Towards Feasible and Effective Load Sharing in a Heterogeneous Computational Grid

Kuo-Chan Huang*, Po-Chi Shih, and Yeh-Ching Chung

*Department of Electronic Commercce**
Hsing Kuo College of Management
No. 89, Yuying Street, Tainan, Taiwan
kchuang@mail.hku.edu.tw

Department of Computer Science
National Tsing Hua University
101, Section 2, Kuang-Fu Road, Hsinchu, Taiwan
shedoh@sslab.cs.nthu.edu.tw , ychung@cs.nthu.edu.tw

Abstract. A grid has to provide strong incentive for participating sites to join and stay in it. Participating sites are concerned with the performance improvement brought by the gird for the jobs of their own local user communities. Feasible and effective load sharing is key to fulfilling such a concern. This paper explores the load-sharing policies concerning feasibility and heterogeneity on computational grids. Several job scheduling and processor allocation policies are proposed and evaluated through a series of simulations using workloads derived from publicly available trace data. The simulation results indicate that the proposed job scheduling and processor allocation policies are feasible and effective in achieving performance improvement on a heterogeneous computational grid.

Keywords: feasibility, load sharing, simulation, heterogeneous grid

1. Introduction

This paper deals with scheduling and allocating independent parallel jobs in a heterogeneous computational grid. Without grid computing local users can only run jobs on the local site. The owners or administrators of different sites are interested in the consequences of participating in a computational grid, whether such participation will result in better service for their local users by improving the job response time. Therefore, we say a computational grid is feasible if it can bring performance improvement and the improvement is achieved in the sense that all participating sites benefit from the collaboration. In this paper that means no participating sites' average response time for their jobs get worse after joining the computational grid.

In addition to feasibility, heterogeneity is another important issue in a computational grid. Many previous works have shown significant performance improvement for multi-site homogeneous grid environment. However, in the real world a grid usually consists of heterogeneous sites which differ at least in the

computing speed. Heterogeneity puts a challenge on designing effective load sharing methods. Methods developed for homogeneous grids have to be improved or even redesigned to make them effective in a heterogeneous environment. This paper addresses the potential benefit of sharing jobs between independent sites in a heterogeneous computational grid environment. To construct a feasible and effective computational grid, appropriate load sharing policies are important. The load sharing policies have to take into account several job scheduling and processor allocation issues. These issues are discussed in this paper, including job scheduling for feasible load sharing benefiting all sites, site selection for processor allocation, multi-site parallel execution. Several job scheduling and processor allocation policies are proposed and evaluated through a series of simulations using workloads derived from publicly available trace data. The simulation results indicate that a significant performance improvement in terms of shorter job response time is achievable.

2. Related Work

Job scheduling for parallel computers has been subject to research for a long time. As for grid computing, previous works discussed several strategies for a grid scheduler. One approach is the modification of traditional list scheduling strategies for usage on grid [1, 2, 3, 4]. Some economic based methods are also being discussed [5, 6, 7, 8]. In this paper we explore non economic scheduling and allocation policies with support for a heterogeneous grid environment.

England and Weissman in [9] analyzed the costs and benefits of load sharing of parallel jobs in the computational grid. Experiments were performed for both homogeneous and heterogeneous grids. However, in their works simulations of a heterogeneous grid only captured the differences in capacities and workload characteristics. The computing speeds of nodes on different sites are assumed to be identical. In this paper we deal with load sharing issues regarding heterogeneous grids in which nodes on different sites may have different computing speeds.

For load sharing there are several methods possible for selecting which site to allocate a job. Earlier simulation studies in our previous work [10] and in the literature [1] showed the best results for a selection policy called *best-fit*. In this policy a particular site is chosen on which a job will leave the least number of free processors if it is allocated to that site. However, these simulation studies are performed based on a computational grid model in which nodes on different sites all run at the same speed. In this paper we explore possible site selection policies for a heterogeneous computational grid. In such a heterogeneous environment nodes on different sites may run at different speeds.

In [11] the authors addressed the scheduling of parallel jobs in a heterogeneous multi-site environment. They also evaluated a scheduling strategy that uses multiple simultaneous requests. However, although dealing with a multi-site environment, the parallel jobs in their studies were not allowed for multi-site parallel execution. Each job was allocated to run within a single site.

The support of multi-site parallel execution [12, 13, 14, 15, 16] on a computational grid has been examined in previous works, concerning the execution of a job in

parallel at different sites. Under the condition of a limited communication overhead, the results from our previous work [10] and from [1, 3, 4] all showed that multi-site parallel execution can improve the overall average response time. The overhead for multi-site parallel execution mainly results from the slower communication between different sites compared to the intra-site communication. This overhead has been modeled by extending the execution time of a job by a certain percentage [2, 3, 10].

In [2] the authors further examined the multi-site scheduling behavior by applying constraints for the job fragmentation during the multi-site scheduling. Two parameters were introduced for the scheduling process. The first parameter *lower bound* restricted the jobs that can be fragmented during the multi-site scheduling by a minimal number of necessary requested processors. The second parameter was implemented as a vector describing the maximal number of job fragments for certain intervals of processor numbers.

However, the simulation studies in the previous works are performed based on a homogeneous computational grid model in which nodes on different sites all run at the same speed. In this paper we explore possible multi-site selection policies for a heterogeneous computational grid. In [17] the authors proposed job scheduling algorithms which allow multi-site parallel execution, and are adaptive and scalable in a heterogeneous computational grid. However, the introduced algorithms require predicted execution time for the submitted jobs. In this paper, we deal with the site selection problem for multi-site parallel execution, requiring no knowledge of predicted job execution time.

3. Computational Grid Model and Experimental Setting

In this section, the computational grid model is introduced on which the evaluations of the proposed policies in this paper are based. In the model, there are several independent computing sites with their own local workload and management system. This paper examines the impact on performance results if the computing sites participate in a computational grid with appropriate job scheduling and processor allocation policies. The computational grid integrates the sites and shares their incoming jobs. Each participating site is a homogeneous parallel computer system. The nodes on each site run at the same speed and are linked with a fast interconnection network that does not favor any specific communication pattern [18]. This means a parallel job can be allocated on any subset of nodes in a site. The parallel computer system uses space-sharing and run the jobs in an exclusive fashion.

The system deals with an on-line scheduling problem without any knowledge of future job submissions. The jobs under consideration are restricted to batch jobs because this job type is dominant on most parallel computer systems running scientific and engineering applications. For the sake of simplicity, in this paper we assume a global grid scheduler which handles all job scheduling and resource allocation activities. The local schedulers are only responsible for starting the jobs after their allocation by the global scheduler. Theoretically a single central scheduler could be a critical limitation concerning efficiency and reliability. However, practical

distributed implementations are possible, in which site-autonomy is still maintained but the resulting schedule would be the same as created by a central scheduler [19].

For simplification and efficient load sharing all computing nodes in the computational grid are assumed to be binary compatible. The grid is heterogeneous in the sense that nodes on different sites may differ in computing speed and different sites may have different numbers of nodes. When load sharing activities occur a job may have to migrate to a remote site for execution. In this case the input data for that job have to be transferred to the target site before the job execution while the output data of the job is transferred back afterwards. This network communication is neglected in our simulation studies as this latency can usually be hidden in pre- and post-fetching phases without regards to the actual job execution phase [19].

In this paper we focus on the area of high throughput computing, improving system's overall throughput with appropriate load sharing policies. Therefore, in our studies the requested number of processors for each job is bound by the total number of processors on the local site from which the job is submitted. The local site which a job is submitted from will be called the *home site* of the job henceforward in this paper. We assume the ability of jobs to run in multi-site mode. That means a job can run in parallel on a node set distributed over different sites when no single site can provide enough free processors for it due to a portion of resources are occupied by some running jobs.

Our simulation studies were based on publicly downloadable workload traces [20]. We used the SDSC's SP2 workload logs[1] on [20] as the input workload in the simulations. The workload log on SDSC's SP2 contains 73496 records collected on a 128-node IBM SP2 machine at San Diego Supercomputer Center (SDSC) from May 1998 to April 2000. After excluding some problematic records based on the *completed* field [20] in the log, the simulations in this paper use 56490 job records as the input workload. The detailed workload characteristics are shown in Table 1.

Table 1. Characteristics of the workload log on SDSC's SP2

	Number of jobs	Maximum execution time (sec.)	Average execution time (sec.)	Maximum number of processors per job	Average number of processors per job
Queue 1	4053	21922	267.13	8	3
Queue 2	6795	64411	6746.27	128	16
Queue 3	26067	118561	5657.81	128	12
Queue 4	19398	64817	5935.92	128	6
Queue 5	177	42262	462.46	50	4
Total	56490				

In the SDSC's SP2 system the jobs in this log are put into five different queues and all these queues share the same 128 processors on the system. In the following simulations this workload log will be used to model the workload on a computational

[1] The JOBLOG data is Copyright 2000 The Regents of the University of California All Rights Reserved.

grid consisting of five different sites whose workloads correspond to the jobs submitted to the five queues respectively. Table 2 shows the configuration of the computational grid under study. The number of processors on each site is determined according to the maximum number of required processors of the jobs belonged to the corresponding queue for that site.

Table 2. Configuration of the computational grid.

	total	site 1	site 2	site 3	site 4	site 5
Number of processors	442	8	128	128	128	50

To simulate the speed difference among participating sites we define a speed vector, speed=(sp1,sp2,sp3,sp4,sp5), to describe the relative computing speeds of all the five sites in the grid, in which the value 1 represents the computing speed resulting in the job execution time in the original workload log. We also define a load vector, load=(ld1,ld2,ld3,ld4,ld5), which is used to derive different loading levels from the original workload data by multiplying the load value ld*i* to the execution times of all jobs at site *i*.

4. Site Selection Policies for Load Sharing in a Heterogeneous Grid

This section explores the potential of a computational grid in improving the performance of user jobs. The following describes the scheduling structures of two system architectures with/without grid computing respectively.

- **Independent clusters**. This architecture corresponds to the situation where no grid computing technologies are involved. The computing resources at different sites are independent and have their own job queues without any load sharing activities among them. Each site's users can only submit jobs to their local site and those jobs would be executed only on that site. This architecture is used as a comparison basis to see what performance gain grid computing can bring.
- **Load-sharing computational grid.** Different sites connected with an interconnection network form a computational grid. In the computational grid, there is a global job scheduler as well as a globally shared job queue. Jobs submitted by users at different sites are automatically redirected to the global queue and the jobs retain the identities of their home sites. In this section, different sites in the computational grid are viewed as different processor pools and each job must be allocated to exactly one site. No jobs can simultaneously use processors on different sites. Support for multi-site parallel execution will be discussed in later sections.

Two kinds of policies are important regarding load sharing in a computational grid: *job scheduling* and *site selection*. Job scheduling determines the sequence of starting execution for the jobs waiting in the queue. It is required in both the *independent clusters* and *computational grid* architectures. On the other hand, site selection

policies are necessary in a computational grid, which choose an appropriate site among a set of candidate sites for allocating a job according to some specified criteria.

The *best-fit* site selection policy has been demonstrated to be the best choice on a homogeneous grid in previous works [1, 10]. In the *best-fit* policy a particular site is chosen for a job on which the job will leave the least number of free processors if it is allocated to that site. As for job scheduling policy, we compared both the FCFS (First-Come-First-Serve) policy and the NJF (Narrowest-Job-First) policy. The NJF policy was shown to outperform other non-FCFS policies, including *conservative backfilling*, *first-available*, *widest-first*, in our previous work [10]. Here, the word "narrowest" means requiring the least number of processors. In this paper we use the average response time of all jobs as the comparison criterion in all simulations, which is defined as:

$$AverageResponseTime = \frac{\sum_{j \in AllJobs} (endTime_j - submitTime_j)}{TotalNumberofJobs}$$

However, in the real world a computational grid is usually heterogeneous, at least, in the aspect of computing speeds at different sites. The *best-fit* site selection policy without considering the speed difference among participating sites may not achieve good performance in a heterogeneous grid, sometimes resulting in even worse performance than the original *independent-site* architecture.

To deal with the site selection issue in a heterogeneous grid, we first propose a two-phase procedure. At the first phase the grid scheduler determines a set of candidate sites among all the sites with enough free processors for a specific job under consideration by filtering out some sites according to a predefined threshold ratio of computing speed. In the filtering process, a lower bound for computing speed is first determined through multiplying the predefined threshold ratio by the computing speed of a single processor on the job's home site, and then any sites with single-processor speed slower than the lower bound are filtered out. Therefore, adjusting the threshold ratio is an effective way in controlling the outcomes of site selection. When setting the threshold ratio to 1 the grid scheduler will only allocate jobs to sites with single-processor speed equal to or faster than their home sites. On the other hand, with the threshold ratio set to zero, all sites with enough free processors are qualified candidates for a job's allocation. Raising the threshold ratio would prevent allocating a job to a site that is much slower than its home site. This could ensure a job's execution time would not be increased too much due to being allocated to a slow site. However, for the same reason a job may consequently need to wait in the queue for a longer time period. On the other hand, lowering the threshold ratio would make it more probable for a job to get allocation quickly at the cost of extended execution time. The combined effects of shortened waiting time and extended execution time are complicated for analysis. At the second phase the grid scheduler adopts a site selection policy to choose an appropriate site from the candidate sites for allocating the job.

Figure 1 compares the performances of two different values, 0 and 1, for the threshold ratio. The results indicate that when the speed difference among sites is

large, speed=(0.6, 0.7, 2.4, 9.5, 4.3), setting the threshold ratio to 1 can enable the *best-fit* policy to make performance improvement in a heterogeneous computational grid compared to the *independent-site* architecture.

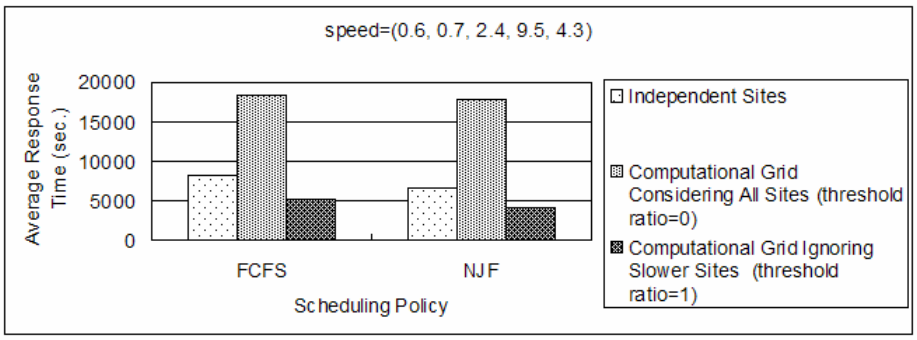

Fig. 1. Performance of *best-fit* policy with large speed difference among participating sites.

Another possible policy for the second phase of the site selection process is called the *fastest one*. The *fastest-one* policy chooses the site with the fastest computing speed among all the sites with enough free processors for a job without consideration of the difference between the number of required processors and a site's free capacity. To deal with the difficulty in determination of an appropriate site selection policy, in this section we propose an *adaptive* policy, which dynamically changes between the *best-fit* and the *fastest-one* policies, trying to make a better choice at each site selection activity. The decision is made based on a calculation of which policy can further accommodate more jobs for immediate execution. Figure 2 shows that the *adaptive* policy has potential for outperforming the *best-fit* and the *fastest-one* policies in some cases.

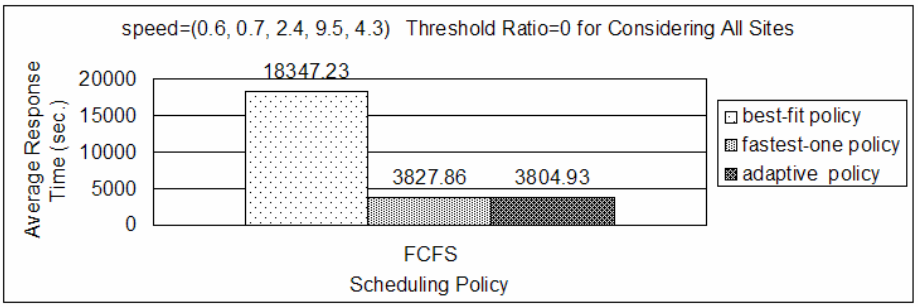

Fig. 2. Performance of the adaptive policy.

We also performed a series of 120 simulations representing all kinds of relative speed sequences for the 5 sites, permutations of speed=(1, 3, 5, 7, 9), in the computational grids. In the 120 simulations, among the three policies the *adaptive* policy is the most stable one. It is never the last one and always quite close to the best one in performance for all the 120 cases, while the other two policies would lead to

poor performance in some cases, being distant from the best and the second policies. Therefore, while it is not clear whether the *best-fit* or the *fastest-one* policy could achieve better performance under current grid configuration and workload, it may be a way for playing safe adopting the proposed *adaptive* policy.

5. Feasible Load Sharing in a Computational Grid

In most current grid systems, participating sites provide their resources for free with the expectation that they can benefit from the load sharing. Therefore, it is important to ensure that the load sharing is feasible in the sense that all sites benefit from it. Feasible load sharing is a good incentive for attracting computing sites to join a computational grid. In this paper, we define the feasibility of load sharing to be such a property which ensures the average job response time of each participating site is improved without exception. In this section we propose a feasible load sharing policy which works as follows. When the grid scheduler chooses the next job from the waiting queue and finds that there exists no single site with enough free processors for this job's immediate execution, instead of simply keeping the job waiting in the queue the grid scheduler inspects the status of the job's home site to see if it is possible to make enough free processors by reclaiming a necessary amount of occupied processors from some of the running remote jobs. If so, it stops the necessary amount of these running remote jobs to produce enough free processors and put the stopped remote jobs back to the front of the waiting queue for being re-scheduled to other sites for execution. This feasible load sharing policy tries to benefit all sites by giving local jobs a higher priority than remote jobs.

For performing the feasible load sharing policy, the grid scheduler maintains a separate waiting queue for each site. Each time it tries to schedule the jobs in one queue as more as it can until no more jobs can be allocated. At this time the grid scheduler moves on to the next queue for another site. Multi-queue is an effective mechanism to ensure that local jobs have higher priority than remote jobs during the processor reclaiming process.

Table 3 evaluates the effects of the feasible load sharing policy in a heterogeneous computational grid with speed=(1, 3, 4, 4, 8) and load=(5, 4, 5, 4, 1). The NJF scheduling policy and the *fastest-one* site selection policy are used in the simulations with the computing speed threshold ratio set to one, ensuring jobs won't be allocated to the sites slower than their home sites. Table 3 shows that with the ordinary load sharing policy site 5 got degraded performance after joining the grid, which may contradict its original expectation. On the other hand, our proposed policy is shown to be able to achieve a somewhat more feasible and acceptable load sharing result in the sense that no sites' performances were sacrificed.

Table 3. Average job response times (sec.) for different load sharing policies.

	Entire grid	Site 1	Site 2	Site 3	Site 4	Site 5
Independent sites	9260	14216	10964	10199	6448	57

Ordinary load sharing policy	4135	191	4758	4799	3881	559
Feasible load sharing policy	4152	193	4750	4798	3939	57

6. Multi-Site Parallel Execution in a Heterogeneous Grid

In the load sharing policies described in the previous sections, different sites in the computational grid are viewed as independent processor pools. Each job can only be allocated to exactly one of these sites. However, one drawback of this multi-pool processor allocation is the very likely internal fragmentation [4] where no pools individually can provide enough resources for a certain job but the job could get enough resources to run if it can simultaneously use more than one pool's resources.

Multi-site parallel execution is traditionally regarded as a mechanism to enable the execution of such jobs requiring large parallelisms that exceed the capacity of any single site. This is a major application area in grid computing called distributed supercomputing [21]. However, multi-site parallel execution could be also beneficial for another application area in grid computing: high throughput computing [21]. In our high throughput computing model in this paper, each job's parallelism is bound by the total capacity of its home site. That means multi-site parallel execution is not inherently necessary for these jobs. However, for high throughput computing a computational grid is used in the space-sharing manner. It is therefore not unusual that upon a job's submission its requested number of processors is not available from any single site due to the occupation of a portion of system resources by some concurrently running jobs. In such a situation, splitting the job up into multi-site parallel execution is promising in shortening the response time of the job through reducing its waiting time. However, in multi-site parallel execution the impact of bandwidth and latency has to be considered as wide area networks are involved. In this paper we summarize the overhead caused by communication and data migration as an increase of the job's runtime [2, 10]. The magnitude of this overhead greatly influences the achievable response time reduction for a job which is allowed to perform multi-site parallel execution.

If a job is performing multi-site parallel execution, the runtime of the job is extended by the overhead which is specified by a parameter p [2]. Therefore the new runtime r^* is:

$$r^* = (1 + p) \times r$$

where r is the runtime for the job running on a single site. As for the site selection issue in multi-site parallel execution, previous works in [1, 10] suggested the *larger-first* policy for a homogeneous grid environment, which repeatedly picks up a site with the largest number of free processors until all the selected sites together can fulfill the requirement of the job to be allocated. As a heterogeneous grid being considered, the speed difference among participating sites should be taken into account. An intuitive heuristic is called the *faster-first* policy, which each time picks up the site with the fastest computing speed instead of the site having the most

amount of free processors. This section develops an *adaptive* site selection policy which dynamically changes between the *larger-first* and the *faster-first* policies based on a calculation of which policy can further accommodate more jobs for immediate single-site execution.

Figure 3 shows that supporting multi-site parallel execution can further improve the performance of a heterogeneous load sharing computational grid when the multi-site overhead p=2. Moreover, our proposed *adaptive* site selection policy outperforms the *larger-first* and the *faster-first* policies significantly. Actually in all the 120 simulations we performed for different speed configurations the *adaptive* policy performs better than the other two policies for each case.

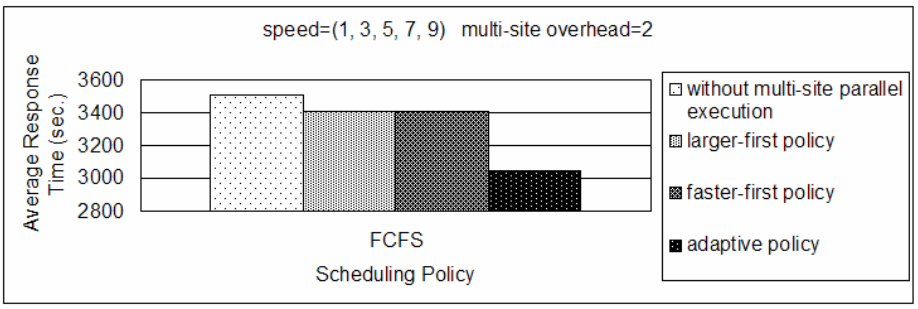

Fig. 3. Performance evaluation of adaptive site selection in multi-site parallel execution

7. Conclusion

Most current grid environments are established through the collaboration among a group of participating sites which volunteer to provide free computing resources. Each participating site usually has its own local user community and computing jobs to take care of. Therefore, feasible load sharing policies that benefit all sites are an important incentive for attracting computing sites to join and stay in a grid environment. Moreover, a grid environment is usually heterogeneous in nature in the real world at least for the different computing speeds at different participating sites. The heterogeneity presents a challenge for effectively arranging load sharing activities in a computational grid. This paper explores the feasibility and effectiveness of load sharing activities in a heterogeneous computational grid. Several issues are discussed including site selection policies for single-site and multi-site parallel execution as well as feasible load sharing mechanisms. For each issue a promising policy is proposed and evaluated in a series of simulations. The quality of scheduling and allocation policies largely depends on the actual grid configuration and workload. The improvements presented in this paper were achieved using example configurations and workloads derived from real traces. The outcome may vary in other configurations and workloads. However, the results show that the proposed policies are capable of significantly improving the overall system performance in terms of average response time for user jobs.

Acknowledgement

The work of this paper is partially supported by National Science Council and National Center for High-Performance Computing under NSC 94-2218-E-007-057, NSC 94-2213-E-432-001 and NCHC-KING_010200 respectively.

References

[1] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of Job-Scheduling Strategies for Grid Computing", *Proceedings of the 7th International Conference on High Performance Computing, HiPC-2000*, pp. 191-202, Bangalore, India, 2000.

[2] C. Ernemann, V. Hamscher, R. Yahyapour, and A. Streit, "Enhanced Algorithms for Multi-Site Scheduling", *Proceedings of 3rd International Workshop Grid 2002, in conjunction with Supercomputing 2002*, pp. 219-231, Baltimore, MD, USA, November 2002.

[3] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour, "On Advantages of Grid Computing for Parallel Job Scheduling", *Proceedings of 2nd IEEE International Symposium on Cluster Computing and the Grid (CC-GRID 2002)*, pp. 39-46, Berlin, Germany, 2002.

[4] C. Ernemann, V. Hamscher, A. Streit, R. Yahyapour, ""On Effects of Machine Configurations on Parallel Job Scheduling in Computational Grids", *Proceedings of International Conference on Architecture of Computing Systems, ARCS 2002*, pp. 169-179, 2002.

[5] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, "Economic Models for Resource Management and Scheduling in Grid Computing", Special Issue on Grid Computing Environments, *The Journal of Concurrency and Computation: Practice and Experience(CCPE)*, May 2002.

[6] R. Buyya, J. Giddy, D. Abramson, "An Evaluation of Economy-Based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications", *Proceedings of the Second Workshop on Active Middleware Services (AMS2000), In conjunction with the Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC 2000)*, Pittsburgh, USA, August 2000.

[7] Y. Zhu, J. Han, Y. Liu, L. M. Ni, C. Hu, J. Huai, "TruGrid: A Self-sustaining Trustworthy Grid", *Proceedings of the First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05)*, pp. 815-821, June 2005.

[8] C. Ernemann, V. Hamscher, R. Yahyapour, "Economic Scheduling in Grid Computing", *the 8th International Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science*; Vol. 2537, pp. 128-152, 2002.

[9] D. England and J. B. Weissman, "Costs and Benefits of Load Sharing in Computational Grid", *10th Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science*, Vol. 3277, June 2004.

[10] K. C. Huang and H. Y. Chang, "An Integrated Processor Allocation and Job Scheduling Approach to Workload Management on Computing Grid", *Proceedings of the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06)*, pp. 703-709, Las Vegas, USA, June 26-29, 2006.

[11] G. Sabin, R. Kettimuthu, A. Rajan and P. Sadayappan, "Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment", *Proceedings of 9th Workshop on Job Scheduling Strategies for Parallel Processing*, June 2003.

[12] M. Brune, J. Gehring, A. Keller, A. Reinefeld, "Managing Clusters of Geographically Distributed High-Performance Computers", *Concurrency – Practice and Experience*, 11(15): 887-911, 1999.
[13] A. I. D. Bucur and D. H. J. Epema, "The Performance of Processor Co-Allocation in Multicluster Systems", *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, pp. 302-, May 2003.
[14] A. I. D. Bucur and D. H. J. Epema, "The Influence of Communication on the Performance of Co-Allocation", *the 7th International Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*; Vol. 2221, pp. 66-86, 2001.
[15] A. I. D. Bucur and D. H. J. Epema, "Local versus Global Schedulers with Processor Co-Allocation in Multicluster Systems", *the 8th International Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science*, pp. 184-204, 2002.
[16] S. Banen, A. I. D. Bucur and D. H. J. Epema, "A Measurement-Based Simulation Study of Processor Co-Allocation in Multicluster Systems", *the 9th Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science*; Vol. 2862, pp. 105-128, 2003.
[17] W. Zhang, A. M. K. Cheng, M. Hu, "Multisite Co-allocation Algorithms for Computational Grid", *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, pp. 8-, April 2006.
[18] D. Feitelson and L. Rudolph, "Parallel Job Scheduling: Issues and Approaches", *Proceedings of IPPS'95 Workshop: Job Scheduling Strategies for Parallel Processing*, pp. 1-18, 1995.
[19] C. Ernemann, V. Hamscher, R. Yahyapour, "Benefits of Global Grid Computing for Job Scheduling," *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*(GRID'04), pp. 374-379, November 2004.
[20] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload/
[21] I. Foster, C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, Inc., 1999.