OAS: A MIP Model for Ordering and Allocating Parallel Jobs on Multi-Cluster Systems

Héctor Blanco
Universidad de Lleida
Escola Politécnica Superior. C/ Jaume II, 69. 25001. Lleida. Spain
hectorblanco@diei.udl.cat

Fernando Guirado
Universidad de Lleida
Escola Politécnica Superior. C/ Jaume II, 69. 25001. Lleida. Spain
f.guirado@diei.udl.cat

Josep Lluís Lérida
Universidad de Lleida
Escola Politécnica Superior. C/ Jaume II, 69. 25001. Lleida. Spain
jlerida@diei.udl.cat

Víctor M. Albornoz
Universidad Santa María
Departamento de Industrias. Av. Santa Maria 6400. Santiago. Chile.
victor.albornoz@usm.cl

ABSTRACT

Multi-cluster environments are composed of multiple clusters of computers 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 heterogeneity of resources and the co-allocation process, which distributes the tasks of parallel jobs across cluster boundaries.

In a previous work, the authors presented a scheduling strategy which selects from the system queue only the set of jobs that fits the available resources, and finds their best possible allocation by a Mixed-Integer Programming model (MIP), considering both the processing and communication requirements of the applications.

In this study, the authors propose a new MIP model that treats a set of jobs in the waiting queue determining their best execution order and allocation, in order to improve the makespan of the set of jobs.

KEYWORDS: Job Scheduling, Multi-Cluster Heterogeneity and Performance, Co-Allocation, Mixed Integer Programming

Acknowledgement: This work was supported by the Ministry of Education and Science of Spain under contract TIN2011-28689-C02 and the CUR of DIUE of GENCAT and the European Social Fund. Also, financial support from the DGIP (Grant USM 28.10.37) and CIDIEN of Univ. Técnica Federico Santa María is acknowledged.

1 Introduction

Computation problems that require the use of more processing resources than those offered by a single cluster 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 with a known topology and more predictable performance [JAA07].

A critical aspect of exploiting the resources in a multi-cluster is the scheduling [BE07]. The scheduler has access to distributed resources across different clusters to allocate those jobs that cannot be assigned to a single cluster. 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 [BE07]. 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 [JLPS05].

Co-allocation scheduling strategies in multi-cluster environments have generated great interest in recent years. The performance of different scheduling strategies using co-allocation was analyzed in [BE07]. This work concludes that unrestricted co-allocation is not recommendable, thus some studies have dealt with co-allocation by developing load-balancing techniques [HFH08][YTCC08], selecting the most powerful processors [NLYW05] or minimizing the inter-cluster link usage [JLPS05] without finding a compromise between them. In order to fill this gap, a new analytical model was presented in [LSG+08] to reduce the parallel jobs execution time by considering both resource availability: processors and communication, .

A common issue in those previous works is that jobs are allocated individually, considering all the available resources in a one-by-one job scheduling process. This methodology does not take into account the set of jobs present in the waiting queue. Thus, allocating the best available resources to a job without considering the requirements of the rest of jobs can reduce the performance of future allocations and accordingly diminishing the overall system performance [SF05]. In order to solve this problem, in [BLG11] the authors presented a scheduling strategy, named *PAS* for Package Allocation 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 the *PAS* strategy comes from the necessity to determine the set of jobs that fit on the available resources. In the present work, in order to overcome this limitation, authors treat not only the best resources to allocate the jobs but also the order in which they must be executed. To evaluate the effectiveness of this approach authors present a MIP model with the ability to determine the best execution order and resources allocation for a set of jobs (in next sections will be referenced as *OAS* for Ordering and Allocation Scheduling). Although the scheduling problem is NP-hard, and thus, the complexity of the MIP model, the result shown that considering a set of jobs take better profit of the resources, improving the overall performance. Thus, the analysis of the obtained results gave us relevant information to develop a new heuristic that will be applied on real environments.

2 Related Work

The research into scheduling parallel jobs has been treated extensively in the literature and multiple heuristics and strategies had been proposed. The scheduling process 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 then the allocation decisions are restricted to those jobs. On the other hand, the off-line mode has knowledge of all the jobs, from beginning to end, and thus can consider the whole set of job for allocation decisions [FRS05].

The on-line mode techniques have the disadvantage of allocating only one job without taking into account the rest of jobs in the waiting queue, and then lacking relevant information that could improve overall system performance. By maintaining the arrival order in the waiting queue, resources that are available may end up not being allocated. The backfilling technique aims to solve this by allowing the smaller jobs from the back of the queue to be moved up [TEF07]. In [SF05], Shmueli et al. proposed a look-ahead optimizing scheduler to generate the local optimal backfill selection by using dynamic programming. Shah et al. in [SQR10] proposed a near optimal job packing algorithm to reduce the chances of job killing and minimize external fragmentation. These approaches tried to map only the jobs that better fill the gaps without considering other packing opportunities with the jobs waiting in the queue. Previous research has shown that slightly modifying the execution order of jobs can improve utilization and offer new optimization opportunities [SF05][BLG11][SKSS02].

The heuristics 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, also called rigid Bulk-Synchronous Parallel jobs [SF05][SHM97]. The meta-scheduling of such jobs on multi-cluster resources is more challenging than traditional scheduling on single-domain systems, due to the dynamic availability of resources with different capabilities in different administrative domains, and the continuos arrival of jobs at the meta-scheduler [ZCmH06]. 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 many scheduling solutions for different optimization criteria: cost, makespan, utilization, etc. Feng et al. [FSZX03] proposed a deadline cost optimization model for scheduling one job with dependent tasks. Buyya et al. [BMAV05] 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 [MLS07][GBS10] heuristics and Genetic Algorithm solutions for scheduling concurrent jobs are proposed. However, those studies assumed independent jobs with no communication restrictions.

The main goal of the present work is to determine the effectiveness of treating a set of jobs at the same time, i.e. those that are waiting in the system queue, by determining their execution order and resources allocation to the multi-cluster environment while minimizing the overall makespan and avoiding inter-cluster link saturation.

3 Ordering & Allocation Scheduling Strategy

In general, the optimization-criteria extensively used in multi-cluster environments is focused on improving performance. Users would like to have the execution done in the minimum time, and resource administrators would like to make the maximum usage of the processing resources. In an environment where parallel jobs arrive with a large inter-arrival time, being the resources mainly free, the allocation mechanism is responsible of improving the performance.

However, in situations with low inter-arrival time, jobs accumulate in the waiting queue thus generating new scheduling opportunities. In these situations, both allocation strategy and execution order are decisive for improving overall performance. In the present work, we propose 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. In order to do that, *OAS* must manage two main challenges: (i) resources can have different capabilities and availabilities and (ii) different tasks in a job can be assigned to different clusters in a co-allocation process. In these circumstances, the allocation mechanism not only has to consider the processing time of the different resources, 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 assumed to be made up of a set of α arbitrary sized clusters with heterogeneous resources. Let $M = \{C_1, C_2, \ldots, C_\alpha\}$ denote the set of Cluster sites; let $R = \{R_1^1, R_2^1, \ldots, R_{n-1}^\alpha, R_n^\alpha\}$ be 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 $\mathcal{L} = \{\mathcal{L}_1..\mathcal{L}_\alpha\}$ denote the set of inter-cluster links and $\mathcal{L}_k$ be the link which connects the site C_k with the central switch.

In heterogeneous and non-dedicated environments, the processing resources capabilities can be different. To measure these differences we use the Effective Power metric (Γ^r) defined in [LSG+08]. 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 that are also called rigid parallel jobs [SF05]. A job j is composed of a fixed number of τ_j tasks that act in a collaborative manner. Each task is comprised of various iterations in which processing alternates with communication and synchronization phases. In our case, each job task was assumed to use an all-to-all communication pattern with similar processing and communicating requirements, following the widely used Bulk-Synchronous Parallel model [SHM97]. Job assignment is static, that is, once the job is mapped into a particular set of nodes, no more re-allocations are performed. Additionally, jobs can be co-allocated by using nodes from different cluster sites in order to better meet the collective needs across the multi-cluster.

Taking this into account the performance of a parallel job in a multi-cluster environment can be modeled by

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

where Te_j denotes the estimated execution time for the parallel job j, Tb_j denotes the base time of j in dedicated resources and ct_j be the time cost that modifies the base time by the set of allocated resources S. The base-time of j in dedicated resources, Tb_j, is assumed to be known based on user-supplied information, experimental data, job profiling or benchmarking techniques.

Time Cost Model In previous studies from literature, the time cost ct_j is obtained from the processing capabilities of the allocated resources without considering communications [JLPS05], or considering a fixed communications penalty when co-allocation is applied [EHS+02]. In contrast, we modeled the time cost of each job 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 of job j produced by the allocated resources, SC_j is the communication slowdown produced by the used inter-cluster links, and σ_j denotes the relevance of the processing time with respect to the communication time for job j. The σ_j value is obtained by characterizing the job along with the base-time Tb_j. Assuming similarity between the job tasks, SP_j is obtained from the slowest processing resource, i.e. that which provides the maximum processing slowdown, as expressed in

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

where SP_j^r denotes the processing slowdown of j obtained from the effective power of allocated 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 $PTBW_j$ denotes the required per-task bandwidth, τ_j is the total number of tasks of j and t_j^k is the number of those tasks allocated to cluster C_k. The first term in the equation is the total bandwidth consumed by the tasks allocated inside cluster C_k, while the second term represents the percentage of communication with other clusters.

When co-allocated jobs consume more bandwidth than the available in a communication link, saturation occurs, and thus, all jobs sharing this link are penalized and then their communication time increases. The degree of saturation of inter-cluster links relates the maximum bandwidth of each link with the bandwidth requirements of the allocated parallel jobs and is calculated by

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

where $SAT^k \geq 1$ when the link $\mathcal{L}_k$ is not saturated. Otherwise the link $\mathcal{L}_k$ is saturated, delaying the jobs that use it, with a slowdown inversely proportional to the degree of saturation as expressed by

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

where communication slowdown SC_j for each parallel job j comes from the most saturated link used, as expressed by

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

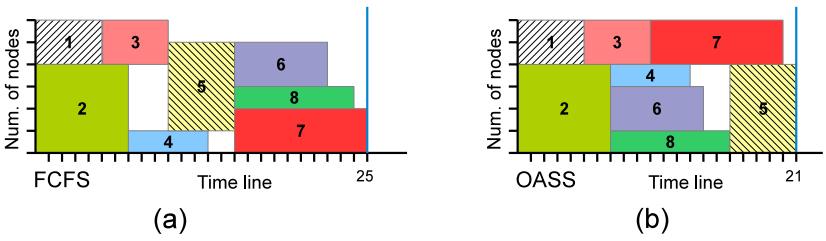

Fig. 1. Representation of job scheduling. (a) one-by-one allocation without considering interactions. (b) allocation grouping tasks, considering interactions

Allocation & Scheduling Mechanism The most common scheduling techniques allocate the jobs one-by-one from the waiting queue, without taking the requirements of further jobs into account. Figure 1(a) represents a *First Come First Served* scheduling, which does not take into account the requirements of further jobs. In situations with high arrival rates, this allocation procedure can produce negative effects on performance by not considering the resource requirements of further jobs. The resource sharing among jobs has a great impact on system performance, especially when co-allocation is applied, since contention for the inter-cluster links can significantly reduce the overall system performance. Several previous studies [SF05][SKSS02] have shown that better performance results can be achieved by allocating groups of jobs, as can be seen in Figure 1(b).

Our proposal treats a set of jobs in the waiting queue, minimizing the overall makespan. The scheduling problem is modeled as a MIP model, considering the most suitable resources and contention from the inter-cluster links.

3.2 The Mixed-Integer Programming model

Mixed-Integer Programming (MIP) is a technique to obtain those solutions that maximize or minimize the value of an objective function subject to some constraints. In this paper, we propose a model to achieve the scheduling that provides the minimum makespan for a set of jobs. This model provides the execution order and resources allocation for each job, minimizing the makespan and avoiding saturation of the inter-cluster links. Ernemann et al. [EHS+02] determined that in some situations a certain threshold on the saturation degree may be allowed. In the present study, for simplicity we restrict the model to those solutions that avoid the saturation, although it could be possible to define a threshold. The proposed MIP model, presented in Figure 2, is described as follows.

Parameters and variables Information about the multi-cluster status and jobs requirements is necessary to find the best allocation (lines 1-9). The multi-cluster structure is described by the set of processing resources R and the set of inter-cluster links ($\mathcal{L}$).The multi-cluster status is represented by the effective CPU power of each resource $r \in R$ (Γ^r) and the maximum available bandwidth for each inter-cluster link $k \in \mathcal{L}$ (MBW_k). The information for each job j corresponds to the number of tasks (τ_j), the job 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.

The decision variables of the model define the order and allocation of each job (lines 13-20). The allocation of a job task is expressed by a binary variable, being $Z_{(j,r)} = 1$ (line 13) when the job j is assigned to the resource r and 0 otherwise. To obtain the job execution order, the allocation domain of each resource is split into time-slots with equal size (lines 14-17), being $X_{(j,r,t)} = 1$ when job j is assigned to resource r in the time-slot t. Let $T = \{T^1, \cdots, T^\theta\}$ denote the set of time-slots and θ be the total number of time-slots used by the set of jobs. This value is provided by the scheduling system as a deadline constraint. Variable $Y_{(j,t)}$ (line 15) is set if job j starts its execution in the time-slot t.

Based on this partition into time-slots, the execution time of job j is defined by the time-slot in which job j starts running, denoted by s_j, and the time-slot in which job j is completed, denoted by f_j. The time-slots occupied by j are calculated considering the characterized job base-time (Tb_j), the processing slowdown (SP_j) produced by the allocated resources (see section 3.1) and the time-slot size (η), as 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 as expressed by equation 3. The slowdown of communication is not considered in this calculation because the model is limited

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.

by constraints to those solutions that avoid the saturation of inter-cluster links. In order to avoid a solution which produces saturation, this must be calculated (lines 19-20). Variable $BW_{j,k,t}$ is the bandwidth consumed by job j on inter-cluster link k on time-slot t. and $ABW_{k,t}$ is the available bandwidth on link k on time-slot t once all the jobs have been allocated.

Objective Function When there are many possible solutions, the objective function defines the quality of each feasible solution. In our model we are interested on minimizing the global makespan. In consequence, makespan optimization can be described in function of the latest completed job expressed by

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

Constraints The constraints contribute to defining the correct solutions to the problem. In our case, we must ensure that all tasks from a parallel application are allocated and start at the same time, by avoiding the saturation of the inter-cluster links. To this end, 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 one parallel job task simultaneously. Constraint set (11) defines the variable $Z_{(j,r)}$. This variable equals 1 when job j is allocated to resource r and equals 0 otherwise. Constraint set (12) ensures that all tasks, τ_j, for the parallel job 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 $Y_{(j,t)}$, which equals 1 when the jth job initiates its execution in the tth time-slot. Otherwise it equals 0. Constraint sets (16) and (17) define the variables s_j and f_j as the time-slot in which the jth job starts running and finishes, respectively. Variable s_j is obtained from variable $Y_{(j,t)}$, while variable f_j is calculated considering the slowest allocated resource. Constraint set (18) defines the execution cost ct_j of each job based on the processing slowdown obtained from the slowest allocated resource. Constraint sets (19) and (20) ensure that the time-slots used by a job are contiguous and in accordance with the time spent on the slowest allocated resource. Constraint set (21) defines the available bandwidth of the kth inter-cluster link in the tth time-slot, $ABW_{k,t}$. 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 was based on a synthetic workload varying the size of the time-slot. For the second study, *OAS* was compared with other heuristics present from the literature.

OAS was implemented by using the *CPLEX* linear mixed integer programming solver package, and the scheduling provided by *OAS* was used on the GridSim simulation framework, characterized as a heterogeneous multi-cluster system. Due to the complexity of the MIP model the environment was limited to 3 clusters, with 2 nodes per cluster and interconnected by a Gigabit network. The heterogeneity characteristic was defined by assigning a different effective power computation capability to each individual cluster with values of $\Gamma_k = \{1.0, 0.75, 0.5\}$ respectively, from greater to lesser computation capability.

4.1 Time-Slot Size Analysis

In order to determine the order in which jobs must be executed, *OAS* includes the concept of time-slot. The time-slot size could limit the quality of the scheduling solution, so an experimental study was performed to evaluate the impact of varying the time-slot size on the results. We defined a workload composed by 8 jobs with different computation and communication requirements, which are representative of parallel applications with very large computational requirements as could be weather prediction, fluid or material simulation, etc.

The time-slot values were in the range $\{15\%, \ldots, 250\%\}$ times the average of the base-time of the jobs from the workload. Figure 3 shows the makespan obtained for each case of the study. As can be observed, when the slot size increases, the makespan increases. This is because a big slot reduces the available resources during long periods of time even if the allocated jobs have finished. Thus, the start time of the next jobs is delayed until the time-slot finishes.

The shortest the time-slot is, the better makespan is obtained. However, the model has to do more evaluations, and the solving time increases as shown in Figure 3. It shows that from sizes lower than 50%, makespan values become stable, which are identified with the optimum makespan possible for this experiment. In the other way the solving time grows up exponentially as the problem size increases. By this, we can conclude that *OAS* is able to reach a near-optimal makespan in a feasible solving time without using the minimum time-slot.

4.2 *OAS* Performance Evaluation

To evaluate the goodness of *OAS*, we compare the results with other techniques present 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 an specific scheduling technique and then limiting any solver advantage. Thus, the first workload *WL-1* was designed to perform well with the techniques that try to match the available resources with the processing requirements from the jobs in the waiting queue. *WL-2* workload was designed to perform well with the techniques that prioritize smaller jobs. Finally, we also use randomly generated workloads *WL-3* to *WL-6*, designed without taking any particular criteria into account. The

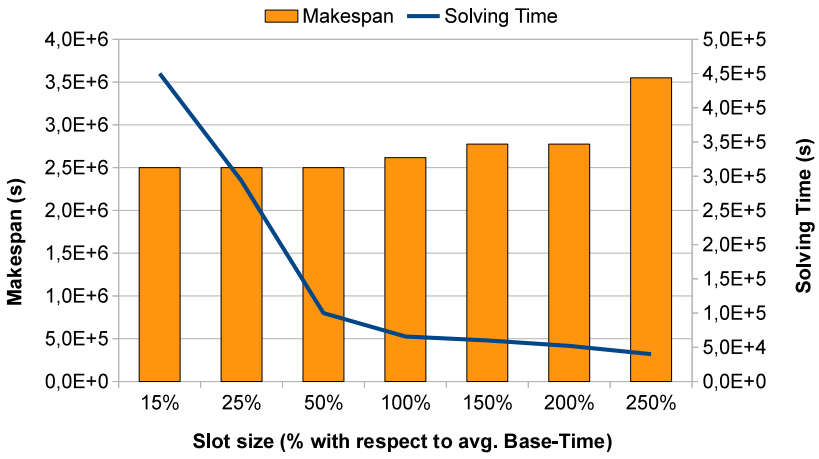

Fig. 3. Comparison for the model slot size.

metric used in the comparison was the makespan, which measures the total time to execute the complete workload and also is able to show the behavior of the selected heuristics when the treated set of jobs has a limited size. The time-slot size was defined to obtain a near-optimal makespan.

The results of this comparison are shown in Figure 4. It can be observed that each technique has a different behavior. The technique that in some workload performs well in another obtains a low makespan. It is important to take into account that *OAS* always obtained good results irrespectively of the workload nature. The reason comes from its ability to have a global vision of the requirements of the whole set of jobs and the available resources. Thus, we can conclude that by defining the correct job execution order it is possible to obtain the scheduling that reduces the final makespan.

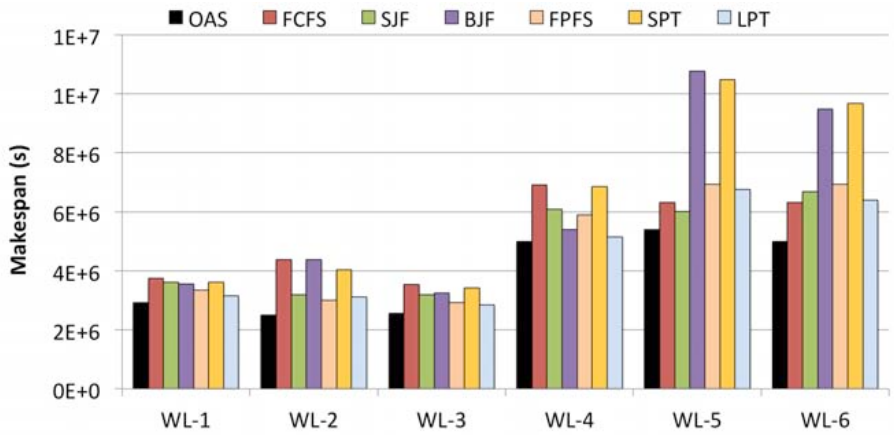

Fig. 4. Comparison of six kinds of workloads.

Finally, analyzing the solver decisions we identified a scheduling pattern. The set of jobs were allocated in a balanced way being the jobs with heavier processing requirements allocated to the most powerful nodes, and those with lower requirements to the nodes with less processing capacity. This knowledge provide us the seed to develop a heuristic able to be used on a real platform.

5 Conclusions

In the present work, we focused on the scheduling process on heterogeneous multi-cluster environments, by applying multiple job allocation and co-allocation when it is necessary. The goal is to determine the goodness 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 confirming that, both execution order and multiple-allocation provide better makespan results. However, the presented MIP model has a large computational complexity, by this we are developing a feasible scheduling heuristic that could be executed in realistic multi-cluster environments obtaining solutions in practical times and treating bigger sets of jobs.

References

[BE07] A.I.D. Bucur and D.H.J. Epema. Schedulling policies for processor coallocation in multicluster systems. *IEEE TPDS*, 18(7):958–972, 2007.

[BLG11] H. Blanco, J.L Lérida, and F. Guirado. Multiple job co-allocation strategy for heterogeneous multi-cluster systems based on linear programming. *Journal of Supercomputing*, 58(3):394–402, 2011.

[BMAV05] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm. *Software Practice and Experience*, 35(5):491–512, 2005.

[EHS+02] C. Ernemann, V. Hamscher, U. Schwiegelshohn, R. Yahyapour, and A. Streit. On advantages of grid computing for parallel job scheduling. In *IEEE/ACM International Conference CCGRID'02*, 2002.

[FRS05] D.G. Feitelson, L. Rudolph, and U. Schiwiegelshohn. Parallel job scheduling - a status report. In *LNCS*, volume 3277, pages 1–16, 2005.

[FSZX03] H. Feng, G. Song, Y. Zheng, and J. Xia. A deadline and budget constrained cost-time optimization algorithm for scheduling dependent tasks in grid computing. In *Grid and Cooperative Computing*, pages 113–120, 2003.

[GBS10] S. K. Garg, R. Buyya, and H. J. Siegel. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems*, 26:1344–1355, 2010.

[HFH08] E.M. Heien, N. Fujimoto, and K. Hagihara. Static load distribution for communicative intensive parallel computing in multiclusters. In *IEEE PDP'08*, pages 321–328, 2008.

[JAA07] B. Javadi, M.K. Akbari, and J.H. Abawajy. A performance model for analysis of heterogeneous multi-cluster systems. *Parallel Computing*, 32(11-12):831–851, 2007.

[JLPS05] W. Jones, W. Ligon, L. Pang, and D. Stanzione. Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. *Journal of Supercomputing*, 34(2):135–163, 2005.

[LSG+08] J.L. Lérida, F. Solsona, F. Giné, J.R. García, and P. Hernández. Resource matching in non-dedicated multicluster environments. In *VECPAR 2008*, pages 160–173, 2008.

[MLS07] E. U. Munir, J. Li, and S. Shi. Qos sufferage heuristic for independent task scheduling in grid. *Information Technology Journal*, 6(8):1166–1170, 2007.

[NLYW05] V.K. Naik, C. Liu, L. Yang, and J. Wagner. Online resource matching for heterogeneous grid environments. In *IEEE/ACM International Conference CCGRID'05*, volume 2, pages 607–614, 2005.

[SF05] E. Shmueli and D.G. Feitelson. Backfilling with lookahead to optimize the packing of parallel jobs. *Journal of Parallel & Distributed Computing*, 65(9):1090–1107, 2005.

[SHM97] D. B. Skillicorn, J. M. D. Hill, and W. F. McColl. *Questions and Answers about BSP*. Oxford University Computing Laboratory, 1997.

[SKSS02] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Selective reservation strategies for backfill job scheduling. In *Job Scheduling Strategies for Parallel Proc. JSSPP'02*, pages 55–71, 2002.

[SQR10] S.M. Hussain Shah, K. Qureshi, and H. Rasheed. Optimal job packing, a backfill scheduling optimization for a cluster of workstations. *Journal of Supercomputing*, 54(3):381–399, 2010.

[TEF07] D. Tsafrir, Y. Etsion, and D.G. Feitelson. Backfilling using system-generated predictions rather than user runtime estimates. In *IEEE Transaction on Parallel and Distributed Systems*, volume 18(6), pages 789–803, 2007.

[YTCC08] C. Yang, H. Tung, K. Chou, and W. Chu. Well-balanced allocation strategy for multiple-cluster computing. In *IEEE International Conference. FTDCS'08*, pages 178–184, 2008.

[ZCmH06] W. Zhang, A.M.K. Cheng, and m. Hu. Multisite co-allocation algorithms for computational grid. In *International Parallel and Distributed Processing Symposium, IPDPS'06*, 2006.