Impact of Communication Uncertainties on Workflow Scheduling in Hybrid Clouds

Luiz F. Bittencourt, Edmundo R. M. Madeira, and Nelson L. S. da Fonseca

Institute of Computing – University of Campinas – UNICAMP
Av. Albert Einstein, 1252, 13083-852 – Campinas – São Paulo – Brazil
e-mail: {bit, edmundo, nfonseca}@ic.unicamp.br

***Abstract*—The so-called hybrid cloud is the composition of an infrastructure that comprises private resources as well as public resources leased from public clouds. Hybrid clouds can be utilized for the execution of applications composed of dependent jobs, usually modeled as workflows. In this scenario, a scheduler must distribute the components of the workflow onto available resources considering the communication demands and the available bandwidth in network links. However, such information can be imprecise, and consequently decisions on resource allocation can be ineffective. In this paper, we evaluate scheduling algorithms in the face of imprecise information on the availability of communication channels. Results showed that schedules are negatively affected by the unforeseen variations in bandwidth during the execution of the application.**

I. Introduction

Cloud computing offers computational capabilities as services in a pay-per-use basis which can be computing power, storage, platforms for software development and execution, or simply a software interface accessible through the network. Depending on the type of service offered, the cloud is commonly classified as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS) [1]. Cloud customers can use/run applications from an SaaS cloud; develop and run their applications on a platform provided by a PaaS cloud; or extend their computational capacity by leasing computing power from an IaaS cloud.

In public clouds, clients are charged on a pay-per-use basis, and therefore it is preferable to use their infrastructure first (private cloud) and lease resources as needed. Such composition of private and the public clouds is called a hybrid cloud. In such clouds, several relevant questions need to be answered for applications to achieve high performance: how many and which resources should be leased to execute an application? Which application (or part of) should be executed on leased resources in order to improve performance and reduce costs? These questions are answered by a schedule produced by a scheduler which makes decisions on resource allocation based on criteria such as execution time and leasing.

Actually, the schedule produced by a scheduler can lead to poor performance when information provided as input to the scheduler is imprecise [2]. Imprecisions on both application demands and resource availability impact the execution time (makespan) of applications. In clouds, large delays introduced in the execution time of the applications imply also on cost increase. In hybrid clouds, uncertainties on the bandwidth availability can impact delays since Internet links are used to transmit data between private and public clouds; and customers usually do not know the network topology of the public clouds. Although virtualization isolates applications in clouds, the bandwidth of inter-cloud and intra-cloud links are shared without isolation, which can potentially create bottlenecks. Network bottlenecks need to be tackled in cloud computing, since they can increase costs and deteriorate performance [3].

To avoid makespan and budget underestimation in the workflow scheduling, clients need to adopt scheduling algorithms robust to uncertainties of input information. Thus, it is important to understand how existing workflow scheduling algorithms for clouds behave when information is not accurate. In this paper, the impact of uncertainties of bandwidth availability on the performance of two workflow scheduling algorithms is evaluated. Results indicate that the impact is more significant on low capacity links and that imprecise information can strongly degrade the performance of schedules.

II. Related Works

Cloud computing emerged from the evolution of paradigms such as service oriented computing, virtualization, and grid computing and several algorithms developed for these systems can be directly applicable to clouds, while others can be adapted. In line with that, several workflow scheduling algorithms developed for clouds were based on characteristics of previous algorithms developed for heterogeneous systems such as grids [4], [5].

A self-adaptive global search optimization technique called particle swarm optimization (PSO) is utilized to schedule workflows in the algorithm proposed in [6]. However, it does not consider workflow deadlines, focusing solely on monetary cost minimization.

Yu et al. [7] proposed a deadline-driven cost-minimization algorithm. The Deadline-MDP (Markov Decision Process) algorithm breaks a DAG into partitions, assigning a maximum finishing time for each partition according to pre-established deadline. Each partition is scheduled to the resource which results in both lowest cost and lowest estimated finishing time.

The Hybrid Cloud Optimized Cost (HCOC) algorithm [8] schedules workflows in hybrid clouds by first attempting a costless local schedule using the Path Clustering Heuristic

(PCH) [9]. If the local schedule does not satisfy the deadline, the algorithm selects jobs to be scheduled on resources in the public cloud. As in MDP, the objective is to minimize the monetary cost obeying the deadline stipulated by the user.

The Partial Critical Paths (PCP) algorithm [10] schedules the workflow backwards, searching for non scheduled partial critical paths. Constraints are added to the scheduling process when the scheduling of a partial critical path fails, making the scheduling process to start again. This algorithm presents the same characteristics of the MDP, however it has higher time complexity, since a potential high number of re-schedulings can occur during the algorithm execution.

Performance prediction is one way to improve the quality of information for the scheduler in grids [11], [12]. However, in hybrid clouds, prediction techniques cannot be applied because they are mainly based on historical usage patterns, therefore relying on constant resource monitoring. Monitoring in public clouds imply on computation on leased processing resources, leading to additional charges. However, performance prediction techniques can be successfully used by cloud provider to foresee usage patterns [13].

Techniques to deal with imprecision of input data have been developed such as dynamic scheduling [14], adaptative scheduling [9], rescheduling [15], and self-adjusting [16]. These approaches are reactive, i.e., they react to fluctuation of the resource availability during the *workflow* execution. The continuous monitoring needed during the workflow execution to perform reactive actions can produce imprecise information due to intrusion effects. Moreover, such approaches can lead to unnecessary job migration and overheads [17].

Batista and Fonseca [17] presented an algorithm based on fuzzy logic to schedule *workflows* in grids under uncertainty of both application demands and resource availability. The authors compared the proposed algorithm with static schedulers for heterogeneous systems. However, the impact of the uncertainty on makespan and budget is neither evaluated for utility grids nor for clouds, which is carried out by the work in the present paper.

III. Scheduling in Clouds

In this section, we define how a workflow can be represented and which system characteristics must be taken into consideration when scheduling dependent jobs. The scheduling algorithms used in the evaluation, namely the Hybrid Cloud Optimized Cost (HCOC) [8] and the Deadline-MDP [7] are also presented.

A. Workflow representation

A directed acyclic graph (DAG) is a common way of representing acyclic workflows. A DAG represents a workflow by associating nodes with jobs and arcs with data dependencies. A topological ordering of a DAG shows its jobs precedences. In this application model, a job can only start after all its precedences have been solved, i.e., after all its parent jobs have finished and sent the necessary data to the machine where the job in question will be executed.

Examples of real application workflows are the AIRSN [18] (Figure 1(a)), the Laser Interferometer Gravitational Wave Observatory (LIGO) [19] (Figure 1(b)), and the Montage [20] (Figure 1(c)). AIRSN normalizes sets of time series of 3D volumes to a standardized coordinate system and applies motion correction and Gaussian smoothing. The LIGO is a project to detect gravitational waves through a network of gravitational-wave detectors. Montage is an image application that makes mosaics from the sky for astronomy research.

In cloud computing, a workflow can represent a composition of services, therefore associating each job of the DAG with a service to be run in the cloud. Nevertheless, we use the term *job* to refer to any computational work to be executed, comprising both services and standard tasks.

B. Hybrid Clouds

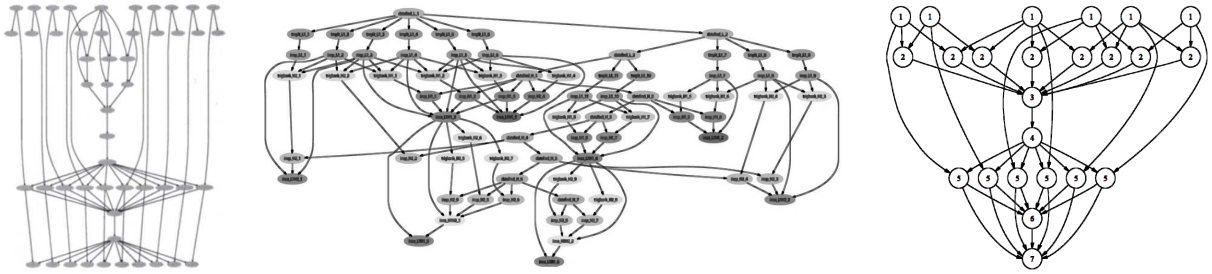
The computational demands of applications such as Montage, LIGO, and AIRSN can exceed the available computational power. The local computational resources can be scarce and the execution of large workflows can be infeasible. Moreover, local resources can be overloaded (with concurrent workflows and/or other jobs). Besides that, the workflow size can vary for the same application. For example, the size of the Montage workflow depends on the number of input images and it can be so large that the local available resources may not be able to handle it. Cloud computing paradigm deals very well with this problem by providing virtually unbounded on-demand resource provisioning.

Figure 2 illustrates a hybrid IaaS cloud scenario. The hybrid cloud is composed of resources inside the user's organization (resources from the private cloud – free of charge) and resources from one or more public IaaS clouds. This composition can be performed in two ways when leasing resources from the public clouds: lease resources on-demand or use long-term reserved resources. When leasing resources on-demand, clients are charged in a pay-per-use basis using a discretized time interval (1 hour, for instance). The same charging scheme is done when using reserved resources, however at a lower price in addition to a fee for the long-term contract (for instance, fee for a one year contract). The leasing terms can be specified in a service level agreement (SLA) stablished between client and provider. A hybrid cloud scheduler must be able to decide which resources should be leased from the public clouds to accomplish the workflow execution within the desired makespan or deadline.

C. Scheduling

To break the workflow in tasks for parallel execution on distributed resources, the scheduler must take into account the data transmitted between different machines. Therefore, intra- and inter-cloud communication needs to be considered, since the available bandwidth can contribute to the decrease in communication delays. Moreover, transmissions to and from the public cloud implies on monetary charges.

Currently, workflow schedulers consider makespans and costs, but uncertainties of information provided are not ad-



(a) AIRSN (b) LIGO (c) Montage

Fig. 1. Examples of workflow applications.

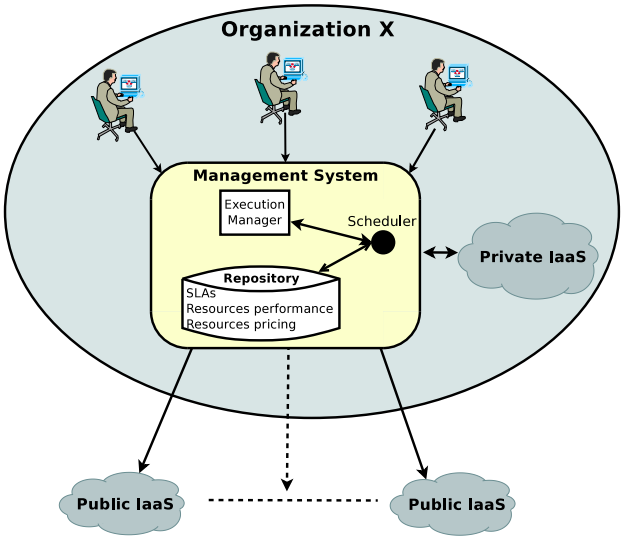


Fig. 2. Hybrid cloud infrastructure.

dressed. In this section, we present two algorithms that schedule workflows aiming at the minimization of cost within a given execution deadline stipulated by the user: the Deadline-MDP [7] and the Hybrid Cloud Optimized Cost (HCOC) [8].

The MDP algorithm was developed for utility grids, and its objective is to minimize costs while meeting user's deadlines. The first step of the MDP algorithm is to divide the jobs from the input DAG $\mathcal{G}$ into two categories: synchronization jobs and simple jobs. A job is a synchronization job if it has more than one parent or child job, and it is a simple job otherwise. After that, MDP creates clusters of jobs, or branches, where each branch has all jobs on the path along between two synchronization jobs. After the partition of the workflow, the overall deadline is distributed between each job v_i in $\mathcal{G}$. The deadline d_{v_i} assigned to v_i is a sub-deadline of the overall deadline $\mathcal{D}$, and it is computed in a way that each sub-deadline leaves enough time for the remaining jobs to run within other sub-deadlines. This deadline distribution considers the bandwidth of communication channels available between the machines running the jobs. Incorrect estimations of the available bandwidth can lead to deadline violations and increases in cost.

After the distribution of the deadline, the MDP finds a local optimal schedule for each partition based on its sub-deadline. It uses two different allocation strategies: the synchronization task scheduling (STS), for synchronization jobs, and the branch task scheduling (BTS), for each branch. When scheduling synchronization jobs with STS, the solution to a single job scheduling problem is simple. The optimal decision is to select the cheapest service that can process the job within the assigned sub-deadline. On the other hand, when branches are scheduled using BTS, the optimal schedule is computed by modeling the problem as a Markov Decision Process. The MDP scheduling is shown in Algorithm 1.

Algorithm 1 MDP Algorithm

```
1: Convert the input DAG G into job partition graph G_p
2: Distribute the deadline D over all vertex ∀v_i ∈ G_p
3: repeat
4:    S ← get unscheduled job partitions whose parent job
      partitions have been scheduled
5:    for all i ∈ S do
6:       query available time slots during ready time and sub-
         deadline on available services
7:       if i is a branch then
8:          compute an optimal schedule for i using BTS
9:       else
10:         compute an optimal schedule for i using STS
11:      end if
12:      adjust sub-deadline of i according to the current
         schedule
13:   end for
14: until all partitions have been scheduled
```

The HCOC is an algorithm developed for hybrid clouds that has the same objectives of MDP: to minimize cost while meeting the deadline. Its first step is to schedule the workflow in the private cloud resources using the path clustering heuristic (PCH) [9]. The PCH algorithm creates clusters of jobs that are in a path of the DAG. These paths are defined based on estimated computation and data transmission times. If the deadline is not met by the PCH initial schedule, HCOC selects iteratively a job to be rescheduled in a resource from the public cloud until the deadline is met. Job selection is based on priorities computed by traversing the DAG backwards, in

which the job farthest from the end of the DAG is selected. This job is then scheduled on the resource which gives the smallest estimated finish time (EFT) among the resources that present: (i) the minimum cost per core, and (ii) at most the same number of cores as the number of job clusters yet to be rescheduled. In HCOC, the selection of the job which will be rescheduled and the selection of the resource from the public cloud are directly influenced by the bandwidth available in the communication channels and, as in MDP, incorrect estimation can lead to deadline violations and increased costs. Algorithm 2 illustrates these steps in more details.

Algorithm 2 HCOC algorithm

1: $\mathcal{R}$ = all resources in the private cloud
2: Schedule $\mathcal{G}$ in the private cloud using PCH
3: **while** $makespan(\mathcal{G}) > \mathcal{D}$ AND $iteration < size(\mathcal{G})$ **do**
4: $iteration = iteration + 1$
5: select node n_i such that its priority is maximum AND $n_i \ni \mathcal{T}$
6: $\mathcal{H} = \mathcal{R}$
7: $\mathcal{T} = \mathcal{T} \cup t$
8: $num\_clusters$ = number of clusters of nodes in $\mathcal{T}$
9: **while** $num\_clusters > 0$ **do**
10: Select resource r_i from the public clouds such that $\frac{price_i}{num\_cores_i \times p_i}$ is minimum AND $num\_cores_i <= num\_clusters$
11: $\mathcal{H} = \mathcal{H} \cup \{r_i\}$
12: $num\_clusters = num\_clusters - num\_cores_i$
13: **end while**
14: **for all** $n_i \in \mathcal{T}$ **do**
15: Schedule n_i in $h_j \in \mathcal{H}$ such that EFT_i is minimum
16: Recalculate jobs attributes for each job
17: **end for**
18: **end while**

Both MDP and HCOC algorithms are suitable for hybrid clouds, since they consider execution costs and makespan of the schedule, as well as deadline estimation established by the user. However, additional care should be taken to avoid that the makespan does not extrapolate the budget and execution deadline for the application. Moreover, schedules given by both algorithms can be subject to uncertainties of the bandwidth estimations. In the next section we evaluate how MDP and HCOC perform in the presence of imprecise input data.

IV. Evaluation of scheduling algorithms

It is assumed that, initially, all input data is precise, i.e., all computation and communication demands from the DAG as well as all processing and bandwidth capacities correspond to those found when running the workflow application. To simulate variability in the presence of uncertainties, we introduce random percentual error in the scheduler input data uniformly distributed in the interval $[-p, +p]$, where p is a value between 0% and 100%. This variation is applied to all links in the hybrid cloud.

Our experiment comprises the following steps:

1) Create an instance of the problem:
   - DAG: The Montage, LIGO, and AIRSN DAGs are used, all with computation demands randomly taken from the interval $[5 \times 10^5, 4 \times 10^6]$ millions of instructions (MI) and communication demands randomly taken from the interval $[60, 500]$ Megabytes.
   - Private clouds have from 1 to 10 resources with processing capacities randomly taken from the interval $[10^4, 10^5]$ MIPS.
   - In public cloud, there are 4 types of resources. Compute unit capacities and leasing costs are equivalent to the Amazon EC2 *small, large, extra large*, and *extra large high CPU* on-demand instance types. Amazon defines a *compute unit*, from which resources processing capacities are derived. In our experiments, each compute unit was randomly taken from the interval $[10, 70]$, referring to the *small* instance type computing power. The 4 types of resources are: *small* (1 core of 1 compute unit, $0.085 per hour); *large* (2 cores of 2 compute units each, $0.34 per hour); *extra large* (4 cores of 2 compute units each, $0.68 per hour); *extra large high cpu* (8 cores of 2.5 compute units each, $0.68 per hour);
   - Define a communication tuple *(public; private; private to public)* that represents different values of bandwidth (in Mbps): the first value correspond to links in the private cloud, the second defines links among resources in the public cloud, and the last one the bandwidth of the links connecting the private and the public clouds (*inter-cloud* bandwidth).
   - Define a deadline $\mathcal{D}$ based on the makespan $\mathcal{M}$ of all jobs from the critical path of the DAG in the fastest resource in the private cloud.
2) Schedule the workflow using PCH considering resources only from the private cloud.
3) Schedule the workflow using HCOC and MDP.
4) For each communication link, introduce a percentual error p in the available bandwidth and recompute the makespan for the schedule given in the previous step.

The above steps were executed 2000 times for each DAG with communication tuples (100; 100; 5), (100; 100; 10), (100; 100; 50), and (100; 100; 100) for the evaluation of the impact of the bandwidth of the inter-cloud link on the makespan and cost. The percentual error p of the available bandwidth varied in the set $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 99\}$. A deadline $\mathcal{D} = 2.5$ was set. The computed metrics were the makespan and monetary costs of the resulting schedule, and the number of times each algorithm produced a schedule with makespan satisfying the deadline (number of solutions found).

Table I shows the number of executions that found solutions within the established deadline for each algorithm, PCH (in the

private cloud), HCOC, and MDP, and for the DAGs Montage (M), LIGO (L), and AIRSN (A), using the four defined network configurations. For HCOC and MDP, it is shown the total number of solutions found by the algorithms, and the number of solutions found excluding instances which PCH was able to find a solution in the private cloud (*excluding PCH sols.*). We observe that the larger the inter-cloud bandwidth is, the higher is the number of solutions found by HCOC and MDP, which attests the importance of the inter-cloud bandwidth for the workflow execution in hybrid clouds. Moreover, HCOC found more solutions than did MDP. In most of the cases, instances for which MDP found a solution were also found by the PCH, i.e., a small number of solutions were produced by MDP besides the solutions given by PCH.

Figure 3 presents the number of disqualified solutions, i.e., the number of solutions given by MDP and HCOC that met the deadline when assuming precise input data, but had their makespan increased beyond the deadline after the percentual error in the bandwidth was introduced. It is evident that uncertainties in the available bandwidth value impact the solutions found by both algorithms. Assuming precision of the input data can yield to a negative impact on the makespan. For example, a 60% uncertainty level of the available bandwidth value when using to schedule MDP the LIGO DAG disqualifies 78% of the 1472 solutions initially found, while when using the HCOC for the same uncertainty disqualifies only 19% of the 1989 solutions found.

Average costs, excluding PCH solutions in the private cloud, for different inter-cloud link capacities are shown in Figure 4. The impact of the imprecise values on costs is presented for all different bandwidth values between private and public clouds, reaching the highest values when the the inter-cloud bandwidth is set to 10 Mbps (for HCOC) and 5 Mbps (for MDP) for an uncertainty level of 99%.

Results showed that the imprecise input on the available bandwidth to the scheduler can significantly affect the makespan and the cost. Such error impacts the number of solutions that meet the deadline and the users' budget, emphasizing the importance of the development of uncertainty-aware schedulers for clouds.

Both MDP and HCOC algorithms, as many other scheduling algorithms in the literature, are strongly based on the premise that input data is precise and all the decision-making process is based on this input data. To mitigate this negative impact, algorithms could consider as input data for each parameter a probability distribution, avoiding reactive approaches such as dynamic scheduling, reducing intrusion and overhead.

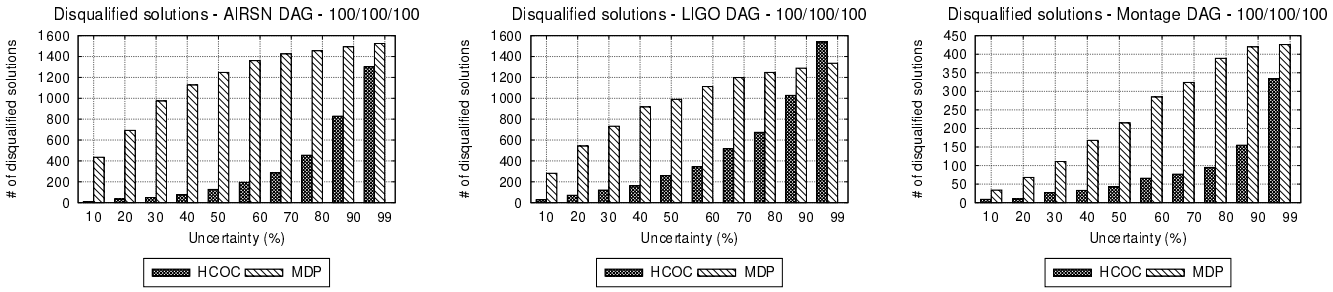
V. Conclusion

The precision of input data provided to the scheduler is essential to the effectiveness of the schedule generated by the scheduling algorithm. However, in clouds such precise data is hard to obtain due to a series of reasons, including: (i) concurrency in communication channels enlarges the makespans and costs; and (ii) lack of knowledge on the infra-structure such as network topology in public clouds.

Results obtained from simulations showed the relevance of precise input data, and that the generated schedule is negatively influenced by imprecise bandwidth estimations. As a consequence, deadline violation, long makespan, and high costs are common. Indeed, the study in this paper evinces the importance of developing scheduling algorithms which consider the quality of information in hybrid clouds

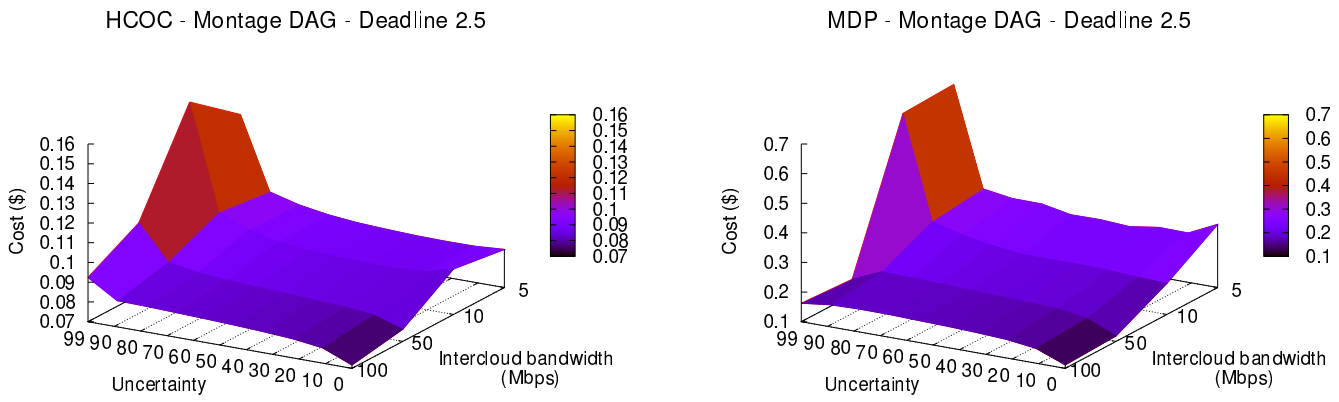
TABLE I
Number of solutions found within the deadline for DAGs Montage (M), LIGO (L), and AIRSN (A).

	100/100/5			**100/100/10**			**100/100/50**			**100/100/100**		
	(M)	(L)	(A)	(M)	(L)	(A)	(M)	(L)	(A)	(M)	(L)	(A)
PCH	1138	1068	1321	1120	1034	1274	1102	1039	1270	1113	1035	1259
HCOC	1643	1218	1811	1624	1419	1826	1616	1973	1945	1612	1989	1954
HCOC / excluding PCH sols.	505	154	490	504	388	552	514	934	675	499	954	695
MDP	985	138	302	992	118	289	991	344	443	1379	1472	1632
MDP / excluding PCH sols.	2	2	6	8	11	8	30	171	76	269	702	533



(a) AIRSN (b) LIGO (c) Montage

Fig. 3. Number of deteriorated solutions after the introduction of uncertainty.



(a) HCOC

(b) MDP

Fig. 4. HCOC and MDP average cost for Montage and four different inter-cloud link configurations, excluding PCH solutions in the private cloud.

As future work, we consider the development of scheduling algorithms that receive a range of probable values for each input parameter rather than a single value.

ACKNOWLEDGMENTS

The authors would like to thank FAPESP (2009/15008-1), CNPq, and RNP for supporting this work.

REFERENCES

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[2] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for scheduling parameter sweep applications in grid environments," in *Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th*, 2000, pp. 349 –363.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, pp. 50–58, Apr. 2010.

[4] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing: state of the art and open problems," Queen's University School of Computing, Kingston, Canada, Tech. Rep., jan. 2006.

[5] J.Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing." *Studies in Computational Intelligence*, vol. 146, pp. 173–214, 2008.

[6] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, april 2010, pp. 400 –407.

[7] J. Yu, R. Buyya, and C. K. Tham, "Cost-based scheduling of scientific workflow applications on utility grids," in *e-Science and Grid Computing*, july 2005, pp. 8 pp. –147.

[8] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, Aug. 2011.

[9] L. F. Bittencourt and E. R. M. Madeira, "A performance-oriented adaptive scheduler for dependent tasks on grids," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 9, pp. 1029–1049, 2008.

[10] S. Abrishami, M. Naghibzadeh, and D. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," in *11th IEEE/ACM International Conference on Grid Computing (GRID 2010)*, oct. 2010, pp. 81 –88.

[11] M. Kalantari and M. K. Akbari, "Grid performance prediction using state-space model," *Concurrency and Computation : Practice and Experience*, vol. 21, no. 9, pp. 1109–1130, Jun. 2009.

[12] H. Gautama and A. van Gemund, "Low-cost static performance prediction of parallel stochastic task compositions," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 78 – 91, jan. 2006.

[13] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Workshop on Cloud Management (CloudMan 2012), NOMS Workshop Proceedings*, USA, April 2012.

[14] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf, "The cactus worm: Experiments with dynamic resource discovery and allocation in a grid environment," *International Journal of High Performance Computing Applications*, vol. 15, p. 2001, 2001.

[15] R. Sakellariou and H. Zhao, "A low-cost rescheduling policy for efficient mapping of workflows on grid systems," *Scientific Programming*, vol. 12, no. 4, pp. 253–262, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1240160.1240165

[16] D. M. Batista, N. L. S. da Fonseca, F. K. Miyazawa, and F. Granelli, "Self-adjustment of resource allocation for grid applications," *Computer Networks*, vol. 52, no. 9, pp. 1762–1781, Jun. 2008.

[17] D. M. Batista and N. L. S. da Fonseca, "Robust scheduler for grid networks under uncertainties of both application demands and resource availability," *Computer Networks*, vol. 55, no. 1, pp. 3–19, 2011.

[18] Y. Zhao, J. Dobson, I. Foster, L. Moreau, and M. Wilde, "A notation and system for expressing and executing cleanly typed workflows on messy scientific data," *SIGMOD Records*, vol. 34, no. 3, pp. 37–43, 2005.

[19] E. Deelman, "Clouds: An opportunity for scientific applications? (keynote in the 2008 Cracow Grid Workshops)," 2008.

[20] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the challenges of scientific workflows," *IEEE Computer*, vol. 40, no. 12, pp. 24 –32, dec. 2007.