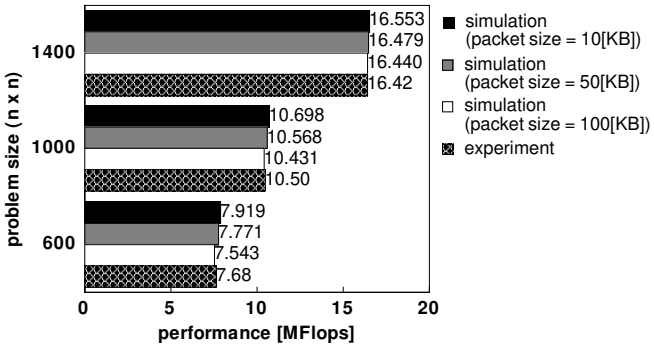

Figure 4: Average Overall Performance of Ninf_Call from Ocha-U

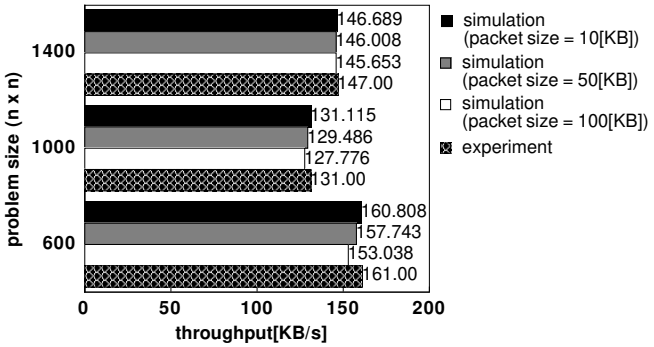

Figure 5: Average Communication Throughput of Ninf_Call from Ocha-U

in Figure 3. This means that the proposed performance evaluation model can effectively simulate the average performance of `Ninf_Calls` invoked by not only the client in a single site but also those in multiple sites.

4.2 Experimental Evaluation of Scheduling Schemes

Figure 8 illustrates an imaginary global computing environment used to test whether simulation of scheduling schemes using our model results in meaningful results. The environment consists of heterogeneous servers interconnected via networks with different communication throughput, and a scheduling system that schedules `Ninf_Calls` invoked by clients. We compare the following three different scheduling schemes employed by the scheduler in the simulation:

1. **RR**
   The scheduler selects the server in a round-robin fashion.

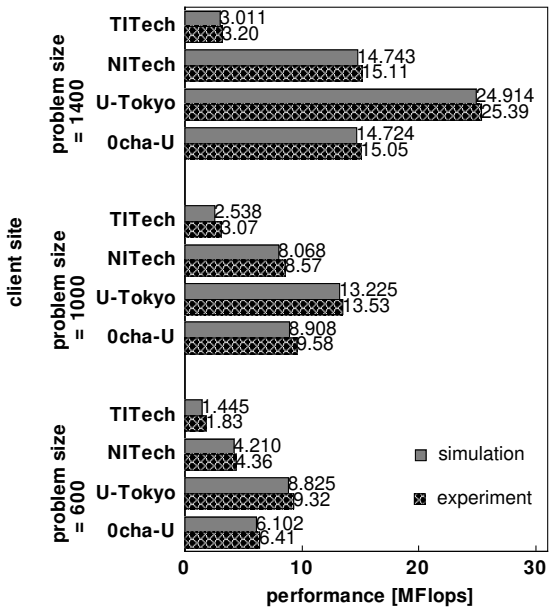

Figure 6: Average Overall Performance of Ninf_Call from Multiple Sites

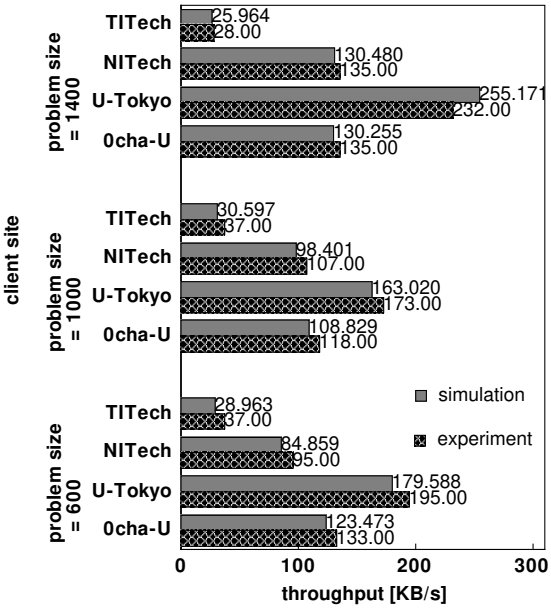

Figure 7: Average Communication Throughput of Ninf_Call from Multiple Sites

2. **LOAD**
   The scheduler employs server information and selects the most lightly loaded server, which has the smallest
   $$\frac{(L+1)}{P}, \tag{8}$$
   where L denotes the number of tasks in the server and P denotes CPU performance of the server, that is, $(L+1)$ means the number of tasks on the server where we assume to dispatch the task to the server.

3. **LOTH**
   The scheduler employs both server and network information. It selects the server that is expected to execute the client's task in the shortest time, or the server which has the smallest
   $$min(\frac{comput.}{\frac{P}{(L+1)}} + \frac{comm.}{T_{act}}), \tag{9}$$
   where "comput." denotes the amount of computation in the task and "comm." denotes the amount of data transmitted on the network with the task.

In this simulation, on each `Ninf_Call` a client invokes either a Linpack or the NasPar EP routine, and this is repeated in a non-overlapping manner. The EP routine [28] performs Monte-Carlo simulations, and has a computational complexity proportional to the number of random numbers generated, and communication complexity of $O(1)$. It means that EP is a computation intensive task while Linpack is a communication intensive task.

Table 3 summarizes the parameters employed for the simulation. We practiced 50 replications in one simulation and calculate average values as a result. Fifty `Ninf_Call`s are invoked in one replication.

Figure 9 shows the average elapsed time, or communication time and calculation time, to complete each `Ninf_Call`. The results show that **RR** performs worst. **RR** causes saturation on the server for the computation intensive task, EP, and saturation on the network for the communication intensive task, Linpack. Also, the performance of **LOAD** is well for EP but not well for Linpack. It means that **LOAD** causes network congestion and degrades the performance for the communication intensive task, Linpack, that is, when several tasks concentrate on a server, saturation occurs on the network and not on the server. Thus, the tasks do not get diverted to other servers, because the server load remains low, and as a result, the overall system performance degrades. This is a phenomenon that we have confirmed in our Ninf benchmarking on the testbed. Finally, **LOTH** performs best both for EP and Linpack, because it employs both server loads and network congestion in its scheduling policy.

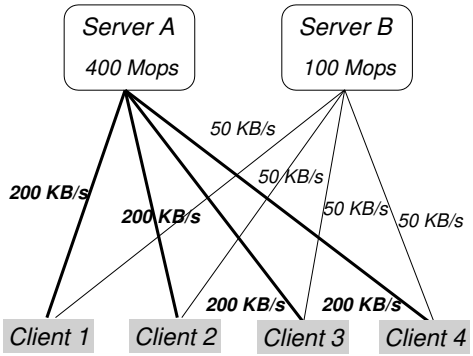

Figure 8: Imaginary Environment for Evaluation of Scheduling Schemes

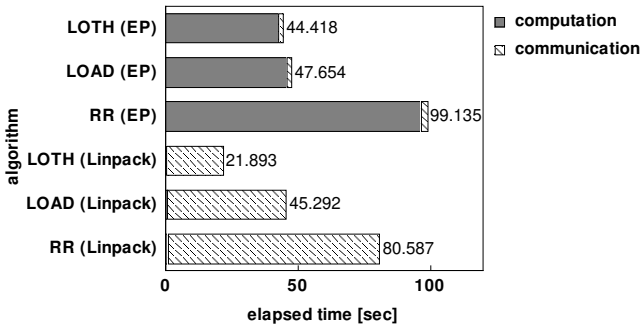

Figure 9: Simulation Results of Scheduling Schemes

5 Conclusions

An appropriate performance evaluation model is necessary not only to theoretically analyze scheduling schemes in a global computing system, but also for effective automated distribution within the WAN. However, the performance evaluation model for the scheduling has not been well established. We have proposed a performance evaluation model for scheduling in global computing systems, based on queueing networks.

The verification of the proposed model exhibited the followings:

1. The proposed model could effectively simulate average overall performance and the average communication throughput of clients' tasks for simple setups of our global computing system, Ninf, in comparison to benchmark results on our testbed.

2. Logical packet size defined in the proposed model can affect accuracy of simulation results. However, we confirmed that simulation cost on the proposed

Table 3: Simulation Parameters Provided for the Evaluation of Scheduling Schemes

Client	avg. logical packet size	100[KB] (exponential distribution)
Server	CPU performance of the server	see Figure 8
	avg. actual utilization on the server	0.1
	avg. task computation size from external processes	10[MFlops] (exponential distribution)
	task arrival from external processes	Poisson
Network	bandwidth of the network	1.5[MB/s]
	avg. actual throughput of the network	see Figure 8
	avg. logical packet size	100[KB] (exponential distribution)
	packet arrival from external processes	Poisson

model could be reduced by defining larger logical packet size without degrading the accuracy.

The experimental evaluation of scheduling schemes on the proposed model exhibited phenomena that we observed for more complex global computing benchmark on Ninf. These phenomena are:

1. Scheduling scheme employing only the information of servers degrades performance because it causes network congestion for communication intensive tasks.

2. Dynamic information of both servers and networks should be employed to achieve effective scheduling.

In the current model, a task invoked from a client is executed sequentially. However, a client will invoke a parallel task, in which subtasks will be executed on different servers concurrently, with a certain application scheduling scheme such as AppLeS [10, 11, 12] in order to achieve high-performance global computing. Furthermore, subtasks in the client's task should be co-allocated to servers to achieve efficient communication among tasks. We plan to construct mechanisms for the parallel task execution in our model. These mechanisms include invocation of parallel tasks at the client, inter-server communication and synchronization, and co-allocation of parallel tasks. Also, for the definition of simulation parameters, we assume that arrivals of tasks and data from external processes follow Poisson manner in this paper. However, this simple modeling may not be sufficient to accurately represent more complicated behavior of the current internet traffic such as fluctuation of communication throughput on the network [29]. The better modeling for the arrival, which employs more sophisticated distribution, is required to improve validity of our performance evaluation model. For scheduling schemes, accurate prediction of server loads and network congestion in a global computing system is required to achieve high-performance computing. We plan to evaluate several prediction schemes on our performance evaluation model by implementing prediction schemes provided in Network Weather Service (NWS) [25, 26] in our simulator.

References

[1] M. Sato, H. Nakada, S. Sekiguchi, S. matsuoka, U. Nagashima, and H. Takagi. Ninf: Network based Information Library for a Global World-Wide Computing Infrastructure. In *Proc. of High Performance Computing and Networking '97, Lecture Notes in Computer Science 1291*. Springer-Verlag, 1997.

[2] A. Takefusa, S. Matsuoka, H. Ogawa, H. Nakada, H. Takagi, M. Sato, S. Sekiguchi, and U. Nagashima. Multi-client LAN/WAN Performance Analysis of Ninf: a High-Performance Global Computing System. In *Proc. of the ACM/IEEE SC97 Conference*, 1997.

[3] H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. In *Proc. of Supercomputing '96*, 1996.

[4] H. Casanova, J. Dongara, C. Johnson, and M. Miller. Application specific tools. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, 1998.

[5] A. S. Grimshaw, W. A. Wulf, and the Legion team. The Legion Vision of a Worldwide Virtual Computer. *Comm. ACM*, 40(1):39–45, 1997.

[6] D. Gannon and A. Grimshaw. Object-based approaches. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, 1998.

[7] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applicatons*, 11(2):115–128, 1997.

[8] I. Foster and C. Kesselman. The Globus Toolkit. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, 1997.

[9] P. Arbenz, W. Gander, and M. Oettli. The Remote Computation System. In *High Performance Computing and Networking, Lecture Notes in Computer Science 1067*, pages 820–825. Springer-Verlag, 1996.

[10] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application-Level Scheduling on Distributed Heterogeneous Networks. In *Proc. of Supercomputing '96*, 1996.

[11] F. Berman and R. Wolski. The AppLeS Project: A Status Report. In *Proc. of the 8th NEC Research Symposium*, 1997.

[12] F. Berman. High-Performance Schedulers. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, 1997.

[13] D. Abramson and J. Giddy. Scheduling Large Parametric Modelling Experiments on a Distributed Meta-Computer. In *PCW '97*, 1997.

[14] M. Livny and R. Raman. High-throughput resource management. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, 1998.

[15] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proc. of the 7th IEEE International Symposium on High Performance Distributed Computing*, 1998.

[16] J. B. Weissman and X. Zhao. Scheduling Parallel Applications in Distributed Networks. *J. of Cluster Computing*, (to appear).

[17] D. G. Feitelson and L. Rudolph, editors. *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*. Springer-Verlag, 1995-1998.

[18] Y. Chow and W. H. Kohler. Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System. *IEEE Trans. on Computers*, C-28(5):354–361, 1979.

[19] D. L. Eager, E. D. Lazowska, and J. Zahorjan. Adaptive Load Sharing in Homogeneous Distributed Systems. *IEEE Trans. on Software Engineering*, SE-12(5):662–675, 1986.

[20] F. Bonomi and A. Kumar. Adaptive Optimal Load Balancing in a Nonhomogeneous Multiserver System with a Central Job Scheduler. *IEEE Trans. on Computers*, 39(10):1232–1250, 1990.

[21] N. G. Shivaratori, P. Krueger, and M. Singhal. Load Distributing for Locally Distributed Systems. *Computer*, 25(12):33–44, 1992.

[22] C. Lu and S. Lau. An Adaptive Load Balancing Algorithm for Heterogeneous Distributed Systems with Multiple Task Classes. In *Proc. of the 16th International Conference on Distributed Computing Systems*, pages 629–636, 1996.

[23] D. Ferrari and S. Zhou. An Empirical Investigation of Load Indices for Load Balancing Applications. In *Proc. Performance '87, the 12th International Symposium on Computer Performance Modeling, Measurement, and Evaluation*, pages 515–528, 1987.

[24] T. Kunz. The Influence of Difference Workload Descriptions on a Heuristic Load Balancing Scheme. *IEEE Trans. on Software Engineering*, 17(7):725–730, 1991.

[25] R. Wolski. Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In *Proc. 6th High-Performance Distributed Computing Conference*, 1997.

[26] R. Wolski, N. Spring, and C. Peterson. Implementing a Performance Forecasting Systems for Metacomputing: The Network Weather Service. In *Proc. of the ACM/IEEE SC97 Conference*, 1997.

[27] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[28] NAS Parallel Benchmarks. http://science.nas.nasa.gov/Software/NPB/.

[29] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. In *Proc. of 1994 ACM SIGCOMM Conference*, pages 257–268, 1998.