An Intelligent Capacity Planning Model for Cloud Market

Shifeng Shang, Yongwei Wu, Bo Wang, Jinlei Jiang, Weimin Zheng
Department of Computer Science and Technology,
Tsinghua National Laboratory for Information Science and Technology,
Tsinghua University, Beijing, China
shangsf@gmail.com,{wuyw,bowang,jjlei,zwm-dcs}@tsinghua.edu.cn

Abstract

Cloud computing is a promising way providing users computing resources. These resources are provided by means of standard computing instances. Currently there are three price schemas including spot, reservation and on-demand in cloud market, but the average difference among these three pricing schema can be as much as 2.7 times even for the same instance type. In the premise of ensuring service level object, how many and which compute instance are needed, which price schema is needed is important for company. In this paper, an intelligent capacity planning model was proposed. Experimental result shows that our model can save more money while the quality of service does not degrade.

1 Introduction

Cloud computing, which refers to services (hardware such as CPUs and storage, platform and application) provisioning and consumption over the Internet in elastic approach, is becoming a hot topic both in academia and industry around the world. A modern compute cloud allows users to share the underlying computing resources (such as CPU, memory and networking bandwidth) in an elastic manner and thus cut down the costs of IT infrastructure[1],more and more companies has been migrating a huge amount of business into compute clouds.

Typically, the price of a cloud computing resource consists of the following parts:

$P = P_c + P_s + P_{in} + P_{out} + P_{tran}$

P_c: the price of the compute instance. For example, Amazon EC2 instances are grouped into six types: Standard, Micro,High-Memory, High-CPU,Cluster and GPU Cluster, the price is counted by hours consumed.

P_s: the price of user data (the computing result through virtual instance or other data) stored on the cloud, the money is count by GB used per month.

$P_{in}P_{out}$: the price of network traffic mainly including uploading data to or downloading data from the cloud, or transferring data between different regions of the same cloud vendor.

P_{tran}: the price of file operations(such as put,get,copy, post, list etc.) within a virtual instance. For example, the P_{tran} of Amazon EC2 is \$0.10 per 1 million file operations.

Currently, resources can be remotely acquired from cloud providers through three different methods, each with its own pricing scheme and associated quality of service.

On-demand Market: This model allows users apply resource on demand at any time, and the user is charged by the hour with no long-term commitments. Pricing is per instance-hour consumed from the time an instance is launched until it is terminated. Each partial instance-hour consumed will be billed as a full hour. This model can frees user from the costs and complexities. But the price of this model is highest among three models. Table 1 shows the services prices of different vendors at on demand schema, where the basic configuration of compute is of 1 GB (=10^9 bytes) RAM, and 40 GB Disk. We

Journal of Internet Services and Information Security, volume: 1, number: 1, pp. 37-45

Price Type	Amazon	IBM	Azure	Google	GoGrid	Rack-space
Compute CPU/hours	$0.085/linux $0.12/windows	$0.139	$0.12	$0.10	$0.10	$0.06
Storage GB/month	$0.15	$0.0792	$0.15	$0.15 0.5GB free	$0.15 10GB free	$0.15
Data Upload GB	$0.10	$0.11	$0.10	$0.12	$free	$0.08
Data Download GB	$0.17 $0.13 if>10TB	$0.11	$0.15	$0.10	$0.29	$0.22

Figure 1: PRICES OF SAME TYPE INSTANCES FROM DIFFERENT VENDORS

can see that the prices of compute range from $0.06 to 0.12 with a maximal difference as much as $0.06 per hour.

Reservation Market: This market enables a cloud resource user to purchase a bundle of resources for a long term (e.g., a whole year, three years etc.), and an one time reservation fee is required in advance so that they can get great discount. When instances previously reserved are used, the user is charged for the use of the cloud resources at a lower price than the instances acquired at the on-demand market.

Spot market. This market is a one side auction market for users to consume resources at a lower and more flexible cost. The price is set by the cloud provider depending on the supply and demand situation of cloud resource market. To use spot Instances, the user should specify the type, region desired, number

Price Method / Instance Type		On-Demand	Reservation	One Time Fee /Year	SPOT
Standard	Small	$0.085/h	$0.03/h	$227.50	$0.029/h
	Large	$0.34/h	$0.12/h	$910	$0.123/h
	Extra Large	$0.68/h	$0.24/h	$1,820	$0.228/h
Micro	Micro	$0.02/h	$0.007/h	$82	0.007/h
High-Memory	Extra Large	$0.50/h	$0.17/h	$1,325	$0.165/h
	Double Extra Large	$1.00/h	$0.34/h	$2,650	$0.434/h
	QuadrupleExtra Large	$2.00/h	$0.68/h	$5,300	$0.875/h
High-CPU	Medium	$0.17/h	$0.06/h	$455	$0.062/h
	Extra Large	$0.68/h	$0.24/h	$1,820	$0.246/h
Cluster	Quadruple Extra Large	$1.60/h	$0.56/h	$4290	$0.55/h
GPU Cluster	Quadruple Extra Large	$2.1/h	$0.74/h	$5630	$0.745/h

Figure 2: THREE PRICING SCHEME OF AMAZON EC2 INSTANCES

of instances required, and the maximum price are willing to pay per instance hour. If the maximum price bid of user exceeds the current spot price, your request will be fulfilled and your instances will run until

either you choose to terminate them or the spot price increases above your maximum price. The usage fee charged is the one set by the cloud resource provider at the time the resource is used. In most case, the price of spot market is lower than on demand, but is a little higher than reservation market.

Table 2 is the price comparison of Amazon's instances using different pricing methods; From the table we can see that the price of instance above is the half of price of below in the table and the average difference can be as high as 2.7 times even for the same instance. [3][21] has proposed a global cloud resources trading market mechanism and users can buy resource from the different vendors.Meanwhile, the resource requirement of many applications such as online video, e-commerce is high peak-to-mean-ratios, recurring and shifting seasonal patterns as well as bursts leading to heavy tail demand distributions[2][4]. How to take full advantage of the elastic of cloud, according to the workload characteristics of client, in the premise of ensuring the service level object , the planning of how many and which compute instance are needed, which price schema is more suitable are vital for the users[5], so users would use less resource wasting and save more money.

The rest of the paper is organized as follows. Section 2 gives an overview of related work. In Section 3 explains the intelligent capacity planning model and Section 4 is the experimental result. The section 5 is the conclusion and future work.

2 Related Work

Capacity Planning includes the process of planning for compute resources required to fulfill current and future requirement. Current research pay more attention on the provider's capacity planning , or short term resource planning in a hybrid environment[6][7] [8][9] [10][11] [13]. However, there is, little research being done on the customer's need for long term capacity planning in a fully cloud computing environment.

There is amount of research on on-demand resource allocation in Grids [12][14] [15][16]. To find out the amount of resources for an application during its execution is there research goal. More important, they focus on the scheduling and not on the capacity planning of the compute resource, these research take less providers costs into account.[18][17] [19][20] [22].

Unlike them, in this paper we proposed a long term capacity planning model for client in an entirely clouds environment. The main concern in the capacity planning of this kind of cloud-based infrastructure is with the different pricing models used by cloud resource providers. The prices of resources can vary significantly from reservation, spot to on-demand usage.In this paper, we propose an intelligent capacity planning model, it take the workload characteristics, different pricing schema into account, can use less resource and save more money.

3 Intelligent Capacity planning Model

Resource planning in cloud is the activity of estimating the computer instance, pricing model, operating system, and network bandwidth necessary according user's application model in order to meet the service level object at lower costs.

In this paper, we mainly concern on the capacity planning activity for client user. First of all, We define following parameters in our model:

λ: the average arrival rate of requests entering the system.

μ: the average rate of serving entering requests.

T_w: the mean wait time of each request in the queue.

ρ: the general utility rate of instances, offered work load rate to a instance, $\rho = \frac{\lambda}{n\mu}$.

n: the number of reservation instances required.

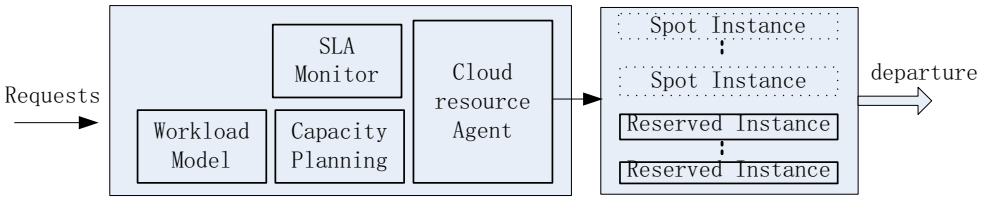

Figure 3: framework of capacity planning model

m: the number of spot instances required.

T: the mean time of a request stay in the system.

r=λ/μ: the average arrivals load in mean service time.

We assume that both arrival intervals of request and service times are Poisson distribution. Furthermore, We also assume that all instances can work together seamless to process all incoming requests that they are waiting in a single queue.

The framework of intelligent capacity planning model is shown in figure 3. The work process of this model can be described as follows:

- From the history information for a old system that the system type and size is similar, characterize the workload. Web workload is unique in its characteristics and some studies [2,5] identified workload properties and invariants, such as the heavy-tailed distributions of file sizes in the Web. It has been also observed that Web traffic is busty in several time scales [20,8].

- Set the threshold. The threshold can be workload of reservation instance, or the mean response time or the arrive rate. We use the queue theory, for example, the mean response time can be set as the threshold T_b. T: the mean waiting time in the system.

$$T = 1/\mu + \left(\frac{r^n}{n!(n\mu(1-\rho)^2}\right) p_0 \tag{1}$$

p_0: the steady-state probabilities of all instances are idles.

$$p_0 = \left(\frac{r^n}{n!(1-\rho)} + \sum_{c=0}^{n-1} \frac{r^c}{c!}\right) \tag{2}$$

l_q: the expected number in the whole queue.

$$L_q = \left(\frac{r^n \rho}{n!(1-\rho)^2}\right) \tag{3}$$

T_w: the mean waiting time in queue, according to Little's law.

$$T_w = \left(\frac{r^n}{n!(n\mu(1-\rho)^2}\right) p_0 \tag{4}$$

- By solving the formula (1), get n, the number of reservation instances to meet the threshold requirement. Because the workload is relative stable under the condition that $T \leq T_b$. We buy n compute instances in a long-term commitment through reservation market.

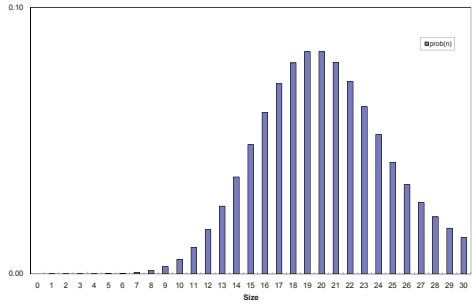

Figure 4: common number of customers

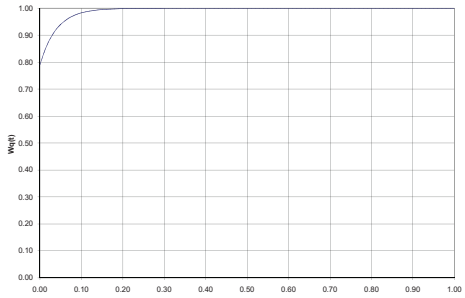

Figure 5: common queue-Wait distribution

- The SLA monitor get the performance parameter. If $T > T_b$ and the response time is higher than the threshold, the capacity planning service calculate the resource should added is m, fast applying the new cloud resource from spot market, the reservation instance and spot instance can work together seamlessly to deal with the peak workload to meet with the service level object.

- $T < T_b$, When the value of response time down to below the threshold, the spot instance temporally added will be released.

4 Experiment

Capacity planning service through SLA monitor to evaluate the service performance. When vast users visit the same service in short time, bad performance can be resulted. This can result in two sides. First, all of the compute instances work too busy to reach fast response to all the requests. Second, due to the extremely slow response, the users maybe become out of the patience. Many factors can change response time to user such as network state, instance workload and the number of user request. In this paper, according to queuing theory, we focus on the reservation instance workload. In common case, reservation instance should keep 20% idle time in order to do some important work. When the percentage of reservation instance workload exceed 80%, we should buy more spot instances to reduce the system workload.

During below experiment, we give arrival rate 100, mean time to complete service 0.2.

In following figure, we use capacity planning model to simulate the service provided by 25 reservation instance. The client arrival complies with Poisson distribution.

The related parameter is that Arrival rate(λ) is 100, service rate(μ) is 5, the service time distribution with mean $1/\mu$, and number of reservation instances in the system is 25. In this case, instance utilization can keep 80.00% or so. For network service, when one client's request comes in, if one of reservation instances keeps idle, this request can be handled at once without any wait. Otherwise, the request must be queued in the queue of requests. When queue becomes long, the reservation instance must allocate much memory. Due to the extremely stability of client visiting, 80% utilization is a safe number. In our system, we control system utilization under 80% all the time.

We also process another M/M/C experiment to reflect the users' visiting action. First, the number of visiting users close to the reservation instance limit. Second, the number of visiting users exceed the service limit. For the first case, all the reservation instances show high workload, but they still can accept the incoming requests. For the second case, $\lambda > n \times \mu$. In order to satisfy the users' visiting, more spot instances must be provide at once, at this time, capacity planning service buy new instance from spot market through cloud resource agent.

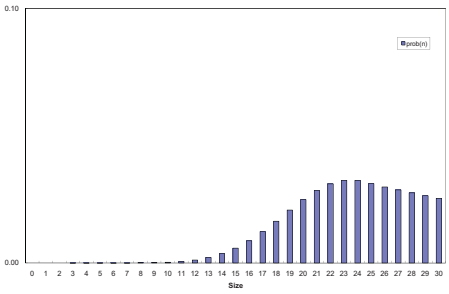

Figure 6: many number of Customers

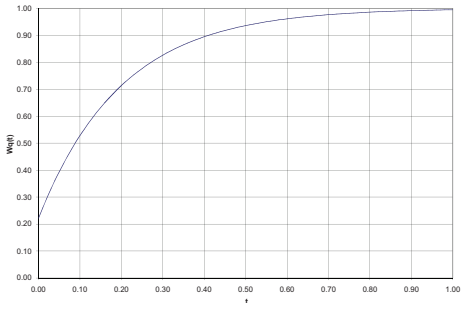

Figure 7: Queue-wait distribution in the case of many users

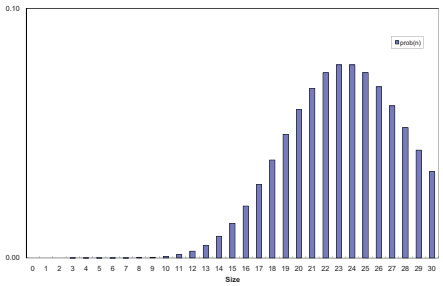

Figure 8: more customers and more instances

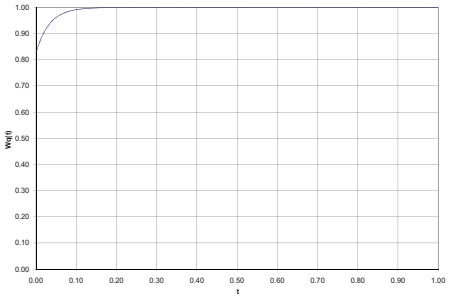

Figure 9: queue wait distribution in the case of more users and more instances

Figure 4 and figure 5 reflect the common case, each user can get the service in very short time. In this case, each reservation instance can show fast response and keep certain idle time to execute other important tasks.

Figure 6 and figure 7 reflect the case that many users send request to reservation instances. When visiting number becomes big and the number of total instances keep constant, the workload become improved and the mean waiting time also becomes long. Another change is that queue wait distribution become longer that the common case.

By adding more spot instances, the mean workload reduces and the mean response time becomes short, correspondingly. In figure 8 and figure 9, we add more 5 spot instances to the service system. By comparison to the common case, this scheme also keeps the idle time 50%, and also keep fast response.

In figure 10 and figure 11, we simulate the special visiting action. In this case, we let the number

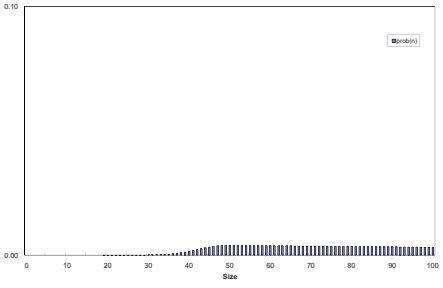

Figure 10: more customers and more instances

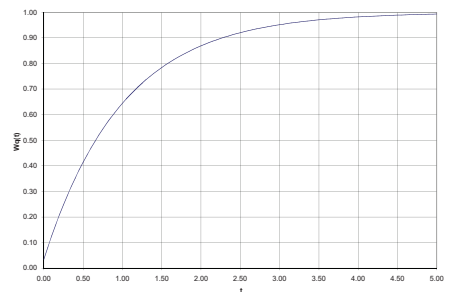

Figure 11: queue wait distribution in the case of more users and more instances

of visiting user improve from 100 to 249 in very short time. At the beginning, the reservation instances suddenly can not handle this sudden event due to the incoming numerous requests. By monitor the workload of each instance, the workload shows the sudden adding. By catching this special case, through computing according to related parameters, we add the number of spot instance up to 50. By this, the whole system can keep common state in short time.

5 Conclusion and Future Work

Nowadays, Cloud computing is becoming a computing buzzword both in industry and academia. A modern compute cloud allows users to share the underlying computing resources (such as CPU, memory and networking bandwidth) in an elastic manner, the user is charged according to the number and hours of compute instance used. There are three types of price schema in cloud market, the difference of different price is big, so capacity planning is crucial for common user. In this paper, a intelligent capacity model based on queue theory, user can conduct capacity planning combined reservation market and spot market. Experiment result shows that this model is flexible and more profitable.

5.1 Acknowledgments

This Work is supported by Natural Science Foundation of China (60803121, 60773145, 60911130371, 90812001, 60963005), National High-Tech R&D (863) Program of China (2009AA01A130, 2006AA01A101, 2006AA01A108, 2006AA01A111, 2006AA01A117).

References

[1] L. Amar, J. Stosser, E. Levy, A. Shiloh, A. Barak, and D. Neumann. Harnessing migrations in a market-based grid os. In *Proc. of the 2008 9th IEEE/ACM International Conference on Grid Computing (GRID'08), Tsukuba, Japan*, pages 85–94. IEEE, September 2008.

[2] James Broberg and Rajkumar Buyya. A multi-commodity flow approach to maximising utility in linked market-based grids. In *Proc. of the 5th international workshop on Middleware for grid computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference (MGC'07), Newport Beach, California, USA*, pages 5:1–5:6. ACM Press, November 2007.

[3] Rajkumar Buyya. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In *Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'09), Shanghai, China*, pages 1–1. IEEE, May 2009.

[4] Artur Czumaj and Amir Ronen. On the expected payment of mechanisms for task allocation. In *Proc. of the 23rd annual ACM symposium on Principles of distributed computing (PODC'04), St. John's, Newfoundland, Canada*, pages 98–106. ACM Press, July 2004.

[5] Moises Goldszmidt, Derek Palma, and Bikash Sabata. On the quantification of e-business capacity. In *Proc. of the 3rd ACM conference on Electronic Commerce (EC'01), Tampa, Florida, USA*, pages 235–244. ACM Press, October 2001.

[6] Jens Grossklags. Experimental economics and experimental computer science: a survey. In *Proc. of the 2007 workshop on Experimental computer science (ExpCS'07), San Diego, California*. ACM Press, June 2007.

[7] Sina Honari, Maziar Gomrokchi, Mojtaba Ebadi, Amin Fos-hati, and Jamal Bentahar. Simulating new markets by introducing new accepting policies for the conventional continuous double auction. In *Proc. of the 2008 Spring simulation multiconference (SpringSim'08), Ottawa, Canada*, pages 89–97. Society for Computer Simulation International, April 2008.

[8] Markus C. Huebscher and Julie A. McCann. A survey of autonomic computing degrees, models, and applications. *ACM Computing Surveys*, 40:7:1–7:28, August 2008.

[9] Woochul Kang, H. Howie Huang, and Andrew Grimshaw. A highly available job execution service in computational service market. In *Proc. of the 8th IEEE/ACM International Conference on Grid Computing (GRID'07), Austin, Texas, USA*, pages 275–282. IEEE, September 2007.

[10] Li Li and Stephen F. Smith. An agent-based framework for dynamic multi-period continuous double auctions in b2b exchanges. In *Proc. of the 4th ACM conference on Electronic commerce (EC'03), San Diego, California, USA*, pages 204–205. ACM Pres, June 2003.

[11] Guanfeng Liu, Yuebin Xu, and Shanzhou Ma. A new resource selection approach based on reputation driven min-min algorithm in the grid economy. In *Proc. of the 2007 Asian technology information program's (ATIP's) 3rd workshop on High performance computing in China: solution approaches to impediments for high performance computing (CHINA HPC '07), Reno, Nevada, USA*, pages 160–167. ACM Press, November 2007.

[12] Jinzhong Niu, Kai Cai, Simon Parsons, Enrico Gerding, and Peter McBurney. Characterizing effective auction mechanisms: insights from the 2007 tac market design competition. In *Proc. of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS'08), Estoril, Portugal*, volume 2, pages 1079–1086. International Foundation for Autonomous Agents and Multiagent Systems, May 2008.

[13] Jinzhong Niu, Kai Cai, Simon Parsons, and Elizabeth Sklar. Reducing price fluctuation in continuous double auctions through pricing policy and shout improvement. In *Proc. of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS'06), Hakodate, Japan*, pages 1143–1150. ACM Press, May 2006.

[14] Simon Parsons, Juan A. Rodriguez-Aguilar, and Mark Klein. Auctions and bidding: A guide for computer scientists. *ACM Computing Surveys*, 43(2):10:1–10:59, February 2011.

[15] S. Phelps, M. Marcinkiewicz, and S. Parsons. A novel method for automatic strategy acquisition in n-player non-zero-sum games. In *Proc. of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS'06), Hakodate, Japan*, pages 705–712, New York, NY, USA, May 2006. ACM.

[16] Rosario M. Piro, Andrea Guarise, and Albert Werbrouck. An economy-based accounting infrastructure for the datagrid. In *Proc. of the 4th International Workshop on Grid Computing (GRID'03), Phoenix, Arizona, USA*, pages 202–204. IEEE, November 2003.

[17] Chris Preist, Claudio Bartolini, and Andrew Byde. Agent-based service composition through simultaneous negotiation in forward and reverse auctions. In *Proc. of the 4th ACM conference on Electronic commerce (EC'03), San Diego, California, USA*, pages 55–63. ACM Press, June 2003.

[18] Chris Preist, Andrew Byde, and Claudio Bartolini. Economic dynamics of agents in multiple auctions. In *Proc. of the 5th international conference on Autonomous agents (AGENTS'01), Montreal, Quebec, Canada*, pages 545–551. ACM Press, May-June 2001.

[19] Alex Rogers, Esther David, Terry R. Payne, and Nicholas R. Jennings. An advanced bidding agent for advertisement selection on public displays. In *Proc. of the 6th international joint conference on Autonomous agents and multiagent systems (AAMAS'07), Honolulu, Hawaii, USA*, pages 51:1–51:8. ACM Press, May 2007.

[20] L. Julian Schvartzman and Michael P. Wellman. Stronger cda strategies through empirical game-theoretic analysis and reinforcement learning. In *Proc. of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09), Budapest, Hungary*, volume 1, pages 249–256. International Foundation for Autonomous Agents and Multiagent Systems, May 2009.

[21] Shifeng Shang, Jinlei Jiang, Yongwei Wu, Guangwen Yang, and Weimin Zheng. A knowledge-based continuous double auction model for cloud market. In *Proc. of the 6th International Conference on Semantics Knowledge and Grid (SKG'10), Beijing, China*, pages 129–134. IEEE, November 2010.

[22] Zhu Tan and John R. Gurd. Market-based grid resource allocation using a stable continuous double auction. In *Proc. of the 8th IEEE/ACM International Conference on Grid Computing (GRID'07), Austin, Texas, USA*, pages 283–290. IEEE, September 2007.

Shifeng Shang is a full-time Ph.D. candidate with Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests are grid computing, CSCW and intelligent systems.

Yongwei Wu received the Ph.D. degree in applied mathematics from the Chinese Academy of Sciences in 2002. He is currently an associate professor of computer science and technology at Tsinghua University, Beijing, China. His research interests include grid and cloud computing, distributed processing, and parallel computing. He is a member of the IEEE.

Bo Wang is a full-time Ph.D. candidate with Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research focuses on parallel computing in multicore system.

Jinlei Jiang is an assistant professor with Department of Computer Science and Technology, Tsinghua University, Beijing, China. From May 2007 to April 2008, he visited Institut fuer Informatik, Technische Universitaet Muenchen, Germany with the Group of Applied Informatics – Cooperative Systems under the sponsorship of Alexander von Humboldt Foundation. Jiang received a PhD degree in computer science and technology from Tsinghua University in January 2004 with an honor of excellent dissertation. His research interests mainly focus on CSCW, workflow management, grid computing and Web services. His research work has appeared in Expert Systems with Applications, Journal of Computer Science and Technology, Lecture Notes in Computer Science, IEEE International Conference on Web Services, ACM Symposium on Applied Computing and so on.

Weimin Zheng received the BS and MS degrees, respectively, in 1970 and 1982 from Tsinghua University, China, where he is currently a professor of computer science and technology. He is the research director of the Institute of High Performance Computing at Tsinghua University, and the managing director of the Chinese Computer Society. His research interests include computer architecture, operating system, storage networks, and distributed computing. He is a member of the IEEE.