Enhancing Prediction on Non-dedicated Clusters*

Joseph Ll. Lérida[1], F. Solsona[1], F. Giné[1], J.R. García[2],
M. Hanzich[2], and P. Hernández[2]

[1] Departamento de Informática e Ingeniería Industrial, Universitat de Lleida, Spain
{jlerida,francesc,sisco}@diei.udl.cat
[2] Departamento de Arquitectura y Sistemas Operativos,
Universitat Autònoma de Barcelona, Spain
{jrgarcia,mauricio,porfidio.hernandez}@aomail.uab.es

Abstract. In this paper, we present a scheduling scheme to estimate the turnaround time of parallel jobs on a heterogeneous and non-dedicated cluster or NoW(Network of Workstations). This scheme is based on an analytical prediction model that establishes the processing and communication slowdown of the execution times of the jobs based on the cluster nodes and links powerful and occupancy. Preservation of the local application responsiveness is also a goal.

We address the impact of inaccuracies in these estimates on the overall system performance. Furthermore, we demonstrate that job scheduling benefits from the accuracy of these estimates. The applicability of our proposal has been proved by measuring the efficiency of our method by comparing the predicted deviations of the parallel jobs in a real environment with respect to the most representative ones of the literature.

The additional cost of obtaining these was also evaluated and compared. The present work is implemented within the CISNE project, a previously developed scheduling framework for non-dedicated and heterogeneous cluster environments.

1 Introduction

Several studies [1] have revealed that a high percentage of the computing resources in NoWs and Clusters environments are idle. The possibility of using this computing power to execute parallel jobs without perturbing the performance of the local users applications on each workstation has led to a proposal for new job schedulers [6, 7, 15].

With the aim of taking advantage of the idle computing resources available across the cluster, in a previous work [6], we developed a new scheduling environment, named CISNE. Entering jobs wait in an input queue to be scheduled and mapped into the cluster. The Scheduler supports backfilling and mapping techniques based on prediction of the turnaround time of parallel jobs. The prediction engine is a simulation tool, integrated into the CISNE scheduling system. Estimation of the future state of the NoW/Cluster is a critical component of our scheduling environment. It was also so in other well-known systems, as in [2, 3, 12, 13].

The first step to estimate the future cluster state is to predict the job run times. In this sense, there are a large number of proposals. The most widely used method is based on a

* This work was supported by the MEyC-Spain under contract TIN2007-64974.

historical system that records the past executions of an application [4, 5, 12]. The second alternative is to use a *simulation system* based on *analytical models*, which by means of the characterization of the environment and the workloads, allows a future state to be estimated [6, 8, 9]. Finally, another typical approach is the combination of *a simulation system* and a *historical scheme.* In this case, the simulation engine approximates the future cluster state based on the search for similarities in previous cluster states stored in the historical database, thus generally obtaining more accurate estimations [3, 13, 14].

Most of the prediction studies only consider the state of the processing resources without insight into job communication. This also applies in [2, 6, 16]. In [6], Hanzich presented a new analytical model that only considered the number of parallel jobs mapped into a non-dedicated cluster in order to estimate the runtime of a parallel job. Additionally, communication issues where not considered. We extend this model by considering the resource heterogeneity and a real measurement of the parallel and local workload.

In [11], Jones presented an application bandwidth-aware model that measures the impact of the saturation of communication links on the runtime of a parallel workload. Recently, Jones in [10], addressed the impact of inaccuracies in user-provided communication requirement estimates on overall system performance. He showed that underestimating the bandwidth requirements specified in the job arguments, gives worse overall system performance than overestimation due to the fact that network saturation occurs when this is the case. In our proposal we also takes communication into account. We add the Jones model to our scheme, and even extend it by also considering the effect of the local applications communications.

In this paper, we present a new analytical model that predicts the parallel job runtime by considering the future non-dedicated and heterogeneous cluster state of the processing and communicating resources, and models the impact of the local activity on the prediction of the parallel job runtime. Preservation of local workload responsiveness is also a goal. Together these are the contributions of this work. We evaluated the effectiveness of our proposals in relation to various estimation techniques from the literature. In general, in-depth experimentation demonstrates the rigorous and low time-costly estimations without excessively damaging the local workload of our proposal.

The outline of this paper is as follows. Section 2 describes the cluster-scheduling environment used in the present work and our proposal for a new estimation model. Next, we describe the experimental environment and analyze the results in section 3. Finally, the conclusions and future work are explained in section 4.

2 Prediction Model for Non-dedicated Clusters

The CISNE system [6] is a scheduling environment for non-dedicated and heterogeneous clusters. CISNE ensures benefits for the parallel applications, while preserving the user task responsiveness. Fig. 1 shows the architecture of CISNE.

Entering jobs wait to be scheduled and mapped by the *Scheduler* in the *Input Queue.* The *Scheduler* is made up of the *Job* and *Node selection* modules. *Job selection* supports two different policies, namely *FCFS* (First-Come-First-Served) and *Backfilling*. *Backfilling* [3] consists of selecting the job in the queue that does not delay the start of

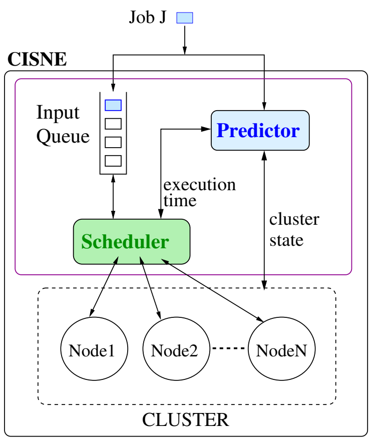


Fig. 1. CISNE architecture

the first job in the Input Queue. To do this, accurate estimations of the job turnaround times must be performed. We implemented two node selection policies, *Normal* and *Uniform*. The *Normal* policy selects the nodes by taking into account only their load. The *Uniform* policy merges CPU and I/O bound parallel applications in the same node as much as possible, with the restriction that nodes should also be evenly loaded.

Estimations of the job turnaround times are performed by the simulator engine (*Predictor*), according to job's priorities (order in the queue), the computational requirements of each parallel job and the cluster *state*. The cluster *state* includes the occupancy of the Memory, CPU and communication links of each cluster resource. The assignment is static, that is, once the job is mapped into a particular set of nodes, no more re-allocations are performed.

The *Predictor* simulates the execution of a parallel job in a non-dedicated cluster. The simulator models a cluster as a collection of heterogeneous and non-dedicated computational resources and allows us to configure several scheduling mechanisms. The input arguments of the simulation engine are the computational resources required by parallel jobs as well as the cluster state and returns the estimated turnaround time. In the present work the *Predictor* was extended with a new analytical model and modified to take more information about the resource nature and their usage into account.

We are interested in arbitrary sized clusters made up of heterogeneous nodes interconnected by means of an internal switch. Formally, a Cluster C is made up by β nodes, this is C=$\{N_1..N_\beta\}$, and a set of β cluster links $\mathscr{L}$, $\mathscr{L}$=$\{L_1..L_\beta\}$, where L_i denotes the cluster link between node N_i and the switch in Cluster C.

Jobs are not supposed to be malleable. That is, they require the same number of processors (one per task) during their lifetime. The computing and communicating requirements of every job task are very similar. Considering this, we define the job execution time, T^e, as follows:

$$T^e = T^p \cdot SP + T^c \cdot SC, \tag{1}$$

where T^p and T^c are the processing and communicating times measured in a dedicated environment. In a real situation, due to the heterogeneity and the non-dedicated property of the resources, T^p and T^c may be lengthened by the *SP* and *SC*, the processing and communication slowdowns respectively. We propose a means for obtaining the SP and SC, which are explained below separately.

2.1 Obtaining SP

In a heterogeneous and non-dedicated environment, the processing capabilities of the constituent nodes depend on its computing power and availability.

We define the *relative computing Power* (P^i) of the node *i*, as the computing power ratio of such a node with regard to the most powerful node in the Cluster. The P^i range is $0 < P^i \leq 1$. $P^i = 1$ means that node *i* is the most powerful node in the Cluster. The *relative computing Power* of each node can be obtained by averaging samples obtained with different applications.

The *Availability* of the node i (A^i), is defined as the percentage of CPU idleness. $A^i \simeq 0$ when 100% of the CPU is occupied and $0 < A^i \leq 1$ otherwise. We define the *Effective computing power* of node *i* (Γ^i) as the product between the *relative computing Power* and the *Availability* of such a node. Formally, Γ^i is defined as follows:

$$\Gamma^i = P^i \cdot A^i, \tag{2}$$

where $0 \leq \Gamma^i \leq 1$. $\Gamma^i = 0$ means that node *i* is unable to execute any task. $\Gamma^i = 1$ tells us that node *i* can execute the task at full speed. Therefore, the processing slowdown of such a node (SP^i) is inversely proportional to its *Effective computing power*, $SP^i = (\Gamma^i)^{-1}$.

As we assume that each job task in our model is very similar in size and they are executing separately, the job execution time is defined as the elapsed execution time of the slowest task. Accordingly, we define the processing time slowdown (*SP*) as the maximum obtained by every forming job task. Formally:

$$SP = max\{SP^i,\ 1 \leq i \leq \rho \leq \beta\}, \tag{3}$$

where ρ is the number of tasks (nodes) making up the job.

2.2 Obtaining SC

Communication characterization is based on the model described by Jones in [11] for homogeneous and dedicated environments. The communication slowdown of each parallel job (*SC*) is defined as the maximum relation between its required bandwidth (*BW*) and the bandwidth finally assigned to the job by the communication system (BW^{as}). BW^{as} is the bandwidth assigned to the job in each mapped link as a consequence of the bandwidth adjustment to the most saturated link. Formally:

$$SC = \left(\frac{BW}{BW^{as}}\right) \tag{4}$$

$SC > 1$ means that the job will suffer delays due to saturated links. When $SC = 1$, the parallel application can communicate at full speed.

2.3 Prediction Engine

When a new parallel job enters the system, or the scheduler needs to know the remaining execution time of any existing parallel job in order to take better scheduling decisions, the simulation engine is activated in order to obtain the remaining execution time of such a job (*Target*). The simulation process is detailed in Algorithm 1. First, the simulator is updated with the current cluster state: the *Input Queue*, the *Target*, the *Running List* and for each cluster node i its *Effective computing power* (Γ^i, defined in section 2.1) and its *link bandwidth requirements*. Next, the simulation engine estimates the execution of all the previous jobs in the system until *Target* was executed. Finally, the remaining execution time $T^e_{rem}(Target)$ of *Target* is returned.

Algorithm 1. Simulation process.

```
1: Input arguments: Input Queue (IQ), Running List (RL, List of executing jobs), $Target$, $\Gamma^i$,
   Cluster (C), $Job\_Selection\_Policy$ and $Node\_Selection\_Policy$.
2: Set $t$ to the simulation start-time. $J = \phi$.
3: while (($RL \neq \{\phi\}$) and ($J \neq Target$)) do
4:    forall ($j \in RL$) Estimate the execution remaining time $T^e_{rem}(j)$, at time $t$.
5:    Select the next job $J = min\{T^e_{rem}(j)\}$ to finish. Let $(t+\Delta t)$ the end time of $J$.
6:    Remove $J$ from $RL$.
7:    forall ($j \in RL$) Obtain its $T^p_{rem}$ and $T^c_{rem}$ at time $(t+\Delta t)$.
8:    forall ($j \in IQ$) Increase the estimated $Waiting\_Time(j)$ in $\Delta t$ time units.
9:    while (($IQ \neq \{\phi\}$) and (available resources in $C$)) do
10:       Select the next job $k \in IQ$ by using the $Job\_Selection\_Policy$.
11:       Select the $C$ nodes to map $k$ by using the $Node\_Selection\_Policy$.
12:       Remove $k$ from $IQ$. Insert $k$ into $RL$.
13:    end while
14:    Set $t$ to $t+\Delta t$.
15: end while
16: return $T^e_{rem}(J)$      // $J = Target$
```

In each simulation step, the simulation engine estimates the execution remaining time of each job in the *RL* by means of the equation 5 (line 4). Let t be current simulation time. The estimated remaining execution time of a job J at time t, $T^e_{rem}(J)$, depends on the remaining processing and communicating times, T^p_{rem} and T^c_{rem}, as well as the current processing and communication slowdown, *SP* and *SC*. This gives,

$$T^e_{rem}(J) = T^p_{rem} \cdot SP + T^c_{rem} \cdot SC, \tag{5}$$

Next, the job J with the minimum estimated remaining time $(min\{T^e_{rem}(J)\})$ was selected (line 5). Let $(t+\Delta t)$ be the time at which the selected job J ends. Then, the remaining processing and communicating times of the rest of jobs in the *RL* should be calculated at time $(t+\Delta t)$ by means the equation 6 (line 7).

The remaining processing time at time $(t+\Delta t)$, T^p_{rem}, is calculated by means of the difference between the remaining processing time at time t, T'^p_{rem}, and the processing time spent during Δt time units with its corresponding slowdown SP', that is $\frac{T^p_{\Delta t}}{SP'}$. This is also true for the remaining communicating time T^c_{rem}.

$$T^p_{rem} = T'^p_{rem} - \frac{T^p_{\Delta t}}{SP'}, \; T^c_{rem} = T'^c_{rem} - \frac{T^c_{\Delta t}}{SC'} \tag{6}$$

The algorithm also obtains the estimated waiting time ($Waiting\_Time(j)$, line 8) of the jobs in the *IQ*. This metric will be very useful in the experimentation to understand the behavior of our proposal. If there exist jobs in the *IQ* and available resources, the simulator tries to schedule the next job based on the configured *Job_Selection_Policy* (lines 9..13). Finally, $T^e_{rem}(Target)$ is returned.

3 Experimentation

Our proposal was evaluated in a real non-dedicated cluster made up of sixteen 3-GHz uniprocessor workstations with 1GB of RAM, interconnected by a 1-Gigabit network. We studied the performance of our method with respect to various scheduling strategies and different processing and communication load levels. Likewise, we compared the accuracy and the cost/complexity of our proposal with other estimation methods described in the literature.

In order to carry out the experimentation process, the local and parallel workloads must be defined. The local workload was represented by a synthetic benchmark (named *local_bench*) that can emulate the usage of 3 different resources: CPU, Memory and Network traffic. The use of these resources was parameterized in a real way. According to the samples collected over a couple of weeks in an open laboratory, *local_bench* was modelled to use 15% CPU, 35% Memory and a 0,5KB/sec LAN, in each node where it was executed.

The parallel workload was a mixture of PVM and MPI applications, with a total of 90 NAS parallel jobs (CG, IS, MG, BT), with a size of 2, 4 or 8 tasks (class A and B), which reached the system under a Poisson distribution. These jobs were merged so that the entire parallel workload had a balanced requirement for computation and communication.

This workload was executed with different combinations of job and node selection policies. More specifically, we combined both *FCFS* and *Backfilling* (*BF*) job selection policies with both *Normal* and *Uniform* node selection policies. In order to compare the accuracy of the different estimation methods of the literature, we tested all the estimation methods by varying the MPL (Maximum Parallelism Level) upper bound between 1..4.

3.1 Experimental Results

First, we evaluate the performance of our analytical proposal for predicting the turnaround time (named from here on *SDPN*: SlowDown-Processing-Networking), under different MPL upper bound values. In Figure 2, *SDPN* is compared with the methods proposed by *Li* [13], *Smith* [14], *Hanzich* [6] and *Historical* [4, 5, 12].

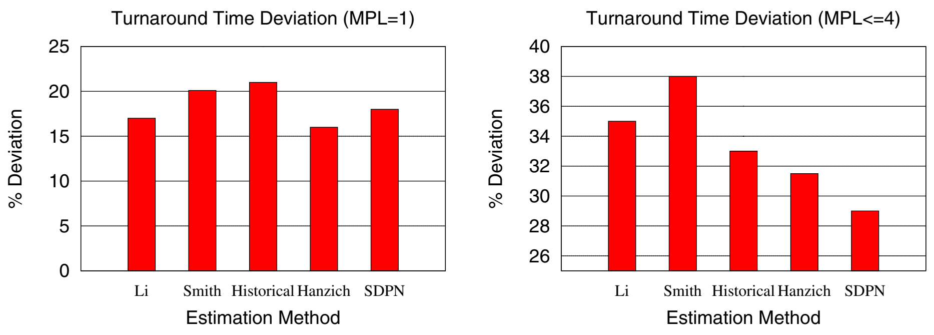


Fig. 2. Turnaround deviation. (left) MPL=1 (right) MPL≤4.

Li and Smith take scheduling decisions by discrete event simulators jointly with the help of a historical database, which only stores information about input parameters and execution time of the previously executed jobs. The *Historical* method tries to infer the future cluster state from a database as a simple replication of the past. *Hanzich* used an analytical model that penalizes the remaining time of the parallel jobs by considering only the degree of parallelism (*MPL*) of the nodes being executed.

The experiments in figure 2 was performed without communication saturation, combining both job selection (FCFS and Backfilling) and node selection (Normal and Uniform) policies. Figure 2 (left) shows the average deviation in estimating the turnaround time of the parallel jobs with a MPL=1 restriction (only one parallel job per node was allowed). The worst behavior belongs for the *Historical* methods. This is so because the waiting queue length increases due to the MPL restriction and the historical repository does not contain enough information to follow the evolution of a job in the waiting queue. Only execution times can be more precisely matched in the database. It can be seen that our proposal, *SDPN*, is slightly worse than *Hanzich*. This means that in dedicated environments the simplest model behaves better. No more sophisticated mechanisms to estimate the future load state, as *SDPN*, are required.

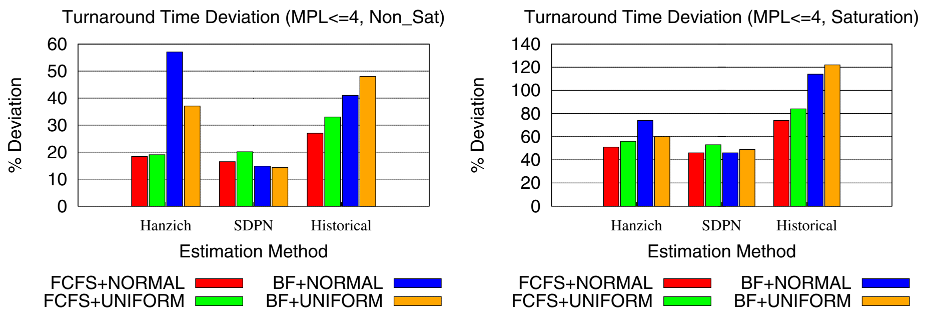
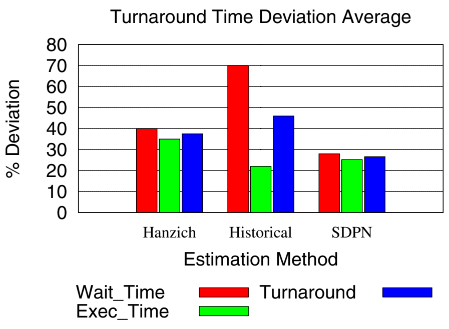


Fig. 3. Turnaround deviation. (left) Non-Saturated links (right) Saturated links.



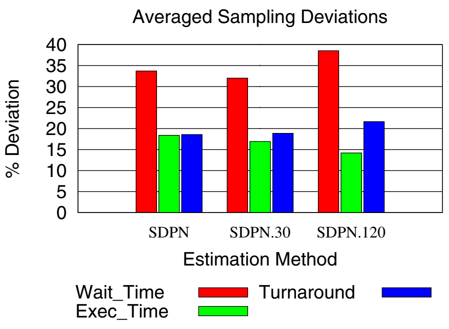


Fig. 4. Waiting and execution time deviations

Figure 2 (right) shows the deviation of the averaged estimated turnaround time with *MPL* ≤ 4. In general, the simulation methods based on analytical models, *Hanzich* and *SDPN*, obtained better results than *Li* and *Smith*. This is because, in order to detect the variability in a non-dedicated environment, the analytical models capture more accurate information and detect the workload variability better than Li and Smith's proposals. In this case, *SDPN* obtained in average a 6% gain with respect to *Hanzich*.

Figure 3 shows the behavior of the different estimation methods for different job selection (FCFS and Backfilling), node selection (Normal and Uniform) policies and *MPL* ≤ 4. In the figure 3 (right) the local tasks saturated 50% of the mapped links. On the left one, the local workload did not saturate any link at all. *SDPN* presented the highest accuracy in almost all the situations. In general, backfilling policies obtained the worst results, except for *SDPN*. This model captured the workload variations much better than the other ones. Note that the *SDPN* behavior for the FCFS case was very similar to *Hanzich*, but not for the backfilling policies where *SDPN* obtains in average a 80% gain. As figure 3 (right) shows, the prediction deviation grew with the presence of saturation. Differences between the scheduling policies were not significant. Nevertheless, the good behavior of our model even with a high saturation level is remarkable. The gain obtained by the *SDPN* method was 8% higher than with the *Hanzich* proposal.

In figure 4 (left), we compare the waiting and the execution time deviations obtained by the *Historical* and the analytical methods. As can be seen, in the historical case, there is no correlation between the results achieved by both metrics. It means that both passed job executions and recent cluster state information have to be taken into account when looking for similarities in the historical data. In the Historical case, as the cluster state does not vary, all the job execution times are similar, and consequently the predictions are very close to real executions. Waiting time predictions are the poorest because no information about the Input Queue is managed by this method. The analytical models were more accurate in both metrics, especially for the *SDPN* method.

In order to find solutions to increase the accuracy, we implemented a mechanism in the *SDPN* method to obtain an averaged resources state of several samples taken over a period of time. Figure 4 (right), shows average deviations of the waiting and execution times for different windowed mean, instantaneous, 30 and 120 seconds. Samples are taken each second. As we can see, the prediction accuracy of the execution time grows

Table 1. Time cost of our estimation proposals

	Simulation	**Historical**	**Hybrid**
Time Cost (milliseconds)	2.8	24	52
Complexity	$O(N_{RL}^2)$	$O(N_{HistDB})$	$O(N_{RL}^2 N_{HistDB})$

by increasing the length of the time period. This situation does not occur for the waiting time. This part of the turnaround time must be investigated in depth because is the main reason for the prediction inaccuracy.

We have measured the time cost spent by the different methods studied in the present work to estimate the turnaround time of a single application. This cost was compared with the complexity associated to each algorithm. The *Simulation* column represents the *Hanzich* and *SDPN* methods and the *Hybrid* column represents the *Li* and *Smith* ones (see table 1). The complexity of the Simulation methods are quadratic with the length of the jobs in the running list (N_{RL}), which is at most 4 jobs, whereas the order of the Historical one is linear with the number of elements in the Historical Repository (N_{HistDB}), which is at least 10^4 times bigger than N_{RL}. Likewise, note that the time cost of the Hybrid one is strongly correlated with its complexity. Finally, we emphasize that the time cost is in the order of milliseconds in all the cases, hardly noticeable with respect to the turnaround time of the parallel jobs.

4 Conclusions and Future Work

In this paper, we presented a prediction scheme to obtain an estimation of the parallel turnaround time in heterogeneous and non-dedicated clusters. Our proposal is based on an analytical model that establishes the processing and communication slowdown of the job's execution time by taking the future cluster state into account.

The model presented was implemented in a prediction engine integrated into the scheduling system of the CISNE platform. We corroborated the usefulness and accuracy of our model in real and widely used and accepted job scheduling policies, such as *Backfilling*. Applicability of our proposal has been proved and compared in a real environment with respect to the most representative models from the literature.

The analytical models gave the best results. In general in situations with no saturation, overtook the *Hanzich* one by a mean average of 40%. In situations with link saturation, our model underestimates the communication slowdown nevertheless the averaged gain was also important, at about 8%. The network parameters and the communication model of the parallel applications must be more deeply analyzed.

Regression models must be developed for obtaining the tendencies in the future cluster state. By doing this, precision can even be increased a bit more, especially in overloaded situations and dynamic load. The elapsed waiting time of the parallel jobs must also be more accurately estimated. The results shows that accurate execution time prediction are not the only factor that determines the accuracy of the waiting time predictions. The variation in the length of the input queue must be reflected in the cluster state at each simulation step. This is a drawback to be solved in future work.

References

1. Acharya, A., Setia, S.: Availability and utility of idle memory in workstation clusters. In: Proceedings of the ACM SIGMETRICS 1999, pp. 35–46 (1999)
2. Buyya, R., Abramson, D., Giddy, J.: Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid. ArXiv Computer Science e-prints (2000)
3. Etsion, Y., Tsafrir, D., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel & Distributed Syst. 18(6), 789–803 (2007)
4. Downey, A.: Predicting queue times on space-sharing parallel computers. In: 11th Intl. Parallel Processing Symp., pp. 209–218 (1997)
5. Aridor, Y., Yom-Tov, E.: A self-optimized job scheduler for heterogeneous server clusters. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2007. LNCS, vol. 4942. Springer, Heidelberg (2008)
6. Hanzich, M., Giné, F., Hernández, P., Solsona, F., Luque, E.: Using on-the-fly simulation for estimating the turnaround time on non-dedicated clusters. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro Par 2006. LNCS, vol. 4128, pp. 117–187. Springer, Heidelberg (2006)
7. Harchol-Balter, M., Li, C., Osogami, T., Scheller-Wolf, A., Squillante, M.S.: Cycle stealing under immediate dispatch task assignment. In: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures, pp. 274–285 (2003)
8. Jarvis, S., Spooner, D., Keung, H.L.C., Cao, J., Saini, S., Nudd, G.: Performance prediction and its use in parallel and distributed computing systems. Future Gener. Comput. Syst. 22(7), 745–754 (2006)
9. Javadi, B., Abawajy, J.: Performance analysis of heterogeneous multi-cluster systems. In: Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW 2005), Washington, DC, USA, pp. 493–500 (2005)
10. Jones, W.: The impact of error in user-provided bandwidth estimates on multi-site parallel job scheduling performance. In: The 19th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2007), Cambridge, Massachusetts (November 2007)
11. Jones, W., Ligon, W., Pang, L., Stanzione, D.: Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. The Journal of Supercomputing 34(2), 135–163 (2005)
12. Lafreniere, B., Sodan, A.: Scopred—scalable user-directed performance prediction using complexity modeling and historical data. In: Feitelson, D.G., Frachtenberg, E., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2005. LNCS, vol. 3834, pp. 62–90. Springer, Heidelberg (2005)
13. Li, H., Groep, D., Templon, J., Wolters, L.: Predicting job start times on clusters. In: 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004) (April 2004)
14. Smith, W., Taylor, V., Foster, I.: Using run-time predictions to estimate queue wait times and improve scheduler performance. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1999, IPPS-WS 1999, and SPDP-WS 1999. LNCS, vol. 1659, pp. 202–219. Springer, Heidelberg (1999)
15. Urgaonkar, B., Shenoy, P.: Sharc: Managing cpu and network bandwidth in shared clusters. IEEE Trans. Parallel Distrib. Syst. 15(1), 2–17 (2004)
16. Wolski, R.: Experiences with predicting resource performance on-line in computational grid settings. ACM SIGMETRICS Performance Evaluation Review 30(4), 41–49 (2003)