Benefits of Global Grid Computing for Job Scheduling

Carsten Ernemann , Volker Hamscher, Ramin Yahyapour
Computer Engineering Institute
Otto-Hahn-Str. 4, 44221 Dortmund, Germany
Email: {carsten.ernemann, volker.hamscher, ramin.yahyapour}@udo.edu

***Abstract*— In addition to other advantages, computational Grids are considered to utilize the participating compute resources more efficiently as well as to improve the response time for user jobs. Due to the lack of common large scale global Grids and corresponding studies on Grid workloads this assumption is not yet verified. In this paper, the effect of geographical distribution of Grid resources on the machine utilization and the average response time is analyzed. To this end, simulations have been performed. The results show a significant benefit for the job scheduling quality due to the participation in a true global Grid. The average weighted response times of all submitted jobs decrease up to about 30%. The results have been verified using different workloads and Grid configurations.**

I. Introduction

Grids are considered to provide multiple advantages to their participants [11]. Despite the fact that the Grid is not limited on compute resources, the typical application of the Grid is currently still the execution of computational intensive tasks on high-performance computers. The resource providers are compute centers that share their processing powers for dedicated user communities. Besides the advantage of accessing locally unavailable resources, the Grid is usually also expected to utilize the existing resources more efficiently. That is, machines are better utilized and users are getting a better quality of service, for instance, a shorter response time. However, to our knowledge this assumption has not yet been verified. In addition, there have been many discussions whether the Grid can in fact improve the quality of service for the end-user at all as HPC resources are already heavily utilized by the existing local user community. Therefore, it is not clear whether the participation in a Grid and the exchange of workload does improve either utilization or response time.

In our work we focus only on the Grid scheduling aspects and try to examine the impact of global Grid computing on the scheduling quality for computational jobs. In detail, the effect of the geographical distribution of the resources on the machine utilization and the average response time for the user is analyzed. Especially the combination of sharing resources for jobs of independent user communities and the effect of the distribution of the participants in different time zones is examined in this paper. To this end, several simulations have been performed to evaluate these effects. The background of this research is given in the following section. The configurations of the conducted simulations are described in Section III and Section IV. The results are presented and discussed in Section V. Finally, the paper ends with a conclusion in Section VI.

II. Background

As mentioned above, a typical example for a computational Grid is currently the collaboration of high-performance computing sites. The participating compute centers have usually a local user community which gains access to additional remote compute resources by joining the Grid. Such a sharing of computational jobs is a major application of Grids. While Grid technology is becoming more common and several computational Grids have already been established, little is yet known about the current and future user demand in such a Grid environment. However, the workload of several single compute centers is publicly available in the Standard Workload Archive [20] and have been examined in several publications [13], [4], [10], [8]. Moreover, these traces are used for the evaluation of different scheduling strategies for single parallel machines [14], [17], [18], [9] and for Grid research [12], [2], [1]. These workload traces include information about all job submissions on a machine for a certain period of time which usually ranges over several months and several thousands of jobs, see Table I. Typically existing large-scale computers from different compute centers are joint in Grids. In most cases no new resources are bought to be dedicated only for Grid use. Therefore, it is reasonable to start with the available workload information from the compute centers to evaluate the impact of Grids on the scheduling quality. In this paper, we especially examine the implications of the fact that the compute resources as well as the user groups are at different geographical locations and therefore reside in different time zones. Requests for computational resources by users vary depending on the time of day, the weekday etc., see [13], [7]. Therefore, the global aspect of the Grid is expected to have an impact on the mentioned resource utilization and, even more interestingly for the user, on the scheduling quality.

III. Model

In this section the model is introduced on which the evaluated scenarios are based. First a specification of the underlying models of sites, machines and jobs is presented, which is followed by an outline of the scheduling system. A more extensive description of different Grid scheduling strategies can be found in [12], [2], [3], [1].

Workload	CTC	SDSC SP2	LANL	KTH
Begin	Wed, 06/26/1996, 16:06:01	Fri, 10/01/1993, 00:00:04	Tue, 10/04/1994, 07:01:13	Mon, 09/23/1996, 12:00:32
End	Sat, 05/31/1997, 18:43:43	Mon, 10/02/1995, 06:50:33	Tue, 09/24/1996, 05:28:21	Fri, 08/29/1997, 08:34:10
Number of Jobs	79302	67631	122305	28487
Number of nodes	430	128	1024	100
Time Zone	GMT-5	GMT-8	GMT-7	GMT+1

TABLE I
THE ORIGINAL WORKLOADS.

A. Machine Model

Our scenario is based on independent computing sites with existing local workload and management systems. We examine the impact on job scheduling results if the computing sites participate in a grid environment. The sites combine their resources and share incoming jobs. For the sake of simplicity, in this paper we consider a central Grid scheduler that controls all resource allocations. The local schedulers are only responsible for starting the jobs after their allocation by the grid scheduler. Note, that for a real world application a single central Grid scheduler would be a critical limitation. However, implementations are possible, in which site-autonomy is still maintained but the resulting schedule would be the same as created by a central Grid scheduler.

We assume that each participating site in the computational grid has a single MPP (Massive Parallel Processor system), which consists of several nodes. Nodes (resources) are single-processor systems with local memory and hard disk. The nodes are linked with a fast interconnection network that does not favor any specific communication pattern inside the machine [6]. The actual mapping of a job on the nodes is neglected [9]. The machines use space-sharing and run the jobs in an exclusive fashion. Moreover, the jobs are not preempted nor time-sharing is used. Therefore, once started a job runs until completion. Furthermore, a job is cancelled if it exceeds its previously requested allocation time.

For simplification all nodes in this study are identical. The machines at the different sites only differ in the number of nodes. The existence of different resource types would limit the number of suitable machines for a job. Typically a preselection phase takes place ahead of the actual scheduling process and generates a reduced set of suitable resources. In our study we neglect this preselection and focus on the job scheduling process.

B. Job Model

Jobs are submitted by independent users at the local sites. The scheduling problem is an on-line scenario without any knowledge of future job submissions. Our evaluations are restricted to batch jobs, as this job type is dominant on most MPP systems. A job request consists of several parameters as e.g. the requested run time and the requested number of resources. After submission a job requests a fixed number of resources that are necessary for starting the job. This number is not changed during the execution of the job. That is, jobs are not moldable nor malleable [9], [5]. It is the task of the scheduling system to allocate the jobs to resources and to determine the starting time. The jobs are executed without further user interaction.

C. Scheduling System

In our examination of a *job sharing* scenario the scheduling algorithm consists of two steps. In the first step the earliest start time of the allocation on all available machines determined and in the second step a machine is selected .

There are several methods possible for selecting machines. Earlier simulations (presented in [12]) showed the best results for a selection strategy called *BestFit*. In this strategy a particular machine is chosen on which the job leaves the least number of free nodes if the new job is allocated to that machine.

Backfilling is used as the scheduling strategy which has been introduced by Lifka [15] and is often in use on parallel machines. This strategy requires knowledge of the expected job execution time and can be applied to any greedy list schedule. If the next job in the list cannot be started due to a lack of available resources, backfilling tries to find another job in the list which can use the idle resources. But it will not postpone the execution of any next jobs in the list. In addition *EASY backfilling* has been used which is an extension to normal backfilling where only the next job is not delayed [19].

Note, that this scenario focuses on the scheduling impact for a job sharing strategy only. That is, jobs are distributed in the Grid but run on a single HPC machine only. That is, input data is only considered to be transferred to the machine before the job execution while output data of the job is transferred back afterwards. This network communication is neglected in our evaluation as this latency can usually be hidden in pre- and post-fetching phases without regards to the actual job execution phase. As shown later in this paper, the average wait time of a job before its execution is at least several minutes. Assuming that HPC resources are usually well-connected, the data overhead will have no impact on most jobs.

IV. EXPERIMENTS

Several simulations with different resource configurations and workload traces have been executed which will be presented in the following sections.

A. Workload Traces

As mentioned earlier, our analysis is based on real workload traces. The used traces and the corresponding information about the resource configuration can be found in Table I.

It is well known that the demand in compute power is not equally distributed over the time of day [16]. The Figure 1 details the time distribution of the CTC-Trace. The resource consumption is unbalanced and has an absolute maximum around noon and an absolute minimum at around 3 am. The other workloads show a similar behavior, but with shifted peaks. While the workloads of KTH, LANL and CTC have their peak utilization during daytime, the peak for SDSC is shifted towards 5 am. All times are given as local time.

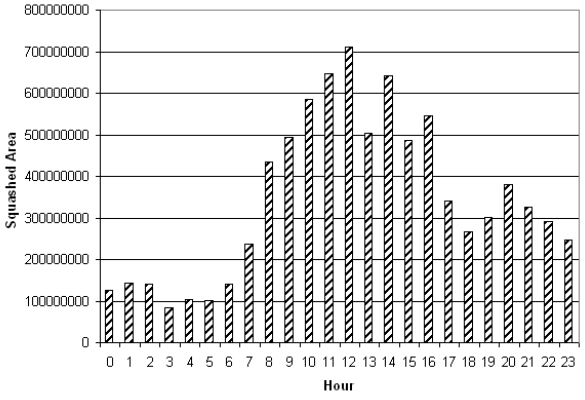

Fig. 1. Aggregated Resource Consumption (Squashed Area) over the Time of Day for all CTC Jobs.

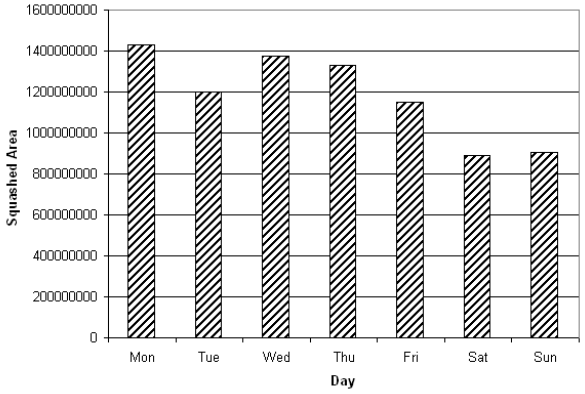

Fig. 2. Aggregated Resource Consumption (Squashed Area) over each Weekday for all CTC Jobs.

Additionally the total resource consumption varies with the day of the week, as shown in Figure 2 for the CTC-workload. This reasons the mentioned synchronization of the used workloads, as shifting the workloads by a day or more would most likely have an impact on the scheduling quality.

As the available workload traces do not cover the same period of time nor start at the same day of week or time of day, modifications are necessary. In our simulation, the workload traces are combined for modeling the workload of a global Grid. Therefore, we transformed the workloads to the effect that all traces begin at the same weekday and at the same time of day. To this end, we removed all jobs until the first Monday at midnight. Note, that the alignment is related to the local time and so the time differences corresponding to the given time zones are maintained. Note also that this modification results in a loss of jobs within each workload. In consideration of the size of the workload traces only marginal number of jobs is affected. The modified number of jobs is presented in Table II.

Workload Name	CTC	SDSC	LANL	KTH
Number of Jobs	79272	67503	120870	28481
% of original	99.99	99.81	98.83	99.98

TABLE II
THE MODIFIED WORKLOADS.

B. Resource Configurations

After introducing the used workloads, the corresponding resource configurations and assumptions will be explained next. We have chosen 3 basic configurations. The first configuration consists of resources in 4 different time zones. The next configuration includes machines within 6 different time zones. In both configurations the examined time zones have been equally distributed. That is, in configuration 1 the time difference between two adjacent resources is 6 hours whereas the difference in configuration 2 is 4 hours.

The last configuration consists of all original machines. So this is the most realistic case as the simulation indicates how the scheduling result is improving dependent on the time zone. For the last resource configuration three different assumptions were used for the simulation. The first variant produces the scheduling result if all machines are in the same time zone. During the second simulation it is assumed that all machines are equally distributed. The last simulation uses the original time zones. Note, that for the third configuration all workloads are cut to span only 11 month caused by the smallest workload trace. This is necessary to assure that each site creates sufficient workload for the whole simulation. In addition, for each configuration scenario a simulation without job sharing has been executed ("*Trace*_single") which is used as a reference. All considered simulation scenarios are summarized in Table III.

In order to compare the simulation results between the first two underlying configurations we have chosen a constant number of 6 machines. Therefore there are two machines located at site 1 and 3 respectively. Both of these two configurations were simulated with all workloads (Table II). Note, that in each case the whole simulation process starts at midnight on Monday for the machine at site 1. The machine at site 2 starts either 4 or 6 hours later.

Note that due to the different offset in time for the first jobs, the startup phase of the schedule is degenerated. While at the

end of the simulation the variance of the schedule length for different sites is considered for the calculation of the metrics, see [1]. Nevertheless the whole amount of workload is rather large in comparison to these marginal effects.

Name	# of Sites	Site Configurations (Workload/Machine,Time Zone)
CTC_single	6	(CTC,0h),(CTC,0h),(CTC,0h), (CTC,0h),(CTC,0h),(CTC,0h)
SDSC_single	6	(SDSC,0h),(SDSC,0h),(SDSC,0h), (SDSC,0h),(SDSC,0h),(SDSC,0h)
KTH_single	6	(KTH,0h),(KTH,0h),(KTH,0h), (KTH,0h),(KTH,0h),(KTH,0h)
LANL_single	6	(LANL,0h),(LANL,0h),(LANL,0h), (LANL,0h),(LANL,0h),(LANL,0h)
CTC_TZ_1	6	(CTC,0h),(CTC,0h),(CTC,0h), (CTC,0h),(CTC,0h),(CTC,0h)
SDSC_TZ_1	6	(SDSC,0h),(SDSC,0h),(SDSC,0h), (SDSC,0h),(SDSC,0h),(SDSC,0h)
KTH_TZ_1	6	(KTH,0h),(KTH,0h),(KTH,0h), (KTH,0h),(KTH,0h),(KTH,0h)
LANL_TZ_1	6	(LANL,0h),(LANL,0h),(LANL,0h), (LANL,0h),(LANL,0h),(LANL,0h)
CTC_TZ_4	6	(CTC,0h),(CTC,0h), (CTC,+6h),(CTC,+12h), (CTC,+12h),(CTC,+18h)
SDSC_TZ_4	6	(SDSC,0h),(SDSC,0h), (SDSC,+6h),(SDSC,+12h), (SDSC,+12h),(SDSC,+18h)
KTH_TZ_4	6	(KTH,0h),(KTH,0h), (KTH,+6h),(KTH,+12h), (KTH,+12h),(KTH,+18h)
LANL_TZ_4	6	(LANL,0h),(LANL,0h), (LANL,+6h),(LANL,12h), (LANL,+12h),(LANL,+18h)
CTC_TZ_6	6	(CTC,0h),(CTC,+4h), (CTC,+8h),(CTC,+12h), (CTC,+16h),(CTC,+20h)
SDSC_TZ_6	6	(SDSC,0h),(SDSC,+4h), (SDSC,+8h),(SDSC,+12h), (SDSC,+16h),(SDSC,+20h)
KTH_TZ_6	6	(KTH,0h),(KTH,+4h), (KTH,+8h),(KTH,+12h), (KTH,+16h),(KTH,+20h)
LANL_TZ_6	6	(LANL,0h),(LANL,+4h), (LANL,+8h),(LANL,+12h), (LANL,+16h),(LANL,+20h)
ALL_single	4	(CTC,0h),(SDSC,0h), (LANL,0h),(KTH,0h)
ALL_TZ_1	4	(CTC,0h),(SDSC,0h), (LANL,0h),(KTH,0h)
ALL_TZ_4	4	(KTH,0h),(CTC,+6h), (LANL,+12h),(SDSC,+18h)
ALL_org	4	(KTH,0h),(CTC,+6h), (LANL,+8h),(SDSC,+9h)

TABLE III
SIMULATED CONFIGURATIONS.

V. EVALUATION

One measure for the schedule quality is the average response time weighted by the job's resource consumption (**AWRT**) [17]. We define it as follows:

$$\mathrm{AWRT} = \frac{\sum_{j \in \mathrm{Jobs}} (\mathrm{SA}_j \cdot (\mathrm{endTime}_j - \mathrm{submitTime}_j))}{\mathrm{Total\_SA}} \quad (1)$$

The resource consumption of a job, called squashed area (**SA**), can be described as the product of the job's run time and the number of requested resources. Therefore large jobs have a larger resource consumption than smaller jobs. The squashed area of a single job j and the total squashed area can be defined as follows:

$$\mathrm{SA}_j = \mathrm{reqResources}_j \cdot (\mathrm{endTime}_j - \mathrm{startTime}_j) \quad (2)$$

$$\mathrm{Total\_SA} = \sum_{j \in \mathrm{Jobs}} \mathrm{SA}_j \quad (3)$$

The response time of a job is weighted by its resource consumption to prevent that smaller jobs have a relatively larger impact on the metric than jobs with a higher resource consumption.

Another criterion for the schedule quality is the slow down (**SD**):

$$\mathrm{SD} = \frac{(\mathrm{endTime}_j - \mathrm{submitTime}_j)}{\mathrm{runtime}_j} \quad (4)$$

Correspondingly, the average weighted slow down is defined as:

$$\mathrm{AWSD} = \frac{\sum_{j \in \mathrm{Jobs}} \mathrm{SA}_j \cdot \mathrm{SD}_j}{\mathrm{Total\_SA}} \quad (5)$$

From the user's point of view the quality of the schedule can be evaluated considering the wait time:

$$\mathrm{WT}_j = \mathrm{startTime}_j - \mathrm{submitTime}_j \quad (6)$$

We define the average weighted wait time (**AWWT**) as follows:

$$\mathrm{AWWT} = \frac{\sum_{j \in \mathrm{Jobs}} (\mathrm{SA}_j \cdot \mathrm{WT}_j)}{\mathrm{Total\_SA}} \quad (7)$$

The simulation results for the CTC workload are presented in Figure 3. The comparison of the three scenarios illustrates the impact of the different temporal offsets on the AWRT. The scenario with 4 sites in 4 time zones with a difference in time of 6 hours between neighbors shows an improvement with a decrease of about 14%.

As shown in Figure 3, the EASY backfilling strategy yields better results than backfilling in almost all cases. Therefore, we limit the following results on the EASY backfilling strategy.

In Table IV the AWRTs are displayed for the 4 uniform workloads.

Trace	_single	_1_TZ	_4_TZ	_6_TZ
CTC	100%	95.6%	85.4%	85.3%
KTH	100%	89.1%	79.2%	78.7%
LANL	100%	87.1%	74.0%	73.2%
SDSC	100%	74.6%	69.8%	70.2%

TABLE IV
COMPARISON OF AWRTS USING EASY BACKFILLING.

Workload	Utilization in%	AWWT in s	AWRT in s	AWSD
CTC_single	55.29	13470	52977	2.0571
CTC_1_TZ	66.34	11127	50634	1.8112
CTC_4_TZ	66.20	5744	45251	1.3463
CTC_6_TZ	66.18	5678	45184	1.3123
KTH_single	58.17	24232	75153	3.2606
KTH_1_TZ	69.80	16074	66995	2.4119
KTH_4_TZ	69.64	8597	59519	1.6676
KTH_6_TZ	69.62	8207	59129	1.605
LANL_single	42.48	2461	11802	1.7069
LANL_1_TZ	55.72	1785	10287	1.4807
LANL_4_TZ	55.67	236	8739	1.0476
LANL_6_TZ	55.66	141	8644	1.0273
SDSC_single	68.87	75940	112929	6.4896
SDSC_1_TZ	82.64	47231	84220	4.5284
SDSC_4_TZ	82.59	41840	78829	3.7995
SDSC_6_TZ	82.58	42334	79323	3.7064

TABLE V
COMPLETE RESULTS.

The decrease of the AWRT varies depending on the number of time zones and the workload between 5% and 30%. In case of the SDSC workload this means that the computing results are more than 9 hours earlier available in the weighted average, see Table V. In general, the participation in the Grid yields an advantage without regard whether the machines are globally distributed or not. However, in all cases in which the sites are distributed over several different time zones, the response time is reduced furthermore.

As pointed out before, the CTC workload has its peak utilization around noon and the minimum utilization at 3 am. Placing the workload in different time zones leads to a combination of high utilization at one site and low utilization at another site. Figure 4 shows the correlation for a scenario with 4 different time zones, each with an offset of +6 hours to the next assigned time zone. This example shows how the shifted workload distribution over different time zones yields an load balancing effect with subsequent reduction of the AWRT.

In addition, scenarios with combinations of different workloads have been examined. Table VI shows the results of the

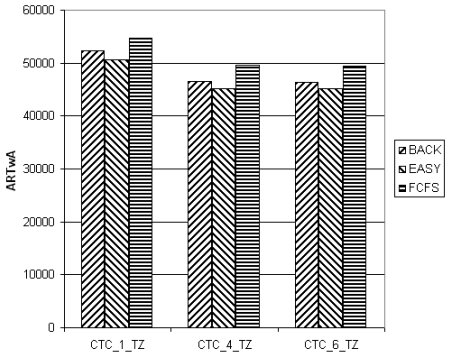

Fig. 3. AWRT for the CTC based scenarios (CTC).

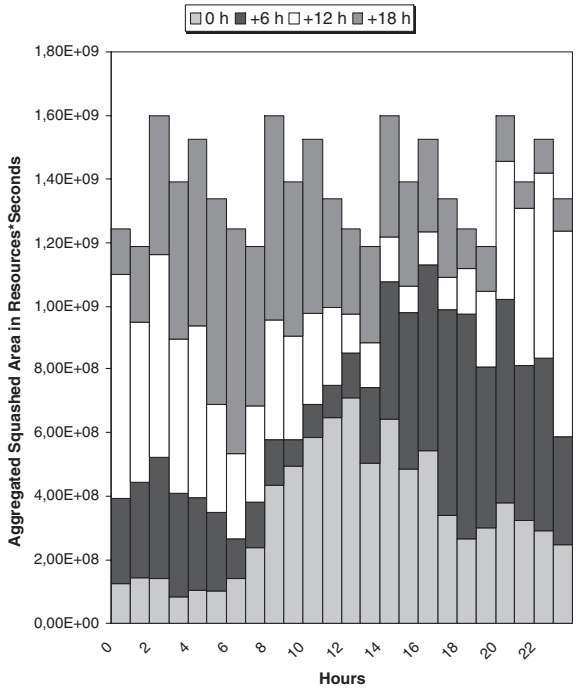

Fig. 4. Aggregated SA for different Time Zones (CTC_4_TZ).

scenarios in which all sites are either located in a single time zone (ALL_1_TZ), or artificially distributed over 4 time zones (ALL_4_TZ), or at their location in reality (ALL_org). For reference, the results are given for participating in a Grid with job sharing or without (single).

First, it can be seen in Table VI that the configuration with the sites located on their original location yield better results than the artificial equal distribution in 4 time zones. However, best results for our example have been achieved with all machines located in the same time zone. This can be explained by the different workload distribution as mentioned before. The SDSC workload has the highest resource demand around 5 am and its lowest resource consumption in the afternoon around 5 pm, see Figure 5. This leads to load balancing effects with the sites CTC and LANL which have their peaks around noon. This causes an additional positive effect on the overall scheduling quality.

Workload	Util. in %	AWWT in s	AWRT in s	AWSD
ALL_1_TZ	60,83	2537	25799	1,484
ALL_1_TZ_single	55,13	8088	29727	2,015
ALL_4_TZ	60,70	3132	26399	1,654
ALL_4_TZ_single	55,03	8096	29746	2,015
ALL_org	60,71	2764	26032	1,528
ALL_org_single	55,03	8095	29743	2,015

TABLE VI
RESULTS WITH AND WITHOUT JOB-SHARING.

Comparing the AWWT of the ALL_org scenario with the ALL_org_single without job sharing, proves the advantage of

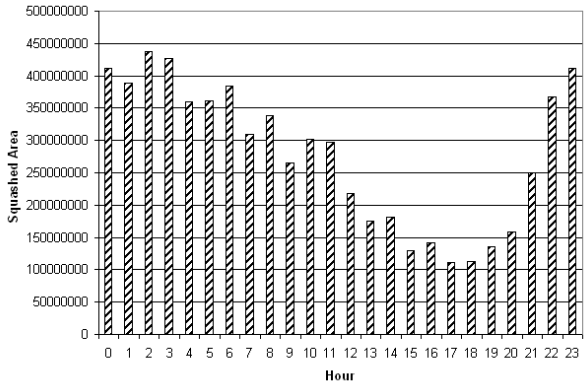

Fig. 5. SA for all SDSC Jobs.

a Grid using job sharing. The AWWT decreases by about 66% for the EASY backfilling algorithm, as shown in Table VI.

Examinations of the resource utilization at the sites showed that the overall utilization is not significantly changed in the configurations with single workloads and homogeneous machines. This indicates that the load balancing effects are caused by under-utilized sites which are unproportionately responsible for the overall improvements in the AWRTs.

However, in the third scenario with combinations of the different workloads there is an increase of utilization from about 55% to 60%. In this case, the different utilizations of the participating sites are balanced.

VI. Conclusion

Simulations have been performed to evaluate the effects of a global Grid constituted by different compute centers which are located in different time zones. In our work we focused on the Grid scheduling aspects and therein showed for the first time that the application of a simple job sharing strategy yields a significant overall advantage. We were able to quantify the effect on the response time. Moreover, the impact of the global distribution of resources being in different time zones has been proven to be an important factor. This shows nicely that a truly global Grid is favorable instead of building several closed Grids in single countries as it is currently implemented by many initiatives. In our examined scenarios, the average weighted response time of all submitted jobs decrease up to about 30%. The results have been verified in several configurations using different workloads and time zone distributions. The presented quantitative examinations highly motivate the participation in a global Grid.

References

[1] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Enhanced Algorithms for Multi-Site Scheduling. In *Proceedings of the 3rd International Workshop on Grid Computing, Baltimore*. Springer–Verlag, Lecture Notes in Computer Science LNCS, 2002.

[2] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. On Advantages of Grid Computing for Parallel Job Scheduling. In *Proc. 2nd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid (CCGRID2002)*, Berlin, May 2002. IEEE Press.

[3] C. Ernemann, V. Hamscher, A. Streit, and R. Yahyapour. On Effects of Machine Configurations on Parallel Job Scheduling in Computational Grids. In *International Conference on Architecture of Computing Systems, ARCS*, pages 169–179, Karlsruhe, April 2002. VDE.

[4] C. Ernemann, B. Song, and R. Yahyapour. Scaling of Workload Traces. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003 Seattle, WA, USA, June 24, 2003*, volume 2862 of *Lecture Notes in Computer Science (LNCS)*, pages 166–183. Springer-Verlag Heidelberg, October 2003.

[5] D. Feitelson. A Survey of Scheduling in Multiprogrammed Parallel Systems. Research report rc 19790 (87657), IBM T.J. Watson Research Center, Yorktown Heights, NY, Feb. 1995.

[6] D. Feitelson and L. Rudolph. Parallel Job Scheduling: Issues and Approaches. In D. Feitelson and L. Rudolph, editors, *IPPS'95 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 1–18. Springer–Verlag, Lecture Notes in Computer Science LNCS 949, 1995.

[7] D. G. Feitelson. Workload Modeling for Performance Evaluation. In M. C. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, pages 114–141. Springer Verlag, 2002. Lect. Notes Comput. Sci. vol. 2459.

[8] D. G. Feitelson and B. Nitzberg. Job Characteristics of a Production Parallel Scientific Workload on the NASA Ames iPSC/860. In D. G. Feitelson and L. Rudolph, editors, *IPPS'95 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 337–360. Springer, Berlin, Lecture Notes in Computer Science LNCS 949, 1995.

[9] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, and K. C. Sevcik. Theory and Practice in Parallel Job Scheduling. *Lecture Notes in Computer Science*, 1291:1–34, 1997.

[10] D. G. Feitelson and A. M. Weil. Utilization and Predictability in Scheduling the IBM SP2 with Backfilling. In *Proc. 12th Int'l Parallel Processing Symp. and the 9th Symp. on Parallel and Distributed Processing*, pages 542–547, Los Alamitos CA, 1998. IEEE Computer Society Press.

[11] I. Foster and C. Kesselman, editors. *The GRID: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[12] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Evaluation of Job-Scheduling Strategies for Grid Computing. In *Proc. 7th Int'l Conf. on High Performance Computing, HiPC-2000*, pages 191–202, Bangalore, Indien, 2000. Springer, Berlin, Lecture Notes in Computer Science LNCS 1971.

[13] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riordan. Modeling of Workload in MPPs. In D. Feitelson and L. Rudolph, editors, *IPPS'97 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 94–116. Springer–Verlag, Lecture Notes in Computer Science LNCS 1291, 1997.

[14] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. On the Design and Evaluation of Job Scheduling Systems. In D. G. Feitelson and L. Rudolph, editors, *IPPS/SPDP'99 Workshop: Job Scheduling Strategies for Parallel Processing*. Springer, Berlin, Lecture Notes in Computer Science, LNCS 1659, 1999.

[15] D. A. Lifka. The ANL/IBM SP Scheduling System. In D. G. Feitelson and L. Rudolph, editors, *IPPS'95 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 295–303. Springer, Berlin, Lecture Notes in Computer Science LNCS 949, 1995.

[16] U. Lublin and D. G. Feitelson. The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. Technical Report 2001-12, Hebrew University, Oct 2001.

[17] U. Schwiegelshohn and R. Yahyapour. Analysis of First-Come-First-Serve Parallel Job Scheduling. In *Proceedings of the 9^{th} SIAM Symposium on Discrete Algorithms*, pages 629–638, January 1998.

[18] U. Schwiegelshohn and R. Yahyapour. Improving First-Come-First-Serve Job Scheduling by Gang Scheduling. In D. Feitelson and L. Rudolph, editors, *IPPS'98 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 180–198. Springer–Verlag, Lecture Notes in Computer Science LNCS 1459, 1998.

[19] J. Skovira, W. Chan, H. Zhou, and D. Lifka. The EASY — LoadLeveler API Project. *Lecture Notes in Computer Science*, 1162:41–47, 1996.

[20] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, Juni 2004.