Multisite Co-allocation Algorithms for Computational Grid

Weizhe Zhang[a], Albert M. K. Cheng[b], Mingzeng Hu [a]
[a] *School of Computer Science and Technology, Harbin Institute of Technology, P.R.China*
{zwz , mzh}@pact518.hit.edu.cn
[b]*Real-Time Systems Laboratory, Department of Computer Science, University of Houston, Houston, TX 77204-3010, USA*
cheng@cs.uh.edu

Abstract

Efficient multisite job scheduling facilitates the cooperation of multi-domain massively parallel processor systems in a computing grid environment. However, co-allocation, heterogeneity, adaptability, and scalability emerge as tough challenges for the design of multisite job scheduling models and algorithms. This paper presents a new multisite job scheduling schema based on the multisite job scheduling model and the performance model for a heterogeneous grid environment. There are three key components: resource selection, reservation, and backfilling. The optimal and greedy-heuristic adaptive resource selection strategies are introduced. The conservative and easy backfilling are incorporated into the backfilling procedure. Experiments indicate that the scheduler and the algorithm are effective and perform better than a non-adaptive algorithm.

1. Introduction

In the last few years, the trends in parallel processing system design and deployment have been to move away from single isolated powerful supercomputers to cooperative networked distributed systems, so called the *grid* [1]. The name grid has been chosen as an analogy to the electric power grid where several power plants provide numerous consumers with electric power that the consumer is not aware of its origin. Similar, it is the goal of the grid to allow scientists and engineers to solve their large-scale challenging applications [2].

Currently, grid systems are classified as *computational*, *data*, and *service* grids [3]. The computational grid category denotes systems that have a higher aggregate computational capacity available for applications than the capacity of any constituent resource in the system. Clearly, a computational grid is mainly of interest for large computational jobs or jobs using a large data set as smaller jobs would usually run locally. It has the potential to provide low average response time for computational jobs and high utility for computational resources. This potential can be realized, however, only if the resources are managed effectively, and especially the computational jobs are scheduled well.

Recent advances in creating the grid resource management infrastructure (e.g., Globus, Legion, Condor-G, and UNICORE), facilitate the deployment of the grid scheduler to schedule jobs onto multiple heterogeneous sites, and promote the investigation of the grid scheduling algorithms. Research into scheduling for the grid environment can be broadly classified into two categories: 1) *application-level scheduling*: the focus is on approaches to optimize the performance of a single job in a grid environment, and 2) *job-level scheduling*: the focus is on the performance optimization across a collection of independent jobs. In this paper, we put the emphasis upon the second branch----independent parallel job scheduling in a real-world computational grid scenario.

Parallel Job scheduling is a complex problem, even in a single parallel computer. However, grid computing systems, compared to the classical parallel computers, pose several technical challenges that introduce an additional degree of complexity to the scheduling problem while amplifying the existing ones. Therefore, it is necessary to point out the intrinsic nature of grid job scheduling that is different from parallel computer scheduling as follows:

1) *Co-allocation* or *multisite* scheduling: a computational grid is typically composed of several sites from geographically distributed organizations (such as TeraGrid [4], E-science [5] and DAS [6]). Parallel jobs should be scheduled to spread to more than one sites in order to run simultaneously on several

1-4244-0054-6/06/$20.00 ©2006 IEEE

sites without considering the resource limitation from one single site. The grid scheduling algorithm should be capable of coordinating these resources from different sites;

2) *Heterogeneity*: in a real-world grid scenario, hardware and software resources from different sites may have a rich diversity. Heterogeneous scheduling issues are highlighted which simply do not occur in "single-chassis" sequential or parallel machines;

3) *Adaptability*: though co-allocation can decrease the response time and the utilization, the makespan of some communication-intensive multisite applications is increased sharply because of the low bandwidth and high latency between the sites. Thus, the scheduling algorithm must adapt to the network performance and determine which parallel jobs should be mapped to single sites or multiple sites;

4) *Scalability*: the scheduler for a single parallel machine has a limited number of resources to control. In comparison, the grid is intended to span over a very large number of systems. Therefore, the scheduling algorithms should avoid the explosion in computational complexity.

Based on the above essences of the grid, this paper concentrates on the heterogeneous multisite job scheduling in real-world computational grid systems - a homogeneous cluster of processors at each site and different sites have different performance characteristics. We propose a multisite scheduling scheme which has been extended from the practically effective, backfilling-based parallel job scheduling strategies. A novel adaptive, multisite, and scalable scheduling algorithm is introduced and performance evaluations which include the average response time, average wait time, and utility are provided.

The organization of this paper is as follows. We introduce related works and clarify our motivation in section 2. Section 3 presents the multisite computing system model and multisite job performance model. The adaptive multisite scheduling schema and algorithms are discussed in detail in section 4. Section 5 defines the performance metrics used for evaluation. Initial performance comparison under different scenarios is presented in section 6. Finally, we conclude this paper and give a discussion in section 7.

2. Related works and our motivation

In this section, we will introduce recent advances in computational grid scheduling. First, in section 2.1 job-level scheduling is described in detail. Our motivation for adaptive, heterogeneous, multisite grid job scheduling is clarified in section 2.2.

2.1 Job-level grid scheduling

Considerable research has been conducted over the last decade on the topic of job scheduling for parallel systems. Much of this research has been presented at the annual Workshops on Job Scheduling Strategies for Parallel Processing [7] and the International Heterogeneous Computing Workshop [8]. Moreover, Feitelson, Rudolph and Schwiegelshohn have written a couple of surveys [9. 10] to report the current art and state for parallel job scheduling on the supercomputer. Recent trends for parallel job scheduling in workstation clusters and the computational grid are summarized in [11], where the grid job scheduling is classified into singlesite (non-co-allocation) and multisite (co-allocation). Section 2.2,1 presents some remarkable works on singlesite grid job scheduling. Also, some important researches on multisite grid job scheduling are introduced in section 2.2.2.

2.2.1. Singlesite job scheduling. As a continuation of metacomputing ideas, singlesite job scheduling permits the jobs to run on the sites from different domains, but it is not allowed for a single job to share the machines by crossing the site boundaries. Abawajy and Dandamudi [12] propose an on-line dynamic scheduling policy that manages multiple job streams on multicluster computing systems with the objectives of improving the mean response time and system utilization. Also, Sabin, Kettimuthu, Rajan and Sadayappan [13] present the idea of multiple simultaneous requests for allocating jobs to the heterogeneous environment. The basic idea is to submit each job to multiple sites, and cancel redundant submissions when one of the sites is able to start the job.

More recently, Ernemann, Hamscher and Yahyapour [14] perform simulations to evaluate the effects of a global grid constituted by the compute centers located in different time zones with a simple two-step job scheduling strategy (Bestfit and Backfilling). Their results show that the average weighted response time of all submitted jobs decrease up to about 30% for a global grid with different time zone distributions comparing with closed Grids in a single country. Also, the benefits of load sharing of parallel jobs in the homogeneous and heterogeneous grid are investigated and a simple scheduling heuristic to select the target machines of migrated jobs is provided [15]. In addition some works on the usage of the genetic algorithm to improve the quality of the grid scheduling are discussed in [16, 17].

2.2.2. Multisite job scheduling. The described restriction of singlesite job scheduling is lifted in a multisite scenario and a job can be executed in more than one site in parallel. In [18, 19], the authors analyzed the problem of executing a parallel application on a multi-cluster environment. They presented some simulations where multisite execution was beneficial compared with job-sharing, even for an additional communication overhead of about 25%. Later, the same authors improved the previous scheduling process by applying constraints for the fragmentation of jobs [20]. Finally, they presented results showing that the use of partitioned configuration did not necessarily imply a performance drawback [21].

Another important research about co-allocation or multisite site scheduling is presented in [22-31]. Bucur and Epema [22-24] assess the influence on the mean response time of the job structure and size, the sizes of the clusters in the system, the ratio of the speeds of local and wide-area communications, and of the presence of a single or of multiple queues in the system. Also, they evaluated different scheduling policies for co-allocation, with unordered requests, in multicluster systems with space sharing for rigid multi-component jobs [25-28]. Furthermore, they use the measure-based trace to evaluate the performance of the scheduling policies, design a dynamic co-allocation service and implement multiple components for a real-world wide-area computer system consisting of five clusters [29-31]. Some other works are performed under different assumptions and constraints. A new scheduling model that permits job migrations is considered to share the dynamic grid environment [32-34]. Snell, Clement, Jackson and Gregory [35] propose advanced reservation strategies for co-allocation.

2.3 Our motivation

Most of the scheduling algorithms described above cover only part of the nature of the grid mentioned in section 1. For example, many singlesite scheduling algorithms in section 2.2.1 are extended from the traditional heterogeneous computing systems in the same domain, which neglects the possibility of co-allocation or multisite computing jobs across the boundary of sites [12-17]. Although many papers in multisite job scheduling have addressed the co-allocation problem indeed, all the sites from different domains are assumed to be homogenous without taking the heterogeneity in a real-world grid scenario into consideration [18-31]. Moreover, less attention is paid on the adaptability and some algorithms lack scalability [16, 17]. Our motivation is to design grid job scheduling algorithms which allow co-allocation, and are adaptive and scalable in a heterogeneous computational grid.

3 Models for multisite job scheduling

In this section, we introduce our model of a multisite computing system for the computational grid. The section is organized as follows. Section 3.1 shows our model of multisite computing system and the constraints for its components. In section 3.2, the performance models for jobs across the multisite are discussed.

3.1 Models for multisite computing systems

Models for multisite computing systems can be divided into four parts: sites, jobs and job queue, local scheduler, and grid scheduler as showed in figure 1.

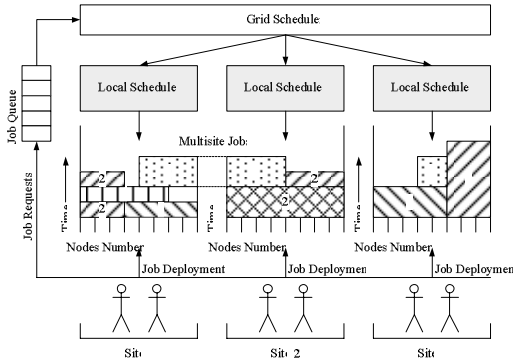

Fig.1. Model for multisite computing system

3.2 Multisite job performance model

An accurate performance prediction model is the foundation of scheduling algorithms' design and evaluation. In this subsection, we introduce the multisite job performance model in a heterogeneous computational grid.

Assume the number of sites is N. Let T_i represent the predicated execution time of parallel jobs on the local, single site i, which is normally provided by grid users; $T_{i->j}$ denotes the time when the job is submitted at the site i but runs at the remote, single site j; $T_{i->(j,\ k.....m)}$ means the job is submitted at the site i but executes simultaneously at sits $j, k....m$, where $1 \le i, j, k, m \le N$.

The heterogeneity of different sites is denoted by a heterogeneous factor h_i, where $0<h_i$. The penalty of multisite execution is denoted by a multisite factor $p_{(j, k...m)}$, where $1 \le p_{(j, k...m)}$. The heterogeneity and penalty of multisite execution in a real system scenario highly depend on the architecture of the sites, nodes'

performance, job communication patterns, and the network configuration between the sites. In our model, we do not consider these effects. All mentioned heterogeneity and multisite penalty are summarized in the heterogeneous and multisite factor, respectively.

When a job is submitted at the site i and transferred to the remote site j, the transport of data requires additional time $t_{trans(i,j)} = s/b_{i,j} + l_{i,j}$, where s is the data size, $b_{i,j}$ and $l_{i,j}$ are network bandwidth and latency, respectively. Now the performance prediction model of $T_{i->j}$ is presented as follows:

$$T_{i->j} = T_i \bullet h_j/h_i + t_{trans(i,j)} \quad (1)$$

Consider the job models of the multisite scenerio in section 3.2.2, all the tasks of a job are terminated at the same time. Thus, the predicted execution time $T_{i->(j,k.....m)}$ should be determined by the worst performance site as follows:

$$T_{i->(j,\ k...m)} = T_i \bullet p_{(j,\ k...m)} \bullet \max(h_j,\ h_k, ...,\ h_m)/h_i + \max(t_{trans(i,j)},\ t_{trans(i,k)}, ..., t_{trans(i,m)}) \quad (2)$$

4 Multisite scheduling algorithms

This section introduces the scheme of multisite scheduling algorithms. We will discuss the complexity of the algorithms in a future paper.

4.1 Multisite scheduling algorithm scheme

In the online scheduling scenario of grid jobs, most job-level grid scheduling algorithms use the First-Come-First-Served (FCFS) policy as the basic scheduling scheme. FCFS provides some kinds of fairness, is easy to implement, and requires very little computational effort [36]. Thus, we employ the FCFS priority strategy in our global job queue. However, FCFS can result in poor scheduling quality such as low system utilization [37] when job requests with large node requirements are submitted. Backfilling [38, 39] is proposed to improve the system utilization and by identifying idle nodes and moving forward smaller jobs that fit into those nodes, without delaying any job with future reservations. Therefore, we use Backfilling to reinforce the FCFS scheme.

Though we use the FCFS plus Backfilling in the scheduling scheme as a traditional parallel supercomputer has done, there are two remarkably different challenges on the design of heterogeneous, multisite job scheduling algorithms: a) The heterogeneity of the sites makes the scheduling decision more complex. If all the nodes of the sites are homogeneous, the execution time of a job in the submitted single site remains best compared with transporting the job to a remote single site or multisite because of the transportation and synchronization penalty. However, the penalty resulting from network communications can be remedied when the remote sites or multisite have better computing ability than the submitted sites. Therefore, the execution time of a job may decreases when the job executes on the remote sites or multisite in a heterogeneous grid. b) Multisite reservation makes it possible to reduce the execution time of a job. Traditionally, the scheduler reserves nodes only if the quantity of node requests can not be satisfied. In the heterogeneous, multisite grid, a job can be reserved in order to acquire better execution time even if it can run immediately. Based on the above consideration, we present a multisite scheduling algorithm scheme as follows:

Algorithm 1. Multisite scheduling algorithm scheme

Input: (1) Job queue (2) Sites aggregate

Output: (1) Mapping results

Variables: Δt inter-schedule interval

Currentjob first unmapped job in the job queue

Status indicate two kinds results of resource selection, instant and reservation execution

1. (Initialization) If the job queue is empty, then wait Δt interval and recheck the status of the job queue; Otherwise, collect the sites' state from the local schedulers and the job requests from the job queue.
2. (Mapping) For each unmapped job request in the current job queue

(a) *CurrentJob* ← the first unmapped job in the current job queue

(b) *Status*← *ResourceSelection* (*CurrentJob*) //return the mapped status and resources for current job

(c) If the mapped *Status* suggests *CurrentJob* should run immediately in the single site or multisite, inform the local schedulers to transport and execute the job; otherwise, call the function *Reserve* (*CurrentJob*) and *Backfill* (*CurrentJob*) sequentially.

(d) Update the information for job queue and sits status

3. (Return) Return the mapping results and go back to step 1.

4.1.1. Resource selection. In this section, we propose two adaptive, multisite resource selection sub-algorithms – optimal and greedy adaptive multisite resource selection.

The common ideas of these two algorithms is as follows: First, if the node requests of a current job can't be satisfied or the predicted job execution time of immediate running is longer than by reservation, the current job is reserved. Second, both are adaptive to different resources whether it is local, remote single

site, or multisite. To the end, better resource allocation and mapping is selected and returned.

However, the difference between the algorithms is remarkable. Optimal resource selection enumerates all the resource combinations for best job performance but lacks scalability while the greedy resource selection employs the greedy heuristic based on the node performance. It is more scalable subject to some kinds of performance penalty.

Algorithm 2. Optimal adaptive multisite resource selection algorithm

Input: (1) *CurrentJob* (2) Sites status

Output:(1) *Status* of instant or reservation execution

(2) Mapping results of the best resource allocation

Variables: $T_{i->j}$ predicted execution time when job submitted at site *i* but run at site *j*

$T_{i->(j,\ k...m)}$ predicted execution time when job submitted at site *i* but run at site *j*, *k*,...*m*

$T'_{i->j}$ predicted execution time when job submitted at site *i* but run at site *j* by reservation

$T'_{i->(j,\ k...m)}$ predicted execution time when job submitted at site *i* but run at site *j*, *k*,..*m* by reservation

$T_{i,\ \text{available}}$ the earliest available time of site *i*

1. (Check resources limitation) If the node request of *Currentjob* exceeds the total node number from all sits, then drop the job and procedure return. Otherwise, if the node request of *Currentjob* exceeds all the idle node number, then *Status*← reservation flag and procedure return; else, go to step 2

2. (Compute instant execution time) Enumerate all the combination on single sites or multisite.

(a) For each single site which the idle nodes are larger than the *Currentjob*'s request, computing $T_{i->j}$

(b) For all the multisite combination that the idle nodes are larger than the *Currentjob*'s request, computing $T_{i->(j,\ k...m)}$

3. (Compute reservation execution time) Enumerate all the combination in single sites or multisite by reserve.

(a) For each site, compute $T_{i,\ \text{available}}$

(b) For each single site which the total nodes are larger than the *Currentjob*'s request, $T'_{i->j} \leftarrow T_{i,\ \text{available}} + T_{i->j}$

(c) For all the multisite combination which the total nodes are larger than the *Currentjob*'s request, $T'_{i->(j,\ k...m)} \leftarrow T_{i->(j,\ k...m)} + \max(T_{j,\ \text{available}}, T_{k,\ \text{available}}, T_{m,\ \text{available}})$

4. (Return) If the shortest predicted execution time belongs to the $T_{i->j}$ or $T_{i->(j,\ k...m)}$, then *Status*← instant flag and return the resource allocation; Otherwise, *Status*← reservation flag and return.

4.1.2. Job reservation and backfilling. The job reservation procedure also can be classified into optimal and greedy. Both algorithms are almost identical with the step 3 of the algorithm 2, so that we do not present the algorithms in detail. Actually, in a real-implementation the multisite job scheduling requires more advanced resource reservation than in traditional supercomputing. Traditional parallel computers are normally in the same domain with identical management policies. Nevertheless, in a multisite grid scenario, the different resources belong to different owners and do not have a common management infrastructure. Also, it lacks cooperation between the local schedulers and grid scheduler, and there is little knowledge about each others policies, priorities, or workload. As a consequence of these conditions, it is hard to realize the intention of multisite co-allocation. Advanced reservation that reserves the resources on different sites may circumvent this problem [39]. Once a job is reserved, there are two common variations to backfilling - easy and conservative [38, 39].

5 Performance Metrics

We use the following metrics to evaluate the performance of multisite scheduling algorithms:

1. Average Weighted Response Time (AWRT):

$$AWRT = \frac{\sum_{j \in Jobs} Cost_j \times (T_{j,end} - T_{j,submit})}{\sum_{j \in Jobs} Cost_j} \qquad (3)$$

2. Average Weighted Wait Time (AWWT):

$$AWWT = \frac{\sum_{j \in Jobs} Cost_j \times (T_{j,start} - T_{j,submit})}{\sum_{j \in Jobs} Cost_j} \qquad (4)$$

In equations (3) and (4), $T_{j,\ end}$, $T_{j,\ start}$ and $T_{j,\ submit}$ represent the end time, start time, and submitted time of job *j*, respectively. The resource consumption of a job $Cost_j = W_i \bullet (T_{j,\ end} - T_{j,\ start})$ is defined as the product of the job's runtime and the number of requested resources, where W_i represents the number of requested resources. The average weighted response time is the sum of all weighted response times divided by the number of all jobs. The wait time of each job is the difference between the start time and the submission time. The weights are defined the same way as for the average weighted response time. AWRT and AWWT have been used to evaluate the quality of scheduling algorithms from the system's and user's viewpoint by many other researchers [14, 18-21, 33].

6 Performance evaluation: adaptive vs. non-adaptive

In this section, we evaluate the performance of multisite job scheduling algorithms and several aspects which would impact the performance are investigated. In particular, we study the performance of the adaptive resource selection against the non-adaptive version. The non-adaptive algorithm assigns strict priorities for site combination selection in order to avoid the penalty of job transfer and multisite communication. Namely, jobs prefer the submitted local site with the highest priority, then remote single sites, and multisites as the lowest one. The experimental results in Figures 2 and 3 show that no matter how small (NISAC 1.0, HIT 1.0, UC 1.0) or large (NISAC 1.4, UC 1.0, HIT 0.6) heterogeneity difference, adaptive site selection significantly outperforms the non-adaptive one. We configure the network bandwidth between sites as 100Kbps, the conservative algorithm is used for backfilling and the revised original trace (JobWorkload1) is adapted as input.

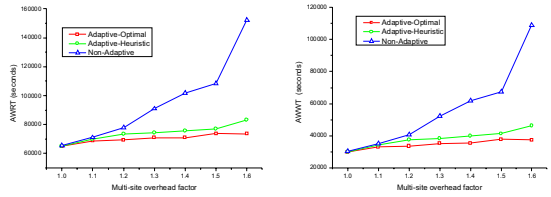

Fig.2. AWRT and AWWT comparisons of adaptive and non-adaptive algorithms with small heterogeneity difference

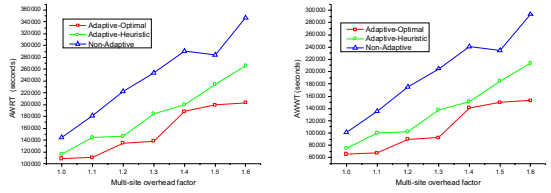

Fig.3 AWRT and AWWT comparisons of adaptive vs. non-adaptive algorithms with large heterogeneity difference

Figures 2 and 3 show the AWRT and AWWT of the optimal adaptive, greedy adaptive, and non-adaptive algorithms as the multisite factor varies from 1.0 to 1.6 with small and large heterogeneity difference. We find that the AWRT and AWWT of the optimal and greedy adaptive algorithms are consistently lower than the non-adaptive one because not only for the single site and multisite but also when the immediate and reservation executions are considered in the adaptive algorithm. Also, with the increment of the multisite factor the difference becomes larger. The reason is that a larger multisite factor provides more possibility of reservation execution. When the multisite factor is 1.6 in Figure 2, the AWRT of the non-adaptive algorithm is nearly 100% higher than the optimal one while the AWWT is nearly 200% higher. Because the greedy adaptive algorithm is a kind of heuristic favoring the high performance resources and does not enumerate all the resource combinations, its AWRT and AWWT are slightly higher than the optimal one. Nevertheless, the AWRT and AWWT of the greedy one are still 82% and 134% lower than the non-adaptive one. Meanwhile, when the multisite factor is 1.6 in Figure 3, the AWRT of the non-adaptive algorithm is still 30% higher than the greedy one and 70% higher than the optimal one. We will report experimental results of the impact of site heterogeneity and network performance, as well as easy versus conservative backfilling strategies, on the performance of the proposed algorithms.

7 Conclusion and future work

In this paper we point out that co-allocation, heterogeneity, adaptability, and scalability are the intrinsic nature of grid job scheduling different from parallel computer scheduling. A multisite computing (co-allocation) system model is introduced in the real-world scenario, which allows the jobs to run across site boundaries. Optimal and greedy multisite scheduling algorithms are proposed to adaptively select and map grid jobs to heterogeneous resource combinations. The greedy multisite scheduling algorithm scales well while the optimal one does not have polynomial time complexity. Initial experimental results show that the adaptability of an algorithm is very important to its performance, as shown by comparing optimal and greedy adaptive algorithms with the non-adaptive version. This work is just a first step to exploit the nature of grid job scheduling and there are still many works remaining for further exploration. For example, many systems are connected to the grid, so the continuous availability and work must be guaranteed. Grid scheduling algorithms can keep parallel jobs running even if some components are impacted by a network or resource failure.

Acknowledgment This work was supported in part by the Natural Science Foundation of China under Grant No.90412001 and the National Grand Fundamental Research 973 Program of China under Grant No.G2005CB321806. Albert M. K. Cheng is also supported by a grant from the Institute for Space Systems Operations.

References

[1] I. Foster. The grid: A new infrastructure for 21st century science. Physics Today, 55(2):42 – 47, 2002.

[2] Ian Foster and Carl Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure (Second Edition). Morgan Kaufmann, 2004.

[3] Arshad Ali, Ashiq Anjum, Atif Mehmood, Richard McClatchey, Ian Willers, Julian Bunn, Harvey Newman, Michael Thomas, Conrad Steenberg A Taxonomy and Survey of Grid Resource Planning and Reservation Systems for Grid Enabled Analysis Environment Proceedings of the 2004 International Symposium on Distributed Computing and Applications to Business Engineering and Science

[4] The TeraGrid project. http://www.teragrid.org/

[5] The Europe Data Grid. http://egee-intranet.web.cern.ch/egee-intranet/gateway.html

[6] The Distributed ASCI Supercomputing's site. http://www.cs.vu.nl/das

[7] Workshops on Job Scheduling Strategies for Parallel Processing, http://www.cs.huji.ac.il/~feit/parsched/

[8] The International Heterogeneous Computing Workshop. http://www.cs.umass.edu/~rsnbrg/hcw2005/hcw05_prev_workshops.html

[9] D. G. Feitelson, A Survey of Scheduling in Multiprogrammed Parallel Systems. Research Report RC 19790 (87657), IBM T. J. Watson Research Center, Oct 1994.

[10] D. G. Feitelson and L. Rudolph, ``Parallel job scheduling: issues and approaches". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), pp. 1-18, Springer-Verlag, 1995. Lecture Notes in Computer Science Vol. 949

[11] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, ``Parallel job scheduling --- a status report". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.), pp. 1-16, Springer-Verlag, 2004. Lecture Notes in Computer Science Vol. 3277.

[12] J. H. Abawajy, S. P. Dandamudi. "Parallel Job Scheduling on Multicluster Computing Systems," cluster, p. 11, IEEE International Conference on Cluster Computing (CLUSTER'03), 2003.

[13] Gerald Sabin, Rajkumar Kettimuthu, Arun Rajan and P. Sadayappan, Scheduling of Parallel Jobs in a Heterogeneous Multisite Environment, Proceedings of 9th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2003), June 2003

[14] C. Ernemann, V. Hamscher, R. Yahyapour "Benefits of Global Grid Computing for Job Scheduling", in 5th IEEE/ACM International Workshop on Grid Computing, in Conjunction with SuperComputing 2004, November 8, 2004, Pittsburgh, USA, pp. 374-379, IEEE press.

[15] Darin England, Jon B. Weissman. Costs and Benefits of Load Sharing in the Computational Grid. Workshop on Job Scheduling Strategies for Parallel Processing with Sigmetrics 2004.

[16] V. Di Martino , M. Mililotti, Sub optimal scheduling in a grid using genetic algorithms, Parallel Computing, v.30 n.5-6, p.553-565, May 2004

[17] Yang Gao, Joshua Zhexue Huang, Hongqiang Rong. Adaptive Grid Job Scheduling with Genetic Algorithm. Future Generation Computer System. Elsevier Press. 2005.21:151-161.

[18] V.Hamscher, U.Schwiegelshohn, A.Streit, R.Yahyapour, "Evaluation of Job-Scheduling Strategies", for Grid Computing (Grid 2000) at 7th International Conference on High Performance Computing (HiPC-2000), Bangalore, India, LNCS 1971, pp. 191 - 202

[19] Ernemann C, Hamscher V, Schwiegelshohn U, Streit A, Yahyapour R. On advantages of Grid computing for parallel job scheduling. Proceedings 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002), May 2002; 39–47.

[20] Ernemann C, Hamscher V, Streit A, Yahyapour R. Enhanced algorithms for multisite scheduling. Proceedings 3rd IEEE/ACM International Workshop on Grid Computing (Grid 2002) at Supercomputing 2002 (Lecture Notes in Computer Science, vol. 2536), Baltimore, MD, 2002. Springer: Berlin, 2002; 219–231.

[21] Ernemann C, Hamscher V, Streit A, Yahyapour R. On effects of machine configurations on parallel job scheduling in computational grids. Proceedings of the 6th Workshop on Parallel Systems and Algorithms. VDE-Verlag, 2002; 169–179.

[22] Bucur, A., Epema, D.: The Influence of the Structure and Sizes of Jobs on the Performance of Co-Allocation. In Feitelson, D., Rudolph, L., eds.: 6th Workshop on Job Scheduling Strategies for Parallel Processing. Volume 1911 of LNCS. Springer-Verlag (2000) 154–173

[23] Bucur, A., Epema, D.: The Influence of Communication on the Performance of Co-Allocation. In Feitelson, D., Rudolph, L., eds.: 7th Workshop on Job Scheduling Strategies for Parallel Processing. Volume 2221 of LNCS. Springer-Verlag (2001) 66–86

[24] Bucur, A., Epema, D.: Local versus Global Queues with Processor Co-Allocation in Multicluster Systems. In Feitelson, D., Rudolph, L., Schwiegelshohn, U., eds.: 8th Workshop on Job Scheduling Strategies for Parallel Processing. Volume 2537 of LNCS. Springer-Verlag (2002) 184–204

[25] A.I.D. Bucur and D.H.J. Epema, Priorities among Multiple Queues for Processor Co-Allocation in Multicluster Systems, 36th Annual Simulation Symp., Orlando, Fl., USA, march-april 2003, 15-27, 2003.

[26] A.I.D. Bucur and D.H.J. Epema, The Maximal Utilization of Processor Co-Allocation in Multicluster Systems, Int'l Parallel and Distributed Processing Symp. (IPDPS 2003), Nice, France, april 2003, 60-69, 2003.

[27] A.I.D. Bucur and D.H.J. Epema, The Performance of Processor Co-Allocation in Multicluster Systems, 3rd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid (CCGrid2003), Tokyo, Japan, may 2003, 302-309, 2003.

[28] A.I.D. Bucur and D.H.J. Epema, Trace-Based Simulations of Processor Co-Allocation Policies in Multiclusters, 12th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC-12), Seattle, Wa, USA, june 2003, 70-79, 2003.

[29] S. Banen, A.I.D. Bucur, and D.H.J. Epema, A Measurement-Based Simulation Study of Processor Co-Allocation in Multicluster Systems, Ninth Workshop on Job Scheduling Strategies for Parallel Processing (in conjunction with HPDC-12), D.G. Feitelson, L. Rudolph and U. Schwiegelshohn (eds), Seattle, USA, june 2003, LNCS 2862, 105-128, 2003

[30] J.M.P. Sinaga, H.H. Mohamed, and D.H.J. Epema, A Dynamic Co-Allocation Service in Multicluster Systems, Tenth Workshop on Job Scheduling Strategies for Parallel Processing (in conjunction with Sigmetrics-Performance 2004), D.G. Feitelson, L. Rudolph and U. Schwiegelshohn (eds), New York, USA, June 2004, LNCS 3277, 194-209, 2005.

[31] H.H. Mohamed and D.H.J. Epema, The Design and Implementation of the KOALA Co-Allocating Grid Scheduler, European Grid Conference, Amsterdam, February 2005, LNCS 3470, 640-650, 2005.

[32] A. Goldman and C. Queiroz. A model for parallel job scheduling on dynamical computer grids. Concurrency and Computation: Practice and Experience, Vol. 16, pp. 461-468, March, 2004.

[33] H. Shan, L. Oliker, R. Biswas, Job Superscheduler Architecture and Performance in Computational Grid Environments, SC2003, Phoenix, AZ, Nov 2003.

[34] H. Shan, L. Oliker, W. Smith, R.Biswas, "Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration, 12th International Conference on Advanced Computing and Communication (ADCOM), Ahmedabad, India, Dec 2004.

[35] Q. Snell, M. Clement, D. Jackson, and C. Gregory, "The performance impact of advance reservation meta-scheduling ". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (eds.), pp. 137--153, Springer Verlag, 2000.

[36] U. Schwiegelshohn and R. Yahyapour. Fairness in Parallel Job Scheduling. Journal of Scheduling, 3(5):297-320. John Wiley, 2000.

[37] J.P. Jones and B. Nitzberg, “Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization,” 5th Workshop on Job Scheduling Strategies for Parallel Processing, 1999

[38] J. Skovira, W. Chan, H. Zhou and D. Lifka, “The EASY-Loadleveller API Project,” Proc. 2nd Workshop on Job Scheduling Strategies for Parallel Processing, Honolulu, Apr. 1996, pp. 41- 47. Lecture Notes in Comp. Sci. Vol. 1162, Springer-Verlag.

[39] D. Feitelson, L. Rudolph, U. Schweigelshohn, K. Sevcik, and P. Wong. “Theory and Practice in Parallel Job Scheduling,” 3rd Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag Lecture Notes in Computer Science, Vol. 1291, pp. 1-34, April 1997.