LMCGRID: A LOW MANAGEMENT COST GRID COMPUTATION MODEL

Y. Yang,* X. Yang,** and C. Zhou*

Abstract

This paper proposes a Low Management Cost computing Grid model (LMCGrid), aiming at harvesting the idle time of computers connected to Internet to run large-scale distributed applications. In LMCGrid, no specific node is designated to manage dynamically changing resources. Despite the absence of the management node, the corresponding information mechanism, task scheduling algorithm and limited task duplicating algorithm naturally evolve the system into equilibrium to perform large-scale distributed computation with low cost. A simulation software package has been developed to verify this model and to assess its performance. The results showed that LMCGrid reasonably distributed loads in a dynamic environment, and fully utilized the computational capacity of idling resources, and that it was feasible to solve large-scale, embarrassingly parallel applications such as parameter sweep and Monte Carlo simulations efficiently.

Key Words

Low management cost, task scheduling, information mechanism, limited task duplicating algorithm, LMCGrid

1. Introduction

In the research of computation grid, one of the key aspects is to use idle computing resources on Internet to process large-scale distributed applications effectively. The challenges confronted by the grid are that it must aggregate adequate resources and utilize them effectively in dynamic Internet. In SuperWeb [1], Javelin [2] and Charlotte [3] grid system, a central proxy is introduced to accomplish all management tasks including responding to users' requests, managing idle resources, scheduling tasks, etc. Central proxy may be the bottleneck and grid may risk single point failure. To eliminate central bottleneck, XtremWeb [4], Bayanihan [5], Javelin++ [6] and Javelin2.0 [7] utilize proxy network instead of single proxy to complete management, in which one proxy manages limited designated resources and all proxies are managed by a central server. However, the costs devoted to management and maintaining coherence as well as synchronization of proxies are high. Studies of above systems indicate that management, no matter central or distributed methods against dynamic and alterable resources in Internet would lead to enormous performance and communication cost and management essentially restricts the utilization of large-scale common resources in Internet.

For the sake of aggregating enormous computing resources without any additional hardware cost, this paper presents Low Management Cost computing Grid (LMCGrid) model and lays out its critical information mechanism and scheduling mechanism. LMCGrid is designed without any central proxy because nodes in LMCGrid are controlled by themselves. The expansibility and usability of LMCGrid are reasonably well when it is confronted with large-scale dynamic computing resources. The rest of this paper is organized as follows: Section 2 describes the architecture of LMCGrid. Section 3 introduces the information mechanisms. The task scheduling is presented and analysed in Section 4. The limited task duplicating algorithm is described in Section 5. Simulations and analysis are introduced in Section 6 and finally we come to a conclusion in Section 7.

2. LMCGrid Architecture

LMCGrid is composed of computing resources connected with each other logically and two computers identified as neighbour node to one another when they are logically connected. All nodes with uniform statuses play the same role in LMCGrid and no management nodes exist. One node in LMCGrid manages its computing resource and does not need to monitor others' statuses. New participant node shares its resource and gains service from grid by establishing neighbouring relations with some nodes already in the system. User can submit tasks at any node and get computing results at the submitting node. Fig. 1 shows the architecture of LMCGrid node.

Distributed applications such as parameter sweep problem and Monte Carlo method are divided into pieces of parallel tasks and submitted on node which is named task source node. Tasks from neighbouring nodes and local nodes are organized as a task queue by the task manager and manager is also in charge of collecting results of local running tasks and returning them to corresponding source

* College of Information Technical Science, Nankai University, Tianjin 300071, China; e-mail: yangyl@nankai.edu.cn, hfx@mail.nankai.edu.cn

** China TravelSky Technology Limited, Beijing 100027, China; e-mail: yang_xue_gang@163.com

(paper no. 202-2496)

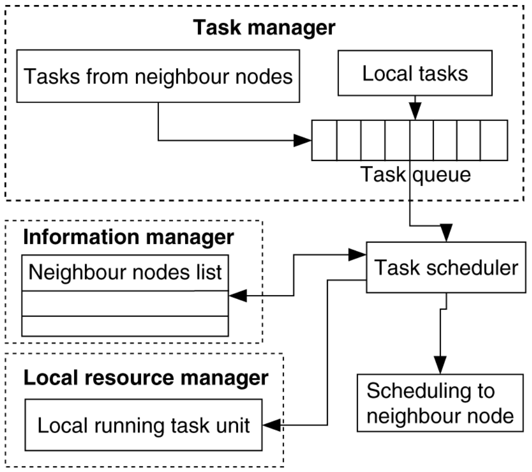


Figure 1. Architecture of node.

nodes. Local information manager maintains a neighbour nodes list, in which information of neighbour nodes is stored, and according to the list, task scheduler delivers tasks to local running unit or to neighbouring nodes, also the list is updated during task scheduling. Local resource manager gets usage status of local resource and decides to suspend, execute or terminate task running on local node.

LMCGrid is confronted two critical challenges due to the absence of global management entry and integrated information of grid. Firstly, it ought to shape proper computing resource areas to serve different source nodes. Secondly, grid should provide persistent and reliable computation power despite network dynamics. Task distributing algorithm and limited duplicating algorithm based on information mechanism are presented to achieve these two goals.

3. Information Mechanisms

A neighbouring nodes list is maintained by every node and it stores the information of local resource and neighbouring nodes which is used to schedule tasks by local nodes.

3.1 Relative Ability and Control Information

Computing resources connected to Internet have many dynamic varying differences, such as different computing cycles, CPU loads and network bandwidth, real-time monitoring of resources statuses would perform at a high cost of communication and computation cycles. However, computing capability is the universal attribute involved in the computer, and we abstract it as two statues: "able" and "disabled" despite its heterogeneity and variability. While "able" status is marked with 1, the "disabled", 0. The following are the definitions of ability:

Definition 1 (Local ability). Let $L(V,V)$ indicate node V's local ability. If node V is running a task for LMCGrid, V is "disabled", or else V is "able".

Definition 2 (Relative ability). Let $R(U,V)$ indicate the relative ability which node V relative to node U, V is a neighbouring node of U, and $R(U,V) = \mathop{||}_{i=1}^{N(V)-1} R(V,V_i)|| L(V,V)$, $N(V)$ is the number of V's neighbouring nodes, V_i is the V's neighbour node, and $V_i \neq U$. $R(U,V)$ is a recursive definition. It represents V's local ability and its all neighbours' ability.

LMCGrid users can submit applications from any node and it would lead to a result that multi-nodes compete against each other to dominate a single node for lack of global management of grid. To resolve the competition we presents control information $C(U,V)$ to mark status of node V relativity to U in terms of competition. $C(U,V)$ can be depicted with following three statuses:

1. *Boundary*: It means that V belongs to another source node's computing area and U cannot distribute task to V, V is the boundary node of U.
2. *Dominate*: It means U firstly serves V and would reject tasks from other nodes.
3. *Normal*: Status other than Boundary and Dominate is marked as Normal.

3.2 Neighbour Nodes List and Information Modification Mechanism

Neighbour nodes list stores the local ability and relative ability of neighbour nodes. Assuming node U's neighbour is V and A, Table 1 shows U's neighbour node list, in which the first item is U's local ability and control information. When new node joins in grid, relative ability is set "able" and control information is normal in its neighbour nodes list.

Table 1
A Sample of Neighbour Nodes List

Node ID	Relative Ability	Control Information
U	$L(U,U)$	$C(U,U)$
V	$R(U,V)$	$C(U,V)$
A	$R(U,A)$	$C(U,A)$

During running of LMCGrid, the node does not detect neighbour's status on its own initiative. Only in case of distributing tasks, completion of local tasks running and entering of new node, the node will inform its dominative node to update information about it. Also, information in list cannot absolutely represent the real status of neighbour nodes. However, LMCGrid task scheduling mechanism guarantees that tasks can be correctly distributed. The modification mechanism of neighbour nodes list eliminates much cost in contrast with method of real-time monitoring resources.

4. Task Scheduling

Based on information manager, task distributing algorithm is presented to distribute tasks to idle nodes smoothly without global information. When node detects tasks in queue as well as "able" node in neighbour nodes list, it would start to schedule task. Algorithm 1 layered out LMCGrid task scheduling mechanism by example of node U:

Algorithm 1: Task scheduling algorithm

1. Node U checks value of $L(U,U)$ and if $L(U,U)$ is 1, U runs task itself and sets $L(U,U)=0$, then goes to step 5. Otherwise, it goes to step 2.
2. Get one node V which $R(U,V)$ is 1 from neighbour nodes list, and send task to V, then set $R(U,V)=0$.
3. When V receives task from U, it then decides whether to accept task according to resource competition algorithm:
   - 3.1 If V can receive task, then V places task in task queue and set $R(V,U)=0$. Also V compares the length of task queue with the number of "able" neighbour nodes in neighbour nodes list. If the length is less than the number, V returns a message that V is still "able" to U, or else it returns nothing. Then it goes to step 5.
   - 3.2 If V refuses to serve U, go to step 2.
4. Node U receives message that node V still can serve U, it sets $R(U,V)=1$. If this makes the relative ability that node U relative to its dominative node changing from 0 to 1, U would return "able" message to its dominative node too.
5. U takes next task from queue and goes to step 1.

Algorithm 2: Resource competition algorithm

1. V looks up the neighbour node list to find who dominates V.
2. If dominative node does not exist, V accepts task and set $C(V,U)=$ Dominate. Then algorithm ends.
3. If $C(V,K)$ equals Dominate and "V's" dominative node is K, compares K with U:
   - 3.1 If K equals U, V accepts the task and ends algorithm.
   - 3.2 If K does not equal U, then informs U that V cannot receive task from it, set $C(V,U)=$ Boundary and $R(V,U)=0$ to avoid distributing tasks to U.
4. If U receives information that V refuses to serve, it sets $C(U,V)=$ Boundary.

Node schedules tasks to its "able" neighbour nodes in turn and tasks submitted by users are spread in network step by step, distributing process can be considered as an extent preferential search process from source node. Resource competition algorithm resolves the competition during the spreading of different source nodes. The spreading and competition processing would form many resource areas for respective source node. However, when boundary node is idle and its dominative node has not sent any tasks to it for a certain time, boundary node would inform its boundary neighbour nodes so that they can compete for node again and resource areas would vary at that time. Resources can be fully utilized during dynamic competition.

5. Limited Task Duplicating Algorithm

Internet comprises of large-scale nodes which may enter, exist and become invalid at any time. Heartbeat inspecting, period checkpoint [8], etc. are presented to avoid uncertainty of changing Internet, but these approaches face great challenges of massive consumption of network bandwidth and CPU cycles. We take limited task duplicating algorithm to deal with variability of network. The following is the description of the algorithm.

Source node organizes tasks submitted by users as a list. Task scheduler distributes a task every time and does not wait for result returning, it goes on to schedule next task in list. When all tasks in list have been scheduled in order, it finishes a scheduling loop and scheduler would process the next loop from the head of list. During scheduling, a returning of task result will trigger removing of corresponding item from list. When scheduling loop executes max scheduling times (MST) times, unfinished tasks in list will be only assigned to local source nodes to run at the MST + 1 times. Algorithm does not need monitoring status of node and task, and so long as the source node works well, the task would be completed successfully even all other nodes running duplications of task go to fail. It is a heuristic algorithms like RR [9] and WQR [10]. The studies of these algorithm show that when the MST equals 3 or more, task achievement efficiency using these heuristic algorithms is not bad than that of DFPLTF [10] and Suffrage-C [11] which need know dynamic information of node ability and task grain [9, 10]. Limited task duplicating algorithm gains reliability and performance at the cost of limited extra computation and our simulation below shows that extra computation is acceptable when duplicating times equal 4 or 5.

6. Simulations and Analysis

A simulation software package has been developed to verify the correctness of LMCGrid and assess its performance. In simulation, task grain and time of transferring task are all weighed as time steps. Task grain is the executing time steps on a node with 1 computing capabilities. A task's running time on a node with 1 computing capabilities is twice longer than that on node with 2 capabilities. Time transferring a task from one node to another is supposed as 1 time step. Performance of LMCGrid is measured by comparing the amount of computation finished by node with its computing capacities when all applications are finished. Based on this we study whether the node resources are fully utilized by LMCGrid, and also the extra computation consumed by limited task duplicating algorithm is simulated and analysed.

LMCGrid is an Internet-based model. To analyse its performance on Internet, a simulation Internet topology based on autonomous systems level is created according to BA model [12]. Fig. 2 demonstrates that it is comprised of 1,000 nodes and its distribution of degree and average number of degree are consistent with Internet autonomous systems-level topology characters [13]. We focus on LMCGrid performance on this complex topology. However, its

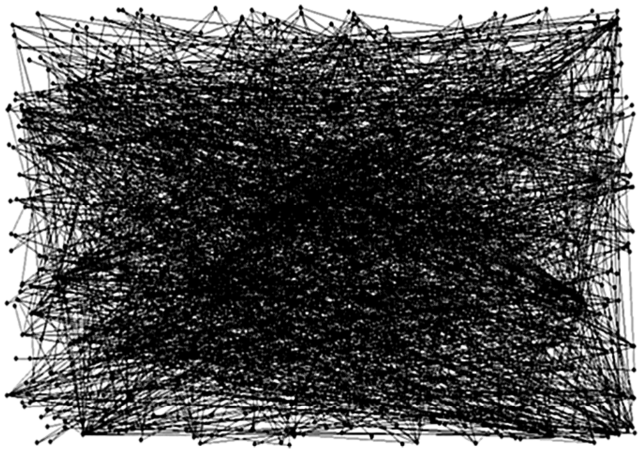


Figure 2. The simulation topology of autonomous systems-level Internet with 1,000 nodes and $<k>=3.59$.

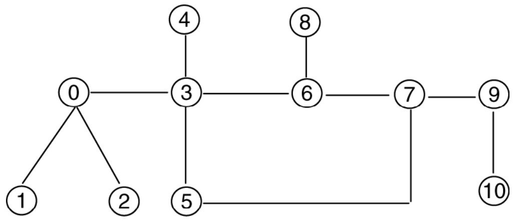


Figure 3. A simple system topology.

simulation results are too complex to lay out it clearly. To display the special characters of LMCGrid, the results on a simple topology in Fig. 3 are presented as well. In Figs. 2 and 3, the line represents the neighbour relation between two nodes.

Topology in Fig. 3 is comprised of circle, line and star-graph, the result on it would represent some universal validities and characters of LMCGrid. Simulation results on simple topology of Fig. 3 are presented in Figs. 4, 5 and 6, in which X axis indicates the node's ID. Y arrow is computation load amount of node, and the figures are average results of 10 times simulation. One thousand tasks are submitted to each source node every time, and the task grain is randomly generated following uniform distribution $U(50,80)$.

Fig. 4 shows the results that all nodes have the same computing capacities, in which nodes 0 and 9 are source nodes. The computation load finished by each node is almost the same and it shows that all nodes' computing capabilities are utilized fully and achieve load balance. Even more, computation load of source nodes 0 and 9 have highest load, while others' loads gradually reduce along with their increasing distance towards source nodes in topology. Obviously, node 8 has the least computation load for its longest distance towards 0 and 9. The above results conform to the principle that tasks should be scheduled to nodes which are close to source nodes.

Fig. 5 shows the results that nodes belong to different computing capacities, in which tasks are submitted on nodes 1 and 10, and computing capacities of node 1 is 3, capabilities of node 4, 5 and 8 are 4, nodes else belong to 1.

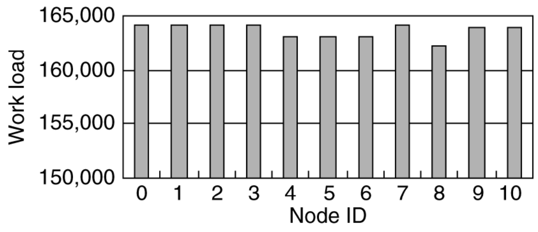


Figure 4. All nodes have same ability.

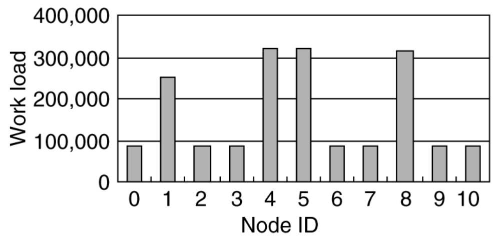


Figure 5. Nodes have different ability.

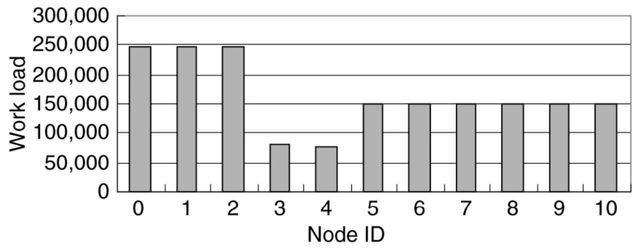


Figure 6. Node invalidation.

Obviously, computation loads of nodes are accordant with their capacities in figure. Computation finished by node with 4 computing capacities is almost four times than that by node with 1 capacity, and that finished by 3-capacity node is three times than that by 1-capacity node.

In Fig. 6, nodes 1 and 7 are source nodes and all nodes have 1 computing capacities. During running of system, node 3 was set to failure suddenly, and due to its invalidation, the network is partitioned into three sub-networks which correctly performed all the same. From then on, sub-network composed of nodes 0, 1 and 2 just worked for source node 2 and ultimate computation of nodes 0, 1 and 2 lay at almost the same level. Sub-network composed of 5, 6, 7, 8, 9 and 10 serves for source 7, in which loads of nodes are almost the same. Node 4 is cut off from network owing to node 3's failure, and as a result, it cannot get any tasks and its load keep fixed from then on. Finally, ultimate loads of nodes 4 and 3 are almost the same in amount.

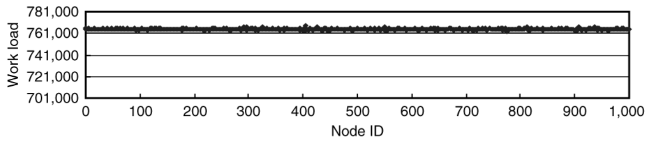


Figure 7. The simulation result with same node ability.

Figs. 7, 8 and 9 are the simulation results utilizing

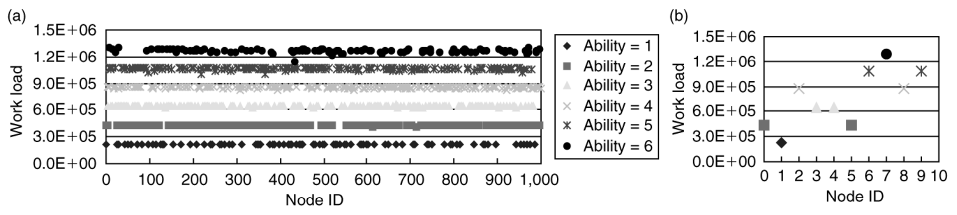


Figure 8. The simulation result with different node ability, (b) shows the local of (a).

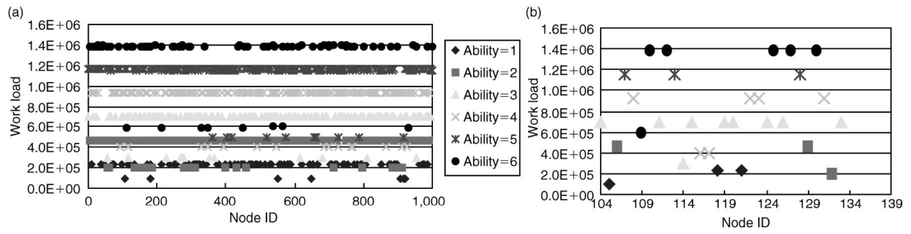


Figure 9. The simulation result of invalidating 50 nodes randomly, (b) shows the local of (a).

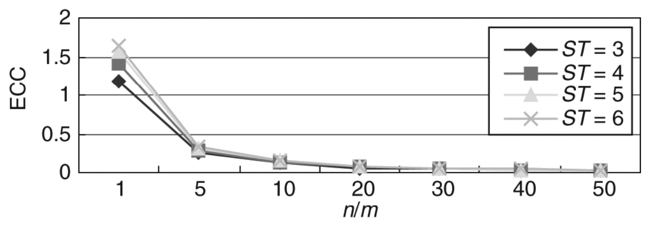


Figure 10. The analysis of extra consuming computation.

the autonomous systems-level Internet topology in Fig. 2 and the results are the average of 10 simulations. Three source nodes were randomly selected and every source node was submitted 50,000 tasks every time. Task grain was generated following uniform distribution $U(4900{,}5100)$.

Fig. 7 shows the results in which all nodes' capacities are 1. It is displayed that computation executed by all nodes lay at almost the same level.

Fig. 8 is the result that nodes belong to different computation capacities, in which capacities of node were randomly generated following uniform distribution $U(1{,}6)$. In the figure, computation loads finished by nodes with same capacities are nearly the same, and nodes whose capacities are larger than 1 have more multiples computation loads than node with 1 capacity. The multiples correspond with comparison between computing capacities, which illustrates that nodes with different capacities are utilized fully in LMCGrid.

Fig. 9 is result comparison when 50 nodes are randomly set to failure during simulation. Nodes capacities were randomly created following $U(1{,}6)$. Despite failure of 5% of nodes in topology, system worked correctly and got all results of tasks in the end. Validate nodes with the same capacities completed almost the same computation, and nodes with distinguishing loads comparing with their capacities were just failed nodes which load remains the same as what they had gained before failure.

We analyse extra cost of task scheduling algorithm according to the rate of extra computation against total computation. The extra consuming computation rate (ECCR) is defined as: $\text{ECCR} = \frac{\text{TRC}-\text{TC}}{\text{TC}}$, and total computation (TC) is the computation sum of all tasks submitted by users, total real computation (TRC) is the computation sum of tasks really executed by grid system. We study the ECCR by simulation on 1,000 nodes Internet autonomous systems-level topology. Figure 10 demonstrates the ECCR trend following the change of n/m when MST is set different values. n is the tasks number and m is nodes number (m equals 1,000). The curves are the average of 10 simulations, and a node is randomly chosen from topology and different number of tasks are submitted on chosen node at a simulation time to generate different values of n/m, and also task computation is generated following $U(4900{,}5100)$. It shows that increasing scheduling MST would lead to more extra computation cost. However, ECCR gradually descends along with the increasing of n/m, and when n/m achieves 10, ECCR is just only 15% notwithstanding scheduling MST is 6, comparing with the cost of methods which need real-time monitor of nodes and tasks, the extra computation in LMCGrid is acceptable.

The results above show that although node just bases on a small set of loose and imprecise information of neighbour nodes, task scheduling algorithm and limited task duplicating algorithm of LMCGrid are able to distribute the tasks from source node to its "able" adjacent nodes to run and avoid network variability by a little of extra computation. Reliable and persistent computing power could be supplied by LMCGrid. Simulations based on some simple topologies such as line, circle and star-graph as well as more complex topologies were performed too, and the results turned out to be similar to the results described in this paper.

7. Conclusion

In this paper, an LMCGrid model is presented, and corresponding information mechanism, task scheduling algorithm as well as limited task duplicating algorithm. Without any global information, task scheduling algorithm based on information mechanism naturally shapes proper resources area to serve different grid users. Limited task duplicating algorithm consuming small extra computing cycles screens the variability of network and eliminates the cost of real-time monitoring nodes and tasks. Simulation was performed to verify the validity and analyse performance essentially and results showed that LMCGrid was able to effectively utilize the large-scale idle computing resources connected to the Internet, and to achieve proper distribution of loads as well as avoiding the network variability entirely. It supplies a simple but effective method for computing of parameter sweep and Monte-Carlo method.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 60473088) and Science and Technology Development Plan Project of Tianjin (No. 05YFGZGX23900). We would like to thank Mr. Xiaoshe Dong, Ms. Qin Fan and Ms. Huifang Ren for their help on this paper.

References

[1] A.D. Alexandrov, M. Ibel, K.E. Schauser *et al.*, SuperWeb: Towards a global web-based parallel computing infrastructure, *Proc. 11th IEEE Int. Parallel Processing Symposium (IPPS)* (Geneva: IEEE Computer Society, 1997), 100–106.

[2] P. Cappello, B. Christiansen, M.F. Ionescu *et al.*, Javelin: Internet-based parallel computing using Java, *Concurrency: Practice and Experience*, *9*(11), 1997, 1139–1160.

[3] A. Baratloo, M. Karaul, H. Karl, & Z. Kedem, An infrastructure for network computing with Java applets, *Concurrency: Practice and Experience*, *10*(11–13), 1998, 1029–1041.

[4] G. Fedak, C. Germain, V. Neri *et al.*, XtremWeb: A generic global computing system, *Proc. 1st Int. Symp. on Cluster Computing and the Grid* (Brisbane: IEEE Computer Society, 2001), 582–587.

[5] L. Sarmenta, *Volunteer computing*, Ph.D. Dissertation, MIT, Massachusetts, 2001.

[6] M.O. Neary, S.P. Brydon, P. Kmiec *et al.*, Javelin++: Scalability issues in global computing, *Concurrency: Practice and Experience*, *12*(8), 2000, 727–753.

[7] M.O. Neary, A. Phipps, S. Richman *et al.*, Javelin 2.0: Java-based parallel computing on the Internet, *Euro-Par 2000* (Munich: Springer, 2000), 1231–1238.

[8] D. Epema, M. Livny *et al.*, A worldwide flock of condors: Load sharing among workstation clusters, *Journal on Future Generations of Computer Systems*, *12*(1), 1996, 53–65.

[9] N. Fujimoto & K. Hagihara, A comparison among grid scheduling algorithms for independent coarse-grained tasks, *SAINT 2004 Workshop on High Performance Grid Computing and Networking* (Tokyo: IEEE Press, 2004), 674–680.

[10] D. Paranhos, W. Cirne, & F. Brasileiro, Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids, *Int. Conf. on Parallel and Distributed Computing* (Klagenfrut: Springer, 2003), 169–180.

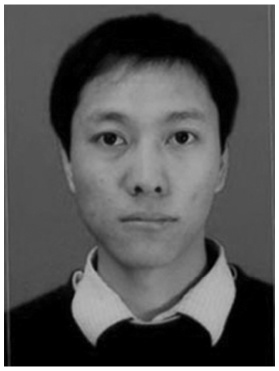
[11] H. Casanova, A. Legrand, D. Zagorodnov *et al.*, Heuristics for scheduling parameter sweep applications in grid environments, *Proc. 9th Heterogeneous Computing Workshop* (Cancun: IEEE Computer Society, 2000), 349–363.

[12] A.L. Barabási & R. Albert, Emergence of scaling in random networks, *Science*, *286*(5439), 1999, 509–512.

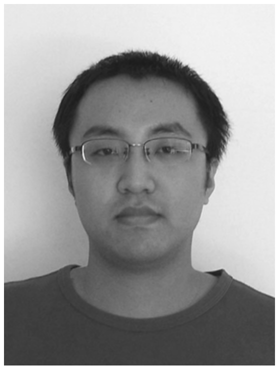
[13] J.S. Wu & Z.R. Di, Complex networks in statistical physics, *Progress in Physics*, *24*(1), 2004, 18–46 (in Chinese with English abstract).

Biographies

Yulu Yang received his B.E. degree from Beijing Agriculture Engineering University, China, in 1984; his M.E. and Ph.D. degrees from Keio University, Japan, in 1993 and 1996, respectively. He works in the Department of Computer Science, Nankai University, China. His research interests include interconnection network, parallel processing, reconfigurable computing and grids.



Xuegang Yang received his B.E. and M.E. degrees from Nankai University, Tianjin, in 2004 and 2007, respectively. He is now an Engineer in China TravelSky Technology Limited, Beijing, China. His research interests include interconnection networks, p2p and parallel processing.



Chao Zhou received his B.E. degree from Nankai University, Tianjin, in 2005. He is now studying in College of Information Technical Science in Nankai University as a postgraduate, Tianjin, China. His research interests include grid computing and parallel processing.