MIP Model Scheduling for Multi-Clusters

Héctor Blanco, Fernando Guirado, Josep Lluís Lérida, and V.M. Albornoz

Universitat de Lleida, Universidad Técnica Federico Santamaría
{hectorblanco,f.guirado,jlerida}@diei.udl.cat,
victor.albornoz@usm.cl

Abstract. Multi-cluster environments are composed of multiple clusters that act collaboratively, thus allowing computational problems that require more resources than those available in a single cluster to be treated. However, the degree of complexity of the scheduling process is greatly increased by the resources heterogeneity and the co-allocation process, which distributes the tasks of parallel jobs across cluster boundaries.

In this paper, the authors propose a new MIP model which determines the best scheduling for all the jobs in the queue, identifying their resource allocation and its execution order to minimize the overall makespan. The results show that the proposed technique produces a highly compact scheduling of the jobs, producing better resources utilization and lower overall makespan. This makes the proposed technique especially useful for environments dealing with limited resources and large applications.

Keywords: Job Scheduling, Multi-Cluster, Co-Allocation, MIP Model.

1 Introduction

Computation problems that require the use of a large amount of processing resources can be solved by the use of multiple clusters in a collaborative manner. These environments, known as multi-clusters, are distinguished from grids by their use of dedicated interconnection networks [1].

In those environments the scheduler has access to distributed resources across different clusters to allocate those jobs that cannot be assigned to a single cluster [2]. This allocation strategy, known as co-allocation, can maximize the job throughput by reducing the queue waiting times, and thus, jobs that would otherwise wait in the queue for local resources can begin its execution earlier, improving system utilization and reducing average queue waiting time [2]. However, mapping jobs across the cluster boundaries can result in rather poor overall performance when co-allocated jobs contend for inter-cluster network bandwidth. Additionally, the heterogeneity of processing and communication resources increases the complexity of the scheduling problem [3].

It is possible to find in the literature multiple studies based on co-allocation. In [2] different scheduling strategies using co-allocation were analyzed, concluding that unrestricted co-allocation is not recommendable. Other studies have dealt with co-allocation by developing load-balancing techniques [4], selecting the most

I. Caragiannis et al. (Eds.): Euro-Par 2012 Workshops, LNCS 7640, pp. 196–206, 2013.

powerful processors [5] or minimizing the inter-cluster link usage [3] without finding a compromise between them. In [6] an analytical model was presented in order to reduce the parallel jobs execution time by considering both resource availability: processors and communication.

A common issue in previous works is that jobs are allocated individually. Thus, allocating the best available resources to a job without considering the requirements of the rest of jobs present in the waiting queue, can reduce the performance of future allocations and the overall system performance [7]. To solve this, in [8] the authors presented a scheduling strategy based on a linear programming model, which brings together the parallel jobs in the waiting queue that *fit* the available resources and allocates them simultaneously.

The main constraint on this strategy is the limitation to the set of jobs that fit on the available resources. In the present work, the authors proposed a new MIP model referenced as *OAS* for Ordering and Allocation Scheduling, capable of determine the best resources allocation and the order in which they must be executed. Although the scheduling problem is NP-hard, the results shown that by considering sets of jobs it is possible to improve the resources utilization and the overall performance.

2 Related Work

In the literature there are multiple strategies for the scheduling process that can be grouped into two categories; on-line and off-line modes. In the on-line mode, only arrived jobs to the system are known, and the allocation decisions are restricted to those jobs. On the other hand, the off-line mode has knowledge of all the jobs considering the whole set of job for allocation decisions [9]. Nevertheless, in on-line systems with high job-interarrival rates, the scheduling problem can be addressed with job-clustering techniques to improve the overall performance, the system utilization, etc.

The on-line techniques allocate only one job without taking into account the rest of the waiting jobs, loosing relevant information to improve the overall system performance. By maintaining the job arrival order, resources that are available may end up not being allocated. The backfilling technique aims to solve this allowing to be moved up smaller jobs from the back of the queue [10]. Shmueli et al. proposed a look-ahead optimizing scheduler to generate the local optimal backfill selection by using dynamic programming [7]. Shah et al. proposed a near optimal job packing algorithm to reduce the chances of job killing and minimize external fragmentation [11]. These approaches tried to map only the jobs that better fill the gaps without considering other packing opportunities. Previous research has shown that slightly modifying the execution order of jobs can improve utilization and offer new optimization opportunities [7][8].

The strategies previously presented are extensively used on Parallel machines and Cluster computing environments. Nevertheless, they are based on specific environment characteristics and in most cases assuming jobs with independent tasks, i.e, without communication constraints. In the present paper, we consider

jobs with a fixed number of processors requirement, known as rigid Bulk-Synchronous Parallel jobs [7]. The meta-scheduling on multi-cluster resources is more challenging than traditional single-domain systems, due to the heterogenous dynamic resources availability in different administrative domains, and the continuous arrival of jobs at the meta-scheduler [12]. Hence, to face the new challenges on multi-cluster environments new heuristics should be proposed.

The research on multi-cluster and grid environments has provided multiple scheduling solutions based on different criteria: cost, makespan, etc. Feng et al. [13] proposed a deadline cost optimization model for scheduling one job. Buyya et al. [14] proposed a greedy approach with deadline and cost constraints for efficient deployment of an individual job. In contrast, the current paper is focused on the concurrent scheduling of many jobs. In [15][16] heuristics and Genetic Algorithm solutions for scheduling concurrent jobs are proposed. However, those studies assumed independent jobs with no communication restrictions.

3 Ordering and Allocation Scheduling Strategy

System performance have different meanings. Final users performance deal with reducing the job execution time. However, system administrators would like to maximize the resources usage. In an environment with large job inter-arrival time, the allocation mechanism is responsible of improving both performances.

However, in situations with low inter-arrival time, jobs accumulate in the waiting queue generating new scheduling opportunities. In these situations, both execution order and allocation strategy are decisive for improving overall performance. In the present work, a new MIP model (OAS), which manages the packing of jobs in the waiting queue to minimize their makespan, thus improving the system utilization and user satisfaction is proposed. In order to do that, OAS deals with two challenges: (i) resources heterogeneity and availability and (ii) tasks from a job can be assigned to different clusters in a co-allocation process. In these circumstances, the allocation not only has to consider the processing time, but also the communication capabilities in order to avoid inter-cluster link saturation, which could produce unpredictable effects on job performance.

3.1 Problem Statement

Multi-Cluster Model. A multicluster environment is made up of a set of α arbitrary sized clusters with heterogeneous resources. Let $M = \{C_1..C_\alpha\}$ is the set of Cluster sites; $R = \{R_1^1, R_2^1..R_{n-1}^\alpha, R_n^\alpha\}$ the set of processing resources of the multi-cluster, being n the total number of nodes. Each cluster is connected to each other by a dedicated link through a central switch. Let $\mathcal{L} = \{\mathcal{L}_1..\mathcal{L}_\alpha\}$ the inter-cluster links being $\mathcal{L}_k$ the link between the site C_k and the central switch.

The processing resources capabilities are represented by the Effective Power metric (Γ^r) defined in [6]. This normalized metric relates the processing power of each resource with its availability. Thus, $\Gamma^r = 1$ when processing resource $r \in R$ has capacity to run tasks at full speed, and otherwise $\Gamma^r < 1$.

Parallel Application Job Model. In this work, we consider parallel application jobs with a fixed number of processing resources requirements known as rigid parallel jobs [7]. A job j is composed of a fixed number of tasks that act in a collaborative manner. Each task τ_j is comprised of various processing, communication and synchronization phases. In our case, each job task uses an all-to-all communication pattern with similar processing and communicating requirements, where all tasks start and finish at the same time, following the Bulk-Synchronous Parallel model. Job assignment is static avoiding re-allocations while the job is being executed. Additionally, jobs can be co-allocated on different clusters in order reduce its execution time.

Thus, the estimated execution time for the parallel job j can be modeled by

$$Te_j = Tb_j \cdot ct_j, \quad \forall j \in J \tag{1}$$

where Tb_j is the base time of j in dedicated resources and ct_j is the time cost factor when job is allocated on resources S. The base-time Tb_j is assumed to be known based on user-supplied information, experimental data, job profiling, etc.

Time Cost Model. In the literature, the time cost ct_j is obtained from the allocated resources without considering communications [3], or considering a fixed communications penalty when co-allocation is applied [17]. In contrast, we modeled the time cost based on heterogeneity of the selected processing resources and the availability of the inter-cluster links used, expressed by

$$ct_j = \sigma_j \cdot SP_j + (1 - \sigma_j) \cdot SC_j, \quad \forall j \in J \tag{2}$$

where SP_j denotes the processing slowdown from resources, SC_j is the communication slowdown by inter-cluster links, and σ_j is the processing time relevance with respect to the communication time. The σ_j is obtained by characterizing the job as the base-time Tb_j. The SP_j factor is obtained from the slowest processing resource, i.e. which provides the maximum processing slowdown,

$$SP_j = \max_{\forall r \in S} \{SP_j^r\}, \quad \forall j \in J \tag{3}$$

where SP_j^r is the processing slowdown from the effective power of resource r.

SC_j evaluates the communication slowdown by inter-cluster link contention. The co-allocation of a parallel job application consumes a certain amount of bandwidth in each inter-cluster link $\mathcal{L}_k$, denoted by BW_j^k, and calculated by

$$BW_j^k = \left(t_j^k \cdot PTBW_j\right) \cdot \left(\frac{\tau_j - t_j^k}{\tau_j - 1}\right), \quad \forall k \in 1 \ldots \alpha, j \in J \tag{4}$$

where $PTBW_j$ is the required per-task bandwidth, τ_j is the total number of tasks of j and t_j^k is the number of those allocated on cluster C_k. The first term in the equation is the total bandwidth consumed by tasks on cluster C_k, while the second is the percentage of communication with other clusters.

When co-allocated jobs use more bandwidth than the available, saturation occurs, and jobs sharing this link are penalized increasing their communication time. The inter-cluster links saturation degree relates the maximum bandwidth of each link with the bandwidth requirements of the allocated parallel jobs

$$SAT^k = \frac{MBW}{\sum_{\forall j}(BW_j^k)}, \quad k \in 1 \ldots \alpha, j \in J \tag{5}$$

when $SAT^k \geq 1$ the link $\mathcal{L}_k$ is not saturated, otherwise is saturated delaying the jobs that use it, with a slowdown expressed by

$$SC_j^k = \begin{cases} (SAT^k)^{-1} \text{ when } SAT^k < 1 \\ 1 \qquad\qquad \text{otherwise} \end{cases} \tag{6}$$

Communication slowdown SC_j comes from the most saturated link used

$$SC_j = \max_{\forall k}\{SC_j^k, \quad \forall j \in J\}, \tag{7}$$

Allocation and Scheduling Mechanism. The most common scheduling techniques allocate the jobs separately, without taking into account the requirements of further jobs. This is represented in Figure 1(a) where a *First Come First Served* scheduling has been applied. Better performance results can be achieved by the set of jobs together, as can be seen in Figure 1(b).

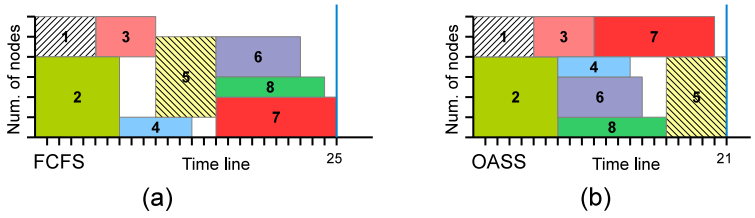


Fig. 1. Job scheduling: (a) FCFS allocation, (b) allocation grouping tasks

3.2 The Mixed-Integer Programming Model

Mixed-Integer Programming (MIP) allows to obtain the solutions that maximize or minimize an objective function under some constraints. In this paper, the objective function is the makespan, thus the obtained solution provides the allocation and execution order for each treated job that reduce their execution time, also minimizing the resources idle time. In [17] was determined that in some situations a certain threshold on the saturation degree may be allowed. For simplicity, in the present study, we restrict the model to those solutions that avoid the saturation, by evaluating the inter-cluster links usage. Next we present the MIP model shown in Figure 2.

Parameters and Variables. First, the multi-cluster environment is described; the set of resources R and their effective power (Γ^r), inter-cluster links ($\mathcal{L}$) and the maximum available bandwidth for each inter-cluster link $k \in \mathcal{L}$ (MBW_k). Next, for each job j: the number of tasks (τ_j), its base-time (Tb_j), the required per-task bandwidth ($PTBW_j$) and the weighting factor (σ_j), which measures the relevance of the processing and communication time (lines 1-9).

The decision variables define the job order and allocation (lines 13-20). The allocation is expressed by a binary variable, $Z_{(j,r)} = 1$ (line 13) when the job j is assigned to r and 0 otherwise. To obtain the job execution order, the allocation

domain of each resource is splitted into time-slots (lines 14-17), being $X_{(j,r,t)} = 1$ when job j is assigned to resource r in the time-slot t. Let $T = \{T^1..T^\theta\}$ be the set of time-slots and θ the total number of time-slots used by the set of jobs. Variable $Y_{(j,t)}$ is set if job j starts its execution in the time-slot t (line 15).

Input Parameters

1. R: Set of processing resources.
2. $\mathcal{L}$: Set of inter-cluster links.
3. Γ^r: Effective power for resource r, $\forall r \in R$
4. MBW_k: Maximum Available bandwidth for each inter-cluster link k, $\forall k \in \mathcal{L}$.
5. J: set of jobs to be allocated.
6. τ_j: number of tasks making up job j, $\forall j \in J$.
7. Tb_j: execution base-time for the job j, $\forall j \in J$.
8. $PTBW_j$: required bandwidth for each jobs task, $\forall j \in J$
9. σ_j: time-processing and -communication weighting factor, $\forall j \in J$.
10. T: Set of time-slots in which job can be assigned.
11. θ: total number of time-slots. Deadline for the set of jobs.
12. η: time-slot size.

Variables

13. $Z_{(j,r)} = 1$ if j is assigned to resource r, $\forall j \in J, r \in R$
14. $X_{(j,r,t)} = 1$ if j is assigned to resource r in slot t, $\forall j \in J, r \in R, t \in T$
15. $Y_{(j,t)} = 1$ if j starts running in slot t, $\forall j \in J, t \in T$
16. s_j: time-slot in which job j starts running, $\forall j \in J$
17. f_j: time-slot in which job j is completed, $\forall j \in J$
18. SP_j is the processing slowdown of job j, $\forall j \in J$
19. $BW_{j,k,t}$: Bandwidth consumed by job j on link k in slot t. $\forall j \in J, k \in \mathcal{L}, t \in T$
20. $ABW_{k,t}$: Available bandwidth on link k, in slot t, $\forall k \in \mathcal{L}, t \in T$

Objective function

21. Minimize the makespan of the set of jobs

Fig. 2. MIP model representation

Based on the time-slot, the job j execution time is determined by its starting time-slot, s_j, and the last used time-slot, f_j. The time-slots occupied by j are calculated considering the job base-time (Tb_j), the processing slowdown (SP_j) that the allocated resources provide and the time-slot size (η), expressed by

$$f_j = s_j + (Tb_j * (SP_j * \sigma_j + (1 - \sigma_j)))/\eta \tag{8}$$

Processing slowdown SP_j (line 18) is calculated by equ. 3. The communication slowdown is not considered because the solutions are those that avoid the saturation of inter-cluster links (lines 19-20). Variable $BW_{j,k,t}$ is the bandwidth consumed on the inter-cluster link k for the time-slot t, and $ABW_{k,t}$ is the available bandwidth on link k on time-slot t once all jobs have been allocated.

Objective Function. When there are many possible solutions, the objective function defines the quality of each feasible solution. The aim of the model is to minimize the global makespan that is determined by the latest completed job

$$minimize\{ \quad \max_{\forall j \in J}(f_j) \quad \} \tag{9}$$

Constraints. The constraints contribute to define the correct solutions to the problem. Thus, the model must ensure that all tasks from a parallel job are allocated and start at the same time. We define the following set of constraints:

$$\sum_{\forall j \in J} X_{(j,r,t)} \leq 1, \quad \forall r, t \tag{10}$$

$$Z_{(j,r)} = \max_{t \in T}(X_{j,r,t}), \quad \forall j, r \tag{11}$$

$$\sum_{\forall r \in R} Z_{(j,r)} = \tau_j, \quad \forall j \tag{12}$$

$$\sum_{\forall r' \in R} (X_{j,r',t}) \geq (\tau_j \cdot X_{j,r,t}), \quad \forall j, r, t \tag{13}$$

$$X_{(j,r,t')} \leq 1 - Y_{(j,t)} \quad \forall j, r \wedge t' \in 1..(t-1) \tag{14}$$

$$\sum_{\forall t \in T} Y_{(j,t)} = 1 \quad \forall j \tag{15}$$

$$s_j = \sum_{\forall t \in T} (Y_{j,t} \cdot t) - 1 \quad \forall j \tag{16}$$

$$f_j = \max_{\forall r \in R, t \in T} (X_{(j,r,t)} \cdot t) \quad \forall j \tag{17}$$

$$ct_j = \sigma_j \cdot SP_j + (1 - \sigma_j) \quad \forall j \tag{18}$$

$$(f_j - s_j) \cdot \eta \leq (Tb_j \cdot ct_j) \quad \forall j \tag{19}$$

$$(\sum_{\forall r \in R, t \in T} X_{j,r,t} \cdot \eta) \geq (Tb_j \cdot ct_j \cdot \tau_j) \quad \forall j \tag{20}$$

$$ABW_{k,t} = MBW_k - \sum_{\forall j \in J} BW_{j,k,t} \quad \forall j, k, t \tag{21}$$

$$ABW_{k,t} \geq 0 \quad \forall k, t \tag{22}$$

$$Z_{(j,r)} \in \{0,1\}, \quad X_{j,r,t} \in \{0,1\}, \quad Y_{(j,t)} \in \{0,1\},$$
$$s_j \geq 0, \quad f_j \geq 0, \quad ct_j \geq 0 \tag{23}$$

Constraint set (10) ensures that a resource can only be allocated to a task simultaneously. Constraint set (11) defines the variable $Z_{(j,r)}$, equals to 1 when job j is allocated to the resource r and 0 otherwise. Constraint set (12) ensures that all tasks τ_j are allocated. Constraint set (13) guarantees that all the tasks of a job are executed at the same time but in different resources. Constraint sets (14) and (15) define the variable $Y_{(j,t)}$, equals to 1 when the jth job initiates its execution in the tth time-slot, otherwise is 0. In constraint (16) and (17) the variables s_j and f_j are defined as the starting and finishing time-slot for the jth job. Variable s_j is obtained from variable $Y_{(j,t)}$ and variable f_j is calculated considering the slowest allocated resource. In constraint set (18), the execution cost ct_j the processing slowdown is obtained from the slowest allocated resource. Constraint sets (19) and (20) ensure that the time-slots are contiguous and in accordance with the time spent on the slowest allocated resource. Constraint set (21) calculates the available bandwidth of the kth inter-cluster link in the tth time-slot, $ABW_{k,t}$. Finally, constraint set (22) guarantees non-saturation of the inter-cluster links in every time-slot.

4 Experimentation

The experimental study was carried out in order to: (1) evaluate the influence of the time-slot duration size on the scheduling solutions and (2) determine the effectiveness of the scheduling solutions provided by *OAS*. The first study used a synthetic workload varying the size of the time-slot. On the second, *OAS* was compared with heuristics from the literature.

OAS was implemented by using the *CPLEX* linear mixed integer programming solver and the solutions were applied on the GridSim simulation framework, characterized as a heterogeneous multi-cluster system. The scheduling of applications with independent tasks is NP-Complete, and MIP models are known to be very computational demanding. Due to this, the environment was limited to 3 clusters, each one with 4 nodes interconnected by a Gigabit network. The heterogeneity was defined by different effective power to each cluster with values of $\Gamma_k = \{1.0, 0.75, 0.5\}$ respectively.

4.1 Time-Slot Size Analysis

To determine the job execution order, *OAS* uses the time-slot concept. The time-slot size could limit the quality of the scheduling solution, so a study was performed to evaluate the impact of the time-slot size on the results. We defined a workload composed by 8 jobs, with high computational requirements ($\sigma_j = 0.7$), with an average job base time of 67×10^4 seconds, with different number of tasks, from 1 to 12 and with different computation and communication requirements, representative of parallel jobs in the fields of weather prediction, fluid simulation, etc. identified by their very large computational requirements.

The time-slot values were in the range {15%..250%} times the average of the jobs base-time from the workload. Figure 3 shows the makespan and the solver execution time for each case of study. As can be seen, when the slot size increases, the makespan increases. This is because bigger slots reduces the resources availability during long periods of time even allocated jobs have finished. Thus, the start time of upcoming jobs is delayed until the time-slot finishes.

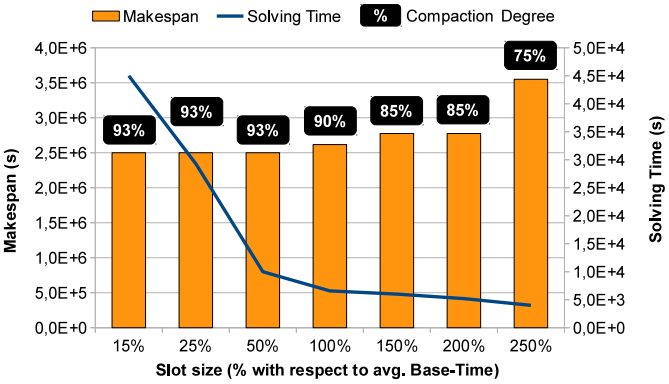


Fig. 3. Comparison for the time-slot size

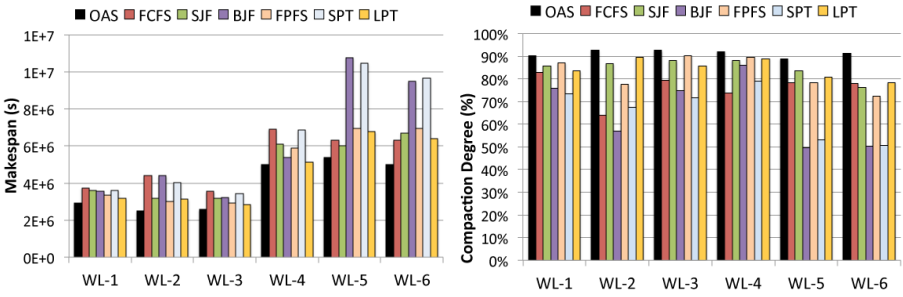
The shortest the time-slot is, the better makespan is obtained. However, the model becomes more complex increasing the solving time. From sizes lower than 50%, makespan values become similar to the minimum makespan. In the other way the solving time grows up exponentially. Figure 3, also show the compaction degree, which measures the percentage of time in which the resources are been used respect the overall execution time. Thus, the higher the compaction degree is, the more exploited and less idle the resources are.

4.2 *OAS* Performance Evaluation

To determine the effectiveness of the solutions provided by *OAS*, we compare the results with some of the most common scheduling techniques in the literature: *First Come First Served* (FCFS), *Short Jobs First* (SJF), *Big Jobs First* (BJF), *Fit Processors First Served* (FPFS), *Short Processing Time* (SPT) and *Long Processing Time* (LPT).

In this experimental study we defined a set of six synthetic workloads composed by 8 jobs with similar characteristics of computational requirements as in the previous experimental study. We defined two of them in order to fit well a specific scheduling technique and thus limiting any solver advantage. Then, the first workload *WL-1* was designed to perform well with the techniques that try to match the available resources with the processing requirements. *WL-2* workload was designed to perform well with the techniques that prioritize smaller jobs. Finally, workloads *WL-3* to *WL-6*, designed without taking any particular criteria into account. The metrics used were the makespan (Figure 4(a)) and the compaction degree (Figure 4(b)). The time-slot size was defined to the 50% of the average job base time.

It can be observed that each technique has a different behavior. The technique that in some workload performs well in another obtains worst makespan and lower compaction degree. But in all cases *OAS* obtained good makespan and compaction results, irrespectively on the workload nature. This is due to its ability to have a global vision on the resources requirements for the whole set of jobs and their availability. Thus, by defining the correct job execution order and task to resource



(a) Makespan Comparison (b) Compaction Degree Comparison

Fig. 4. Comparison of six kinds of workloads

allocation, it is possible to obtain the better scheduling decisions that reduces the makespan, increasing the compaction degree and improving the resources usage.

5 Conclusions

In the present work, we focused on the scheduling process of BSP parallel applications on heterogeneous multi-cluster environments, by applying multiple job allocation and co-allocation when it is necessary. The goal is to determine the effectiveness of the job execution order and the multiple-job allocation with processing and communication resource considerations, and thus, a *Mixed-Integer Programming* model had been developed. The results were compared with other scheduling techniques from the literature, obtaining better makespan results and resources usage. However, the presented MIP model has a great computational complexity. By this, we are developing a low complexity scheduling heuristic with similar goals.

Acknowledgment. This work was supported by the MECS of Spain under contract TIN2011-28689-C02 and the CUR of DIUE of GENCAT and the European Social Fund. Also, DGIP (Grant USM 28.10.37) and CIDIEN of Univ. Técnica Federico Santa María.

References

1. Javadi, B., Akbari, M.K., Abawajy, J.H.: A performance Model for Analysis of Heterogeneous Multi-Cluster Systems. Parallel Computing 32(11-12), 831–851 (2006)
2. Bucur, A.I.D., Epema, D.H.J.: Schedulling Policies for Processor Coallocation in Multicluster Systems. IEEE TPDS 18(7), 958–972 (2007)
3. Jones, W., Ligon, W., Pang, L., Stanzione, D.: Characterization of Bandwidth-Aware Meta-Schedulers for Co-Allocating Jobs Across Multiple Clusters. Journal of Supercomputing 34(2), 135–163 (2005)
4. Yang, C., Tung, H., Chou, K., Chu, W.: Well-Balanced Allocation Strategy for Multiple-Cluster Computing. In: IEEE Int. Conf. FTDCS 2008, pp. 178–184 (2008)
5. Naik, V.K., Liu, C., Yang, L., Wagner, J.: Online Resource Matching for Heterogeneous Grid Environments. In: Int. Conf. CCGRID 2005, vol. 2, pp. 607–614 (2005)
6. Lérida, J.L., Solsona, F., Giné, F., García, J.R., Hernández, P.: Resource Matching in Non-dedicated Multicluster Environments. In: Palma, J.M.L.M., Amestoy, P.R., Daydé, M., Mattoso, M., Lopes, J.C. (eds.) VECPAR 2008. LNCS, vol. 5336, pp. 160–173. Springer, Heidelberg (2008)
7. Shmueli, E., Feitelson, D.G.: Backfilling with Lookahead to Optimize the Packing of Parallel Jobs. J. Parallel Distrib. Comput. 65(9), 1090–1107 (2005)
8. Blanco, H., Lérida, J.L., Cores, F., Guirado, F.: Multiple Job Co-Allocation Strategy for Heterogeneous Multi-Cluster Systems Based on Linear Programming. Journal of Supercomputing 58(3), 394–402 (2011)
9. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U.: Parallel Job Scheduling — A Status Report. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 1–16. Springer, Heidelberg (2005)

10. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE TPDS 18(6), 789–803 (2007)
11. Hussain, S., Qureshi, K.: Optimal job packing, a backfill scheduling optimization for a cluster of workstations. Journal of Supercomputing 54(3), 381–399 (2010)
12. Zhang, W., Cheng, A.M.K., Hu, M.: Multisite co-allocation algorithms for computational grid. In: IPDPS 2006, pp. 335–335 (2006)
13. Feng, H., Song, G., Zheng, Y., Xia, J.: A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Dependent Tasks in Grid Computing. In: Li, M., Sun, X.-H., Deng, Q., Ni, J. (eds.) GCC 2003. LNCS, vol. 3033, pp. 113–120. Springer, Heidelberg (2004)
14. Buyya, R., Murshed, M., Abramson, D., Venugopal, S.: Scheduling parameter sweep applications on global Grids: a deadline and budget constrained cost-time optimization algorithm. Softw. Pract. Exper. 35(5), 491–512 (2005)
15. Munir, E.U., Li, J., Shi, S.: QoS sufferage heuristic for independent task scheduling in grid. Information Technology Journal 6(8), 1166–1170 (2007)
16. Garg, S., Buyya, R., Siegel, H.: Time and cost trade-off management for scheduling parallel applications on Utility Grids. Future Generation Computer Systems 26(8), 1344–1355 (2010)
17. Ernemann, C., Hamscher, V., Schwiegelshohn, U., Yahyapour, R., Streit, A.: On Advantages of Grid Computing for Parallel Job Scheduling. In: Int. Conf. CCGRID 2002 (2002)
18. Li, H., Groep, D., Wolters, L.: Workload Characteristics of a Multi-cluster Supercomputer. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 176–193. Springer, Heidelberg (2005)