Using Relative Costs in Workflow Scheduling to Cope with Input Data Uncertainty

Luiz F. Bittencourt
Institute of Computing
University of Campinas
Av. Albert Einstein, 1251
Campinas, São Paulo, Brazil
bit@ic.unicamp.br

Rizos Sakellariou
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
rizos@cs.man.ac.uk

Edmundo R. M. Madeira
Institute of Computing
University of Campinas
Av. Albert Einstein, 1251
Campinas, São Paulo, Brazil
edmundo@ic.unicamp.br

ABSTRACT

Grids and clouds are utilized for the execution of applications composed of dependent tasks, usually modeled as workflows. To efficiently run the application, a scheduler must distribute the components of the workflow in the available resources using information about duration of tasks and communication between tasks in the workflow. However, such information may be subject to imprecisions, thus not reflecting what is observed during the execution. In this paper we propose a simple way of representing the costs of the components in a workflow in order to reduce the impact of uncertainties introduced by wrong estimations, and also to ease the application specification for the user. Evaluation shows that the use of relative costs in tasks and dependencies can improve in many cases the resulting schedule when compared to cases where the input data carries an uncertainty of 20% and 50%.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Cloud Computing*; D.4.1 [**Operating Systems**]: Process Management—*Scheduling*

General Terms

Algorithms

Keywords

Uncertainty, scheduling, workflow

1. INTRODUCTION

Scheduling algorithms rely on information about how long tasks and data transmissions take to occur during the application execution [10]. In systems such as grids and clouds, this application information is combined with information about the processing capacities and bandwidth available in the heterogeneous resources in order to estimate the execution time of the application components on different resources, allowing the scheduler to perform the decision making according to an objective function. Combining this information, the scheduler is capable of estimating the execution time for the application as a whole [13].

While the estimation of tasks execution times is important for any application, the estimation of data transmission times is especially important when dependent tasks are executed. Nowadays, a wide variety of scientific and business applications can be modeled as workflows [17]. These applications can take advantage of distributed systems such as grids and clouds to speed up their execution, and they are the focus of this work.

The input data for a workflow scheduler can be separated in two sets: the first one with information about the application components and their dependencies, and the second one with information about the performance of the resources (processors and links) in the target system. The information in the first set can be obtained from two sources: (i) the workflow programming model, where the user is responsible for estimating the "size" (or cost) of each task and each data dependency; and (ii) from previous executions of the same application, if available. The information about resource performance can be obtained from its hardware characteristics as well as from benchmarks. An important aspect of both sets is that they are prone to inaccuracy, introducing uncertainty to the scheduling process [4]. The imprecision in the input information given to the scheduler is associated to the concept of *quality of information* [7].

In this paper we focus on the first set of information, which provides data about the application. Scheduling algorithms generally consider this information to be precise, using it during the decision making process without taking into account possible variations of the task duration and data transmission times. However, it is hard to estimate task computation demands or data dependency sizes for all application components. Because of that, estimates given by the scheduler for the workflow execution duration can be often impaired by low quality of information.

In this paper we propose a novel way of interpreting the input data given by the user for the workflow execution. In the proposed approach, the scheduler does not need to know the estimated cost of each task or data dependency, but only their relative costs. With this, the user can provide to the scheduler only a list of tasks and data dependencies sorted by their expected duration, however without any numeric value.

Note that we are not dealing with performance fluctuations in the resources, and therefore dynamic rescheduling is not effective. As a first evaluation of the proposal, we used it with the well known heuristic called *Heterogeneous Earliest Finish Time* (HEFT) [18]. Preliminary results show that providing only a relation of costs for tasks and transmission times can result in schedules in many cases with lower makespan than when the input data carries an uncertainty of 50%.

This paper is organized as follows. Section 2 presents related work. In Section 3 we present some background and motivation behind this work. The proposed use of relative costs in workflow scheduling is presented in Section 4, along with a description on how it is applied to HEFT. The evaluation is shown in Section 5, and the conclusion is presented in Section 6.

2. RELATED WORK

Scheduling in both homogeneous and heterogeneous distributed systems is vastly studied. Mostly, scheduling efforts have been directed to heuristics [18] and meta-heuristics [12]. There are also works in approximation algorithms for scheduling [15], and mixing integer linear programming with heuristics [11]. Extensive description of the scheduling problem and solutions for computational grids were done by Akl and Dong [10] and Yu et al. [13].

Batista and Fonseca [4] presented a fuzzy-based algorithm to deal with uncertainties for the workflow scheduling problem in grids. The authors compare their algorithm with static ones, including HEFT. However, only a superficial evaluation of HEFT performance in the presence of uncertainty, with a 4-node workflow, was performed. In the present work we extend this evaluation, showing HEFT performance in a variety of scenarios with uncertainty, and also how it performs when our relative cost approach is used.

Other approaches for uncertainty, such as dynamic scheduling [1], adaptive scheduling [6], re-scheduling [16], and self-adjusting [5] can be found in the literature. Such approaches react to fluctuations on the resource performance during the application execution, i.e, they are /em reactive. The continuous monitoring needed during the application execution to perform reactive actions can produce imprecise information due to intrusion effects and may also lead to unnecessary job migration and overheads [4].

Internet-based computing scheduling (IC-scheduling) [14] has the goal of maximizing the number of eligible tasks for execution at each scheduling step. While many traditional scheduling algorithms assume to have precise information about the costs of the workflow components, IC-scheduling assumes no knowledge about such costs. Our proposal falls in the middle of these two approaches, assuming high level knowledge about the costs of the workflow components.

We propose an approach to avoid reactive actions by producing a reasonable schedule in the face of uncertainty in the input data about the application. In this case, since in the considered scenario the problem does not lie with resource performance, reactive approaches like dynamic rescheduling would be ineffective because rescheduling would be continuously performed using imprecise input data about the application. Moreover, in a cloud computing scenario, reactive techniques could lead to more expensive execution if the rescheduling results in the utilization of resources yet to be leased. Our proposal is new in the sense that it does not rely on numeric data provided as input to the scheduler, but only on relative values among tasks and data transmission times.

3. BACKGROUND

Grids and clouds are heterogeneous distributed systems that exist over shared networking and computing resources. Grids are shared distributed systems that allow users to perform computation over a set of heterogeneous processors to collaborate towards a common objective. On the other hand, clouds are systems that isolate resources through virtualization [3] and that can be used in a pay-per-use basis. A remarkable characteristic of the cloud computing paradigm is *elasticity*, which provides on-demand extension of a private pool of resources during high-peak loads. With this, the cloud client pays only the effective use of those resources, avoiding upfront investments, maintenance costs, and idleness during low-peak demand.

The elasticity provided by clouds allows the composition of the so-called *hybrid clouds*, where the user has its own private cloud (which can be a computational grid) and leases resources from public clouds when the private pool of resources is not enough to attend the demand. During the composition process, the scheduler has a main role in deciding whether a task should execute in the private cloud or in the public cloud. This decision making process is done based on data about both characteristics of the application and of the computational resources, supported by an objective function to be optimized.

The application model considered in this paper is a workflow represented by a Directed Acyclic Graph (DAG) $G = (T, E)$, where each node $t \in T$ is a task and each edge $e \in E$ represents a data dependency [1]. In the DAG, each node and edge has, respectively, a computation cost and a communication cost (amount of data) associated. Since grids and clouds are heterogeneous, the processing time of each task and the data transmission times for each dependency are computed based also on the knowledge about the performance of each resource.

Incorrect estimations related to the application components (i.e., nodes and edges costs) are one point of uncertainty that affects the quality of information. Such estimates can come from the programming model or from a history of executions of the same application. In both cases, imprecise data will lead the scheduler to perform the decision making using information about tasks and data transmission durations that will not turn out to be true during execution. As a consequence, the estimated execution time of the application is distorted.

The uncertainty introduced in these estimations leads to a negative impact in the schedule at running time, consequently delaying the application execution as well as rising monetary costs when using the elasticity provided by one or more public clouds. In this scenario, mechanisms to avoid makespan augmentation and budget overrun should be developed. Uncertainty-aware schedulers are one way of reducing misleading information to heavily impact the applications makespan, also minimizing unforeseen monetary costs in the application execution in hybrid clouds. In this paper we address the problem of imprecise estimation of application component costs (i.e., size of tasks and data de-

[1]The terms DAG and workflow are used interchangeably in this text.

pendencies). By considering only relative costs among the workflow components, we avoid numeric estimations for the size of each component, which is a source of uncertainty.

4. THE RELATIVE COST SCHEDULING

We propose that scheduler's decision making relies on information of relative cost (or relative size), instead of relying on numeric information about tasks and data dependency sizes. With this, the scheduler input should not represent a numeric estimation of how long a task will take to run, but a hint if a task will take longer than the other tasks in the workflow. The same is valid for data dependencies among tasks. The motivation behind this proposal is that it is easier to the user to sort tasks by their duration than to numerically estimate their duration. More than that, a numeric estimation given by the user is likely to be inaccurate, introducing uncertainty and worsening the scheduling results. By providing a higher level information to the scheduler, we aim at avoiding decisions carried out over imprecise data (assumed to be precise by the scheduler), and we expect the resulting schedule to be better than when the user numeric estimations are not reliable.

4.1 Relative cost assignment

In order to avoid precise information about the workflow components to be required from the user, we propose three different approaches based on the relative costs of tasks and dependencies: *relative costs of tasks and edges* (RCTE), *relative costs with CCR* (RC-CCR), and *relative costs altogether* (RCA).

4.1.1 Relative costs of tasks and edges (RCTE)

In the relative costs of tasks and edges (RCTE) approach, tasks and edges are separately sorted into two sets. To accomplish this, the user must separate the tasks of the DAG into sets T_i such that all tasks in each T_i have the same relative cost. This can be done by assigning a relative cost $w_i = i$ to all tasks in T_i. In practice, this means that the user thinks all tasks in a set T_i have the same execution times. We define a partially ordered set (poset) $\mathcal{T} = \{T_1, T_2, ..., T_p\}$, $p \leq |T|$, where $w_{t_i} < w_{t_{i+1}}$, $1 \leq i \leq p-1$, which is used as the scheduler input. Similarly, let E_j be a subset of E such that all edges in E_j have the same weight $w_j = j$. We define a poset $\mathcal{E} = \{E_1, E_2, ..., E_q\}$, $q \leq |E|$, where $w_{e_j} < w_{e_{j+1}}$, $1 \leq j \leq q-1$, as another input to the scheduler.

A hypothetical example of a Montage workflow with relative costs given by the user is shown in Figure 1. Tasks in the second level have relative cost of 1 (the fastest execution), thus being in the set $T_1 \in \mathcal{T}$. Tasks from the fifth level would be in T_2, meaning that they would take longer to run then tasks from the second level, and so on. The same reasoning is applied to the edges of the DAG, separating them in another set. For example, all edges linking tasks from the first level to the second level have relative cost of 1, thus they would be in the set first set E_1 of the partially ordered set $\mathcal{E}$.

Thus, in RCTE, the scheduler receives as input a set of tasks sorted by their weights, where a task with larger weight means only that it "takes longer to run" than tasks with smaller weights if they are placed in the same processor. The same is valid for the data dependencies: a dependency with larger weight should take longer to be transmitted than

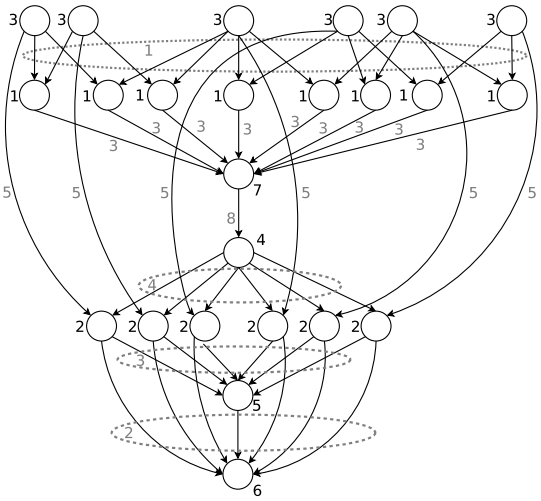

Figure 1: Montage DAG with relative costs.

dependencies with smaller weights when the transmission is done over the same link. Note that both the posets $\mathcal{T}$ and $\mathcal{E}$ can be automatically computed from a DAG with numeric costs associated to its nodes and edges, which is a common input of workflow scheduling algorithms, relieving the user from manually specifying the relative costs.

4.1.2 Relative costs with CCR (RC-CCR)

We assume that the user provides additional information about the application along with the relative costs: the computation to communication ratio (CCR) of the workflow. Given a DAG $G = (T, E)$, the CCR is defined in Equation 1.

$$CCR = \frac{\sum_{t \in T} w_t}{\sum_{e \in E} w_e}. \quad (1)$$

In this approach, after assigning weights w_t and w_e separately for tasks and edges in the same way as in the RCTE approach, the weight of each task is multiplied by a factor $c = \frac{CCR(|T|+1)}{|E|+1}$, computed as follows:

$$\begin{aligned} CCR\frac{1}{|T|}\sum_{t \in T} w_t &= c\frac{1}{|E|}\sum_{e \in E} w_e \\ \frac{CCR}{|T|}\left(\frac{|T|^2+|T|}{2}\right) &= \frac{c}{|E|}\left(\frac{|E|^2+|E|}{2}\right) \\ &\vdots \\ c(|E|+1) &= CCR(|T|+1) \\ c &= \frac{CCR(|T|+1)}{|E|+1} \end{aligned}$$

where CCR is the communication to computation ratio given by the user, $|T|$ is the number of tasks, and $|E|$ is the number of edges in the DAG. After the multiplication of each task relative weight by c, the generated DAG with relative costs will have the CCR given by the user.

4.1.3 Relative costs altogether (RCA)

In this approach, tasks and edges are sorted in a single set and weighted according to the same rules of the RCTE approach.

4.2 Relative HEFT

The three ways of "relaxing" the input data presented above are intended to make it easier for the user to provide to the scheduler useful information about the application. These relative costs are used as input for the scheduling algorithm. In this paper we evaluated how this relative assignment approaches perform when used within the HEFT algorithm.

The *Heterogeneous Earliest Finish Time* heuristic (HEFT) [18] has been one of the most often used, having the advantage of simplicity and producing generally good schedules with a short makespan. HEFT is essentially a list scheduling heuristic that constructs first a priority list of tasks and then makes locally optimal allocation decisions for each task on the basis of the task's estimated finish time. The priority value $rank_u$ is computed by traversing the DAG backwards and using information about the application and the target system. The $rank_u$ for each task t_i is defined in Equation 2.

$$rank_u(t_i) = \overline{w_i} + \max_{t_j \in suc(t_i)} (\overline{c_{i,j}} + rank_u(t_j)), \qquad (2)$$

where $\overline{w_i}$ is the average execution time of t_i in all resources, $suc(t_i)$ is the set of immediate successors of t_i, and $\overline{c_{i,j}}$ is the average communication cost between t_i and t_j. With this, the $rank_u$ of a task t is the weight of t plus the maximum value resulted from the weight of each successor of t added to the weight of the edge that connects this successor to t. After that, tasks are scheduled in non-decreasing order of $rank_u$ in the resource which results in the smallest estimated finish time (EFT).

With the introduction of our relative weights, the $\overline{w_i}$ and $\overline{c_{i,j}}$ values are actually computed over the relative costs given to each task and each dependency according to one of the three proposed approaches. For example, following the notation in Section 4, one has to replace $\overline{w_i}$ with the average over all w_{t_i}, $w_{t_i} \in T$, and $\overline{c_{i,j}}$ with the average over all w_{e_j}, $w_{e_j} \in E$, to use the versions based on relative costs. As a consequence, the estimated finish time (EFT) computed by the standard version of HEFT cannot be assumed to be an actual estimation of the finish time for the workflow, since no data about tasks or communication sizes are known by the algorithm.

5. EVALUATION

We performed the evaluation by scheduling a set of real world workflow applications, namely Montage (24 tasks) [9]; AIRSN (51 tasks) [19]; LIGO (166 tasks) [8]; Chimera-1 (43 tasks) [2]; and Chimera-2 (123 tasks) [2]. The maximum number of computational resources were set to 10 and to 50. A maximum of 50 resources means that the simulation was run with from 2 to 50 resources, with this quantity randomly taken. Each resource had its processing capacity randomly taken in the interval $(10, 100)$ from a uniform distribution. These resources were fully connected by a heterogeneous network, where each link between two resources had its bandwidth randomly taken in the interval $(10, 100)$ from a uniform distribution. For each type of DAG, we performed simulations using communication to computation ratios (CCR) of 0.5, 1.0, and 2.0. The metric is the makespan, and the presented averages are over 500 runs.

In order to evaluate our relative cost proposal, we have set up scenarios where HEFT was executed under uncertainty in the workflow input data. To model this uncertainty, we used the *quality of information* concept [7]. We assumed that, initially, all estimates were 100% correct, i.e., the computation and communication costs associated to the DAG and given to the scheduler match what is found during the workflow execution. After running HEFT and performing the schedule, to simulate the variability of the quality of information in the presence of uncertainty, we introduced a percentage error to each task and edge cost. The error is uniformly distributed in the interval $[-p, +p]$, where p is a value between 0% and 100%. After that, we recomputed the makespan of the schedule to obtain the final workflow makespan affected by the uncertainty. We have run simulations for $p = 0$ (i.e., the HEFT without modification), $p = 20$ and $p = 50$ (named HEFT / p=20 and HEFT / p=50 in the result graphs).

The proposed approaches were evaluated using the same DAGs generated for the HEFT scheduling, but the nodes and edges costs were replaced by relative values as described in Section 4.

Figure 2 presents results for all application DAGs. For the Montage DAG, we observe that for any CCR and for both 10 and 50 maximum number of resources, the quality of information is important, since the makespan is incorrectly estimated when $p = 20$ and $p = 50$. On the other hand, we observe that the proposed approach RCA (relative costs altogether) is able to perform better than when $p = 50$ in all cases, and even better than when $p = 20$ in some cases, approximating the makespan given by HEFT without uncertainty ($p = 0$) when there are at most 50 resources.

For the other DAGs, the quality of information importance remains clear, with the average makespan always increasing with p. For example, the uncertainty $p = 50$ introduced an error in the makespan estimation of up to 14% for the LIGO DAG when there were up to 10 resources, up to 17% for the DAG Chimera-1, and up to 15% for the DAG Chimera-2.

Regarding the proposed relative costs input data, for the AIRSN DAG all the relative approaches performed better than HEFT with $p = 50$ for $CCR = 1.0$ and $CCR = 2.0$. When $CCR = 0.5$, only the RCTE approach resulted in higher average makespan than HEFT with $p = 50$. A slightly different behavior is observed for the LIGO DAG, where the proposed relative costs performed better than $p = 50$ and equivalently to $p = 20$ only when there were at most 50 resources. When there were at most 10 resources, RC-CCR and RCA performed slightly worse than $p = 50$ for $CCR = 2.0$, while the RCTE was worse than $p = 50$ for all CCRs. On the other hand, RCA performed quite well for all DAGs when $CCR = 0.5$.

The variation in the quality of information was observed to have a negative impact on the HEFT algorithm. The evaluation showed that the input data provided for the scheduler, when subject to uncertainties, can significantly affect the application makespan when compared to the estimated by the scheduler. In grids, this error can lead to delays in the workflow execution. In clouds, besides the delay in the execution, this can also compromise the application budget by taking longer to run in paid resources.

6. CONCLUSION

In the workflow scheduling, data about the computational costs of application components is important in the quality of the generated schedule. However, such data is hard to obtain in a precise manner due to lack of knowledge about

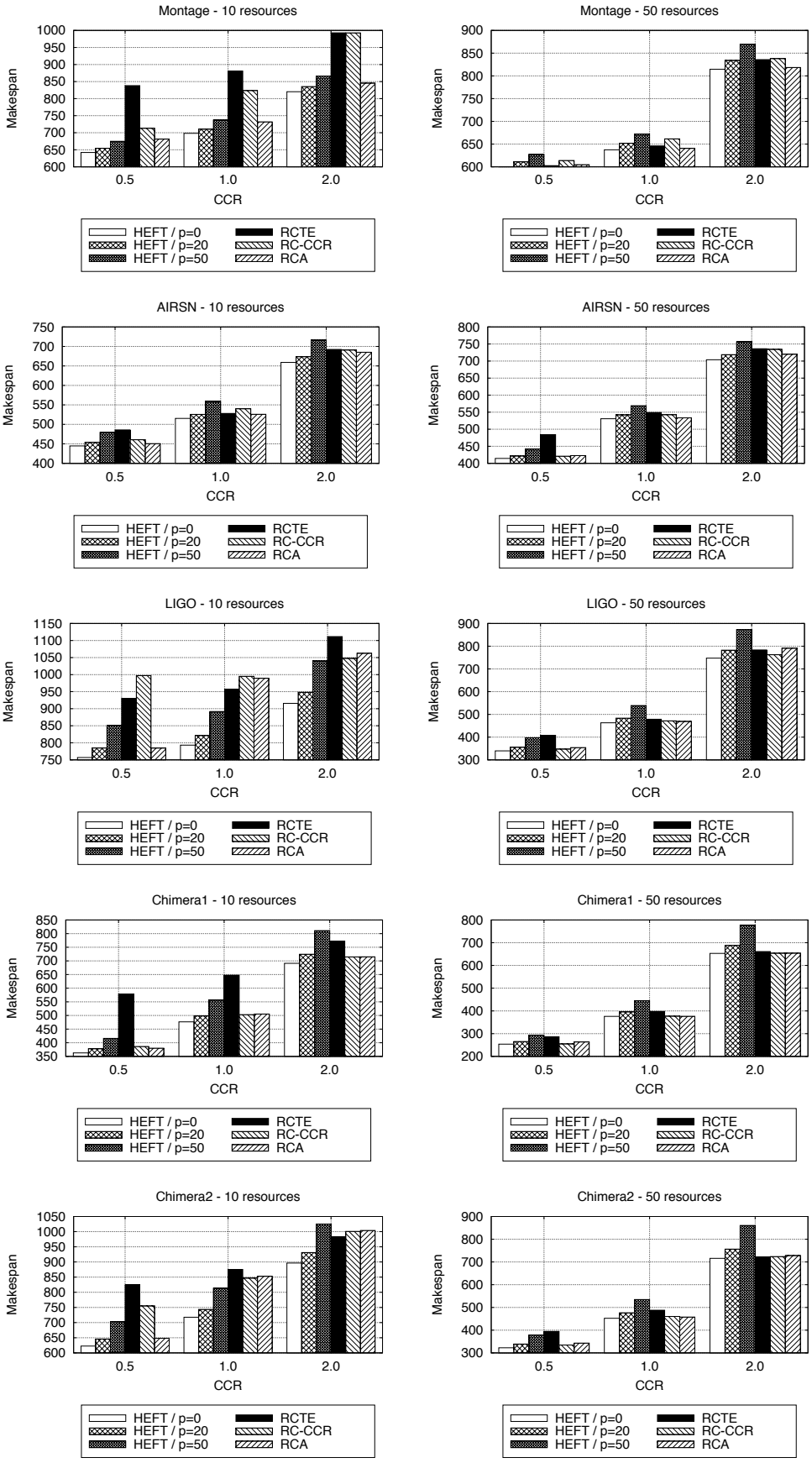

Figure 2: Average makespan for each DAG.

the application or varying behavior according to the application input data. Since most workflow scheduling algorithms have their decision based on computational costs of the application, this information must be supplied by the user or from estimations based on past executions. In both cases, providing numeric values and expecting them to help the scheduler may be actually harmful, introducing uncertainty and worsening the application makespan. The more imprecise is the data, the more noticeable is the negative impact in the makespan.

In this paper we argue that this input data about the computational costs can be provided as a relation among components instead of numeric values that represent the expected duration of each workflow component. We propose three different ways of specifying this relative costs. Evaluation results show that specifying relative costs can actually improve the schedule in most cases when compared to scenarios where the numeric estimation given to the scheduler is up to 50% uncertain. This leads us toward the investigation of new semantics to specify schedulers input data, avoiding the requirement of numeric values, which are unlikely to be precise.

7. ACKNOWLEDGMENTS

The first and third author would like to thank FAPESP (2009/15008-1) and CNPq for the financial support.

8. REFERENCES

[1] 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*, 15:2001, 2001.

[2] J. Annis, Y. Zhao, J. Voeckler, M. Wilde, S. Kent, and I. Foster. Applying Chimera virtual data concepts to cluster finding in the Sloan Sky Survey. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pp 1-14, Baltimore, USA, 2002. IEEE Computer Society Press.

[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*, 53:50–58, Apr. 2010.

[4] 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*, 55(1):3–19, 2011.

[5] D. M. Batista, N. L. S. da Fonseca, F. K. Miyazawa, and F. Granelli. Self-adjustment of resource allocation for grid applications. *Computer Networks*, 52(9):1762–1781, June 2008.

[6] L. F. Bittencourt and E. R. M. Madeira. A performance-oriented adaptive scheduler for dependent tasks on grids. *Concurrency and Computation: Practice and Experience*, 20(9):1029–1049, 2008.

[7] 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*, pages 349 –363, 2000.

[8] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda. GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, Edinburgh, UK, 2002. IEEE Computer Society.

[9] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

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

[11] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira. Workflow Scheduling for SaaS / PaaS Cloud Providers Considering Two SLA Levels. In *IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, Maui, USA, 2012. IEEE.

[12] M. Grajcar. Genetic list scheduling algorithm for scheduling and allocation on a loosely coupled heterogeneous multiprocessor system. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, New York, USA, 1999. ACM Press.

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

[14] G. Malewicz, A. L. Rosenberg, and M. Yurkewych. Toward a theory for scheduling DAGs in internet-based computing. *IEEE Transactions on Computers*, 55:757–768, 2006.

[15] C. Phillips, C. Stein, and J. Wein. Task scheduling in networks. *SIAM Journal on Discrete Mathematics*, 10(4):573–598, 1997.

[16] R. Sakellariou and H. Zhao. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. *Scientific Programming*, 12(4):253–262, Dec. 2004.

[17] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields. *Workflows for e-Science. Scientific Workflows for Grids*. Springer, 2007.

[18] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.

[19] 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 Record*, 34(3):37–43, 2005.