Computer Science Review 30 (2018) 31–54

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Scheduling in distributed systems: A cloud computing perspective

Luiz F. Bittencourt [a,*], Alfredo Goldman [b], Edmundo R.M. Madeira [a], Nelson L.S. da Fonseca [a], Rizos Sakellariou [c]

[a] Institute of Computing, University of Campinas, Brazil
[b] Instituto de Matemática e Estatística, Universidade de São Paulo, Brazil
[c] School of Computer Science, University of Manchester, UK

ARTICLE INFO

Article history:
Received 25 April 2018
Received in revised form 15 August 2018
Accepted 17 August 2018
Available online 13 September 2018

Keywords:
Distributed systems
Scheduling
Cloud computing

ABSTRACT

Scheduling is essentially a decision-making process that enables resource sharing among a number of activities by determining their execution order on the set of available resources. The emergence of distributed systems brought new challenges on scheduling in computer systems, including clusters, grids, and more recently clouds. On the other hand, the plethora of research makes it hard for both newcomers researchers to understand the relationship among different scheduling problems and strategies proposed in the literature, which hampers the identification of new and relevant research avenues. In this paper we introduce a classification of the scheduling problem in distributed systems by presenting a taxonomy that incorporates recent developments, especially those in cloud computing. We review the scheduling literature to corroborate the taxonomy and analyze the interest in different branches of the proposed taxonomy. Finally, we identify relevant future directions in scheduling for distributed systems.

Contents

* Corresponding author.
E-mail addresses: bit@ic.unicamp.br (L.F. Bittencourt), gold@ime.usp.br (A. Goldman), edmundo@ic.unicamp.br (E.R.M. Madeira), nfonseca@ic.unicamp.br (N.L.S. da Fonseca), rizos@cs.man.ac.uk (R. Sakellariou).

https://doi.org/10.1016/j.cosrev.2018.08.002
1574-0137/

1. Introduction

The scheduling problem arises in countless areas, and it evolves over time along with industry and technology [1]. With the development of computers, scheduling in computer processors received great attention [2,3], being the most common objective the minimization of task completion times, also known as makespan. Besides some peculiarities, the basic principles remained the same as in scheduling activities among machines in production. In supercomputers, multiprocessor scheduling considers several parallel processors with the same capacity. In addition, the data source is considered to be centralized and connected by a high speed channel between processors, in a way that activities (or jobs) can exchange messages quickly.

More recently, computer networks allowed clusters of homogeneous computers to act as a multiprocessor computer with distributed data sources. However, when compared to supercomputers, clusters initially had a slow communication channel between processors, which made data exchange among processors more expensive. The scheduling of jobs in computing clusters led to another branch of research: the scheduling in distributed computer systems. With improvements in computer networks, the connection among computing nodes in clusters became faster. On the other hand, new applications demanded more and more bandwidth, storing and exchanging massive volumes of data. Multimedia and e-Science are examples of applications that handle large data sets nowadays, putting in evidence the importance of communications to improve performance and support quality of service offering in distributed systems.

Grid computing emerged in the late 90's as a heterogeneous collaborative distributed system [4] evolved from homogeneous distributed computing platforms. Grids are shared systems that enclose potentially any computing device connected to a network, from workstations to clusters. Computing grids are infrastructures that enable resource sharing by establishing use policies as well as security rules, which compose the so called Virtual Organizations (VOs) [4].

Cloud computing offers computing resources, often virtualized, as services to the users, hiding technical aspects regarding resource management [5]. Therefore, clusters and grids can be part of datacenters in the cloud computing infrastructure, demanding new optimization objectives and variables common in green computing [6] and utility computing [7]. Kwok and Ahmad stated in 1999 [3] that considering heterogeneous platforms was a challenging direction to extend scheduling algorithms. As a consequence of the popularization of these platforms in grid and cloud computing, novel scheduling concepts appeared in the literature [8–12]. While on the one hand fundamental scheduling aspects remain unchanged, on the other hand different optimization objectives ballooned the scheduling literature in the past decade. Such swell in the field brought so much information that it became challenging the recognition of the exact contribution of new results. Since challenges in scheduling still exist, Smith argues in [13] that scheduling is not a fully solved problem, he stated that *"Scheduling techniques that properly account for uncertainty, enable controlled solution change, and support efficient negotiation and refinement of constraints are crucial prerequisites, and the need to operate in the context of multiple self-interested agents is a given."*. This statement matches certain characteristics in virtualization and service in cloud computing, as we shall describe in the upcoming sections.

In a nutshell, this paper has three main contributions:

1. Propose a taxonomy for scheduling in distributed systems and introduce a taxonomy extension to cover cloud computing schedulers.
2. Classify the literature in the proposed taxonomy;
3. Identify relevant future directions for scheduling in distributed systems.

Due to its wide application, there exist a variety of approaches to the scheduling problem. This paper presents directives for scheduling researchers to identify and classify their work, as well as to provide them with an overview of existing approaches associated to their research by presenting a broad view of the scheduling problem in distributed systems as well as introducing existing works. We first introduce the problem of scheduling in distributed systems, covering advances in scheduling in cluster and grid computing. Then, an overview of the proposed taxonomy is presented, followed by the state-of-the-art in each branch of the taxonomy tree. After that, we focus on cloud computing and detail similarities and differences of scheduling in clouds with scheduling in previously existing distributed systems. We introduce a taxonomy of scheduling in cloud computing, extending the pre-cloud taxonomy. Finally, we discuss research challenges that were inherited from grid and cluster computing by the cloud computing paradigm, as well as new problems to be tackled.

This paper is organized as follows. Section 2 discusses previous work that have addressed reviews and surveys of scheduling in distributed computing. Section 3 introduces basic concepts to define scheduling problem. Section 4 briefly introduces the whole taxonomy discussed in this paper, highlighting branches that were introduced to cover scheduling in cloud computing. Section 5 presents a taxonomy of schedulers previously to the advent of cloud computing (*pre-cloud* taxonomy). Section 6 discusses and classifies the scheduling taxonomy in cloud computing (*cloud taxonomy*). Section 7 reviews the cloud computing scheduling literature according to the proposed taxonomy. Future directions in scheduling for distributed systems are discussed in Section 8, and Section 9 presents the concluding remarks.

2. Related work

In computer science, with the constant networking and middleware development, scheduling in distributed processing systems is one of the topics which has gained attention in the last two decades. Casavant and Kuhl [14] present a taxonomy of scheduling in general purpose distributed systems. The classification presented by the authors include local and global, static and dynamic, distributed and non-distributed, cooperative and non-cooperative scheduling, as well as some approaches to solve the problem, such as optimal and sub-optimal, heuristic, and approximate. This presented classification is complete in some sense, and it is still valid nowadays. However the current state of distributed systems indeed demands the addition of new branches in this taxonomy.

Kwok and Ahmad [3] survey static scheduling algorithms for allocating tasks connected as directed task graphs (DAGs) into multiprocessors. The authors presented a simplified taxonomy for

approaches to the problem, as well as the description and classification of 27 scheduling algorithms. The DAG scheduling algorithms for multiprocessors have been adapted for scheduling in distributed systems, incorporating intrinsic characteristics of such systems for an enhanced performance. Therefore, Kwok and Ahmad presented static scheduling algorithms for multiprocessors, which are also applicable to distributed systems, and their classification. In this paper we review extensions of those algorithms as well as the their classification by including heterogeneous systems, dynamic scheduling algorithms, scheduling algorithms in modern distributed environments, and new scheduling techniques.

In the last decades, after Kwok and Ahmad's work, other surveys and taxonomies for solutions to the scheduling problem for parallel systems have been developed. Most of these works focus on heterogeneous distributed systems [15], which Ahmad and Kwok considered as one of the most challenging directions to follow [3].

Job scheduling strategies for grid computing are evaluated by Hamscher et al. in [11]. The authors present common scheduling structures, such as centralized, decentralized, and hierarchical. Within each scheduling structure, they present and evaluate 4 processor selection strategies and three scheduling algorithms, namely *First-Come-First-Serve* (FCFS), *Random*, and *Backfill*. After this work, many dynamic scheduling strategies were developed to tackle with the grid dynamicity.

As Feitelson et al. claim in their scheduling review [16], parallel job scheduling reviews are needed in a regular basis. The purpose of their short review was to introduce clusters and grids into the parallel job scheduling literature. Indeed, the authors present an introduction to job scheduling in grids, highlighting differences between a parallel computer and the grid. They point out cross-domain load balancing and co-allocations as two main concerns when scheduling in grids. In our work, we introduce a classification of schedulers in distributed systems that comprises a more extensive view of grid computing algorithms. Moreover, we highlight new requirements for the cloud computing emergent paradigm as well as its differences to grid computing.

Wieczorek et al. [17] present a taxonomy in the scheduling problem for workflows considering multiple criteria optimization in grid computing environments. The authors separate the multi-criteria scheduling taxonomy in 5 *facets*, namely *scheduling process, scheduling criteria, resource model, task model*, and *workflow model*, each facet describing the problem from a different point of view. These facets are expanded to classify existing works in a smaller granularity, pointing out where current research can be expanded and the work in each facet.

We highlight two conclusions achieved by the authors in [17] which are touched by contributions given by our survey: (i) "grid workflow scheduling problem is still not fully addressed by existing work"; and (ii) "there are almost no workflow scheduling approaches which are based on an adaptive cost model for criteria". As a contribution to (i), in this survey we expand the general distributed system scheduling to comprise it. As a contribution to (ii), we include scheduling taxonomies for utility grids and cloud computing environments.

Besides specific reviews of the scheduling literature, some other general surveys include scheduling taxonomies and classification. In [18], Yu and Buyya classify workflow management systems for grid computing. The presented classification includes some taxonomy branches that have influence in schedulers, such as the workflow structure (DAG or non-DAG), workflow QoS constraints, and information retrieval coordination. Regarding the scheduling itself, the classification includes system *architecture, decision making, planning scheme, strategies, and performance estimation* [18]. In this paper we cover this classification and further expand it to a more general view of the scheduling problem, comprising independent tasks and recent advances in distributed systems, such as cloud computing. Venugopal et al. review the literature on data grids in [19]. Concerning scheduling in such systems, the authors present a brief taxonomy that includes classifications regarding *application model, scope, data replication, utility function, and locality* [19].

In the last decade, cloud computing taxonomies started to appear. In [20], Hilley describes a bunch of existing services that provide cloud computing infrastructures, classifying them inside a proposed taxonomy. Oliveira et al. [21] classify the cloud computing paradigm regarding many aspects, such as architecture, pricing, privacy, and technology used. Other recent review papers in the scheduling literature have focused on specificities of scheduling, as for example meta-heuristics [22], scheduling techniques [23], workflow costs [24], virtual machines [25], workflows in clouds [26], evolutionary approaches, [27]. In this paper we introduce more general cloud computing classification aspects that affect the development of scheduling algorithms, which are not addressed by those papers.

Scheduling has been a focus of research with the high attention given by the community to grid computing in the last decade and to cloud computing in the last few years. With this, algorithms and techniques were freely developed, aggregating new concepts from the service oriented computing and aspects from the constantly changing distributed systems environments. As shown in this section, many advances were achieved in the last decades. Also, excellent reviews which unify topics surveyed in this paper exist, however, updates are necessary in subtopics such as grid and cloud computing. In this work, we classify new developments in the scheduling field in what concerns the distributed computing systems evolution, aiming at fitting topics developed in the last decade and future directions. This paper brings an organized overview of the scheduling problem advancements in distributed computer systems, serving as a reference for the development of algorithms for cloud computing and the upcoming distributed computing paradigms.

3. Basic concepts

The definition of the scheduling problem given by Pinedo in [1] is as follows:

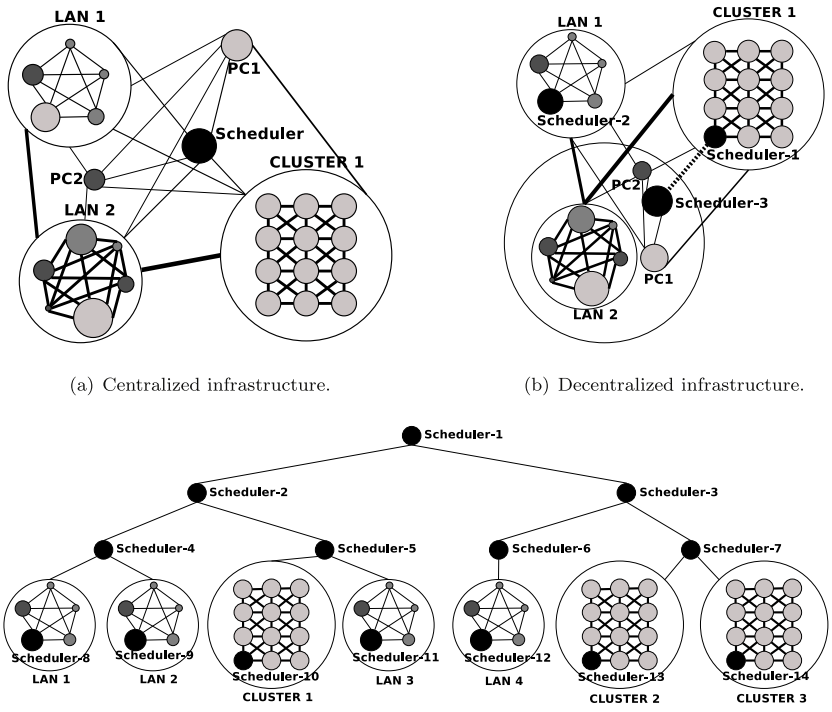
"*Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives.*"

There exist many other definitions to the general scheduling problem in the literature [28]. The scheduling problem is NP-Complete, although there exist polynomial time solutions for few scenarios [1].

In a distributed system, the scheduler has the role of choosing in which computational resource each task (or job) will execute, therefore distributing jobs to be executed concurrently. The execution time of each job, and consequently the completion time of all jobs, is intrinsic to the generated schedule. The scheduler organizes jobs in resources queues usually following an objective function. The most common objective found in the distributed systems literature is minimize the completion time, or makespan, achieved by proper allocation of tasks in the available resources. Common objective functions are examined in more details in Section 5.5.

3.1. Scheduling model

Here we describe a scheduling model that is the basis to the scheduling problems in distributed systems, as previously modeled in [29]. Consider a distributed system composed of a set of computational resources $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$, with associated processing capacities $p_{r_i} \in \mathbb{R}^+$, connected by network links. Each resource r_i has a set of links $\mathcal{L}_i = \{l_{i,1}, l_{i,2}, \ldots, l_{i,m}\}$, $1 \leq m \leq k$,



(a) Centralized infrastructure.

(b) Decentralized infrastructure.

(c) Hierarchical infrastructure.

Fig. 1. Architectural organization of distributed systems and schedulers.

where $l_{i,j} \in \mathbb{R}^+$ is the available bandwidth in the link between resources r_i and r_j, with $l_{i,i} = \infty$. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be the set of application jobs submitted by users and to be executed in $\mathcal{R}$. The scheduler is responsible for assigning a sequencing of jobs in $\mathcal{A}$ onto the resources in $\mathcal{R}$ according to one or more objectives.

According to the well-known, general scheduling problem description by Pinedo [1], the scheduling problem is commonly described as a triplet $\alpha \mid \beta \mid \gamma$. The α field, with a single entry, is a description of the execution environment. The β field can have from zero to multiple entries, and it describes what is the processing to be done. Lastly, the γ field entry is the description of the objective to be optimized. In this paper we are mainly interested in discussing the scheduling problem for two different α's [1]:

1. $\alpha = P_m$: Identical machines in parallel. Execution environment is composed of m identical machines in parallel, as usually found in a cluster computing infrastructure.
2. $\alpha = R_m$: Unrelated machines in parallel. There are m different machines in parallel, and machine i executes job j at speed $v_{i,j}$, as in a grid or cloud computing infrastructure. Note that, given two application jobs a_1 and a_2 and two computational resources r_1 and r_2, $v_{a_1,r_1} > v_{a_2,r_1} \not\Longrightarrow v_{a_1,r_2} > v_{a_2,r_2}$.

As utility computing becomes more popular, we propose the introduction of a new field to the representation of the scheduling problem: $\alpha \mid \beta \mid \gamma \mid \mu$, where the μ field describes the charging models adopted in the system. For example, μ can be an hourly-based charge, an auction system, per-transaction charge, data-access charge, and so on. This is well-suited to the cloud computing paradigm, in which many different computing services can be offered with a variety of charging models, as well as for the upcoming utility-based computing and networking infrastructures.

3.2. System organization

In distributed systems, the overall system organization may have influence in the scheduler scope. This organization can be a result of the infra-structure architecture or existing policies and access controls. Regarding the system management organization, it can be classified in three categories: *centralized*, *decentralized*, and *hierarchical* (Fig. 1).

In a *centralized* management, the distributed system has a central entity that has information about the whole system and is responsible for managing its resources and jobs execution. On the other hand, a *decentralized* management is accomplished through distributed algorithms that manage the system without a central entity. In such systems, management roles may be distributed over resources in the system [30]. Scheduling in a decentralized system is made by more than one scheduler, where each scheduler has information about part of the system and may exchange information with other schedulers to make decisions in a cooperative way. A decentralized non-cooperative distributed system considers that each scheduler is responsible for achieving its own objectives, regardless the objectives of other schedulers. A *hierarchical* system structures the management in a tree-like structure. Therefore, machines at a given level can communicate with machines at a level immediately above or below it in the hierarchy [30]. With this, the distributed system management is shared, with entities that are higher in the hierarchy taking high-level decisions (e.g., in which WAN a job will execute), while entities at the bottom make the low-level decisions (e.g., in which machine a job will be executed).

Note that the centralized architecture can be seen as a specific case of the decentralized model where only one scheduler exists. In addition, algorithms for the centralized architecture can independently execute in each scheduler of a decentralized (or hierarchical) system. In this case, a scheduler component could be responsible for the communication among different schedulers to arrange jobs execution within different boundaries. See Fig. 2 for a taxonomy of the scheduler organization in distributed systems.

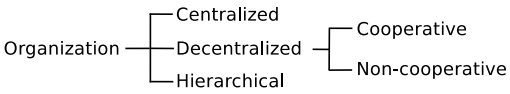


Fig. 2. Taxonomy for schedulers organization in distributed systems.

A more detailed view of distributed systems organization can be found in [19].

4. Taxonomy overview

This section introduces the taxonomy presented in this paper without discussing details. Fig. 3 presents the full taxonomy, where each branch up to the third level presents a short text notation, which can be extended to deeper branches in the taxonomy, and that is used later in this paper to classify the current literature.

The taxonomy brings a classification according to different factors derived from schedulers developed for distributed systems that existed before the advent of cloud computing, which we call *pre-cloud schedulers*, and those developed for cloud computing infrastructures, the *cloud schedulers*. For pre-cloud schedulers, taxonomy branches are the scheduler ***organization***, as discussed in the previous section, ***input***, ***frequency***, ***target***, ***application***, ***optimization***, and ***output***. For cloud schedulers, we update the taxonomy (branches in red in Fig. 3) to reflect characteristics that were introduced to address cloud computing scheduling, namely branches about ***monetary costs*** and ***virtual machines*** under the *input* branch, ***clouds*** under the *target* branch, ***virtualized*** application models, scheduler ***objective***, and ***location***. Detailed discussion on each branch of the taxonomy is presented in the upcoming sections.

5. Pre-cloud scheduling taxonomy

5.1. Input classification

The information available to the scheduler drives the decision making process. This input information is subject to variations in *accuracy* and *semantics* (different characteristics influencing the scheduling process), as shown in Fig. 4.

5.1.1. Input accuracy

The scheduling algorithm may expect the input data to be *precise*, *uncertain*, or *stochastic*. When the input data is considered to be *precise*, the algorithm is developed to work better within conditions in which both the input data about jobs and about resources reflect the real execution behavior. Input data is said to be stochastic when it is expected to follow some generalized probability distribution. If the input data is assumed to be *uncertain*, the algorithm can make use of mechanisms to deal with different kinds of uncertainty, including *application demands* and *resources availability* uncertainty. Schedulers commonly assume that application communication and computation costs are known. Therefore, we consider that application specification can present uncertainty regarding its *communication* and *computation (processing)* demands, and also *other* demands as further possible application requirements (memory, storage, etc.). The input accuracy taxonomy is shown in Fig. 5.

Derbal presents in [31] an entropy-based quantification of input data uncertainty regarding resources performance. Batista and Fonseca deal with both application and resource availability uncertainties in [32]. Examples of studies in stochastic scheduling are Schopf [33] and earlier by Pinedo [34]. Precise input data is commonly found in the literature, such as in works by Topcuoglu et al. [8] and Kwok et al. [3].

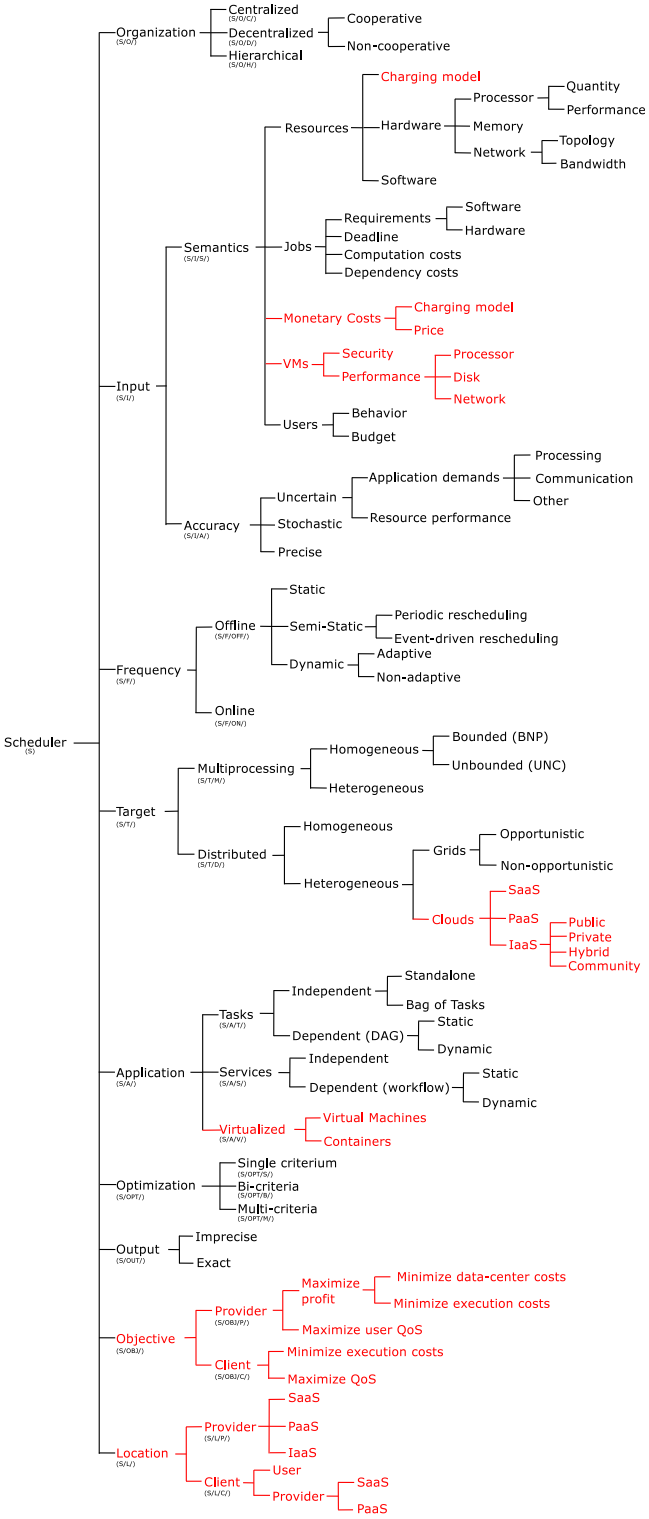


Fig. 3. Overview of the proposed taxonomy. A short text notation is shown below branches up to the third level, which can be extended to deeper branches in the taxonomy.

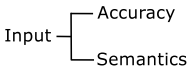


Fig. 4. Input taxonomy.

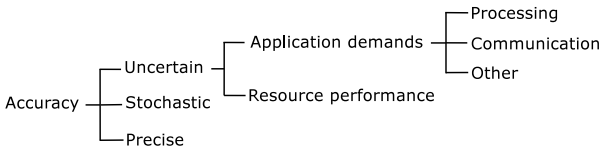


Fig. 5. Input accuracy taxonomy.

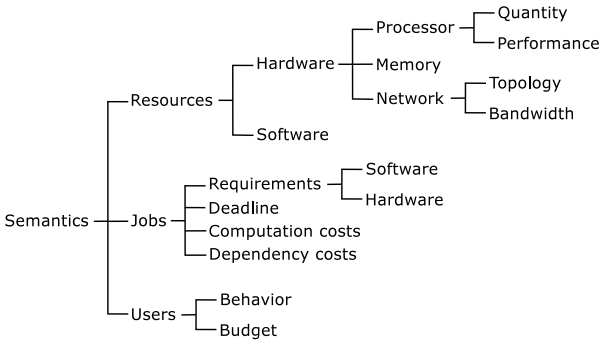


Fig. 6. Input semantics taxonomy.

5.1.2. Input semantics

The input data semantics taxonomy (Fig. 6) classifies the *meaning* of the information provided to the scheduling algorithm. This information is used by the scheduler to perform the decision making following pre-defined objectives. The semantics of the input data can be related to *resources*, *jobs*, or *users*. Information about *resources* are divided into *software* and *hardware*. Hardware information commonly comprises *processor*, *memory*, and *bandwidth*. The *quantity* and *performance* of processors are usually considered as input data. Within the *software* branch below *resources*, applications, services, and libraries available on each resource can be provided to the scheduler.

A job may have *hardware* and *software* requirements. Therefore, the scheduler can match software and hardware requirements from jobs with the available hardware and software on each resource. A job may also have a *deadline*, that is, a maximum completion time. An important information used by the scheduler is the *computation cost* of a job, which influences the decision on where it will be executed. Another important information for coupled jobs are their *dependencies costs*, which influences where dependent jobs should be placed in order to minimize their data transmission times or any other objective.

The *user* who submits a job may have a *budget* to follow. This information is usual when the target system is a utility grid or a cloud and the scheduler must use it to check if a given schedule obeys this user's constraint. Another information regarding *users* is their *behavior*. This can be useful in a shared system (such as grids), in which the scheduler may use information about which applications are most often executed by a user *A*, therefore estimating how long they take to run and how much *CPU* they consume. For example, a scheduler may expect that a job from user *A* running for 30 minutes is about to finish, since user *A* jobs usually do not take longer than half an hour.

The execution time of a job is, in general, highly dependent on the processor on which it runs. Because of that, scheduling algorithms usually consider the processing capacities as an important, and often determinant, characteristic when deciding where each job will execute. Research assuming that the scheduler has information about resources bandwidth, memory, and processors quantity and performance, as well as jobs computation and communication costs, is very common [35–39]. Information about software installed in the resources is also considered in works where jobs have software restrictions [40]. Ramamrithan et al. present a set of heuristics to schedule jobs with deadlines and resources requirements in [41]. Chaves et al. present an integer programming formulation for scheduling grid applications with software requirements in [42]. In [39], den Bossche et al. deal with the problem of scheduling deadline-constrained workloads, thus considering jobs deadlines as input data. Yu and Buyya in [12] consider the user's budget when scheduling workflows in grids. In [43], Fox and Gannon overviews contributions on workflow in grid systems. In [44], Gomes and Costa analyze desktop users' application behavior in opportunistic grids, extracting the local application CPU usage to predict bursts durations, creating profiles to be used in scheduling decisions.

5.2. Frequency

The scheduling frequency is associated to how many times the scheduler is executed within a period of time. In a general scope, the scheduling algorithm can be classified as *offline* or *online*. An online algorithm assumes that its input is a data sequence of unknown size, handling parts of the data without having the whole input at the start of its execution. Therefore, an online scheduling algorithm does not know the entire set of jobs being scheduled at a given time (or even the whole set of resources where the jobs will execute). Rather, it schedules a job without information about the next incoming jobs. Thus, the decision maker becomes aware of the existence of a job only when the job is released and presented to it [1]. An online scheduling algorithm can be aided by some information about the input stream, such as which probability distribution the input should follow. On the other hand, in the more common paradigm of *offline scheduling*, the information regarding all jobs is known *a priori*. Note that the online paradigm can have an intersection with the offline paradigm if arriving jobs are grouped into sets before they are scheduled, in a *batch-mode* scheduling [45].

In another branch, offline scheduling can be classified as *static* or *dynamic*. Static schedulers perform scheduling at *compile time*. This means that the scheduling algorithm decision making is accomplished using information available before any job is sent to the resources execution queues [3], and no interference is made until the end of the application execution. Conversely, a dynamic scheduler uses information which is updated during jobs execution, thus having an up-to-date prospect of the system as a whole. To do this, the dynamic scheduler dispatches a set of jobs to execution and observes the behavior of the system, gathering information to schedule the next set of jobs. This information gathering is usually done when a resource becomes idle [9]. As extensions from these two general classifications, *semi-static* (also called *hybrid* [45]) and *adaptive* approaches exist as well [10,38]. In a semi-static approach, an initial mapping is done and remappings are performed when necessary. This rescheduling can be done *periodically* or after observing some *event*, such as resource performance changes during execution or application input data changes [38]. Adaptive approaches alter the dynamicity according to the observed behavior of the system, therefore being able to act as a highly dynamic algorithm, which updates the system information frequently, or even as an almost static one. The taxonomy regarding the scheduling frequency is presented in Fig. 7.

Offline algorithms are most commonly found in the literature than online algorithms. Examples of well-known static offline scheduling algorithms are the *Heterogeneous Earliest Finish Time* (HEFT) [8] and the *Dynamic Critical Path* (DCP) [2]. A semi-static algorithm with periodic rescheduling is presented by Kwok et al.

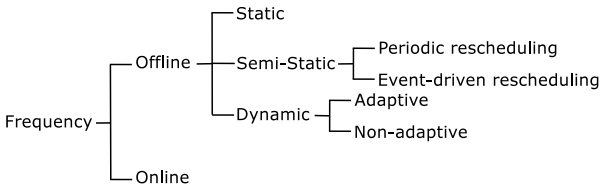


Fig. 7. Taxonomy according to the scheduling frequency.

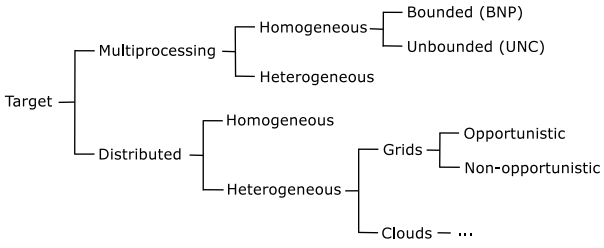


Fig. 8. Taxonomy according to the target system.

in [38]. Lucchese et al. present a semi-static algorithm with event-driven rescheduling in [9]. A round-based adaptive dynamic algorithm is presented by Bittencourt and Madeira in [10]. Maheswaran et al. present non-adaptive dynamic approaches in [45], as well as online scheduling algorithms. Other examples of online algorithms include [46–48].

5.3. Target system

The target system for a scheduler is associated to the computational architecture of the system where the scheduled jobs will run. Therefore, another factor that has influence on a scheduler design is information about the target system to make decisions. Two main characteristics considered during scheduling are the resources processing capacity and link bandwidth value interconnecting the processing resources. Our classification regarding the target system is shown in Fig. 8.

Although multiprocessing system may be apart from the distributed systems definition since they are shared memory systems, the study of scheduling algorithms for multiprocessors contributes to distributed systems in two aspects: (i) algorithms for multiprocessing systems can be adapted to work in distributed systems architectures; and (ii) multiprocessing systems may be part of distributed systems such as clusters, grids, or clouds. Scheduling targeting multiprocessing systems can be split in *homogeneous multiprocessing* or *heterogeneous multiprocessing*. Studies in unbounded homogeneous multiprocessing systems are common, such as the analysis made by Papadimitriou and Yannakakis in [49]. The so called unbounded schedulers assume that the number of processors is always greater than the number of tasks being scheduled, and this kind of algorithm had previously been classified as *unbounded number of clusters* (UNC) [3]. Conversely, scheduling algorithms considering bounded homogeneous multiprocessors, previously classified as *bounded number of processors* (BNP) [3], do not make such assumption. Examples include the *Modified Critical Path* (MCP) and the *Earliest Task First* (ETF). Heterogeneous multiprocessing systems were less explored in the past than its homogeneous counterpart.

Considering the scheduling in distributed systems, i.e., with no shared memory, our classification comprises both *homogeneous* and *heterogeneous* systems. Works in homogeneous distributed systems generally consider them as computing clusters with identical machines interconnected by a homogeneous network. Chang and Oldham present models for dynamic task allocation in distributed homogeneous systems in [50]. A study on scheduling of parallel jobs in homogeneous distributed systems is presented by Karatza and Hilzes in [37]. Hagras and Janeček present an algorithm *called critical nodes parent trees* (CNPT) [36]. Large homogeneous datacenters gave momentum to scheduling in homogeneous systems [51], including compositions of systems such as in multicluster grids [52].

Scheduling in heterogeneous distributed systems received substantial attention after the popularization of grid computing and, more recently, the cloud computing paradigms. Buyya et al. present the Nimrod-G [53], which is a system for resource management and scheduling for grids. In addition, various simulators were developed to aid algorithms evaluation. Among the most cited ones are GridSim [54], developed by Buyya and Manzur, and the SimGrid [55], developed by Casanova.

Grid computing originally started as an opportunistic computing community-based platform for parallel processing. Heymann et al. proposed an algorithm for adaptive scheduling of master–worker applications in opportunistic grids [56]. Another work that considers opportunistic grids is presented by Dutot in [57]. An architecture for scheduling in opportunistic grids was developed by Netto et al. [58].

Grids evolved from desktop-based cycle-stealing mechanisms to more powerful processing platforms composed of shared heterogeneous beowulfs and clusters. Concerning those non-opportunistic grids, Boeres et al. [59] study the integration of static and dynamic algorithms for heterogeneous distributed systems, presenting experiments using hybrid (or semi-static) algorithms. Casanova et al. presented heuristics for the scheduling of parameter sweep applications in grids [60]. He et al. described a min–min heuristic for task scheduling in grids [61].

After the maturing of the computational grid, the cloud computing emerged as a new computing paradigm in distributed systems. The cloud branches in the taxonomy are discussed in Section 6.

5.4. Application

Application characteristics, as the target system, also have major influence in the development of scheduling algorithms. Therefore, it is worthwhile classifying the application models considered in the scheduling literature. Our proposed scheduling classification scheme according to the application to be scheduled is shown in Fig. 9. In the most general classification, we separate the application model in *tasks* or *services*. A *task* is a self-contained job that carries its own executable code. A task may also carry the necessary input data for its execution (or a reference to its location). A *service* is a job which is actually an invocation to executable code that is available through an interface on the computational resource [62]. Therefore, a service call carries information about parameters and may also carry information on input data.

Both tasks and services can be classified as *dependent* or *independent*. Independent tasks or services perform no communication, while the dependent ones present communication dependencies among them. *Independent tasks* can be either *standalone* or *bag of tasks* (BoT). While a standalone task has no relation to other tasks arriving to the scheduler, as the ones studied in [45], tasks within a bag of tasks have similarities among them. An example of a class of BoT are the parameter sweep applications, where the same task is executed many times with different input parameters. Casanova et al. present algorithms for the scheduling of parameter sweep applications in [60]. Dependent tasks can be modeled as directed acyclic graphs (DAGs), in which nodes represent tasks and arcs represent communication dependencies among tasks. In our classification we separate DAGs into two classes: *static DAGs* and *dynamic DAGs*. Static DAGs have all their tasks run during their

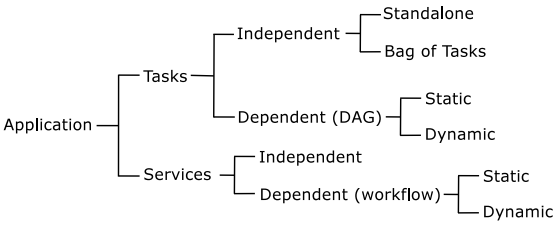


Fig. 9. Taxonomy according to the application model.

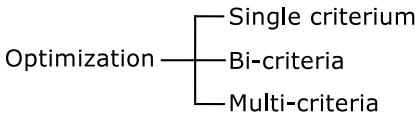


Fig. 10. Taxonomy according to the optimization of the objective function.

execution, such the ones scheduled by the algorithm presented by Zhao and Sakellariou [63]. On the other hand, dynamic DAGs may change during the execution: conditional tasks may not be run or new tasks may be added to the graph as a result of iterations in the DAG definition [64]. Dynamic DAGs can also be modeled as Petri nets [18].

The *service oriented computing* (SOC) paradigm has expanded more recently. The same classes definition applied for tasks can be applied for services. However, we do not separate services in *standalone* or *bag of services*, since those terms are not used when dealing with service scheduling. When regarding to dependent services, which are also commonly modeled as DAGs, we call them *workflows*. This term is now widely used when referring a set of dependent services and scientific applications to be scheduled in grids or clouds, as well as it is more ample and may include cycles.

5.5. Objective function optimization

A scheduler can optimize different objectives when scheduling applications and their jobs in distributed systems. The minimization of the makespan is the most common optimization objective found in the scheduling literature [1]. In [65], Dong and Akl classify the objective functions in either *resource centric* or *application centric*. Although this classification indeed reflects some aspects of the scheduling objectives, we prefer to approach this taxonomy in a different way because some objective functions cannot be said either resource centric or application centric. For example, minimizing data movement [66] can be thought as centered on both the resources pool and the application itself.

We separate the scheduler objective function optimization in distributed systems into three branches: single criterium, bi-criteria, and multi-criteria, as shown in Fig. 10.

Single criterium scheduling algorithms try to find a schedule that optimizes one criterium. In this class of scheduling algorithms, it is possible to identify the most common objectives: minimize execution time [8], minimize communication [66], maximize throughput [67], maximize load balancing [68], and maximize utilization [69]. Bi- and multi-criteria algorithms focus on finding the best schedule following two or more (often conflicting) objectives. Optimizing more than one objective often leads to tradeoffs with no clear distinction among possible solutions to reach a single optimal solution. With no solution that is better than all the others, the algorithm must actually generate a set of solutions that are *non-dominated*, meaning that for any solution in this set, there is no other solution that can improve one objective without worsening another. The *Pareto front* is a set composed of non-dominated solutions.

It is common to trade-off between two objectives in scheduling, yielding bi-criteria scheduling algorithms that optimize, for example, reliability and execution time [70], workflow bi-criteria optimization [71], robustness and makespan [72], service instantiation costs and makespan [73], execution time and costs [74], and resources utilization and makespan [75]. Multi-criteria scheduling also frequently involves optimizing execution time along with a set of other criteria [17].

Optimization techniques. Classification and evaluation of scheduling techniques in distributed systems have been addressed before in the literature [3,11,14]. The naïve approach to perform the scheduling is to try all possible combinations and return the best one. This is feasible only if the size of the input instance (e.g. amount of tasks and/or dependencies and amount of computational resources) is sufficiently small, since there exist an exponential number of combinations and, consequently, large instances are unfeasible within a reasonable wall time. That is why scheduling is often approached by using sub-optimal techniques, such as heuristics [76] and meta-heuristics [35], as well as combinatorial optimization methods [77] and approximation algorithms [78].

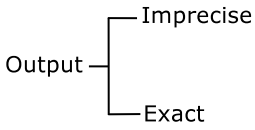


Fig. 11. Taxonomy according to the scheduling output.

5.6. Output

A scheduling algorithm produces a schedule can be either is *exact* or *imprecise*, depending on the expected input. When the produced schedule is *exact*, it is a consequence of an *exact* input data, thus producing a final schedule that will be fully obeyed. On the other hand, if the schedule produced is assumed to be *imprecise*, the algorithm should make assumptions to deal with stochastic and uncertain input data, which includes application demands and resources availability imprecisely specified or estimated. The scheduling output taxonomy is shown in Fig. 11.

Many scheduling algorithms consider that the produced schedule is exact, as a result of the input data to be precise [3,8,60]. Some algorithms assume that other mechanisms are responsible for improving the scheduling quality if the input data is not accurate (or imprecise). Such mechanisms include reactive rescheduling, as in dynamic or semi-static scheduling approaches. Note that the imprecisions can result from user specification or performance measurement/prediction systems, such as in [44].

6. Scheduling in cloud computing

Hitherto we described characteristics of the scheduling problem and target distributed systems in general. Although the presented overview is broad, the concepts are directly applicable to specific computer systems. Henceforth, we specialize this broader view to the cloud computing paradigm. In this section, we present how the scheduling problem in cloud computing has been developed, and discuss scheduling peculiarities and major challenges in cloud computing. Moreover, we extend some aspects of the proposed taxonomy to incorporate cloud computing characteristics and present how to tackle with schedulers objectives in clouds.

Scheduling algorithms and their associate performance are strongly influenced by the input data and specific characteristics

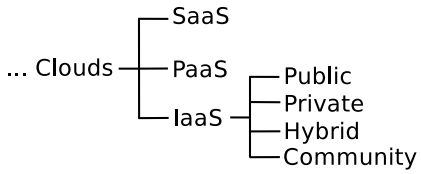


Fig. 12. Cloud extension of the taxonomy according to the target system.

of the target system. As different service models implemented in cloud computing have specific objectives, the design of new scheduling algorithms is a clear need to produce efficient services. Actually, it is common that conflicting objectives can be part of a single optimization goal.

6.1. Cloud computing service models

In the cloud computing model, users are not aware of the specific technology employed to enable service provision. Such abstraction results in the provision of services over the Internet which are dynamic and scalable [79]. The cloud computing business model allows clients to extend their computing platforms by leasing virtualized computational resources offered as services by cloud computing providers through the Internet. This resource leasing provides elasticity to the client computational power by running software or development platforms, or by leasing virtualized hardware. Cloud computing has three canonical models: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). Other service models encountered in the literature are often specialization or combination of these three models.

The IaaS is a popular model from a resource management perspective, thus scheduling in such systems has received great attention in the research community [39,80,81]. In a nutshell, IaaS is a cloud computing model where virtual servers are made accessible to clients through the Internet. In terms of access policy, an IaaS can be *public*, *private*, or *hybrid*. A *public IaaS* offers servers in a pay-per-use basis, while a *private IaaS* can be seen as a virtualized and more user-transparent cluster. As a combination of these two, a *hybrid IaaS* comprises both public and private IaaS clouds, offering private resources that can be expanded with paid public cloud resources, realizing the so-called *elasticity* [29]. Another model considered in the literature is the *Community Cloud*, being a composition of private clouds sharing their resources through their administrative borders. To cover these types of clouds, an extension of the target system taxonomy is shown in Fig. 12.

Clearly, the classification of cloud access in public, private, hybrid, or community can be extended for PaaS and SaaS. In special, community clouds inherit from grid computing many challenges related to administrative aspects of access control that are discussed in a multi-cloud framework [82]. In our classification, we put these types of cloud access explicitly under IaaS since it is at the infrastructure level that the resource sharing effectively occurs.

6.2. Client and provider perspectives

Cloud management, which includes scheduling, must deal with two different perspectives: one from the cloud client and the other from de cloud provider. Cloud clients demand quality of service and low prices, while cloud providers focus on offering QoS and making profit. The development of algorithms for each perspective requires different approaches to achieve such different optimization objectives.

Fig. 13(a) illustrates the client perspective, in which users have access to a private or in-organization task/service submission interface. The management system is responsible for gathering and storing information about available clouds, as well as their services and prices. In this perspective, clients can utilize services from a private cloud (if available) and/or from multiple public clouds, which is consistent with the taxonomy shown in Fig. 12. In what concerns the scheduler, it must deal with the trade-off between quality of service and price, which depends on users requirements and objectives.

The provider perspective is illustrated in Fig. 13(b). The service interface can be located in the cloud provider or in a client broker, usually offered as a portal, a set of tools or as an API (application programming interface). Based on the client/application requirements defined in the SLAs, the cloud provider must decide the best resource configurations to be used according to its profitability principles.[1] Depending on the service level offered by the provider interface (IaaS, PaaS, SaaS), the scheduler will cope with different inputs for the decision-making. In the case of an IaaS provider, the scheduler is responsible for allocating virtual machine requests on the physical substrate. In the case of SaaS or PaaS provider, the scheduler allocates client applications and/or application requests to the virtualized computing resources.

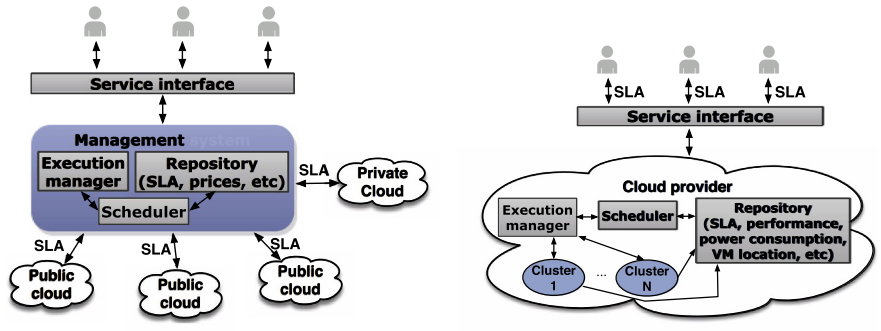
A third perspective, illustrated in Fig. 14, combines both client and provider perspectives, in which a public cloud provider makes use of other public clouds to provide services to its clients. In this perspective, two SLA layers exist: one between the provider and its clients, and one between the provider and the other cloud providers. In this scenario, different terms can exist at each SLA level, which must be translated from one layer into QoS requirements at the other layer so the scheduler can allocate client requests. Therefore, both client and provider perspectives co-exist, and the scheduler must make decisions on how to get the best QoS according to the current demand, but also should focus on profitability when acting as a client from other public clouds. Fig. 15 presents a taxonomy for the scheduler location in the cloud.

6.3. Scheduler "feedstock"

A fundamental difference between clouds and previous distributed systems is the wide adoption of virtualization to offer computing resources to the users. To achieve efficient server consolidation, VM allocation (or placement) must be performed, i.e., a scheduling decision that is different from the application scheduling. In this section, we denominate *scheduler feedstock* as the type of computer software being considered by the scheduler: virtual machines or applications. Thus, when developing schedulers for cloud computing, we must specify if the scheduler is a *VM scheduler* or an *application scheduler*. These two different resource allocation facets in clouds have implications that were not present in grids and clusters. Based on this, we derive a new branch in the application model taxonomy in Fig. 16(a) (previously shown in Fig. 9). Moreover, we extend the input semantics taxonomy (previously shown in Fig. 6) to cover input data about virtualized environments. The new branch of the input semantics is shown in Fig. 16(b).

The VM scheduler needs information on the resource requirements of the VMs being allocated, namely security constraints and hardware requirements (CPU performance and number of cores, amount of RAM, etc.). Another relevant information for the scheduler is the charging model of the VMs. The charging model

[1] As a market-oriented business, the provider can focus on different strategies, from strong confidence relations with its clients to lower level services. The discussion on business/market strategies is very complex and is out of the scope of this paper.



(a) Client perspective. (b) Provider perspective.

Fig. 13. Client and provider perspectives in cloud computing.

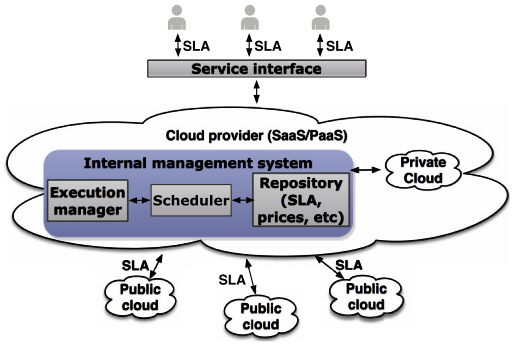


Fig. 14. Cloud management with 2 (or more) SLA layers.

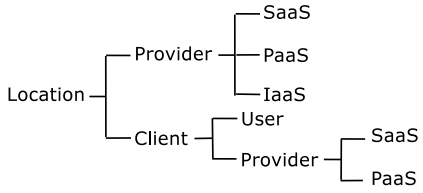
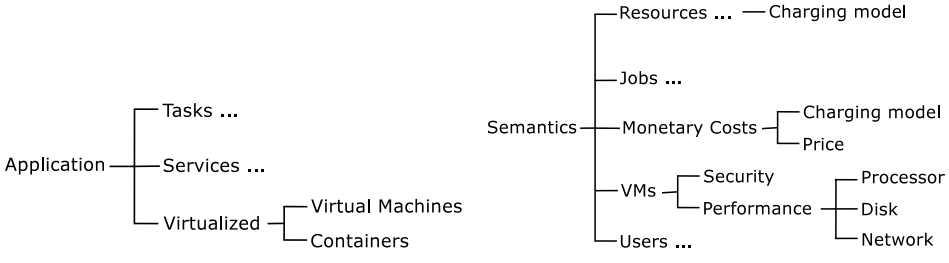


Fig. 15. Taxonomy in clouds according to the scheduler location.

have implications in the expected duration of the allocation and also in its preemption rules [83]. For example, Amazon EC2 spot instances can be interrupted by the provider without prior notice, while doing that for on-demand instances would mean an SLA violation from the provider. Taking this information into account can help the VM scheduler to properly place VMs onto the physical machines.

To deal with the users requests, an application scheduler is necessary when the scheduler is located in the cloud client, as shown in Fig. 13(a). This scheduler decides on which VMs the applications will run. If the VMs are already deployed and running, the application can be sent for execution. Otherwise, the private cloud and/or the public clouds that will be used to run the application must deploy new virtual machines according to the application scheduler output. Thus, the virtual machine scheduler present in the cloud (Fig. 13(b)) must be invoked to decide on which physical machine the new VM(s) will be placed. After that, all VMs can be switched on and then the application components can run.

When considering IaaS providers, one important aspect that is made clear from the two scenarios in Fig. 13(a) and (b) is that application and VM schedulers information exchange is limited. The application scheduler can ask for a certain amount of VMs with a set of characteristics to the VM scheduler, and no further information is exchanged. In other words, the VM scheduler has no specific information about the application(s) that will run on those VMs. As a consequence, the resulting VM scheduling in the physical resources can, in practice, lead to low resource utilization even if all physical resources in a machine are currently allocated to virtual machines. However, if we consider, for instance, a two or more SLA layers scenario with an SaaS or PaaS provider (Fig. 14), the application scheduler and the private cloud VM scheduler are under the same administrative domain. Therefore, information about the computing demands of the application being scheduled can serve as input to the VM scheduler to improve allocation. By considering the application demands and the VM requirements altogether, the VM scheduler can, for example, allocate a VM that have low CPU utilization and high I/O demands in the same physical machine of a VM with high CPU utilization and low I/O demands, avoiding bottlenecks that could impair quality of service, as well as maximizing hardware utilization.



(a) Application model. (b) Input semantics.

Fig. 16. Application model and input semantics taxonomy extensions for cloud computing.

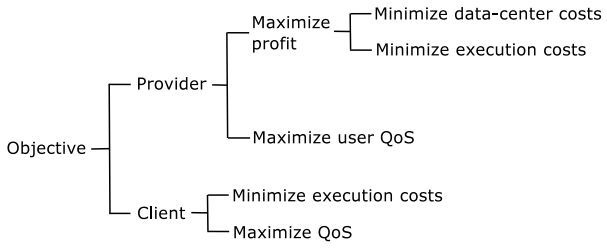


Fig. 17. Taxonomy in clouds according to the scheduler objectives.

6.4. Scheduler objectives

In clouds, the objectives pursued by a scheduling algorithm often depend on where the scheduler is running. Different approaches and algorithms must be implemented to achieve these objectives in a variety of scenarios. Each scheduling approach, according to the scheduler location, takes as input information of diverse granularity and semantics. In order to reach an objective, the scheduler must optimize one or more criteria following an objective function (see Fig. 10). Thus, a single objective can be pursued by an algorithm which optimizes one criterium, but can be one among multiple objectives to optimize (multiple criteria). The proposed taxonomy for scheduler objectives in clouds is presented in Fig. 17. Its details are discussed in the next sections.

6.4.1. Providers

Cloud providers must supply its clients with virtualized computing resources according to a service level agreement. In the case of IaaS providers, it receives VM requests from its clients and must schedule them on the underlying physical machines following the resource requirements of these VMs, such as amount of processing cores, performance, disk space and speed, networking requirements, and so on. The virtual machine placement decided by the scheduler will have direct impact in the datacenter utilization and VM performance observed by the clients. Therefore, a scheduler at this level is concerned with minimizing datacenter costs subject to VM specifications in the SLAs established with the clients. From the input semantics for clouds (Fig. 16(b)), we observe that this SLA can include security and hardware requirements, characteristics of the charging model to be followed, and application demands.

To *minimize datacenter costs*, cloud providers can rely on algorithms that, for instance, minimize the amount of active machines. This can be achieved by scheduling VMs on the minimum possible amount of physical machines as soon as the desired (or ensured by SLA contracts) quality of service is maintained. However, as the clients want elasticity, the number of running VMs changes over time, increasing with new VM requests and decreasing when clients turn their VMs off. This leads to allocations where hardware is underutilized, whereas virtual machine migration can remediate this situation. Thus, the scheduling frequency (Fig. 7) of VMs in the datacenter must consider how volatile the VM requests are. For instance, a periodic rescheduling or event-driven rescheduling can be utilized to maintain a high hardware utilization.

One way of minimizing running costs is through profiling running virtual machines on-the-fly, aiming at discovering the VMs resource usage patterns. This can lead to a better reallocation of virtual machines, avoiding to collocate virtual machines that will potentially create bottlenecks in the underlying hardware. For example, two I/O intensive virtual machines could be placed in different physical hardware, while two VMs that have strong communication relations could be placed in the same physical machine. The VM characterization problem is yet a challenge that can be addressed through hardware and software monitoring in order to find use patterns, and the development of new algorithms should consider such patterns to avoid overload is desirable. However, privacy and security concerns must be taken into consideration when monitoring VMs.

We identified two main objectives from schedulers located in cloud providers: maximize profit and maximize quality of service. Profit maximization can be achieved by adopting more specific scheduling objectives, such as minimizing the datacenter running costs and minimizing the tasks execution costs. Consolidation can be utilized for both objectives: cloud providers can adopt energy-aware resource allocation policies to minimize datacenter running costs, or they can follow a consolidation objective function that focuses on minimizing the amount of cloud resources (rented from other providers as well) necessary to run a set of tasks.

Energy consumption has attracted attention from the resource allocation community in cloud data centers [84–87] as part of the so-called *Green Computing* concept [6]. Energy-aware resource allocation commonly takes the form of a VM placement optimization in IaaS data centers, where the VM allocation is determined by quality of service requirements (usually defined in an SLA with the cloud client) and consolidation of VMs into the physical machines available. Therefore, consolidation helps in reducing the amount of machines powered on, and thus resource allocation collaborates to reduce power consumption. Energy-aware allocation can also take into account power consumption of network devices, aiming at placing VMs closer to data sources to reduce network traffic [86].

Clouds offering platform (PaaS) and software (SaaS) as services need a computational infrastructure to host the platform and run the software services. These cloud providers can own the whole infrastructure or they can rely on IaaS providers to achieve elasticity, shrinking their infrastructure and reducing costs whenever possible. In this scenario, the SaaS or PaaS provider composes a hybrid cloud to run services offered to its clients, therefore resulting in two SLA layers. As in other businesses, different strategies can be used by the SaaS/PaaS provider to attract clients, such as improved service quality with higher charges or cheap services with lower service quality. The scheduling algorithm must work with different optimization functions depending on the desired strategy. For example, if the cloud provider wants to offer a better service, the scheduler may focus on the maximization of pre-established quality criteria of the underlying IaaS providers. On the other hand, the cloud provider can focus on reducing its running costs, where a cost-oriented scheduler could be used to minimize the budget when renting IaaS resources according to client SLAs. As a consequence, PaaS and SaaS providers can follow the objectives *minimize execution costs* and *maximize user QoS*, as shown in Fig. 17.

Clearly, some of the objectives described here can overlap with each other. How and when to combine such objectives, as well as algorithms for such combinations, are still challenges to be addressed. An even more challenging objective to be considered by a scheduler is the minimization of maintenance and personnel costs, which involves man-hour costs and expected lifetime of datacenter components.

6.4.2. Clients

The final cloud client can be a user that utilizes exclusively either private or public cloud resources, or a user that composes a hybrid cloud to run his/her applications. In both cases, the scheduler objective should be to reduce costs while receiving a satisfactory QoS according to the user/applications requirements. Client QoS in clouds is often measured in terms of response time, performance, and availability, but other requirements regarding resources configuration can also be part of the SLA contract. As in other market niches, scheduling from client perspective usually involves a trade-off between budget and quality (QoS parameters).

From the client perspective, research has been focused in two main objectives: maximize QoS (often minimize the makespan)

and minimize the monetary costs involved when using the public cloud. These objectives can be pursued separately, when only one of them matters for the scheduling, or together, when both objectives are related. One example is the minimization of the monetary costs but at the same time respecting a maximum makespan for the workload.

When we consider the client as the owner of a private cloud, scheduling objectives can have similarities with the scheduling from the provider perspective in the sense that the number of active physical resources should be minimized, avoiding unnecessary monetary costs. However, scheduling in the private cloud is mainly focused on performance metrics instead of profitability. This leads to the development of multi-criteria approaches to deal with the scheduling in hybrid clouds, which needs to address both private cloud objectives and cost minimization to compose the hybrid cloud.

7. Literature review on scheduling in clouds

This section presents a literature review of the scheduling problem in cloud computing. We followed a simple search protocol using online indexing systems.[2] Searches were performed for each year, from 2010 to 2016, using the filtering tools available in the indexing online systems. Variations of the terms *cloud computing* and *scheduling* were appended to the proposed taxonomy branch names to find relevant work for the taxonomy branch problems. For example, to search papers on scheduling for SaaS, the search term *scheduling cloud SaaS* was used. Results were collected and manually filtered to remove manuscripts from other areas (e.g. Physics/weather or works focused on e-Science applications, not in the scheduling problem). Most relevant papers were then selected having as criteria the focus of the text (should be focused on scheduling), venue relevance, and/or number of citations, resulting in a list of 18 papers published in 2010, 14 in 2011, 12 in 2012, 16 in 2013, 30 in 2014, and 11 in 2015, and 13 in 2016, totaling 114 selected research papers we considered relevant to the scheduling problem in cloud computing systems.

Selected papers were classified according to the taxonomy proposed in this paper, considering both the cloud (Section 6) and pre-cloud (Section 5) taxonomy branches. This detailed classification is shown in Tables 1–4, where column titles are the short text notations introduced in the full taxonomy overview in the beginning of this paper (Fig. 3). In these tables, a filled circle means the research paper listed in the row is relevant to the branch listed in that column. A filled circle does not necessarily mean the work in that row explicitly mentions the branch keyword, but states that the branch is somehow considered by that research. These tables aim at helping readers to find unexplored branches and combinations of topics that lack research, but also help in finding relevant related work that can be used as comparison and/or extensions to the readers research.

From Tables 1–4, we highlight the following conclusions based on the classification resulted from the proposed taxonomy:

Organization. The scheduling problem is commonly approached from a centralized perspective, where a central scheduler has information about the system as a whole.

Input accuracy. In the input *accuracy* taxonomy, most scheduling work assumes data input is accurate, while only a few papers consider stochastic or uncertain inputs. Between 2014 and 2016, works considering uncertainty in the input data received more attention than in previous years.

Input semantics. Pre-cloud *semantics* of the input data highlights the established assumption that schedulers often consider as input information about the computing resources as well as about the jobs to be run. Information about the user is less common, with a few works considering budget, for instance.

Frequency. The most common scheduling frequency for algorithms is offline/static. A few works study online schedulers, and some works study semi-static or dynamic schedulers.

Application. Regarding the application model considered by the scheduler in the pre-cloud taxonomy, most works focus on tasks, both dependent and independent. However, a good amount of research is also dedicated to services, with most of them focusing on workflows (dependent services).

Optimization. Scheduling research in computing, for a long time, focused on a single criteria, often makespan. In this review of recent literature, cloud computing scheduling brought the focus to bi- and multi-criteria scheduling approaches, where costs and resource utilization play an important role.

Output. As a consequence of the common assumption that input data is precise, the scheduler output is more commonly considered to be also exact, with imprecise outputs receiving a little bit more attention in the last years.

Target. IaaS clouds are the most common target system considered by scheduling research in clouds. As IaaS is often a basis of PaaS and SaaS, one could expect this focus on IaaS. On the other hand, PaaS and SaaS also need higher-level schedulers that can better understand application requirements.

Location. Literature often considers the cloud client itself as the host of the scheduler, where information about application and users are used to take the best scheduling decision. When the scheduler is assumed to run in the provider, a broker is commonly the system assumed to be hosting its decision-making process (and having an IaaS as the target system). IaaS clouds are also often considered as hosting the scheduler, even though more recently authors considered it fewer times than in cloud computing's earlier years.

Input semantics cloud. Many schedulers in cloud assume they have information about the VMs capacities, specially processing and network. Also, as an important cloud aspect, monetary costs (mostly price) is also a common input data for schedulers in cloud.

Objective. As many works consider the scheduler to be in the client, the objectives considered by the scheduler optimization in cloud research take into account the client point of view: application execution costs and applications QoS. Nevertheless, few works have also taken into account the provider's point of view by considering data-center and execution costs.

Application cloud. Virtual machines and containers were introduced recently, and schedulers in clouds can consider them as the "application" to be scheduler. Only a few works have considered virtual machines or containers scheduling, with most of them being dedicated to VM allocation in cloud data centers.

8. Future directions

Schedulers will need to keep evolving along with distributed systems to properly account new characteristics that arise [13]. In this section we highlight scheduling challenges that still exist in current distributed system, supporting them with information from the literature review above, as well as devise problems that will be important to be addressed in the future of distributed

[2] Google Scholar, IEEExplore, ACM Digital Library, Scopus.

Table 1
Reviewed research articles from 2014 to 2016 classified considering the **cloud branches** of the proposed taxonomy.

	Cloud taxonomy branches																								
	S/T/.../C/SaaS	S/T/.../C/PaaS	S/T/.../C/IaaS/Public	S/T/.../C/IaaS/Private	S/T/.../C/IaaS/Hybrid	S/T/.../C/IaaS/Community	S/OBJ/P/MP/Min. DC costs	S/OBJ/P/MP/Min. ex. costs	S/OBJ/P/Max. user QoS	S/OBJ/C/Min. ex. costs	S/OBJ/C/Max. QoS	S/I/S/MC/Charging Model	S/I/S/MC/Price	S/A/V/Virtual machines	S/A/V/Containers	S/I/S/VMs/Security	S/I/S/VMs/P/Processor	S/I/S/VMs/P/Disk	S/I/S/VMs/P/Network	S/L/P/SaaS	S/L/P/PaaS	S/L/P/IaaS	S/L/C/User	S/L/C/P/SaaS	S/L/C/P/PaaS-Broker
2016																									
Abdullahi, M. et al. [88]	○	○	○	●	○	●	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○	○	●
Bryk, P. et al. [89]	○	○	●	●	○	○	○	○	○	●	●	○	●	○	○	○	●	●	●	○	○	○	●	○	○
Deldari, A et al. [90]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Gupta, I. et al. [91]	○	○	○	○	●	●	○	●	○	○	●	○	○	○	○	○	●	○	●	○	○	○	○	●	●
He, T-Q. et al. [92]	○	○	○	●	○	●	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	○	○	●
Juarez, E. et al. [93]	○	○	○	●	○	○	●	○	●	○	○	○	○	○	○	○	●	●	○	●	○	○	○	○	○
Kliazovich, D. et al. [94]	○	○	●	●	●	●	○	○	○	●	●	○	○	○	○	○	●	○	●	○	●	○	●	○	●
Liu, L. et al. [95]	○	○	●	●	●	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	○	○	●
Maheshwari, K. et al. [96]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	○	●
Omezzine, E. et al. [97]	○	○	●	○	○	○	○	●	●	○	○	○	●	○	○	○	●	○	○	●	●	○	○	○	○
Sedaghat2016, M. et al. [98]	○	○	●	●	○	○	○	●	○	○	○	○	○	●	●	○	●	○	●	○	○	●	○	○	○
Tchernykh, A. et al. [99]	○	○	●	●	○	○	●	○	●	○	○	○	●	○	○	○	●	○	○	○	●	●	○	○	○
Zhang, R. et al. [100]	○	○	●	○	○	○	○	○	○	●	○	○	●	○	○	○	●	○	●	○	○	○	○	○	●
2015																									
Amalarethinam and Beena [101]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Bijon, K. et al. [102]	○	○	●	●	○	○	●	○	●	○	○	○	○	●	○	●	○	○	○	○	○	●	○	○	○
Genez, T. et al. [103]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Hoenisch, P. et al. [104]	○	○	○	○	●	○	○	○	○	●	○	○	●	○	○	○	●	○	○	○	○	○	●	○	●
Huang, D. et al. [105]	○	○	○	○	●	●	○	○	○	○	●	○	○	○	○	○	●	○	●	○	○	●	○	●	○
Lee, Y. C. et al. [106]	○	○	●	●	○	○	●	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	●
Lin, X. et al. [107]	●	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	●	●	○
Malawski, M. et al. [108]	○	○	○	○	●	○	○	○	○	●	○	○	●	○	○	○	●	●	●	○	○	○	●	○	●
Panda and Jana [109]	○	○	○	○	●	●	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	○	●
Sahni and Vidyarthi [110]	○	○	●	●	○	○	○	○	○	●	○	○	●	○	○	○	●	○	○	○	○	○	●	○	●
Vieira, C. et al. [111]	○	○	●	●	○	○	○	○	○	●	●	●	●	●	○	○	●	○	○	○	○	○	○	○	●
2014																									
Bellavista, P. et al. [112]	○	○	●	●	●	○	●	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	●
Calheiros and Buyya [113]	○	○	●	●	○	○	●	○	○	●	○	●	●	○	○	○	●	○	○	○	○	○	●	○	●
Di, S. et al. [114]	○	○	●	●	○	○	○	○	○	○	●	○	●	○	○	○	●	●	●	○	○	○	●	○	●
Duan, R. et al. [115]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	●	●	○	●	○	○	○	○
Frîncu, M.E. [116]	○	●	○	○	○	○	○	●	○	○	●	○	○	○	○	○	○	○	○	○	●	○	○	○	○
Kang, S. [117]	○	○	○	○	●	●	○	○	●	○	○	○	○	○	○	○	●	○	●	○	○	○	○	○	●
Genez, T. et al. [118]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Genez, T. et al. [119]	○	○	○	○	●	○	●	○	○	●	●	○	●	○	○	○	●	○	●	○	○	●	○	○	○
Kune, R. et al. [120]	○	●	○	●	○	○	○	○	●	○	○	○	○	○	○	○	○	●	●	○	●	○	○	○	●
Lin, W et al. [121]	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○	●	○	●	○	○	○	○	○	○
Liu, Z. et al. [122]	○	●	○	○	○	○	○	○	●	○	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Maguluri and Srikant [123]	○	○	●	●	○	○	○	○	●	○	○	○	○	●	○	○	●	●	●	○	○	●	○	○	○
Poola, D. et al. [124]	○	○	●	○	○	○	○	○	○	●	○	●	●	○	○	○	●	○	○	○	●	○	●	○	●

(*continued on next page*)

Table 1 (*continued*)

	Cloud taxonomy branches																							
	S/T/.../C/SaaS	S/T/.../C/PaaS	S/T/.../C/IaaS/Public	S/T/.../C/IaaS/Private	S/T/.../C/IaaS/Hybrid	S/T/.../C/IaaS/Community	S/OBJ/P/MP/Min. DC costs	S/OBJ/P/MP/Min. ex. costs	S/OBJ/P/Max. user QoS	S/OBJ/C/Min. ex. costs	S/OBJ/C/Max. QoS	S/I/S/MC/Charging Model	S/I/S/MC/Price	S/A/V/Virtual machines	S/A/V/Containers	S/I/S/VMs/Security	S/I/S/VMs/P/Processor	S/I/S/VMs/P/Disk	S/I/S/VMs/P/Network	S/L/P/SaaS	S/L/P/PaaS	S/L/P/IaaS	S/L/C/User	S/L/C/P/SaaS	S/L/C/P/PaaS-Broker
Poola, D. et al. [125]	○	○	●	○	○	○	○	○	○	●	○	○	●	○	○	○	●	○	●	○	○	○	●	○	○
Raju, R. et al. [126]	○	○	○	●	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○	●	○	○	○	○
Rodriguez and Buyya [80]	○	○	●	●	●	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●	○	○	○	●
Sandhu and Sood [127]	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○	○	●	●	●	○	○	●	○	○	○
Shie, M.-R. et al. [128]	○	○	●	●	○	○	○	○	○	●	○	●	●	○	○	○	○	○	○	○	○	●	○	○	●
Somasundaram and Govindarajan [129]	○	○	●	●	●	●	○	○	○	○	●	○	●	○	○	○	●	○	●	○	○	○	○	○	●
Tchernykh, A et al. [87]	○	○	●	●	○	○	●	○	●	○	○	○	●	○	○	○	●	○	○	○	●	●	○	○	○
Tsai, C.-W. et al. [130]	○	○	●	●	○	○	○	○	○	○	●	○	○	○	○	○	●	○	○	○	●	○	●	○	●
Vieira, C. et al. [131]	○	○	●	●	○	○	○	○	○	●	●	●	●	●	○	○	●	○	○	○	○	○	○	○	●
Wang, X. et al. [86]	○	○	●	●	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○
Wu, C-M. et al. [85]	○	○	●	●	○	○	●	○	●	○	○	○	○	○	●	○	○	●	○	○	○	●	○	○	○
Wu, L. et al. [132]	○	○	●	●	○	○	○	●	●	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○	○
Xiang, X. et al. [133]	○	○	●	●	○	●	●	○	○	○	○	○	○	○	●	○	○	●	○	●	●	○	○	●	○
Zhang, C et al. [134]	●	○	○	○	○	○	○	○	●	○	○	○	○	○	●	○	○	●	○	○	●	○	○	○	○
Zhang, F. et al. [135]	○	○	○	●	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○
Zhu, X. et al. [136]	○	○	●	●	○	○	●	○	●	○	●	○	○	○	●	○	○	●	○	●	●	○	○	●	○
Zuo, X. et al. [137]	○	○	○	○	●	○	○	●	○	○	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○

Table 2
Reviewed research articles from 2014 to 2016 classified considering the **pre-cloud branches** of the proposed taxonomy.

	Pre-cloud taxonomy branches																				
	S/O/Centralized	S/O/Decentralized	S/O/Hierarchical	S/I/A/Uncertain	S/I/A/Stochastic	S/I/A/Precise	S/I/S/Resources	S/I/S/Jobs	S/I/S/Users	S/F/Online	S/F/Offline/Static	S/F/Offline/Semi-Static	S/F/Offline/Dynamic	S/A/T/Independent	S/A/T/Dependent	S/A/S/Independent	S/A/S/Dependent	S/OPT/Single criterium	S/OPT/Bi-criteria	S/OPT/Multi-criteria	S/OUT/Imprecise	S/OUT/Exact
2016																						
Abdullahi, M. et al. [88]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	●	○	○	○	●
Bryk, P. et al. [89]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Deldari, A. et al. [90]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Gupta, I. et al. [91]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
He, T.-Q. et al. [92]	○	○	●	○	○	●	●	●	○	●	○	○	○	○	●	○	○	○	●	○	○	●
Juarez, E. et al. [93]	●	○	○	○	○	●	●	●	○	○	○	○	●	○	●	○	○	○	●	○	○	●
Kliazovich, D. et al. [94]	●	○	○	●	○	●	●	●	○	○	○	○	○	○	●	○	○	○	●	○	○	●
Liu, L. et al. [95]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	●	○	○	●	○
Maheshwari, K. et al. [96]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Omezzine, E. et al. [97]	○	○	●	○	○	●	●	●	●	○	○	●	○	●	○	○	○	○	○	●	○	●
Sedaghat, M. et al. [98]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	●	○	○	○	●
Tchernykh, A. et al. [99]	●	○	○	○	○	●	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	●
Zhang, R. et al. [100]	●	○	○	○	○	●	●	●	○	●	○	○	○	●	○	○	○	●	○	○	○	●
2015																						
Amalarethinam and Beena [101]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	●	○	●	○	○	●
Bijon, K. [102]	●	○	○	○	○	●	●	●	●	○	●	○	○	○	○	○	○	●	○	○	○	●
Genez, T. et al. [103]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	○	●	○	●	○
Hoenisch, P. et al. [104]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	●	○	○	●	○	●
Huang, D. et al. [105]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Lee, Y.C. et al. [106]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Lin, X. et al. [107]	●	○	○	○	○	●	●	●	○	○	○	●	○	●	○	○	○	●	○	○	○	●
Malawski, M. et al. [108]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	○	○	○	●
Panda and Jana [109]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Sahni and Vidyarthi [110]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Vieira, C. et al. [111]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	●
2014																						
Bellavista, P. et al. [112]	●	○	○	○	○	●	●	○	○	○	○	●	○	○	○	○	○	●	○	○	○	●
Calheiros and Buyya [113]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Di, S. et al. [114]	●	○	○	●	○	○	●	●	●	○	○	○	●	●	○	○	○	○	●	○	●	○
Duan, R. et al. [115]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	○	●	○	●
Frîncu, M.E. [116]	●	○	○	○	○	●	●	●	○	○	○	●	○	○	○	○	●	○	○	●	○	●
Kang, S. [117]	●	○	○	○	○	●	●	●	○	○	●	○	●	○	○	○	○	●	○	○	○	●
Genez, T. et al. [118]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	○	●	○	●	○
Genez, T. et al. [119]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	○	●	○	●	○
Kune, R. et al. [120]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	●	○	○	○	●
Lin, W. et al. [121]	●	○	○	○	○	●	●	○	○	○	●	○	○	●	○	○	○	●	○	○	○	●
Liu, Z. et al. [122]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	●	○	○	○	●
Maguluri and Srikant [123]	●	○	○	○	●	●	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	●
Poola, D. et al. [125]	●	○	○	●	○	○	●	●	●	○	●	○	○	○	●	○	○	○	○	●	●	○
Poola, D. et al. [124]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Raju, R. et al. [126]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	●	○	○	○	●
Rodriguez and Buyya [80]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	●	○	○	●	○
Sandhu and Sood [127]	○	○	●	●	○	○	●	●	○	○	●	○	○	●	○	○	○	●	○	○	●	○
Shie, M.-R. et al. [128]	○	●	○	○	○	●	○	●	○	○	●	○	○	●	○	○	○	●	○	○	○	○
Somasundaram and Govindarajan [129]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Tchernykh, A. et al. [87]	●	○	○	○	○	●	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○
Tsai, C.-W. et al. [130]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	●	○	○	○	●
Vieira, C. et al. [131]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	●
Wang, X. et al. [86]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	○	●	○	●
Wu, C.-M. et al. [85]	●	○	○	○	○	○	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	●
Wu, L. et al. [132]	●	○	○	○	○	●	○	○	○	○	●	○	○	○	○	●	○	○	○	●	○	●
Xiang, X. et al. [133]	○	●	○	○	●	○	●	○	○	●	○	○	○	○	○	○	○	○	○	●	○	○
Zhang, C. et al. [134]	●	○	○	●	○	○	●	●	○	○	○	●	○	○	○	○	○	●	○	○	○	○
Zhang, F. et al. [135]	○	○	○	○	●	○	●	●	○	○	●	○	○	○	●	○	○	○	●	○	●	○
Zhu, X. et al. [136]	●	○	○	○	○	●	●	●	○	○	○	●	○	●	○	○	○	●	○	○	○	●
Zuo, X. et al. [137]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	●	○	●

systems. The discussion presented here is supported by data presented as supplementary material for this article.

Input data. One of the main practical challenges that are present in job scheduling in computer systems is regarding the quality of the input data. In practice, although some applications can reliably estimate job execution time from past executions, for many other applications jobs running times are either unknown or a rough estimation, leading to imprecision in the schedule results. In pre-cloud systems, this can result in problems such

Table 3
Reviewed research articles from 2010 to 2013 classified considering the **cloud branches** of the proposed taxonomy.

	Cloud taxonomy branches																								
	S/T/.../C/SaaS	S/T/.../C/PaaS	S/T/.../C/IaaS/Public	S/T/.../C/IaaS/Private	S/T/.../C/IaaS/Hybrid	S/T/.../C/IaaS/Community	S/OBJ/P/MP/Min. DC costs	S/OBJ/P/MP/Min. ex. costs	S/OBJ/P/Max. user QoS	S/OBJ/C/Min. ex. costs	S/OBJ/C/Max. QoS	S/I/S/MC/Charging Model	S/I/S/MC/Price	S/A/V/Virtual machines	S/A/V/Containers	S/I/S/VMs/Security	S/I/S/VMs/P/Processor	S/I/S/VMs/P/Disk	S/I/S/VMs/P/Network	S/L/P/SaaS	S/L/P/PaaS	S/L/P/IaaS	S/L/C/User	S/L/C/P/SaaS	S/L/C/P/PaaS-Broker
2013																									
Abrishami, S. et al. [138]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	○	●
Dhillon, J. S. et al. [139]	○	○	●	●	○	○	○	○	●	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○
Ergu, D. et al. [140]	○	○	○	●	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○
Genez, T. et al. [141]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Ghribi, C. et al. [84]	○	○	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○
Huang, Y. et al. [142]	○	○	○	○	○	●	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	●	○	○	○
Jain, N. et al. [143]	○	○	○	●	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○
Kessaci, Y. et al. [144]	○	○	●	●	○	○	●	●	○	○	○	○	●	○	○	○	○	●	○	○	○	●	○	○	○
Lucas-Simarro, J.L. et al. [145]	○	○	○	○	●	●	○	○	○	●	●	●	●	●	○	○	○	●	●	○	○	○	○	○	●
Marcon, D. et al. [146]	○	○	●	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	●	○	○	●	○	○
Pereira, W. et al. [147]	○	○	●	○	○	○	○	○	○	●	●	●	●	○	○	○	●	●	●	○	○	○	●	○	●
Rahman, M. et al. [148]	○	○	○	○	●	○	○	○	○	●	●	○	○	○	○	○	●	○	○	○	○	○	●	○	●
Shen, S. et al. [149]	○	○	●	○	●	○	○	○	○	●	○	●	●	○	○	○	●	○	○	○	○	○	●	○	○
Takouna, I. et al. [150]	○	○	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	○	○	●	○	○	●	○	○
Van den Bossche, R. [151]	○	○	●	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	○	○	●
Wu, Z. et al. [152]	○	●	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	○	●
2012																									
Biran, O. et al. [153]	○	○	●	●	○	○	○	○	●	○	●	○	○	●	○	○	●	○	●	○	○	●	○	●	○
Bessai, K. et al. [74]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	●	●	○	○
Bittencourt, L. et al. [81]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Genez, T. et al. [77]	○	○	●	●	●	○	○	●	○	○	●	○	●	○	○	○	●	○	●	○	○	○	○	●	●
Huang, Q. et al. [154]	○	○	○	●	○	○	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○
Lee, Y. et al. [155]	○	○	●	●	●	○	○	●	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	●	●
Leitner, P. et al. [156]	○	○	●	○	○	○	○	●	○	○	○	●	●	○	○	○	○	○	○	●	●	○	○	○	○
Li, J. et al. [157]	○	○	●	●	○	●	●	○	○	○	●	○	●	○	○	○	●	○	●	○	○	●	○	○	●
Lingfang, Z. [158]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	●	●	○	○
Moschakis and Karatza [159]	○	○	●	●	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○
Oliveira, D. et al. [160]	○	○	●	●	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Wang, W. et al. [161]	○	○	●	●	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○
2011																									
Bittencourt and Madeira [29]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Frîncu, M. et al. [162]	○	○	○	○	●	●	●	○	●	○	○	○	○	●	●	○	○	○	○	○	○	○	●	○	●
Garg, S. et al. [163]	○	○	●	●	○	○	●	○	○	○	○	○	●	○	○	○	●	○	●	○	○	●	○	○	○
Jin, J. et al. [164]	○	○	●	●	○	○	○	○	●	○	○	○	○	○	○	○	○	●	●	○	○	●	○	○	○
Juhnke, E. et al. [165]	○	○	○	○	●	●	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○	●
Lago, D. [166]	○	○	●	●	○	○	●	○	●	○	○	○	○	●	○	○	●	○	○	○	○	○	●	○	○
Lee, G. et al. [167]	○	○	●	●	○	○	○	○	●	○	○	○	●	○	○	○	●	●	○	○	●	○	●	○	○
Li, C.-C. et al. [168]	○	○	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○	○
Li, W. et al. [169]	○	○	○	○	●	●	○	●	●	○	○	○	●	●	○	○	●	●	○	○	○	○	○	○	●

(*continued on next page*)

Table 3 (*continued*)

	Cloud taxonomy branches																							
	S/T/.../C/SaaS	S/T/.../C/PaaS	S/T/.../C/IaaS/Public	S/T/.../C/IaaS/Private	S/T/.../C/IaaS/Hybrid	S/T/.../C/IaaS/Community	S/OBJ/P/MP/Min. DC costs	S/OBJ/P/MP/Min. ex. costs	S/OBJ/P/Max. user QoS	S/OBJ/C/Min. ex. costs	S/OBJ/C/Max. QoS	S/I/S/MC/Charging Model	S/I/S/MC/Price	S/A/V/Virtual machines	S/A/V/Containers	S/I/S/VMs/Security	S/I/S/VMs/P/Processor	S/I/S/VMs/P/Disk	S/I/S/VMs/P/Network	S/L/P/SaaS	S/L/P/PaaS	S/L/P/IaaS	S/L/C/User	S/L/C/P/SaaS	S/L/C/P/PaaS-Broker
Mezmas, M. et al. [170]	○	○	●	●	○	○	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○
Prodan, R. et al. [171]	○	○	●	●	○	○	○	●	○	●	●	○	●	○	○	○	○	○	○	○	○	○	●	●
Van den Bossche, R. et al. [172]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	●	●	○	○	○	●	●
Vecchiola, C. et al. [173]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	●
Wu, L. et al. [174]	○	○	●	●	○	○	○	●	●	○	○	○	●	○	○	○	○	○	●	●	○	○	○	○
2010																								
Dörnemann, T. et al. [175]	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	●	○	●	○	○	○	○	●
Fang, Y. et al. [176]	○	○	●	●	○	○	●	○	●	○	○	○	○	●	○	○	○	○	○	○	●	○	○	○
Hu, J. et al. [177]	○	○	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	○	○	○	○	●	○	○
Huang and Huang [178]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	○
Li and Guo [179]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○
Li, J. et al. [180]	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	●	●	●	○	○	○	○	●
Liu, K. et al. [181]	○	○	●	●	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	○	●
Liu, S. et al. [182]	○	●	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●
Moon, J.H. et al. [183]	○	○	○	●	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	○	○	○
Oprescu and Kielmann [184]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	○	○	○	○	○	○	○	●
Pandey, S. et al. [35]	○	○	●	○	○	○	○	○	○	●	○	○	●	○	○	○	●	○	●	○	○	○	●	○
Salehi and Buyya [185]	○	○	○	○	●	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	●
Selvarani et al. [186]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	○	○	○	○	●	○
Van den Bossche, R. et al. [39]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	●
Wei, G. et al. [187]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○
Wu, Z. et al. [188]	○	○	●	○	○	○	○	○	○	●	●	○	●	○	○	○	●	○	●	○	○	○	●	○
Zhang and Li [189]	○	○	●	●	○	○	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○
Zhong, H. et al. [190]	○	○	●	●	○	○	●	○	○	○	○	○	○	○	●	○	○	●	○	●	○	●	○	○

Table 4
Reviewed research articles from 2010 to 2013 classified considering the **pre-cloud branches** of the proposed taxonomy.

	Pre-cloud taxonomy branches																					
	S/O/Centralized	S/O/Decentralized	S/O/Hierarchical	S/IA/Uncertain	S/I/A/Stochastic	S/I/A/Precise	S/I/S/Resources	S/I/S/Jobs	S/I/S/Users	S/F/Online	S/F/Offline/Static	S/F/Offline/Semi-Static	S/F/Offline/Dynamic	S/A/T/Independent	S/A/T/Dependent	S/A/S/Independent	S/A/S/Dependent	S/OPT/Single criterium	S/OPT/Bi-criteria	S/OPT/Multi-criteria	S/OUT/Imprecise	S/OUT/Exact
2013																						
Abrishami, S. et al. [138]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Dhillon, J.S. et al. [139]	○	●	○	○	○	●	●	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●
Ergu, D. et al. [140]	●	○	○	○	○	○	●	●	○	○	●	○	○	●	○	○	○	○	○	●	○	●
Genez, T. et al. [141]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Ghribi, C. et al. [84]	●	○	○	○	○	○	●	○	○	○	●	●	○	○	○	○	○	●	○	○	○	●
Huang, Y. et al. [142]	○	●	○	○	○	●	●	●	○	●	●	○	○	○	●	○	○	●	○	○	○	●
Jain, N. et al. [143]	●	○	○	○	○	●	●	○	●	○	●	○	○	●	○	○	○	●	○	○	○	●
Kessaci, Y. et al. [144]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	○	●	○	●
Lucas-Simarro, J.L. et al. [145]	●	○	○	○	○	●	●	●	●	○	○	●	○	○	○	○	○	○	○	●	○	●
Marcon, D. et al. [146]	●	○	○	○	○	●	●	●	●	○	○	●	○	○	○	○	○	○	○	●	○	●
Pereira, W. et al. [147]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Rahman, M. et al. [148]	●	○	○	○	○	●	●	●	○	○	○	○	●	○	●	○	○	○	●	○	○	●
Shen, S. et al. [149]	●	○	○	○	○	●	●	●	○	●	○	○	○	●	○	○	○	●	○	○	○	●
Takouna, I. et al. [150]	●	○	○	○	○	●	●	○	○	○	○	●	○	○	○	○	○	●	●	○	○	○
Van den Bossche, R. [151]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Wu, Z. et al. [152]	○	○	●	○	○	●	●	●	○	○	●	○	●	○	●	○	○	○	●	○	○	●
2012																						
Biran, O. et al. [153]	●	○	○	○	○	●	●	○	○	○	●	○	○	○	○	○	○	●	○	○	○	●
Bessai, K. et al. [74]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Bittencourt, L. et al. [81]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Genez, T. et al. [77]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Huang, Q. et al. [154]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Lee, Y. et al. [155]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	●	●	○	○	○	●
Leitner, P. et al. [156]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Li, J. et al. [157]	○	●	○	○	○	●	●	●	○	○	○	○	●	○	●	○	○	○	●	○	○	●
Lingfang, Z. [158]	●	○	○	○	○	●	●	●	●	○	●	○	○	○	●	○	○	○	●	○	○	●
Moschakis and Karatza [159]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	●	○	○	○	●
Oliveira, D. et al. [160]	●	○	○	●	○	○	●	●	●	○	○	○	●	○	●	○	○	○	○	●	●	○
Wang, W. et al. [161]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	●	○	○	○	●
2011																						
Bittencourt and Madeira [29]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Frîncu, M. et al. [162]	●	○	○	○	○	●	●	○	○	○	○	●	○	○	○	●	○	○	○	●	○	●
Garg, S. et al. [163]	○	●	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Jin, J. et al. [164]	●	○	○	○	○	●	●	●	○	○	○	○	●	○	●	○	○	●	○	○	○	●
Juhnke, E. et al. [165]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	●	○	●	○	○	●
Lago, D. [166]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	○	●	○	○	○	●
Lee, G. et al. [167]	●	○	○	○	○	●	●	○	○	○	○	○	●	○	●	○	○	●	○	○	○	●
Li, C-C. et al. [168]	●	○	○	○	○	●	●	○	○	○	○	○	●	○	○	○	○	●	○	○	○	●
Li, W. et al. [169]	●	○	○	○	○	●	●	○	○	○	○	○	●	○	○	○	○	○	●	○	○	●
Mezmas, M. et al. [170]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Prodan, R. et al. [171]	○	●	○	○	○	●	●	●	○	○	○	○	●	●	●	○	○	○	●	○	○	●
Van den Bossche, R. et al. [172]	○	●	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Vecchiola, C. et al. [173]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●
Wu, L. et al. [174]	●	○	○	○	○	●	○	○	●	○	●	○	○	○	○	●	○	○	○	●	○	●
2010																						
Dörnemann, T. et al. [175]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	●	●	○	○	○	●
Fang, Y. et al. [176]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	●	○	○	○	●
Hu, J. et al. [177]	●	○	○	○	○	●	●	○	○	○	●	○	○	○	○	○	○	○	○	●	○	●
Huang and Huang [178]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	○	○	●	○	○	●
Li and Guo [179]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	○	●	○	●	○	○	○	●
Li, J. et al. [180]	●	○	○	●	○	○	●	●	○	○	○	●	○	○	●	○	○	●	○	○	●	○
Liu, K. et al. [181]	●	○	○	○	○	●	●	●	●	○	●	○	○	○	○	○	●	○	●	○	○	●
Liu, S. et al. [182]	●	○	○	○	●	○	○	●	○	●	○	○	○	○	○	●	○	○	●	○	○	●
Moon, J.H. et al. [183]	●	○	○	○	○	●	○	●	○	○	●	○	○	○	○	●	○	●	○	○	○	●
Oprescu and Kielmann [184]	●	○	○	●	○	○	●	●	●	○	○	●	○	●	○	○	○	○	●	○	●	○
Pandey, S. et al. [35]	●	○	○	○	○	●	●	●	○	○	○	●	○	○	●	○	●	○	○	○	○	●
Salehi and Buyya [185]	●	○	○	○	○	●	○	○	○	○	○	○	●	●	○	○	○	○	●	○	○	●
Selvarani et al. [186]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Van den Bossche, R. et al. [39]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Wei, G. et al. [187]	●	○	○	○	○	●	●	●	○	○	●	○	○	○	●	○	●	○	●	○	○	●
Wu, Z. et al. [188]	●	○	○	○	○	●	●	●	○	○	○	●	○	○	●	○	●	○	○	○	○	●
Zhang and Li [189]	●	○	○	○	○	●	●	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●
Zhong et al. [190]	●	○	○	○	○	●	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	●

as delayed application execution, increased makespan, deadline misses, among others. In clouds, an additional direct consequence of such unhandled uncertainty is increased costs. Uncertainties can be a result of no previous knowledge about jobs and resources, but also from resource contention when concurrency is present. In the latter, network contention is of special importance when data dependencies and data transfers exist among jobs of the same user or from users sharing the same infrastructure. Most schedulers do not take resource contention into consideration, what can actually bring more uncertainties to the application execution times. Schedulers that are robust to such uncertainties are necessary to avoid the discussed pitfalls and to increase reliability on schedulers estimates.

QoS and QoE. Many schedulers take into account multiple criteria in the decision making. These criteria are commonly associated with quality of service metrics, which in turn are associated with a higher level requirement from the user. To allow a scheduler to autonomously adapt itself, it must understand those higher level requirements, referred as quality of experience (QoE). This involves translating the user's input data (requirements) into lower quality of service levels that can be monitored and adapted according to the user quality of experience. Currently, QoS is challenging for schedulers due to variable requirements from different applications, and this challenge becomes even greater when QoS metrics observed are the same, but a different perception of quality can come from different users for the same application.

Edge/Fog computing. As cloud computing became more mature, the emergence of models to cover requirements not fulfilled by clouds also evolved. One promising distributed system architecture evolution is bringing computing capacity to the edges of the network to reduce latencies [191], as for example in Fog Computing [192]. As computing capacity is moved closer to the user, but cloud computing do still exists to provide larger capacity, a hierarchical system emerges, with each level having its constraints and capabilities. For instance, in Fog Computing *Cloudlets* closer to the user can fulfill lower-latency requirements and locally optimized decisions, while cloud computing keeps having its role as a centralized, high-capacity computing facility. This hierarchical heterogeneous system introduces new variables to be considered by the schedulers, with latency becoming a relevant optimization criterium.

Mobility. Mobile applications are more and more complex as smart devices become ubiquitous, with users often carrying more than one (sometimes multiple with wearable computing) processing-capable, networked devices. Such applications often rely on data storage and processing capacities from the cloud to provide relevant functionalities to mobile users. In this context, mobile users have at a given time and location a set of data and processing tasks that is a subset of all his/her needs. Edge computing, e.g. Fog Computing, associated with mobility can be used to maintain these subsets closer to the user. This is desirable to reduce network traffic and associated delays and latencies [193,194]. To make this effective, resource allocation and scheduling play an important role, and modeling the scheduling problem introducing time and mobility variables is a challenge that arises. These models would have to consider user positioning and movement predictions in order to be able to move data in advance and reduce user's perception of latency in a variety of mobility scenarios and edge resource availability.

Service levels. Schedulers are usually developed for specific service levels. The reviewed literature presents an imbalance in studies for IaaS and other service levels, which suggests schedulers that can take advantage of different service levels are yet to be developed. Moreover, studies to understand the relation between costs and resource utilization at the different cloud service levels could help in reducing cloud costs from the user perspective.

Charging models. Besides being service-level-specific, cloud offerings also bring a variety of charging models with different charging intervals. Notwithstanding, most schedulers do not take the charging models into account. Two traditional charging models for virtual machines are *on-demand* and *reserved* instances. Auction-style instances with variable prices also exist. While some applications can have strict requirements that better match with a charging model that offers a more reliable instance, other applications might be prone to failures, where cheaper instances could be useful. A comprehensive scheduling model that captures the relation between charging models, charging intervals, and pricing would help in establishing lower execution costs within the applications QoS requirements.

IoT and BigData. Internet of Things and Big Data challenges are also relevant to scheduling in distributed systems. With predictions on the number of devices connected to the Internet reaching 50–100 billion [195], the amount of data produced, transferred, and processed will also increase unprecedentedly. Raw data generated by the plethora of connected devices can be stored for further processing or future access, or it can be processed before storage, being reduced to more condensed, meaningful data. The on-the-fly processing of such stream of data is referred as *stream processing* or *complex event processing*. In this kind of processing, incoming data passes through a set of operators to be filtered/aggregated and stored. The way these operators are allocated to the available computing resources have impact in the delay observed between when the data input has taken place and its resulting computation result is obtained. With billions of devices data streams realizing a set of operations, the scale of the scheduling problem can make current decision-making methods unfeasible. Developing fast, online scheduling that can handle big data is a challenging issue that can impact the future of the Internet of Things. Online schedulers are much less common in the past literature than offline schedulers, which makes the development of effective fast online scheduling even more challenging.

Robustness. Models that capture the relation between scheduling and resilience, specially for large applications, can reduce execution times and costs. Any changes in the execution of an application, such as changes in application topology, resources configuration, software configuration, input data, or data sources can turn a successful execution into a failure. An application scheduler that is able to capture failure occurrences from a resilience model and avoid allocations that can potentially fail is a challenging research topic which results can benefit the execution of large applications by avoiding interruption, and consequent waste of money and time.

Online optimization. Most schedulers assume the sequence of jobs is known a-priori. Online scheduling algorithms that capture many characteristics of tasks and systems (such offline schedulers do) are still a challenge in distributed systems. One of the difficulties is that previously dispatched jobs might need to be relocated in order to achieve the aim of a given objective function. Process migration can be costly and complicated, but with the adoption of virtualization (VMs and containers), migration is feasible. Online schedulers that include migration costs and the impact of migration in the running applications can bring online schedulers to attention, with more comprehensive online schedulers becoming relevant.

VM placement. Clouds provide, through virtual machines, resource sharing among tenants. Although resource sharing through VMs promotes software isolation improves resource utilization, it does

no guarantee performance isolation. Virtual machine allocation, or virtual machine placement, algorithms that can avoid performance interference among cloud stakeholders are a challenge because of lack of information or knowledge about the applications users are going to deploy. Moreover, a trade-off between performance isolation and cost is present: a guarantee of performance isolation can actually mean less users/applications per host, reducing resource utilization and consequently increasing infrastructure costs. Application profiling can be used to improve input information for the VM allocation algorithms, but care must be taken not to violate users privacy.

Security and privacy. With the resource sharing and transparent data/processing offloading promoted by virtualization, security and privacy are increasing concerns. As the user cannot control with whom his VM will share the physical resources, the co-located VMs can potentially be owned by anyone around the globe. Although security is a general concern on networked environments, resource allocation and scheduling can take it into account to reduce risks according to application and users requirements. This introduces new constraints to the scheduling problem, what can again result in poorer resource utilization and increased costs.

9. Conclusion

The classification of scheduling solutions is challenging due to the large amount of literature as well as the number of variations of the problem. These very same reasons make the identification of relevant topics in the scheduling problem an important task to be regularly performed by both young and experienced researchers in the field.

In this paper we review several aspects of the scheduling literature for distributed systems, and proposed a taxonomy that encompasses the scheduler organization in the system, the scheduler input and output data, the frequency the scheduler runs, the application model and target system, and the scheduling objectives. The taxonomy is extended to include branches that appeared with the more recent cloud computing paradigm, bringing new target system aspects, the utility perspectives in terms of target system and scheduling objectives, as well as different application model resulted from the cloud virtualized characteristic.

To corroborate the taxonomy, we reviewed a total of 114 papers from the scheduling literature, identifying research according to the proposed taxonomy branches. This resulted in an analysis of the latest scheduling research in distributed systems from a variety of aspects. As a result from the literature review, we also discussed trends and challenges that can be focus of research in the upcoming years.

The work of classifying scheduling research does not come to a foreseeable end. The need for classification and survey of scheduling literature for distributed systems will remain in the future, especially considering the internet of things, mobility, and big data resulting from the ever increasing number of connected devices, as well as the continued existence of computing as a utility service.

Acknowledgments

This work was partially funded by the following agencies/grants: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, and grants #2009/15008-1 and #2015/16332-8, São Paulo Research Foundation (FAPESP), Brazil.

Conflict of interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cosrev.2018.08.002.

References

[1] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer Publishing Company, Incorporated, New York, 2008.
[2] Y.K. Kwok, I. Ahmad, Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors, IEEE Trans. Parallel Distrib. Syst. 7 (5) (1996) 506–521.
[3] Y.K. Kwok, I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Comput. Surv. 31 (1999) 406–471.
[4] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organization, Int. J. High Perform. Comput. Appl. 15 (3) (2001) 200–222.
[5] L.F. Bittencourt, E.R.M. Madeira, N.L.S. da Fonseca, Resource management and scheduling, in: Cloud Services, Networking, and Management, John Wiley & Sons, Inc, 2015, pp. 243–267.
[6] P. Kurp, Green computing, Commun. ACM 51 (10) (2008) 11–13.
[7] M.A. Rappa, The utility business model and the future of computing services, IBM Syst. J. 43 (1) (2004) 32–42.
[8] H. Topcuoglu, S. Hariri, M.Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[9] F. de Oliveira Lucchese, E.J.H. Yero, F.S. Sambatti, M.A.A. Henriques, An adaptive scheduler for grids, J. Grid Comput. 4 (1) (2006) 1–17.
[10] L.F. Bittencourt, E.R.M. Madeira, A performance-oriented adaptive scheduler for dependent tasks on grids, Concurr. Comput.: Pract. Exper. 20 (9) (2008) 1029–1049.
[11] V. Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour, Evaluation of job-scheduling strategies for grid computing, in: GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing, Springer-Verlag, London, UK, 2000, pp. 191–202.
[12] J. Yu, R. Buyya, Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms, Sci. Program. 14 (3,4) (2006) 217–230.
[13] S. Smith, Is scheduling a solved problem? in: G. Kendall, E.K. Burke, S. Petrovic, M. Gendreau (Eds.), Multidisciplinary Scheduling: Theory and Applications, Springer US, 2005, pp. 3–17.
[14] T. Casavant, J. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Softw. Eng. 14 (2) (1988) 141–154.
[15] C. Jiang, C. Wang, X. Liu, Y. Zhao, A survey of job scheduling in grids, in: Proceedings of the Joint 9th Asia-Pacific Web and 8th International Conference on Web-Age Information Management Conference on Advances in Data and Web Management, APWeb/WAIM'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 419–427.
[16] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, Parallel job scheduling — a status report, in: Proceedings of the 10th international conference on Job Scheduling Strategies for Parallel Processing, JSSPP'04, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 1–16.
[17] Wolfgang, M. Wieczorek, A. Hoheisel, R. Prodan, Taxonomies of the multi-criteria grid workflow scheduling problem, in: Grid Middleware and Services, Springer US, 2008, pp. 237–264.
[18] J. Yu, R. Buyya, A taxonomy of workflow management systems for grid computing, J. Grid Comput. 3 (2005) 171–200 10.1007/s10723-005-9010-8.
[19] S. Venugopal, R. Buyya, K. Ramamohanarao, A taxonomy of data grids for distributed data sharing, management, and processing, ACM Comput. Surv. 38 (1) (2006) 3.
[20] D. Hilley, Cloud Computing: a Taxonomy of Platform and Infrastructure-Level Offerings, Georgia Institute of Technology. College of Computing, 2009.
[21] D. Oliveira, F.A. Baião, M. Mattoso, Towards a taxonomy for cloud computing from an e-science perspective, in: N. Antonopoulos, L. Gillam (Eds.), Cloud Computing, in: Computer Communications and Networks, Springer London, 2010, pp. 47–62.
[22] C.W. Tsai, J. Rodrigues, Metaheuristic scheduling for cloud: a survey, Syst. J. IEEE 8 (1) (2014) 279–291.
[23] S. Shaw, A. Singh, A survey on scheduling and load balancing techniques in cloud computing environment, in: Computer and Communication Technology, ICCCT, 2014 International Conference on, 2014, pp. 87–95.

[24] E.N. Alkhanak, S.P. Lee, S.U.R. Khan, Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities, Future Gener. Comput. Syst. 50 (2015) 3–21 Quality of Service in Grid and Cloud 2015.
[25] I. Pietri, R. Sakellariou, Mapping virtual machines onto physical machines in cloud computing: a survey, ACM Comput. Surv. 49 (3) (2016) 49:1–49:30.
[26] F. Wu, Q. Wu, Y. Tan, Workflow scheduling in cloud: a survey, J. Supercomput. (2015) 1–46.
[27] Z.H. Zhan, X.F. Liu, Y.J. Gong, J. Zhang, H.S.H. Chung, Y. Li, Cloud computing resource scheduling and a survey of its evolutionary approaches, ACM Comput. Surv. 47 (4) (2015) 63:1–63:33.
[28] R. Haupt, A survey of priority rule-based scheduling, OR Spectrum 11 (1) (1989) 3–16.
[29] L.F. Bittencourt, E.R.M. Madeira, Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds, J. Internet Serv. Appl. 2 (3) (2011) 207–227.
[30] K. Krauter, K.K.I.R. Buyya, M.M. It, A taxonomy and survey of grid resource management systems for distributed computing, Softw. - Pract. Exp. 32 (2) (2002) 135–164.
[31] Y. Derbal, Entropic grid scheduling, J. Grid Comput. 4 (4) (2006) 373–394.
[32] D.M. Batista, N.L.S. da Fonseca, Robust scheduler for grid networks under uncertainties of both application demands and resource availability, Comput. Netw. 55 (1) (2011) 3–19.
[33] J.M. Schopf, F. Berman, Stochastic scheduling, SC Conference (1999) 48.
[34] M. Pinedo, Stochastic scheduling with release dates and due dates, Oper. Res. 31 (3) (1983) 559–572.
[35] S. Pandey, L. Wu, S.M. Guru, R. Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments, in: Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, AINA '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 400–407.
[36] T. Hagras, J. Janecek, A high performance, low complexity algorithm for compile-time job scheduling in homogeneous computing environments, in: Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on, 2003, pp. 149–155.
[37] H.D. Karatza, R.C.H. Jr., Parallel job scheduling in homogeneous distributed systems, Simulation 79 (5–6) (2003) 287–298.
[38] Y.K. Kwok, A.A. Maciejewski, H.J. Siegel, I. Ahmad, A. Ghafoor, A semi-static approach to mapping dynamic iterative tasks onto heterogeneous computing systems, J. Parallel Distrib. Comput. 66 (1) (2006) 77–98.
[39] R. Van den Bossche, K. Vanmechelen, J. Broeckhove, Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads, in: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, pp. 228–235.
[40] K. Cooper, A. Dasgupta, K. Kennedy, C. Koelbel, A. Mandal, G. Marin, M. Mazina, J. Mellor-Crummey, F. Berman, H. Casanova, A. Chien, H. Dail, X. Liu, A. Olugbile, O. Sievert, H. Xia, L. Johnsson, B. Liu, M. Patel, D. Reed, W. Deng, C. Mendes, Z. Shi, A. YarKhan, J. Dongarra, New grid scheduling and rescheduling methods in the grads project, in: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, 2004, pp. 199.
[41] K. Ramamritham, J. Stankovic, W. Zhao, Distributed scheduling of tasks with deadlines and resource requirements, IEEE Trans. Comput. 38 (8) (1989) 1110–1123.
[42] C.G. Chaves, D.M. Batista, N.L.S. Fonseca, Scheduling grid applications with software requirements, IEEE Latin America Trans. 9 (4) (2011) 578–585.
[43] G.C. Fox, D. Gannon, Workflow in grid systems: editorials, Concurr. Comput. : Pract. Exper. 18 (10) (2006) 1009–1019.
[44] R. de Aquino Gomes, F.M. Costa, An approach to enhance the efficiency of opportunistic grids, Concurr. Comput.: Pract. Exper. (2011).
[45] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, in: 8th Heterogeneous Computing Workshop, IEEE Computer Society, Washington, DC, USA, 1999, pp. 30–44.
[46] C. Castillo, G.N. Rouskas, K. Harfoush, On the design of online scheduling algorithms for advance reservations and QoS in grids, in: Parallel and Distributed Processing Symposium, International, 2007, p. 36.
[47] U. Schwiegelshohn, A. Tchernykh, R. Yahyapour, Online scheduling in grids, in: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, 2008, pp. 1–10.
[48] B. Hamidzadeh, L.Y. Kit, D. Lilja, Dynamic task scheduling using online optimization, Parallel Distrib. Syst. IEEE Trans. 11 (11) (2000) 1151–1163.
[49] C. Papadimitriou, M. Yannakakis, Towards an architecture-independent analysis of parallel algorithms, in: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC '88, ACM, New York, NY, USA, 1988, pp. 510–513.
[50] H. Chang, W. Oldham, Dynamic task allocation models for large distributed computing systems, Parallel Distrib. Syst. IEEE Trans. 6 (12) (1995) 1301–1315.
[51] Q. Tang, S.K.S. Gupta, G. Varsamopoulos, Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach, IEEE Trans. Parallel Distrib. Syst. 19 (2008) 1458–1472.
[52] L. He, S.A. Jarvis, D.P. Spooner, X. Chen, G.R. Nudd, Dynamic scheduling of parallel jobs with QoS demands in multiclusters and grids, in: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 402–409.
[53] R. Buyya, D. Abramson, J. Giddy, Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid, in: High-Performance Computing in the Asia-Pacific Region, International Conference on, vol. 1, 2000, p. 283.
[54] R. Buyya, M. Murshed, Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, Concurr. Comput.: Pract. Exper. 14 (13–15) (2002) 1175–1220.
[55] H. Casanova, Simgrid: a toolkit for the simulation of application scheduling, in: Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, IEEE, 2001, pp. 430–437.
[56] E. Heymann, M. Senar, E. Luque, M. Livny, Adaptive scheduling for master-worker applications on the computational grid, in: R. Buyya, M. Baker (Eds.), Grid Computing GRID 2000, in: Lecture Notes in Computer Science, vol. 1971, Springer Berlin / Heidelberg, 2000, pp. 214–227.
[57] P. françois Dutot, M.A.S. Netto, A. Goldman, F. Kon, Scheduling moldable BSP tasks, in: Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2005), Lecture Notes in Computer Science, Springer, 2005, pp. 157–172.
[58] M.A.S. Netto, A. Goldman, P.F. Dutot, A flexible architecture for scheduling parallel applications on opportunistic computer networks, RT-MAC-2006-01, Department of Computer Science, University of São Paulo, 2006.
[59] C. Boeres, A. Lima, V.E.F. Rebello, Hybrid task scheduling: integrating static and dynamic heuristics, in: Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing, SBAC-PAD '03, IEEE Computer Society, Washington, DC, USA, 2003, p. 199.
[60] H. Casanova, D. Zagorodnov, F. Berman, A. Legrand, Heuristics for scheduling parameter sweep applications in grid environments, in: HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop, IEEE Computer Society, Washington, EUA, 2000, p. 349.
[61] X. He, X. Sun, G. von Laszewski, QoS guided min-min heuristic for grid task scheduling, J. Comput. Sci. Tech. 18 (2003) 442–451 10.1007/BF02948918.
[62] L. Qi, H. Jin, I. Foster, J. Gawor, Provisioning for dynamic instantiation of community services, IEEE Internet Comput. 12 (2) (2008) 29–36.
[63] H. Zhao, R. Sakellariou, A hybrid heuristic for DAG scheduling on heterogeneous systems, in: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, 2004, p. 111.
[64] F.R.L. Cicerre, E.R.M. Madeira, L.E. Buzato, A hierarchical process execution support for grid computing, in: Proceedings of the 2nd Workshop on Middleware for Grid Computing, MGC '04, ACM, New York, NY, USA, 2004, pp. 87–92.
[65] S.G. Akl, F. Dong, Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, 2006–504, School of Computing —Queen's University, Kingston, Ontario, Canada, 2006, p. 55.
[66] M. Tanaka, O. Tatebe, Workflow scheduling to minimize data movement using multi-constraint graph partitioning, in: Cluster Computing and the Grid, IEEE International Symposium on, IEEE Computer Society, Los Alamitos, CA, USA, 2012, pp. 65–72.
[67] A. Bar-Noy, S. Guha, Approximating the throughput of multiple machines in real-time scheduling, SIAM J. Comput. 31 (2) (2002) 331–352.
[68] B.A. Shirazi, K.M. Kavi, A.R. Hurson (Eds.), Scheduling and load balancing in parallel and distributed systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.
[69] A. Bucur, D. Epema, The maximal utilization of processor co-allocation in multicluster systems, in: Parallel and Distributed Processing Symposium, 2003. Proceedings. International, 2003, 10 p.
[70] A. Doğan, F. Özgüner, Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems, Comput. J. 48 (3) (2005) 300–314.
[71] M. Wieczorek, S. Podlipnig, R. Prodan, T. Fahringer, Bi-criteria scheduling of scientific workflows for the grid, in: Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on, 2008, pp. 9–16.
[72] L.C. Canon, E. Jeannot, Scheduling strategies for the bicriteria optimization of the robustness and makespan, in: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, 2008, pp. 1–8.
[73] L.F. Bittencourt, C.R. Senna, E.R.M. Madeira, Bicriteria service scheduling with dynamic instantiation for workflow execution on grids, in: N. Abdennadher, D. Petcu (Eds.), Advances in Grid and Pervasive Computing, in: Lecture Notes in Computer Science, vol. 5529, Springer Berlin / Heidelberg, 2009, pp. 177–188.
[74] K. Bessai, S. Youcef, A. Oulamara, C. Godart, S. Nurcan, Bi-criteria workflow tasks allocation and scheduling in cloud computing environments, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 638–645.

[75] L.F. Bittencourt, R. Sakellariou, E.R.M. Madeira, Using relative costs in workflow scheduling to cope with input data uncertainty, in: Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science, MGC '12, ACM, New York, NY, USA, 2012, pp. 8:1–8:6.
[76] L.C. Canon, E. Jeannot, R. Sakellariou, W. Zheng, Comparative evaluation of the robustness of dag scheduling heuristics, in: S. Gorlatch, P. Fragopoulou, T. Priol (Eds.), Grid Computing, Springer US, 2008, pp. 73–84.
[77] T.A.L. Genez, L.F. Bittencourt, 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, IEEE, 2012.
[78] S. Albagli-Kim, H. Shachnai, T. Tamir, Scheduling jobs with dwindling resource requirements in clouds, in: INFOCOM, 2014 Proceedings IEEE, 2014, pp. 601–609.
[79] P. Mell, T. Grance, The NIST Definition of Cloud Computing 15, National Institute of Standards and Technology (NIST), 2009.
[80] M. Rodriguez, R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, Cloud Comput., IEEE Trans. 2 (2) (2014) 222–235.
[81] L.F. Bittencourt, E.R.M. Madeira, N.L.S. Da Fonseca, Scheduling in hybrid clouds, IEEE Commun. Mag. 50 (9) (2012) 42–47.
[82] R.N. Calheiros, C. Vecchiola, D. Karunamoorthy, R. Buyya, The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds, Future Gener. Comput. Syst. 28 (6) (2012) 861–870.
[83] C.C.A. Vieira, L.F. Bittencourt, E.R.M. Madeira, Towards a PaaS architecture for resource allocation in IaaS providers considering different charging models, in: J. Altmann, K. Vanmechelen, O.F. Rana (Eds.), Economics of Grids, Clouds, Systems, and Services, in: Lecture Notes in Computer Science, vol. 8193, Springer International Publishing, 2013, pp. 185–196.
[84] C. Ghribi, M. Hadji, D. Zeghlache, Energy efficient VM scheduling for cloud data centers: exact allocation and migration algorithms, in: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, 2013, pp. 671–678.
[85] C.M. Wu, R.S. Chang, H.Y. Chan, A green energy-efficient scheduling algorithm using the {DVFS} technique for cloud datacenters, Future Gener. Comput. Syst. 37 (2014) 141–147.
[86] X. Wang, Y. Wang, Y. Cui, A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing, Future Gener. Comput. Syst. 36 (2014) 91–101.
[87] A. Tchernykh, L. Lozano, P. Bouvry, J. Pecero, U. Schwiegelshohn, S. Nesmachnow, Energy-aware online scheduling: Ensuring quality of service for IaaS clouds, in: High Performance Computing Simulation (HPCS), 2014 International Conference on, 2014, pp. 911–918.
[88] M. Abdullahi, M.A. Ngadi, S.M. Abdulhamid, Symbiotic organism search optimization based task scheduling in cloud computing environment, Future Gener. Comput. Syst. 56 (2016) 640–650.
[89] P. Bryk, M. Malawski, G. Juve, E. Deelman, Storage-aware algorithms for scheduling of workflow ensembles in clouds, J. Grid Comput. 14 (2) (2016) 359–378.
[90] A. Deldari, M. Naghibzadeh, S. Abrishami, Cca: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud, J. Supercomput. (2016) 1–26.
[91] I. Gupta, M.S. Kumar, P.K. Jana, Compute-intensive workflow scheduling in multi-cloud environment, in: 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI, 2016, pp. 315–321.
[92] T.Q. He, L.J. Cai, Z.Y. Deng, T. Meng, X. Wang, Queuing-oriented job optimizing scheduling in cloud mapreduce, in: International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Springer, 2016, pp. 435–446.
[93] F. Juarez, J. Ejarque, R.M. Badia, Dynamic energy-aware scheduling for parallel task-based application in cloud computing, Future Gener. Comput. Syst. (2016).
[94] D. Kliazovich, J.E. Pecero, A. Tchernykh, P. Bouvry, S.U. Khan, A.Y. Zomaya, CA-DAG: Modeling communication-aware applications for scheduling in cloud computing, J. Grid Comput. 14 (1) (2016) 23–39.
[95] L. Liu, M. Zhang, R. Buyya, Q. Fan, Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing, Concurr. Comput.: Pract. Exper. (2016) n/a–n/a CPE-16-0064.R2.
[96] K. Maheshwari, E.S. Jung, J. Meng, V. Morozov, V. Vishwanath, R. Kettimuthu, Workflow performance improvement using model-based scheduling over multiple clusters and clouds, Future Gener. Comput. Syst. 54 (2016) 206–218.
[97] A. Omezzine, N.B.B. Saoud, S. Tazi, G. Cooperman, Negotiation based scheduling for an efficient saas provisioning in the cloud, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), 2016, pp. 33–40.
[98] M. Sedaghat, E. Wadbro, J. Wilkes, S.D. Luna, O. Seleznjev, E. Elmroth, Diehard: reliable scheduling to survive correlated failures in cloud data centers, in: 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 52–59.
[99] A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J.E. Pecero, S. Nesmachnow, A.Y. Drozdov, Online bi-objective scheduling for iaas clouds ensuring quality of service, J. Grid Comput. 14 (1) (2016) 5–22.
[100] R. Zhang, K. Wu, M. Li, J. Wang, Online resource scheduling under concave pricing for cloud computing, IEEE Trans. Parallel Distrib. Syst. 27 (4) (2016) 1131–1145.
[101] D.I.G. Amalarethinam, T.L.A. Beena, Customer facilitated cost-based scheduling (CFCSC) in cloud, Procedia Comput. Sci. 46 (2015) 660–667 Proceedings of the International Conference on Information and Communication Technologies, {ICICT}.
[102] K. Bijon, R. Krishnan, R. Sandhu, Mitigating multi-tenancy risks in IaaS cloud through constraints-driven virtual resource scheduling, in: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, SACMAT '15, ACM, New York, NY, USA, 2015, pp. 63–74.
[103] T.A.L. Genez, L. Bittencourt, N. Fonseca, E. Madeira, Estimation of the available bandwidth in inter-cloud links for task scheduling in hybrid clouds, IEEE Trans. Cloud Comput. PP (99) (2015) 1–1.
[104] P. Hoenisch, C. Hochreiner, D. Schuller, S. Schulte, J. Mendling, S. Dustdar, Cost-efficient scheduling of elastic processes in hybrid clouds, in: 8th IEEE International Conference on Cloud Computing, 2015.
[105] D. Huang, P. Du, C. Zhu, H. Zhang, X. Liu, Multi-resource packing for job scheduling in virtual machine based cloud environment, in: Service-Oriented System Engineering, SOSE, 2015 IEEE Symposium on, 2015, pp. 216–221.
[106] Y.C. Lee, H. Han, A.Y. Zomaya, M. Yousif, Resource-efficient workflow scheduling in clouds, Knowl.-Based Syst. 80 (2015) 153–162, 25th anniversary of Knowledge-Based Systems.
[107] X. Lin, Y. Wang, Q. Xie, M. Pedram, Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment, IEEE Trans. Serv. Comput. 8 (2) (2015) 175–186.
[108] M. Malawski, K. Figiela, M. Bubak, E. Deelman, J. Nabrzyski, Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization, Sci. Program. 2015 (2015).
[109] S.K. Panda, P.K. Jana, Efficient task scheduling algorithms for heterogeneous multi-cloud environment, J. Supercomput. 71 (4) (2015) 1505–1533.
[110] J. Sahni, D. Vidyarthi, A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment, Cloud Comput. IEEE Trans. PP (99) (2015) 1–1.
[111] C.C.A. Vieira, L.F. Bittencourt, E.R.M. Madeira, A scheduling strategy based on redundancy of service requests on IaaS providers, in: Parallel, Distributed and Network-Based Processing, PDP, 2015 23rd Euromicro International Conference on, 2015, pp. 497–504.
[112] P. Bellavista, A. Corradi, A. Reale, N. Ticca, Priority-based resource scheduling in distributed stream processing systems for big data applications, in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, 2014, pp. 363–370.
[113] R. Calheiros, R. Buyya, Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through DVFS, in: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, 2014, pp. 342–349.
[114] S. Di, C.L. Wang, F. Cappello, Adaptive algorithm for minimizing cloud task length with prediction errors, Cloud Comput. IEEE Trans. 2 (2) (2014) 194–207.
[115] R. Duan, R. Prodan, X. Li, Thermal-aware scheduling of batch jobs in geographically distributed data centers, Cloud Comput. IEEE Trans. 2 (1) (2014) 71–84.
[116] M.E. Frîncu, Scheduling highly available applications on cloud environments, Future Gener. Comput. Syst. 32 (2014) 138–153.
[117] S. Kang, B. Veeravalli, K.M.M. Aung, Scheduling multiple divisible loads in a multi-cloud system, in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, 2014, pp. 371–378.
[118] T.A.L. Genez, L.F. Bittencourt, N.L.S. da Fonseca, E.R.M. Madeira, Refining the estimation of the available bandwidth in inter-cloud links for task scheduling, in: Global Communications Conference, GLOBECOM, 2014 IEEE, 2014, pp. 1127–1132.
[119] T.A.L. Genez, L.F. Bittencourt, E.R.M. Madeira, On the performance-cost tradeoff for workflow scheduling in hybrid clouds, in: Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on, 2013, pp. 411–416.
[120] R. Kune, P.K. Konugurthi, A. Agarwal, R.R. Chillarige, R. Buyya, Genetic algorithm based data-aware group scheduling for big data clouds, in: Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing, BDC '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 96–104.
[121] W. Lin, C. Liang, J.Z. Wang, R. Buyya, Bandwidth-aware divisible task scheduling for cloud computing, Softw. - Pract. Exp. 44 (2) (2014) 163–174.
[122] Z. Liu, W. Qu, W. Liu, Z. Li, Y. Xu, Resource preprocessing and optimal task scheduling in cloud computing environments, Concurr. Comput.: Pract. Exper. (2014) n/a–n/a.
[123] S. Maguluri, R. Srikant, Scheduling jobs with unknown duration in clouds, Netw. IEEE/ACM Trans. 22 (6) (2014) 1938–1951.
[124] D. Poola, K. Ramamohanarao, R. Buyya, Fault-tolerant workflow scheduling using spot instances on clouds, Procedia Comput. Sci. 29 (2014) 523–533, 2014 International Conference on Computational Science.

[125] D. Poola, S. Garg, R. Buyya, Y. Yang, K. Ramamohanarao, Robust scheduling of scientific workflows with deadline and budget constraints in clouds, in: Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, 2014, pp. 858–865.
[126] R. Raju, J. Amudhavel, M. Pavithra, S. Anuja, B. Abinaya, A heuristic fault tolerant mapreduce framework for minimizing makespan in hybrid cloud environment, in: Green Computing Communication and Electrical Engineering, ICGCCEE, 2014 International Conference on, 2014, pp. 1–4.
[127] R. Sandhu, S. Sood, Scheduling of big data applications on distributed cloud based on QoS parameters, Cluster Comput. (2014) 1–12.
[128] M.R. Shie, C.Y. Liu, Y.F. Lee, Y.C. Lin, K.C. Lai, Distributed scheduling approach based on game theory in the federated cloud, in: Information Science and Applications, ICISA, 2014 International Conference on, 2014, pp. 1–4.
[129] T.S. Somasundaram, K. Govindarajan, CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud, Future Gener. Comput. Syst. 34 (2014) 47–65.
[130] C.W. Tsai, W.C. Huang, M.H. Chiang, M.C. Chiang, C.S. Yang, A hyper-heuristic scheduling algorithm for cloud, Cloud Comput. IEEE Trans. 2 (2) (2014) 236–250.
[131] C.C.A. Vieira, L.F. Bittencourt, E.R.M. Madeira, Reducing costs in cloud application execution using redundancy-based scheduling, in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, 2014, pp. 117–126.
[132] L. Wu, S. Garg, S. Versteeg, R. Buyya, SLA-based resource provisioning for hosted software-as-a-service applications in cloud computing environments, IEEE Trans. Serv. Comput. 7 (3) (2014) 465–485.
[133] X. Xiang, C. Lin, F. Chen, X. Chen, Greening geo-distributed data centers by joint optimization of request routing and virtual machine scheduling, in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, 2014, pp. 1–10.
[134] C. Zhang, J. Yao, Z. Qi, M. Yu, H. Guan, vGASA: adaptive scheduling algorithm of virtualized GPU resource in cloud gaming, Parallel Distrib. Syst. IEEE Trans. 25 (11) (2014) 3036–3045.
[135] F. Zhang, J. Cao, K. Li, S.U. Khan, K. Hwang, Multi-objective scheduling of many tasks in cloud platforms, Future Gener. Comput. Syst. 37 (2014) 309–320.
[136] X. Zhu, L. Yang, H. Chen, J. Wang, S. Yin, X. Liu, Real-time tasks oriented energy-aware scheduling in virtualized clouds, Cloud Comput. IEEE Trans. 2 (2) (2014) 168–180.
[137] X. Zuo, G. Zhang, W. Tan, Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud, Autom. Sci. Eng. IEEE Trans. 11 (2) (2014) 564–573.
[138] S. Abrishami, M. Naghibzadeh, D.H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Gener. Comput. Syst. 29 (1) (2013) 158–169.
[139] J. Dhillon, S. Purini, S. Kashyap, Virtual machine coscheduling: a game theoretic approach, in: Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on, 2013, pp. 227–234.
[140] D. Ergu, G. Kou, Y. Peng, Y. Shi, Y. Shi, The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment, J. Supercomput. 64 (3) (2013) 835–848.
[141] T. Genez, L. Bittencourt, E. Madeira, Using time discretization to schedule scientific workflows in multiple cloud providers, in: Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on, 2013, pp. 123–130.
[142] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, B. Hirsbrunner, Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm, Future Gener. Comput. Syst. 29 (1) (2013) 402–415.
[143] N. Jain, I. Menache, J. Naor, J. Yaniv, A truthful mechanism for value-based scheduling in cloud computing, Theory Comput. Syst. 54 (3) (2014) 388–406.
[144] Y. Kessaci, N. Melab, E.G. Talbi, A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation, Cluster Comput. 16 (3) (2013) 451–468.
[145] J.L. Lucas-Simarro, R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente, Scheduling strategies for optimal service deployment across multiple clouds, Future Gener. Comput. Syst. 29 (6) (2013) 1431–1441.
[146] D. Stefani Marcon, R. Ruas Oliveira, M. Cardoso Neves, L. Salete Buriol, L. Gaspary, M. Pilla Barcellos, Trust-based grouping for cloud datacenters: improving security in shared infrastructures, in: IFIP Networking Conference, 2013, 2013, pp. 1–9.
[147] W.F. Pereira, L.F. Bittencourt, N.L.S. da Fonseca, Scheduler for data-intensive workflows in public clouds, in: Cloud Computing and Communications (LatinCloud), 2nd IEEE Latin American Conference on, 2013, pp. 41–46.
[148] M. Rahman, R. Hassan, R. Ranjan, R. Buyya, Adaptive workflow scheduling for dynamic grid and cloud computing environment, Concurr. Comput.: Pract. Exper. 25 (13) (2013) 1816–1842.
[149] S. Shen, K. Deng, A. Iosup, D. Epema, Scheduling jobs in the cloud using on-demand and reserved instances, in: F. Wolf, B. Mohr, D. an Mey (Eds.), Euro-Par 2013 Parallel Processing, in: Lecture Notes in Computer Science, vol. 8097, Springer Berlin Heidelberg, 2013, pp. 242–254.
[150] I. Takouna, R. Rojas-Cessa, K. Sachs, C. Meinel, Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers, in: IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC, 2013, pp. 251–255.
[151] R.V. den Bossche, K. Vanmechelen, J. Broeckhove, Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds, Future Gener. Comput. Syst. 29 (4) (2013) 973–985.
[152] Z. Wu, X. Liu, Z. Ni, D. Yuan, Y. Yang, A market-oriented hierarchical scheduling strategy in cloud workflow systems, J. Supercomput. 63 (1) (2013) 256–293.
[153] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, E. Silvera, A stable network-aware vm placement for cloud systems, in: Cluster, Cloud and Grid Computing, CCGrid, 2012 12th IEEE/ACM International Symposium on, 2012, pp. 498–506.
[154] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, X. Huang, Enhanced energy-efficient scheduling for parallel applications in cloud, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ccgrid 2012, CCGRID '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 781–786.
[155] Y.C. Lee, C. Wang, A.Y. Zomaya, B.B. Zhou, Profit-driven scheduling for cloud services with data access awareness, J. Parallel Distrib. Comput. 72 (4) (2012) 591–602.
[156] P. Leitner, W. Hummer, B. Satzger, C. Inzinger, S. Dustdar, Cost-efficient and application SLA-aware client side request scheduling in an infrastructure-as-a-service cloud, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 213–220.
[157] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, Z. Gu, Online optimization for scheduling preemptable tasks on IaaS cloud systems, J. Parallel Distrib. Comput. 72 (5) (2012) 666–677.
[158] L. Zeng, B. Veeravalli, X. Li, ScaleStar: budget conscious scheduling precedence-constrained many-task workflow applications in cloud, in: Advanced Information Networking and Applications, AINA, 2012 IEEE 26th International Conference on, 2012, pp. 534–541.
[159] I. Moschakis, H. Karatza, Evaluation of gang scheduling performance and cost in a cloud computing system, J. Supercomput. 59 (2) (2012) 975–992.
[160] D. de Oliveira, K. Ocaña, F. Baião, M. Mattoso, A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds, J. Grid Comput. 10 (3) (2012) 521–552.
[161] W. Wang, G. Zeng, D. Tang, J. Yao, Cloud-DLS: Dynamic trusted scheduling for Cloud computing, Expert Syst. Appl. 39 (3) (2012) 2321–2329.
[162] M. Frincu, C. Craciun, Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments, in: Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, 2011, pp. 267–274.
[163] S.K. Garg, C.S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of {HPC} applications on distributed Cloud-oriented data centers, J. Parallel Distrib. Comput. 71 (6) (2011) 732–749 Special Issue on Cloud Computing.
[164] J. Jin, J. Luo, A. Song, F. Dong, R. Xiong, Bar: an efficient data locality driven task scheduling algorithm for cloud computing, in: Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, in: CCGRID '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 295–304.
[165] E. Juhnke, T. Dornemann, D. Bock, B. Freisleben, Multi-objective scheduling of BPEL workflows in geographically distributed clouds, in: Cloud Computing (CLOUD), 2011 IEEE International Conference on, 2011, pp. 412–419.
[166] D.G.d. Lago, E.R.M. Madeira, L.F. Bittencourt, Power-aware virtual machine scheduling on clouds using active cooling control and DVFS, in: Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science, MGC '11, ACM, New York, NY, USA, 2011, pp. 2:1–2:6.
[167] G. Lee, B.G. Chun, H. Katz, Heterogeneity-aware resource allocation and scheduling in the cloud, in: Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'11, USENIX Association, Berkeley, CA, USA, 2011 4–4.
[168] C.C. Lin, P. Liu, J.J. Wu, Energy-aware virtual machine dynamic provision and scheduling for cloud computing, in: Cloud Computing (CLOUD), 2011 IEEE International Conference on, 2011, pp. 736–737.
[169] W. Li, J. Tordsson, E. Elmroth, Modeling for dynamic cloud scheduling via migration of virtual machines, in: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, 2011, pp. 163–171.
[170] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E.G. Talbi, A. Zomaya, D. Tuyttens, A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, J. Parallel Distrib. Comput. 71 (11) (2011) 1497–1508.
[171] R. Prodan, M. Wieczorek, H. Fard, Double auction-based scheduling of scientific applications in distributed grid and cloud environments, J. Grid Comput. 9 (4) (2011) 531–548.

[172] R. Van den Bossche, K. Vanmechelen, J. Broeckhove, Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds, in: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, 2011, pp. 320–327.
[173] C. Vecchiola, R.N. Calheiros, D. Karunamoorthy, R. Buyya, Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka, Future Gener. Comput. Syst. (Aceito para publicação / disponível online) (2011).
[174] L. Wu, S. Garg, R. Buyya, SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments, in: Cluster, Cloud and Grid Computing, CCGrid, 2011 11th IEEE/ACM International Symposium on, 2011, pp. 195–204.
[175] T. Dörnemann, E. Juhnke, T. Noll, D. Seiler, B. Freisleben, Data flow driven scheduling of BPEL workflows using cloud resources, in: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, pp. 196–203.
[176] Y. Fang, F. Wang, J. Ge, A task scheduling algorithm based on load balancing in cloud computing, in: F. Wang, Z. Gong, X. Luo, J. Lei (Eds.), Web Information Systems and Mining, in: Lecture Notes in Computer Science, vol. 6318, Springer Berlin Heidelberg, 2010, pp. 271–277.
[177] J. Hu, J. Gu, G. Sun, T. Zhao, A scheduling strategy on load balancing of virtual machine resources in cloud computing environment, in: Parallel Architectures, Algorithms and Programming, PAAP, 2010 Third International Symposium on, 2010, pp. 89–96.
[178] Q.y. Huang, T.l. Huang, An optimistic job scheduling strategy based on QoS for cloud computing, in: Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on, 2010, pp. 673–675.
[179] Q. Li, Y. Guo, Optimization of resource scheduling in cloud computing, in: Symbolic and Numeric Algorithms for Scientific Computing, SYNASC, 2010 12th International Symposium on, 2010, pp. 315–320.
[180] J. Li, M. Qiu, J.W. Niu, Y. Chen, Z. Ming, Adaptive resource allocation for preemptable jobs in cloud systems, in: Intelligent Systems Design and Applications, ISDA, 2010 10th International Conference on, 2010, pp. 31–36.
[181] K. Liu, H. Jin, J. Chen, X. Liu, D. Yuan, Y. Yang, A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform, Int. J. High Perform. Comput. Appl. 24 (4) (2010) 445–456 arXiv:http://hpc.sagepub.com/content/24/4/445.full.pdf+html.
[182] S. Liu, G. Quan, S. Ren, On-line scheduling of real-time services for cloud computing, in: Services (SERVICES-1), 2010 6th World Congress on, 2010, pp. 459–464.
[183] H.J. Moon, Y. Chi, H. Hacigümüs, SLA-aware profit optimization in cloud services via resource scheduling, in: Services (SERVICES-1), 2010 6th World Congress on, 2010, pp. 152–153.
[184] A.M. Oprescu, T. Kielmann, Bag-of-Tasks scheduling under budget constraints, in: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 2010, pp. 351–359.
[185] M. Salehi, R. Buyya, Adapting market-oriented scheduling policies for cloud computing, in: C.H. Hsu, L. Yang, J. Park, S.S. Yeo (Eds.), Algorithms and Architectures for Parallel Processing, in: Lecture Notes in Computer Science, vol. 6081, Springer Berlin Heidelberg, 2010, pp. 351–362.
[186] S. Selvarani, G. Sadhasivam, Improved cost-based algorithm for task scheduling in cloud computing, in: Computational Intelligence and Computing Research, ICCIC, 2010 IEEE International Conference on, 2010, pp. 1–5.
[187] G. Wei, A.V. Vasilakos, Y. Zheng, N. Xiong, A game-theoretic method of fair resource allocation for cloud computing services, J. Supercomput. 54 (2) (2009) 252–269.
[188] Z. Wu, Z. Ni, L. Gu, X. Liu, A revised discrete particle swarm optimization for cloud workflow scheduling, in: Computational Intelligence and Security, CIS, 2010 International Conference on, 2010, pp. 184–188.
[189] L.M. Zhang, K. Li, Y.Q. Zhang, Green task scheduling algorithms with speeds optimization on heterogeneous cloud servers, in: Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing, IEEE Computer Society, Washington, DC, USA, 2010, pp. 76–80.
[190] H. Zhong, K. Tao, X. Zhang, An approach to optimized resource scheduling algorithm for open-source cloud systems, in: ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual, 2010, pp. 124–129.
[191] L.F. Bittencourt, O. Rana, I. Petri, Cloud computing at the edges, in: M. Helfert, V. Méndez Muñoz, D. Ferguson (Eds.), Cloud Computing and Services Science, Springer International Publishing, Cham, 2016, pp. 3–12.
[192] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.
[193] L.F. Bittencourt, M.M. Lopes, I. Petri, O.F. Rana, Towards virtual machine migration in fog computing, in: 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015, pp. 1–8.
[194] L.F. Bittencourt, J. Diaz-Montes, R. Buyya, O.F. Rana, M. Parashar, Mobility-aware application scheduling in fog computing, IEEE Cloud Comput. 4 (2) (2017) 26–35.
[195] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Context aware computing for the internet of things: a survey, Commun. Surv. Tutor. IEEE 16 (1) (2014) 414–454.