Cloud Resource Scheduling Taxonomy

ABSTRACT

The growth and development of commercial and scientific applications in the cloud demand the creation of efficient resource management systems to coordinate the resources while addressing the heterogeneity of services, the inter-dependencies, and unpredictability of load posed by the users. We present a resource scheduling taxonomy that originates from the experience of the authors in utilizing and managing multi-cloud environments. This study is backed up by a literature review that targets not only virtual machines but also container and Function as a Service frameworks. It justifies a proposed resource provider focused Y-cloud taxonomy and introduces an overview of existing scheduling techniques in cloud computing. As a result, this work can lead to a better understanding of the complex field of scheduling for clouds in general. Furthermore, the study promotes through the Y-cloud taxonomy, the vision of a layered scheduling architecture that will be useful for the implementation of application and resource-based scheduling frameworks in support of the NIST Big Data Reference Architecture.

ACM Reference Format:

. 2022. Cloud Resource Scheduling Taxonomy. In *Accepted at the 2022 Emerging Researchers National (ERN) Conference in STEM, Washington, D.C., due to COVID-19, the conference is postponed until 2023.* ACM, New York, NY, USA, 35 pages.

1 INTRODUCTION

Cloud computing has emerged as a computing paradigm to fulfill large-scale application requirements in domains including science, e-commerce, lifestyle, and many other fields. According to the definition of NIST, *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction* [120].

Sustaining efficient resource provisioning and utilization in clouds is a formidable challenge. Poor resource management results in high costs that are amplified by long term and dynamic resource usage we see in many cloud applications. Hence, scheduling plays an important role in improving resource utilization and optimization. Consequently, resource scheduling is an important service of any cloud framework as it is responsible for orchestrating the resources to both cloud providers and cloud users in an efficient manner.

In this paper, we contribute to the argument that scheduling in the cloud requires a multi-layered approach that not only schedules tasks and jobs but also integrates resource provisioning and dynamic resource adaptation during the runtime of cloud applications. Information has to be passed between the various layers that comprise this scheduling architecture for clouds to guide the optimal placement onto resources. Hence, a cloud-based scheduling model is more comprehensive than previous classical scheduling approaches as it is conducted on scales and types of resources that were previously not considered. Scheduling is not only done on the task, job, and cluster-level but integrates the data center and even regional and global data centers while adding on-demand resource needs. To work towards a layered scheduling model we have introduced a Y-Cloud-Taxonomy that allows us to work towards the identification and implementation of scheduling models and algorithms at different junction points. Furthermore, this study already contributed considerably to the identification of services that assist in the formulation of the scheduling needs and interfaces with the NIST Big Data Reference Architecture (NIST-BDRA) [1] definitions currently under development [153].

The paper is structured as follows. In Section 2 the terminology used in the paper is introduced. Next, we present in Section 3 an architecture view and taxonomy that we derived from the practical experience with FutureGrid [79], FutureSystem, and Software such as Cloudmesh [160], Virtual Clusters [133], and Rain [71, 155, 159] while working on hybrid and multi-cloud frameworks.

This view is backed up by an extensive literature review presented in Sections 4 and their classification based on the taxonomy introduced in Section 3. Lastly, we provide some concluding remarks in Section 5.

1.1 Contributions

The contributions of this paper are the following:

- We introduce a resource provider focussed Y-Cloud Taxonomy 3.2 that establishes a provider view associating physical, resource, and connectivity models for clouds with each other (Section 3.2). This view helps to implement a layered scheduling approach.
- We identify specific characteristics we face in cloud computing that provide specific scheduling challenges motivated by the use of clouds.
- We introduce a detailed general classification of cloud scheduling while analyzing clouds in regards to the cloud infrastructure, the models to describe and utilize the cloud infrastructure efficiently, and categorize scheduling frameworks and algorithms to address the many scheduling problems arising in the cloud.
- Based on the lessons learned while being a resource provider for clouds, a developer and a researcher of cloud software and applications we identified that a layered and phased scheduling model is beneficial. The benefits of such a model include the separation of scheduling concerns between infrastructure, platform, software, and function as a Service while at the same time projecting a holistic approach.

ERN'22, due to COVID-19, the conference is postponed until 2023, Washington, D.C.
2022.

- We provide a systematic survey of cloud scheduling approaches and associate them with the presented scheduling taxonomy.
- We identify areas that have not yet been addressed by this paper and outline future activities.

2 TERMINOLOGY AND BASIC CONCEPTS

In this section, terminology and basic concepts related to cloud and scheduling are discussed.

2.1 General Scheduling Terminology for Clouds

We use the following terminology for Cloud computing and Resource scheduling:

Cloud Computing is according to the definition of NIST, Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [120].

Cloud Resource is a resource offered by a cloud provider on which cloud services are run as part of the implementation of a cloud application that may use this resource.

Cloud Service is a service offered by a cloud provider or developed as part of an application utilizing cloud resources and exposing the functionality as a service.

Cloud Application is an application that uses cloud services and resources for its instantiation and execution.

Resource Provisioning in the Cloud is the process of allocating resources demanded by services and applications running in the cloud.

Resource Scheduling in the Cloud refers to the mapping of resources to fulfill the cloud service requirements.

Cloud Scheduler refers to a service that maps basic cloud scheduling units such as virtual machines, containers, functions, and data onto cloud resources to utilize them while leveraging a scheduling policy.

Cloud Scheduling Policy refers to a policy employed by the scheduler to derive decisions as to how to guide a scheduling algorithm.

Cloud Scheduling Algorithm refers to an algorithm that includes cloud scheduling units and policies (as defined next), and resources as input and determines an optimized mapping of cloud scheduling units to cloud resources.

2.2 Scheduling Units

The traditional units for scheduling include processes, tasks, and jobs. However, in the cloud, it is beneficial to consider an enhanced set of scheduling units. These units must include scheduling of virtual machines, containers, functions, platforms, clusters, services, and other infrastructure or services used by the clients or cloud-related services. Naturally, such units can be abstracted into tasks that are coordinated as part of cloud workflows.

Hence, we distinguish the following scheduling units related to cloud computing:

Task is an abstract unit to be run on a cloud that may have complex resource requirements attached to them and may itself be built from other tasks.

Job is a computational activity made up of several tasks that may require different processing capabilities while resolving the resource requirements as part of a scheduling process.

Function is a small computational unit executed as service with precisely specified resource requirements to run on a cloud. Please note that to distinguish them from the common term we also refer to them as Function as a Service.

Application is a software solution for solving a (large) problem in a computational infrastructure. Applications may require splitting the use of any combination of tasks, jobs, services, and functions while using Cloud resources to solve the requirements of the applications. The allocation of resources is usually referred to as application deployment.

Workflow contains a combination of Tasks, Jobs, Functions, and applications with dependencies assuring the order of execution.

Tasks, services, functions, and applications must be mapped onto cloud resources to be able to be executed. The association of such resources is typically conducted in the resource provisioning. We list next the terminology related to provisioning:

Resource is a basic computational entity that can be used to fulfill the requirements of the application's execution. Resources have specific characteristics such as CPU, memory, software, disks, etc. Various performance and policy parameters are associated with a resource, among them, the data speed, the processing speed, space, and workload, which change over time, as well as cost, authentication, and authorization policies.

Deployment is a series of jobs that deploy services onto the cloud that can be used for subsequent use as part of an application or service.

Container is an agglomeration of software that includes all packages and dependencies so it can be run easily on cloud computing resources due to its standardized specification.

Virtual machine (VM) is a simple software program that simulates the functions of a physical machine.

Virtual cluster is an agglomeration of virtual services that build the core of a computational resource hosted in the cloud. A virtual cluster can be comprised out of many resources including virtual machines, containers, Platform as a Service frameworks, data services, and resources, and more. A virtual cluster may be associated with an application and optimized for its use. Just as containers or virtual machines, a virtual cluster can be created, suspended, resumed, or terminated.

Scheduler is a process that decides which task and process should be accessed and run at a specific time by the resources. Schedulers help to keep the performance of the cloud at the highest level by using optimization strategies. Based on the scope of resources involved in the scheduling decision we distinguish between global, regional, and local schedulers.

Task, Job, Application, Service, Function scheduling is to allocate resources to a particular scheduling unit so they can be executed. Limited resource availability and their cost motivate the development of optimized scheduling algorithms to address the problem of task scheduling.

Provisioning is a process to aggregate resources and services that are used as part of the application or software service-related infrastructure setup. Provisioning helps users to simplify the resource management tasks while accessing resources that are hosted in the cloud and made available to the user through provisioning.

3 SCHEDULING TAXONOMY FOR CLOUDS

In this section, we introduce a scheduling taxonomy for clouds. The taxonomy integrates the classical service-oriented cloud architectures defined by NIST [120].

First, we will introduce a motivation for introducing the concept of layered scheduling that motivates the use of the taxonomy in the separate layers.

Second, we will be introducing a resource provider focused Y-Cloud taxonomy that deals with showcasing the relationship between cloud resources, their physical instantiation and their connectivity in a layered fashion depicted as a Y-diagram.

Next, we present in the taxonomy classifications.

3.1 Layered scheduling

The NIST cloud model promotes an easy to understand separation between infrastructure, platforms, and software as a service. This separation motivates a scheduling taxonomy separated by the different layers in which service providers and users attempt to place compute, data and other services in order to optimize the use of the infrastructure as is showcased in Figure 1, in which we added also Function as a Service (FaaS) as it is going to be playing a major role in upcoming cloud Software as a Service offerings, just as platforms did.

A platform provider may utilize insights of the infrastructure to offer to the users an optimized platform placement, while a software provider or application user may utilize information from the platform and or the infrastructure to offer to schedule on levels accessible to them. To facilitate the scheduling on the lower levels, scheduling information has to be passed along to them to provide enough information to the provider to integrate scheduling of resources that are not under direct control by the developer and users.

Thus one strategy to develop scheduling algorithms for the cloud is to integrate the service boundaries of the layered cloud architecture into conducting a multi-layered scheduling approach. In this approach, we separate scheduling concerns related to resources, platform, function and application scheduling as showcased in Figure 2. As most recently the FaaS model has gained traction we added it to Figure 2 to indicate

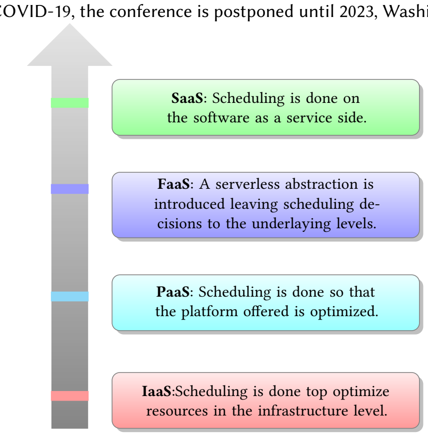

Figure 1: Multi-phase scheduling in a hierarchical resource model with less scheduling control and needs in the higher service levels by the user [151].

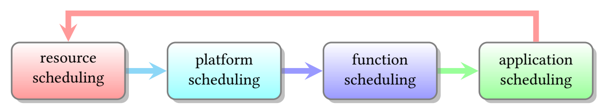

Figure 2: Multi-layered scheduling in a hierarchical resource model motivated by the NIST cloud architecture [151].

that through the use of resource bound functions scheduling decisions propagated to the infrastructure provider level become easier. Hence to optimize usage of the infrastructure, cloud providers have integrated the use of functions in their portfolio in addition to the original NIST model. This integration allows for the better potential of utilizing the infrastructure by scheduling small well-defined functions with limited resource needs.

Certainly, the goal of hiding the scheduling decisions between each layer is still important to reduce complexity exposed to the users and developers, but if enough information between the layers is exchanged, this information can lead to good scheduling decisions on each of the layers.

When putting together, we distinguish several aspects that relate to cloud scheduling. This includes metrics, cloud scheduling models, the cloud infrastructure, and algorithms specifically designed to address clouds as seen in Figure 3. These aspects are elaborated in more detail next.

3.2 Resource Provider Focused Y-Cloud Taxonomy

To showcase the interaction between the different layers more clearly we like to refer the reader to the Y-cloud scheduling diagram introduced by Laszewski in [151].

In this taxonomy, we are concerned about how resources are placed on physical models and are interconnected with each other to facilitate scheduling algorithms. Figure 4 depicts the different models that are an integral part of this taxonomy. It includes the

Physical Model representing major physical resource layers to enable a hierarchical scheduling strategy across multiple data centers, racks, servers, and computing cores.

Resource Model representing resource-based scheduling decisions while dealing with containers and functions, virtual machines and jobs, virtual clusters, provider-managed resources, and multi-region provider-managed resources.

Connectivity Model introducing connectivity between components when addressing scheduling. This includes components such as memory, processes, connectivity to distributed resources, hyper-graphs to formulate hierarchies of provider-based resources, and region enhanced hyper-graphs. The connectivity model allows us to leverage classical scheduling algorithms while applying such models and leveraging established or new scheduling algorithms for these models.

3.3 Cloud Scheduling Model

Now that we have identified the resource provider focused Y-Cloud taxonomy we can identify some important classifications that govern the scheduling decisions to effectively use these resources. This includes metrics that influence the scheduling. Traditional scheduling metrics and attributes for scheduling algorithms are shown in Figure 5. They include typically cost, time, space, reliability, energy, and security.

When looking into the specifics of these metrics applied to cloud computing we can easily identify more details for these traditional metrics that apply to the various infrastructure components that constitute a cloud including compute, data, energy, quality of service, and security. We depict some of the major attributes that influence the scheduling decisions in Figure 6. Furthermore, each of the attributes in the categories Compute, Data, Security, Energy, and Quality of service can be combined if not already included in the specific scheduling attribute. For example, to identify a scheduling model based on virtual machines, attributes such as those in data, energy, QoS, or security may be introduced in the scheduling decision.

This information can now be used to define provisioning and service scheduling as categorized next.

3.4 Challenges in Cloud Scheduling

It is important to understand that cloud scheduling is going beyond traditional scheduling approaches. For this reason, we need to look at specific challenges we face that will lead us to features that need to be addressed by scheduling solutions for the cloud and build significant requirements to be addressed by scheduling solutions

Some of the obvious cloud characteristics and challenges are listed next and are summarized in Figure 7.

Large scale: Clouds offer a large number of resources to its users that need to be optimally utilized under the quality of service constraints set by providers and users. A cloud involving a plethora of resources spanning across the globe is obviously a huge infrastructure. The range of functions, tasks, jobs, and applications need to be scheduled at any point of time onto available resources. Handling them on such scale requires efficient resource management. As such, scheduling becomes a complex endeavor, integrating dynamic and multi-faceted scheduling.

Dynamic nature of clouds: Clouds encompass a dynamically changing resource environment in which resources belong to different administrative domains keep on joining and leaving the clouds. Hence, scheduling must be adaptive and address the dynamic resource availability.

Heterogeneous providers and services: There is no single cloud. We have to recognize that the competitive nature in the cloud market promotes not only heterogeneous cloud providers but heterogeneous cloud services that may compete with each other and either offer the same or customized services targeting a particular user community. Resources in clouds are highly diversified in nature, capacity, working style, and administrative domains. The inclusion of different resource providers with the desire to lock customers into their services and products makes heterogeneous multi-cloud scheduling a formidable challenge.

Highly diversified: Due to the large diverse set of applications (but also infrastructure) smart strategies to schedule such applications on the required resources are needed.

Decentralized: The resources in the cloud are distributed among various data centers, rack, and servers. Although they may belong to a provider, they can still be utilized across provider boundaries and even within the same provider regions, calling for a high degree of decentralization.

Limited control by users: Due to the fundamental nature of the cloud, access to low-level scheduling mechanisms is often hidden and only available to the provider. On the other hand, users still have their own scheduling requirements in regards to, for example, cost and deadlines.

Dynamic loads: Due to the size of the user community sporadic burst on resource requirements lead to challenges to adjust provisioned resources and schedule application onto them.

Security concerns: Another important requirement for scheduling is the ability to integrate issues such as privacy and security considerations as the provider needs to assure that local laws, as well as, the general privacy and security concerns are addressed. This is especially of concern when government or health providers need to schedule resources in a cloud for their application needs, making it necessary to distinguish problems that can be executed on public vs private clouds through scheduling but also through policy decisions that integrate with scheduling algorithms.

Thus, we need to distinguish many scheduling challenges, one of which is governed by differentiating users and providers. Here, on the one hand, we focus on cloud providers that try to utilize in the best possible way to utilize the existing resources for the customers under optimization constraints such as cost, high availability, fault tolerance for the providing cloud resources and services. On the other hand, we have customers that expect quality assurances, but also have their own constraints such as deadlines, cost, and implicit requirements from their applications including data placement and management that may influence the scheduling decision.

In both cases, we need to address the challenge of provisioning resources and also the challenge of scheduling services onto these resources. Although they can be done independently, it is obvious that interrelationship between them is needed in case of re-provisioning and dynamic adaptation to dynamic loads placed on the resources.

In both cases under-utilization prevents a resource from performing optimally, incurring idle time, whereas over-utilization causes a resource to degrade the node's performance.

3.5 Taxonomy Classification of Resource Scheduling Algorithms

Next, we present in Figure 8 a classification of resource scheduling algorithms that we found while reviewing a significant set of literature related to cloud computing. We focus in Figure 8 on a relevant subset while focussing on VM placement while considering QoS parameters to guide the scheduling task. An additional classification is based on the type of algorithm used for the scheduling task. Dependent on the locality and large scale of the scheduling task in many cases a deterministic approach is not suitable. Hence, different algorithm categories are listed in Figure 9.

4 LITERATURE REVIEW OF CLOUD RESOURCE SCHEDULING ALGORITHMS

In this section, we conduct an exemplary but extensive literature review of cloud scheduling to confirm the **taxonomy categories**. As part of this review, we present several tables to identify the categories from research and frameworks we reviewed and are related to cloud scheduling. We augmented each table with a first column that is highlighted and refers to the cloud scheduling taxonomy category we identified for this work.

To provide an additional guide we introduce several topical sections focusing and grouped the literature based on its main contribution to these groups. However, we avoided a double listing of the research in multiple groups as much as possible to keep the tables small.

As a result we organize this section by scheduling categories related to dynamic scheduling (Section 4.1), cloud metric-based scheduling with emphasize (Section 4.2) on energy (Section 4.2.1), network (Section 4.2.2), cost (Section 4.2.3), time (Section 4.2.4), reliability (Section 4.2.5), security (Section 4.2.6), and heuristics (Section 4.3).

As High-Performance Computing in the cloud is also a service offered by several providers, we also need to be aware of HPC in the cloud (Section 4.5) and scientific workflows (Section 4.6) that is going to become a field of interest for the scientific community. This is motivated by the fact that transition to cloud services takes place in academic and commercial settings and is explicitly an area of interest for NIST as discussed in the Big Data Reference Architecture Working Group while leveraging activities from the community including the past Grid community.

In this section, we also review papers with emphasis on scheduling in public clouds (Section 4.7), containers (Section 4.8), function as a service (Section 4.9) as well as distributed resource providers (Section 4.10) which can utilize a service mesh (Section 4.12).

4.1 Dynamic Scheduling

In literature, we find the distinction between static and dynamic cloud scheduling algorithms. In static scheduling, resources are scheduled once, while in dynamic scheduling updates are applied constantly to find better resource utilization during runtime.

The latter is often motivated by the need for scalability [104] across and within data centers or increased fault tolerance [142]. Association of other metrics into the dynamic scheduling approach is common while including power, network bandwidth and the integration of sophisticated service level agreements [142].

In many cases, not only the cloud user but also the cloud provider can benefit from dynamic scheduling [141].

We find that it can be beneficial to separate the scheduling task in multiple steps such as shown in [135]. Here, live migration for correlated VMs is optimizing on data, compute, and bandwidth conducted in several steps. Other cloud metrics such as price [144] are also common and will be addressed in Section 4.2.3. To address the scale problem many such algorithms use heuristics as showcased in Section 4.3.

Table 1 lists several efforts related to dynamic scheduling while focusing on virtual machine placement.

4.2 Cloud Metric-based Scheduling

Due to the complexity of cloud environments, many different metrics are used to guide the scheduling of virtual machines, containers, platforms, tasks, batch jobs, and workflows (see Figure 5). Next, we review examples of literature that integrates such metrics into their scheduling algorithm.

4.2.1 Energy Aware Scheduling. Energy consumption is a key issue for cloud providers due to the enormous cost associated with operating hyper-scale and large cloud data centers. By using server consolidation, optimizing operation on physical machines, energy consumption can be reduced in contrast to smaller-scale infrastructure. Also, while using dynamic voltage scaling of processors, energy consumption can be reduced as shown in [60, 161, 162] by slowing down the services.

Various scheduling methods such as to minimize the total makespan [51], developing dynamic meta-heuristics [53], fractal mathematics [73], and machine learning clustering and stochastic [59] have been utilized to optimize energy-aware scheduling. Multiple metrics must be included to correlate, for example CPU, RAM, and bandwidth [178].

These features, for example, could be utilized to dynamically adapt to peak loads [73] while making processors faster during such periods. Furthermore, migration [50] has naturally an impact on energy cost. Energy cost in multi-cloud and hybrid-cloud data centers in the clouds are discussed in [65, 82, 83, 128] while at the same time increasing the cloud provider broker's revenue.

Others create models to predict the energy consumption of each virtual machine [105]. This requires the ability to properly monitor the underlying server farms in a cloud data center as discussed in [149]. Integration of historical or previous program executions while recording their energy consumption can also be utilized [99]. Others focus on predicting future resource consumption needs [65].

A comparison of energy-aware scheduling algorithms in cloud computing is shown in Table 2 and 3.

4.2.2 Network Aware Scheduling. Clouds promote large-scale network traffic to, from, and within clouds. Thus network-aware scheduling must be considered for scheduling. This not only contains moving data in and out of the cloud data center but may also contain message exchanges between complex distributed applications that run in cloud data centers in a distributed fashion.

Minimizing the distance between data providers and data consumers while, for example,replicating data [38] can save a significant amount of traffic and has long been applied on the internet as one of its beneficial strategies. Service level agreements (SLA) [58] are playing an important role to achieve proper utilization as part of the scheduling effort. Treating the network as shared scarce resource [129] motivates the development of network-based scheduling algorithms. Also in network-aware scheduling, we find the distinction between static [56] and dynamic scheduling at runtime so we can deal with traffic bursts.

A variety of traditional scheduling metrics (see Figure 5) are often used to improve scheduling while considering network traffic. An example is demonstrated in [169] to optimize traffic in virtual clusters. Scheduling across multiple layers is especially of benefit for networking [54]. Scheduling of platforms such as Hadoop, offers advantages when networking is integrated [106]. Having access to lower-level infrastructure such as offered by OpenStack, presents opportunities to include Network Function Virtualization (NVF) [115].

Table 4 shows examples of network-aware scheduling algorithms in cloud computing.

4.2.3 Cost Aware Scheduling. Cost in clouds arises by using the data center facilities. These costs are passed along to the users.

Through shared use of the facilities and keeping under-utilization low, clouds can have an advantageous cost-performance ratio compared to on-premise compute and data centers. Costs for such centers include hardware operation, costs such as energy and equipment, as well as, operating costs, such as software licensing and update and personnel costs. Dependent on the hardware and software used, cloud providers offer a tiered cost model that allows users to assess the need for data, speed, and reliability as part of their cost analysis. Other options such as the selection of renewable energy use within the data center in case of energy conscious customers may also play a role.

Cost aware scheduling has been applied to virtual machines [172], tasks [171, 179], workflows [44, 45], as well as high-throughput [170] computing and use of data placement. Revenue maximization [174] has not only been applied to metrics such as latency [85], but is also useful via advanced Dynamic Voltage and Frequency Scaling (DVFS) [60, 168] due to reducing the high energy costs with little performance reduction. This also could be achieved through delayed execution [52] or relaxation of deadlines [176]. Other strategies include the introduction of penalties as part of SLA [166]. Typical resource utilization such as optimizing processor sharing [112] data placements [112], have been known to decrease cost. Also, dynamic adaptations at run-time allow reduction of cost [47]

To allow customers to decide the usage of various services including compute, data, function, and platform, most publish extensive cost schemes that can then be integrated into customers scheduling decisions.

Table 5 presents a comparison of cost-aware scheduling algorithms.

4.2.4 Time-based Scheduling. Cloud users have the desire to reduce the time it takes to execute their applications and fulfill deadlines [46]. Besides virtual machine and time-based scheduling, it is also important to integrate data-aware scheduling to reduce access time to the data [148]. Historical data [140] or proxies [76] for execution times help designing time-aware scheduling algorithms. We find algorithms that integrate deadline constraints [113], completion time [167] with fairness, low downtime to improve time for execution [81], and delay bounds [173].

Table 6 presents a comparison of time-aware scheduling algorithms.

4.2.5 Reliability Aware Scheduling. Cloud Users and providers need the guarantee of reliability. Thus, many cloud scheduling efforts address how to increase reliability. Strategies such as replication of data and compute services are common practice. This often comes at a price and increased cost may occur when reliability is of concern. The distributed nature of clouds makes it a formidable challenge to offer reliability. However at the same time, while providing (a) large scale data centers to offer cloud services with (b) highly specialized operating staff and (c) abilities to replicate and migrate workloads to other services, it increases reliability when compared to on-premise data centers. This is often due to the larger efficiency of the cloud data centers regarding the overall cost for its users.

Various studies have been conducted to analyze the effect of reliability on clouds.

This includes reliability assessment models [116], integration of communication and networks [101], increase of resource availability [110]. Trade-offs between different scheduling metrics such as energy and reliability have also been studied [136].

A comparison of reliability and scheduling is given in Table 7.

4.2.6 Security-based Scheduling. Security is a key feature cloud users and providers require for cloud infrastructure to be useful for many applications.

Virtual machine scheduling requires the need for isolation, that can be controlled by security policies [37]. Isolation can also apply to the incoming and outgoing data [102, 103]. Risks occurring by inspecting the connections among VMs [132] can be analyzed and integrated into scheduling strategies. To enable trust between components in the cloud key exchanges have been proposed [114].

Multiple possibly contradictory scheduling objectives need to be also considered in many scheduling frameworks.

An example included the cost it takes to provide security and integrate it adequately in security scheduling frameworks [102, 164, 175]. Furthermore, as many edge devices need to interface with cloud services due to their computational and data limitations, privacy-preserving solutions to interface between clouds and mobile and edge devices have been considered [55].

Security-based scheduling algorithms are presented (see Table 8).

4.3 Heuristic-based Scheduling

Heuristic methods help to design efficient algorithms in the case where deterministic methods can not be applied. We provide here a small sample of heuristics applied to clouds as found in the literature. This includes particle swarm optimization [126], multi-objective genetic algorithm-based [84, 121], colony optimization with swarm intelligence [118], bee colony [111], artificial neural networks [107], simulated annealing [143], game-theory [84], and Game theory by minimizing the Pareto dominance and makespan [134]. Other heuristics utilize classical models such as using the critical path in multi-phase scheduling algorithms [35]. Besides virtual machines we often also find workflows to be the scheduling unit in heuristics [57].

A comparison of heuristic-based scheduling algorithm is provided in Tables 9 and 10.

4.4 ML-based Scheduling

Recently, Machine Learning, and especially Deep Reinforcement Learning (DeepRL)-based approaches have become quite popular due to significant progress in the field. These methods can also be applied to automatically learn to schedule more efficiently in a cloud while it can also adapt to system changes. Various studies have been conducted to analyze the effect of ML-based scheduling approach in the Cloud computing environment. This includes deep reinforcement approach [61, 117], Q-learning model [177] and Q network model [165]. Markov's decision-based approach [49] is also studied to handle the uncertainty to provide an optimal decision at the time of scheduling. A comparison of ML-based scheduling algorithms is presented in Table 11.

4.5 HPC and Cloud Computing Scheduling

Next, we review scheduling classifications related to traditional High-Performance Computing (HPC). It is important to recognize, that HPC and its frameworks must not be excluded as part of cloud scheduling review due to its exposure for scientific application in industry and academia. More importantly, HPC is now also offered as one of the supported compute services in public cloud providers. When looking at the services offered and needed we distinguish HPC batch queuing in the cloud, cloud bursting of on-premise HPC tasks, container isolation, on-demand platforms, and bare-metal provisioning.

HPC Batch Queuing in the Cloud. Cloud providers offer specialized high-performance super-computing systems to customers with computation needs that can only be fulfilled by large scale specialized hardware. Grand challenge problems are often motivators for such hardware. In the industry, we, for example, find computational fluid dynamics, and modeling of biochemical processes as one of its drivers. Example offerings for HPC in the cloud are provided by AWS [40], Azure [122], Google [87], but also other less prominent clouds such as Penguin Computing HPC in the cloud [29], and SabalCore [31].

Cloud Bursting of On-Premise HPC tasks. The on-premise HPC systems are often over-utilized and thus the situation of resource starvation motivates the provider to gain additional resources in the cloud. For this reason, many batch queuing system allows the integration of cloud resources in such a fashion that task and workflows may be executed in the cloud through the integration of commercial or on-premise cloud resources. In this case, the term cloud bursting is used [23, 25]. Examples for the integration in prominent HPC scheduling includes Slurm [18], Univa Grid Engine [19], PBSpro [127], LSF [100], Moab [36].

Container Isolation. Due to the usage of queuing systems it is also possible to provide in part an improved container framework while executing containerized tasks as part of the queuing system. An example would be to utilize all cores in a compute-server that is allocated with a queuing system processor. This feature can be integrated into many queuing systems while using Singularity [33].

On Demand Platforms. Resource starvation in academic clouds and supercomputing centers motivate also the ability to run platforms that would typically run also in the cloud but provide an alternative if run locally in the existing HPC centers. A good example is Hadoop that can be run through myhadoop [108] in HPC centers [32].

Bare Metal Provisioning. In other cases it may be better to provide bare-metal provisioning capabilities in case existing platform or cloud abstraction may not be sufficient. Academic efforts such as FutureGrid [79] now followed by Comet [133] and Chameleon Cloud [24] are good examples for it. Commercial efforts in this regard include OpenStack Ironic [28], IBM [27], AWS [20] and Rackspace [30].

Table 12 presents a selected comparison of the different batch resource management systems.

4.6 Workflow Scheduling Frameworks

In the previous sections, we already pointed out several workflow related scheduling algorithms while using specific metrics to conduct the scheduling. Also, we can integrate virtual machines, containers, and tasks. The main intent of the cloud workflow is to automate repeated tasks in a reliable way.

It is important to recognize that previous academic task-based workflow engines can easily be modified to be integrated into the scheduling needs of clouds. This includes traditionally, workflow schedulers distinguished by DAG and non-DAG scheduling strategies [68, 69, 138, 154, 156–158]. This includes the ability to integrate virtual machines, containers, and functions into task abstractions as demonstrated for example in [64, 152] recently. A very recent effort includes the development of cloudmesh cloud that provides abstractions to interface with VMs, FaaS, PaaS, and containers [152]. This effort also includes needed information such as the cost of virtual machines that can be leveraged into the use and development of cost-based scheduling algorithms.

Furthermore, It is often an overlooked fact that as part of the scheduling within existing HPC batch queuing systems workflow abstractions exist that could be utilized to integrate cloud scheduling tasks. Such queuing systems include [36, 100, 127, 147]. These systems can naturally be utilized to facilitate cloud bursting, as well as the scheduling of IaaS, PaaS, and FaaS, as is successfully demonstrated on Comet [133] at Sandiego Super Computing Center.

Additionally, we see another important aspect, where cloud providers host their workflow services in the cloud. This includes efforts such as Nintex [124], Amazon Simple Workflow (ASW), Hadoop streaming [42] for map-reduce workflows are used. For example, ASW intends to provide support to build, run, and scale background jobs that have parallel or sequential steps. The main intent of Spark streaming [43] is to provide scalable, high-throughput and fault-tolerant stream processing of live data streams. Argo [109], a container-native workflow engine is used for orchestrating parallel jobs on Kubernetes. This will become a rich area of research as workflows need to utilize resources and efficient workflow schedulers to utilize cloud resources is an important goal.

Other trends are discussed in [77, 137**?**].

4.7 Scheduling in Public Cloud Providers

Next, we compare scheduling methods and needs offered in public cloud service providers. This includes AWS, Azure, Google, Rackspace, but also academic clouds such as FutureGrid and FutureSystems Comet, Jetstream, and Chameleon Cloud.

It is important to recognize that today public cloud providers offer not only virtual machines to the users, but a large variety of compute, data, and analytical services. Some of them may even use bare metal while others are having heightened security demands, to, for example, fulfill heath care or government isolation needs as part of the infrastructure. All these issues naturally influence the scheduling efforts which need to be addressed by the provider. In many cases, we do not find sufficient information on how such scheduling is conducted due to security and company secrets.

However, we find metrics that users can utilize to formulate their strategies as we have introduced in the previous section if such metrics are communicated to the users. This typically includes cost and allows to leverage for example virtual machine with reliability constraints such as AWS spot pricing compared to regular pricing [3]. Cost also motivates users to suspend the usage of VMs instead of running them without concern. This has happened to the authors of this paper, where in a class a student, refused to shut down experimental virtual machines and within two weeks consumed thousands of compute hours on an academic cloud, while the actual calculation was irrelevant.

One of the schedulers provided by public clouds are job and instance schedulers that promote start and stop times for the resources used [5, 10, 21, 22]. Such schedulers can integrate functions, data and compute instances. More sophisticated schedulers can switch workloads between cloud data centers [2].

In [5] cloud load-balancer, round-robin and least connections-based algorithms are commonly used so that workload could be distributed equally on all resources. As one of the original tasks of clouds was hosting of Web services under traffic load, public clouds include strategies that scale up and down the services-based on such loads and allocate resources dynamically.

Other providers have focused on making use of multi-cloud virtual machine placements while offering optimization strategies for workflows [4] including a detailed analysis of cost metrics [26]

Other efforts such as [79] have early on uniquely focused on scheduling bare-metal resources between the use of HPC and clouds while running HPC queuing systems on the same resources as cloud resources. Dynamic provisioning allowed resources to be provisioned to the one or the other by demand. In [133] the re-provisioning is even done with the help of a traditional batch queuing system enabling a sophisticated scheduling approach

Table 13 depicts examples as used in public cloud providers.

4.8 Scheduling in Container Frameworks

Container schedulers provide mechanisms to fine-tune the selection processes of containers onto distributed resources [7, 67]. Typically a default scheduling policy is provided. Policies might place new services on hosts with the fewest currently active services.

Based on the Y-diagram we need to distinguish two different services. First, scheduling on the same server and second scheduling on several servers that are treated typically as one abstract cloud resource

For the first scheduling task, we need to consider data management to efficiently utilize the memory hierarchy, but also, for example, execution deadlines or privacy concerns to organize the computation tasks as required. In the distributed case we also need to integrate communication-related issues. We focus next on the distributed frameworks in more detail we focus on Docker Swarm, Kubernetes, Singularity, and Mesos.

Docker Swarm. Docker Swarm is a clustering and scheduling tool for Docker containers [9] across compute servers. In a docker swarm, we distinguish manager nodes and worker nodes. The manager uses load balancing to place the containers onto the worker nodes. Once a task is placed on a server it is executed there. Docker swarm uses a single scheduling strategy [8].

Kubernetes. Kubernetes is an open-source orchestrator developed by Google for automating container management and deployment [11]. The basic deployable object is a Pod that consists of one or more containers running in a shared context. An API is used to declare policies and scalability constraints. The Kubernetes scheduler is topology-aware and workload aware which can be integrated into the policy constraints to expose availability, performance, and capacity. Auto-scaling, load balancing, and secrets management are also provided by Kubernetes.

Singularity. Singularity can be using a variety of container frameworks as backend. It allows the use of containers without being a superuser. Due to this, singularity is a popular choice for running containers on traditional HPC systems [33]. Due to this scheduling can be supported directly by the under-laying queuing system.

Mesos. Mesos [13, 98] provides an API for resource management and scheduling in data centers. Mesos abstracts CPU, memory, storage, and other compute resources. It integrates fault-tolerance. Mesos provides a thin resource sharing layer that helps to furnish fine-grained sharing by providing common interfaces among different cluster frameworks. Its goal is improved utilization, respond quickly to workload changes, by maintaining a system's capability in terms of scalability and robustness.

Community Efforts. Many community efforts to improve container scheduling are conducted. This includes, for example, the use of genetic algorithm [91], container, and host selection policies for cloud deployment models [94] with SLA's, the characterization of applications [119] scheduling of virtual clusters [74], and migration [78], and systems integrating multiple schedulers such as Nomad which offer service scheduler, batch scheduler, and a systems scheduler while focusing on the support of long-running jobs [14].

Table 14 shows the comparison of existing work related to container scheduling.

4.9 Function Scheduling Algorithms

To further improve scheduling on cloud resources, the concept of Function as a Services was introduced. It allows the invocation of small functions with limited resource constraints on servers [151]. For example, a minimum execution time per request is five minutes provided by AWS lambda and Azure functions [17]. It supports managed user-defined functions on highly available infrastructure in a unified fashion [123]. This also allows the scheduling of workflows comprised out of functions [39]. In [80] we discuss the status of serverless computing and function as a service in Industry and research. Serverless computing is considered the backend for running FaaS at runtime. System allocation and other resource management activities are provided by the backend. Thus the users have not to worry about activities conducted by the server. Hence, the name serverless computing. Through the use of FaaS and serverless computing, cost can be reduced by more efficiently scheduling smaller tasks on resources.

Several FaaS frameworks exist that can be used on public clouds but also self-hosted clouds or network of workstations.

Scheduling in FaaS is provided by triggers. Such triggers offer a publish-subscribe model in which events are conducted, once the trigger is fired. This includes triggers for time, data, and executions. Time-based scheduling is supported by cron. These frameworks are supported by all major public clouds including AWS lambda [6], Google cloud functions [88], Azure Function [48]. This can also be combined with simple workflow scheduling as introduced in pipelines as part of, for example as used in Jenkins [75].

Other open-source frameworks such as Apache OpenWhisk [16] allow users to install FaaS services on their own infrastructure.

An important aspect of scheduling in FaaS is that the deployment of the function itself does cost time. At times the start-up time for the function is substantial. In such cases, workflows can be leveraged to assure that before the function is executed a cache is set up in which the function is deployed [34]. Thus it is important to assure that deployment times are minimized and when multiple function calls are conducted the deployment is done only once.

In production Clouds such as AWS we also use the term cold, warm, and hot for classifying the preparation of the software to execute a function repeatedly. In these environments a function becomes cold after a particular time, meaning that it needs to restart its functional requirements before the actual function can be executed.

4.10 Scheduling Among Distributed Resources and Providers

Users may have the desire to not only use services on one cloud but multiple clouds. This is motivated largely by avoiding vendor lock-in, unique service offerings, or combining services from different vendors.

Such scheduling efforts can be as simple as switching the cloud provider such as promoted in Cloudmesh [160]. Other efforts such as Eagle, provide a hybrid data center scheduler for data-parallel programs [70]; Hopper [130], a job scheduler that trades off existing and speculated job scheduling decisions; Tetris [90], a cluster scheduler that aims to match multi-resource task requirements with resource availability; Fawkes [86] a multi-cluster systems for map-reduce; Omega [131] with optimistic concurrency control; OurGrid [41, 62] for worldwide computing platform with isolated environments; Sparrow [125] and fine-grained task scheduling scheduler.

We can also find more prominent schedulers such a contains Apache's Hadoop YARN [150] which acts as a resource management system to for example schedule Hadoop distributed processing framework considering QoS, scalability, higher efficiency, and fair resource usage.

We contrast different resource management systems, used for maintaining resources in distributed environments such as Clouds (see Table 15).

4.11 Data-based Scheduling

In cloud applications, it is not only important to integrate compute services into the scheduling decision, but also data. Here we distinguish typically two models. Bring the data to compute services or bring the compute services to the data. A comprehensive scheduling algorithm must be deciding which of these approaches is most amenable to the problem. Metrics such as bandwidth, latency, but also the size of the data are to be integrated into such decision processes.

Just as with compute services other metrics can be integrated into the scheduling algorithm to optimize the strategy. Most recently the introduction of Function as a Service is providing the potential for further improving the utilization of data services while concurrently scheduling many functions to operate on data in parallel, streaming, and high-performance [146].

4.12 Service Mashups

To support scheduling across clouds and services, service mashups can be used. This includes long-standing efforts such as Cloudmesh [160], which targets the creation of reproducible environments to easily manage virtual machines, bare-metal provisioned operating systems, platform deployments and more recently data services in a multi-cloud environment. It is a goal to integrate custom schedulers in such service mashups. Another example is Terraform [95] which focuses on reproducible environments.

4.13 Simulators

When developing scheduling frameworks it is important to evaluate them before use. For this reason, it is useful to simulate cloud environments before deploying them in real-time. As many researchers do not have access to large scale clouds this is often the only application we find for many papers and thus they stay more theoretical in nature. Efforts such as FutureGrid [79] that provided the earliest multi-cloud environment, ChameleonCloud [24], and CloudLab [63] did and do provide environments in which clouds can be deployed and algorithms can be tested.

For simulation-based efforts we find that the following can be of help TPC-H [145], BigDataBench [163], Google Cluster traces [89], DCSim [66] and CloudSim [96].

4.14 Selection of a Scheduling Framework

As we can see from the discussion in this paper, the number of parameters and considerations to select a scheduling algorithm fine-tuned for the particular cloud is a significant challenge. Not only do we need to identify the parameters that will lead to a better schedule, but we also need to understand the scale and scope of compute, data, network and also other inputs that may not be available to users but only available to cloud providers. Hence as much information as possible must be exposed between the different layers so that smart scheduling algorithms can be developed and address quality of service assertions in time and space.

One thing we observed is that some papers do present a fair number of simulations, but a precise formula for communicating the complexity of the algorithm may not be provided. We believe in a future research activity such a complexity analysis that needs to be conducted while including parameters use to determine the could needs. Furthermore, an automated simulation framework that predicts the performance of the scheduler based on its input can be derived. The need for this is motivated by the potential time constraints we have. Certainly, we do not have the luxury to wait for a scheduling decision if it takes longer to derive it than the actual calculation or data analysis takes. Furthermore, we envision a REST service based on the NIST Big Data Reference Architecture principles that would allow us to stage an analytics service in which multiple competing algorithms can be evaluated to make the selection process of the scheduler more transparent. Not only would this framework be able to present the complexity but also instantiate or look up a benchmark for the particular scheduling problem formulated.

5 CONCLUSION AND LESSONS LEARNED

In this section, we summarize some of the lessons we learned from our activities.

More than VMs. Due to the shift and enhancement of clouds from VM to containers and FaaS, we must consider also new scheduling strategies as motivated by new cloud compute service offerings utilizing them. This offers several opportunities for research activities.

Energy. Energy costs for data centers are enormous and this plays a significant role for providers, but also for users to which energy costs are passed along. Not only good scheduling algorithms are needed, but the design of the data center close to cheap energy is an important issue.

Y-Diagram. The Y-diagram promotes scheduling across scale and models. This allows creating a hierarchy of interfacing scheduling approaches for integrated and layered scheduling between resources at different scales.

Multi-Metric and Multi-Objectivity. Scheduling algorithms must use multiple metrics and multiple objectives to provide effective scheduling decisions. In many cases, contradictory scheduling goals such as reliability vs cost are to be considered.

Policy driven. Due to multi-metric and multi-objective scheduling goals modern schedulers will expose them through policies to users and providers.

Iterative Optimization in Layers. Due to the complexity of the scheduling efforts motivated by out Y-diagram, a layered scheduling approach seems appropriate.

Security and Privacy. We need to deal more stringently with security and privacy as part of scheduling needs.

Fault tolerance and Risk Analysis. As part of the policy-driven service level agreements with the cloud providers schedule must include the ability to integrate fault tolerance while leveraging risk analysis.

Traditional Scheduling. Naturally we need to deal with scheduling with traditional issues such as load balancing, congestion, and service spikes. However, they are amplified by formidable resource management issues in hyper-scale data centers.

5.1 Future Directions

In this section, we summarize some of the future directions we need to address.

Integration for data. While we focused mostly on compute resource scheduling an additional study is needed to more explicitly address aspects that integrate Big Data and of it while addressing scheduling aspects.

Analytics Services. While FaaS provides the ability to schedule resource-restricted functions the next level of schedulers will address Analytics as a Service (AaaS) where more resource bound functions are cast and exposed to cloud users as analytical calculations.

Edge Computing. Due to the increase of computational power of edge devices scheduling algorithms must include the power available on these devices instead of sending all the requests to a cloud. Billions of cellphones today already conduct a significant amount of computation thus scheduling must balance between activities that can take place on the edge or needs to be conducted in the cloud.

Scheduling Challenges Arising form use of Containers. By using virtualization technologies such as virtual machines the cloud provides the illusion of hardware resources but introduces a cost to also virtualize the operating system.

Containers, however, use virtualization within the operating system level. Multiple containers run on the top of the operating system kernel. Hence, a container is a lightweight approach to implement the virtualization technology leveraging the underlying OS. The memory consumption by containers is less than the resources required to boot a virtual machine with its virtualized OS. As an example, we point out Kubernetes [12] where containers within a pod [11] share an IP address and find each other via localhost. Communication among them is done by inter-process communications, such as SystemV semaphores or POSIX shared memory. Containers in different pods cannot communicate directly as they have distinct IP addresses. Kubernetes commonly uses flannel to accomplish container networking. Containers are joined in a virtual network. Kubernetes provides mechanisms to utilize several pre-existing scheduling algorithms but also provides the ability to replace them with customized approaches.

The challenge here is to assure that containers between users do not create security or violate privacy issues. Also, the access to potentially elevated system privileges may cause other Therefore systems such as Singularity offer users an isolated use of containers within traditional HPC queuing systems to mitigate that issue. Still once on such a system, we still have to be aware of elevated privileges, and containers may only be offered in limited form to its users. Once this has been clarified, also for containers the typical quality assertions during its use apply just as for virtual machines. Such challenges must be integrated into a scheduling strategy when adding containerized cloud resources.

Challenges in Function as a Service. The *Function as a Service* model allows developers to build and execute their requirements. Functions are uploaded to FaaS infrastructure and services and triggered by events. Due to resource limitations, they provide significant information for the underlying layers to provide more efficient resources. However, monitoring tools and fault tolerance have to be carefully integrated to avoid FaaS failures based on resource starvation or an excess of resources used. Also, more intense functions may require splitting them up in smaller so they can be fulfilled resource constraints of the FaaS framework. Naturally, while splitting up a larger model into smaller functions increases the overhead. Such limitations must be understood by the developer in order not to create a function that is impossible to schedule.

In this paper, we have surveyed important classes related to scheduling in cloud computing. After introducing the needed terminology, we presented a comprehensive taxonomy for cloud scheduling including a Y-cloud taxonomy. A layered and phased scheduling model is presented that differentiates the concerns between infrastructure, platform, servers and function as a service model. A comprehensive investigation has been conducted to verify that the taxonomy is valid and that existing scheduling techniques motivate its validity.

POSTFACE

We realize that although we have analyzed a large number of papers, there are more papers in that area of cloud scheduling available. The papers cited here are used as examples to showcase some important features related to cloud scheduling. We appreciate it if you inform us about other efforts as we intend to collect them for further updates to this paper. We like to especially pay attention to papers that may motivate us to refine the taxonomy. Please send us your reference in BibTeXformat while pointing out how your paper enhances the taxonomy. The contribution can be sent either to `laszewski@gmail.com`, or via a GitHub pull request at https://github.com/cyberaide/paper-cloud-scheduling/blob/master/vonlaszewski.bib and https://github.com/cyberaide/paper-cloud-scheduling/blob/master/cloud-scheduling.bib

ACKNOWLEDGMENT

We like to thank the reviewers for helping us improve this paper with their thoughtful comments.

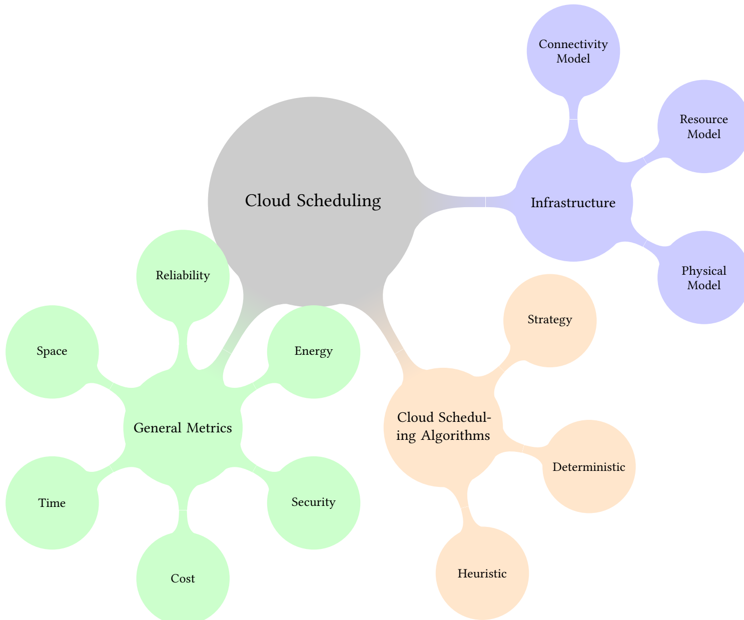

Figure 3: Cloud scheduling

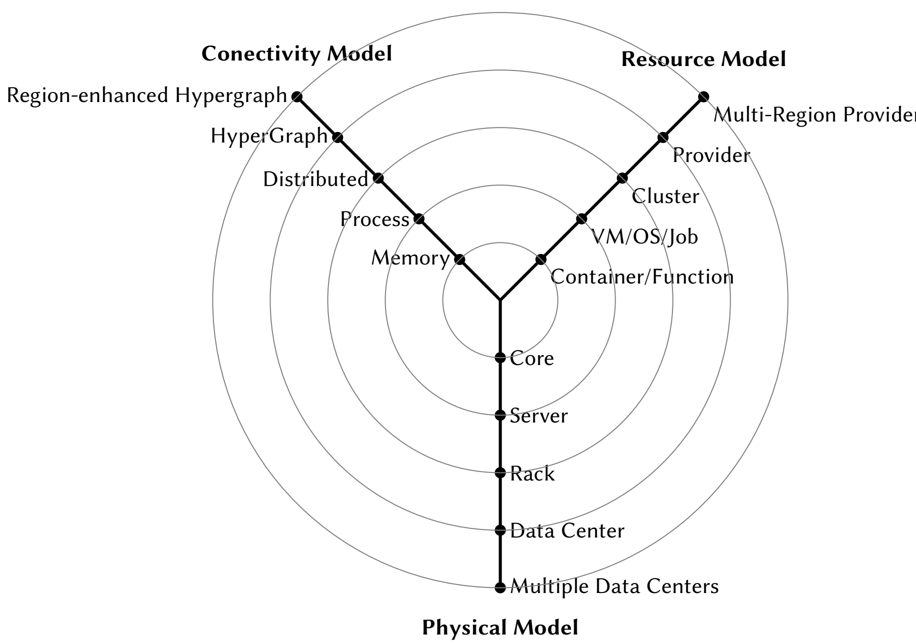

Figure 4: Resource Provider Focused Y-Cloud Taxonomy [151]

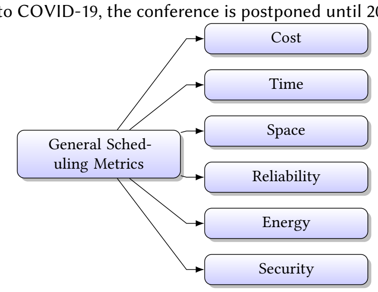

Figure 5: Traditional Compute Scheduling Metrics

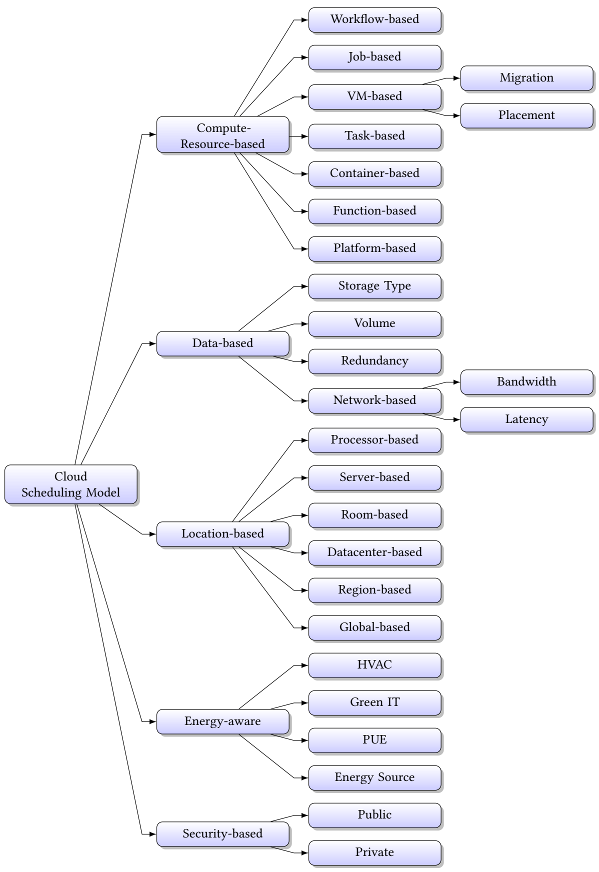

Figure 6: Cloud Scheduling Models

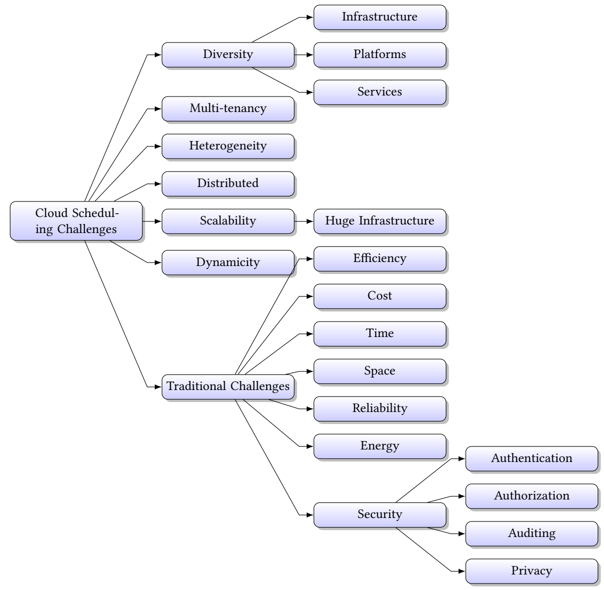

Figure 7: Scheduling challenges applied to all levels

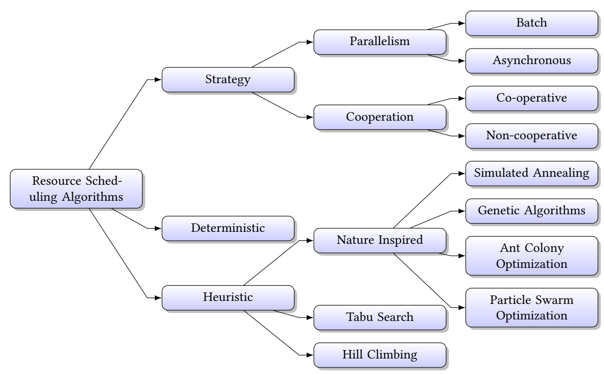

Figure 8: A subset of Resource Scheduling Algorithms

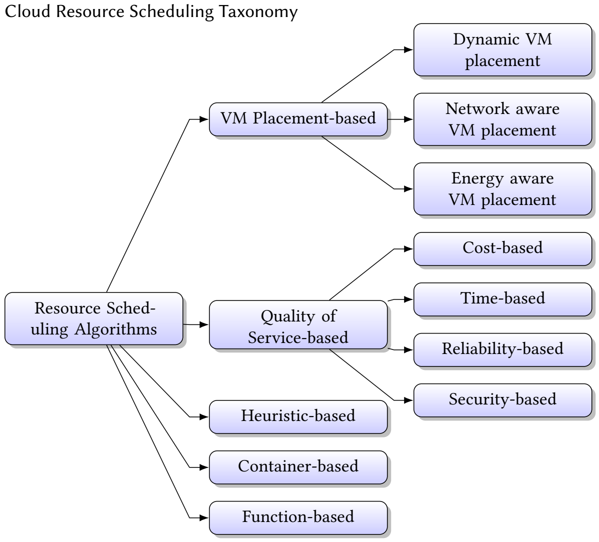

Figure 9: Classification of Resource Scheduling Algorithms

REFERENCES

[1] [n.d.]. NIST Big Data Interoperability Framework: Volume 6, Reference Architecture.
[2] 2014. Microsoft Azure. https://azure.microsoft.com/en-in/services/scheduler/
[3] 2015. Amazon EC2. https://aws.amazon.com/ec2/
[4] 2016. Cloud Sigma. https://www.cloudsigma.com/features/
[5] 2016. Rackspace. https://www.rackspace.com/en-in/why-rackspace
[6] 2018. AWS Lambda. https://aws.amazon.com/lambda/
[7] 2018. Containers. https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-scheduling-and-orchestration
[8] 2018. Docker Swarm. https://semaphoreci.com/community/tutorials/scheduling-services-on-a-docker-swarm-mode-cluster
[9] 2018. Docker Swarm Engine. https://docs.docker.com/engine/swarm/manage-nodes
[10] 2018. Google App Engine. https://cloud.google.com/appengine/docs/
[11] 2018. Kubernates. https://github.com/kubernetes/kubernetes
[12] 2018. Kubernates. https://kubernetes.io/docs/concepts/
[13] 2018. Mesos. http://mesos.apache.org/getting-started/
[14] 2018. Nomad. https://www.nomadproject.io/docs/internals/scheduling.html
[15] 2018. OpenPBS: The Portable Batch System Software. https://www.pbspro.org/
[16] 2018. OpenWhisk. https://openwhisk.apache.org/
[17] 2018. Serverless Computing: Architecture, Challenges, and Prospects. https://www.apriorit.com/dev-blog/551-serverless-computing
[18] 2018. Slurm Workload Manager. https://slurm.schedmd.com/
[19] 2018. Univa. http://www.univa.com/
[20] 2019. AWS. https://aws.amazon.com/about-aws/whats-new/2019/02/introducing-five-new-amazon-ec2-bare-metal-instances/
[21] 2019. AWS Instance Scheduler. https://aws.amazon.com/solutions/instance-scheduler/
[22] 2019. Azure Scheduler. https://azure.microsoft.com/en-in/services/scheduler/
[23] 2019. Bursting HPC. https://insidehpc.com/2018/04/universities-step-cloud-bursting/
[24] 2019. Chameleoncloud. https://www.chameleoncloud.org/
[25] 2019. Cloud Bursting. https://www.nextplatform.com/2018/02/26/adaptive-approach-bursting-hpc-cloud/
[26] 2019. Cloud metrics. https://www.rightscale.com/about-cloud-management/cloud-cost-optimization/cloud-pricing-comparison
[27] 2019. IBM Bare Metal. https://www.ibm.com/cloud/bare-metal-servers
[28] 2019. Openstack Ironic. https://docs.openstack.org/ironic/latest/
[29] 2019. POD HPC Cloud. https://www.penguincomputing.com/pod-hpc-cloud
[30] 2019. Rackspace. https://www.rackspace.com/en-us/cloud/servers/onmetal
[31] 2019. Sabalcore. http://www.sabalcore.com/
[32] 2019. San Diego Supercomputer Center. https://www.sdsc.edu/support/user_guides/tutorials/hadoop.html
[33] 2019. Singularity. https://singularity.lbl.gov/docs-hpc
[34] Cristina L. Abad, Edwin F. Boza, and Erwin van Eyk. 2018. Package-Aware Scheduling of FaaS Functions. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering* (Berlin, Germany) *(ICPE '18)*. ACM, New York, NY, USA, 101–106. https://doi.org/10.1145/3185768.3186294
[35] Saeid Abrishami, Mahmoud Naghibzadeh, and Dick HJ Epema. 2013. Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. *Future Generation Computer Systems* 29, 1 (2013), 158–169.
[36] Adaptive Computing. 2019. Moab. Web Page. http://www.adaptivecomputing.com/products/
[37] Zaina Afoulki, Aline Bousquet, and Jonathan Rouzaud-Cornabas. 2011. *A security-aware scheduler for virtual machines on iaas clouds*. Raport de Recherche RR-2011-08. University of D'Orleans, France.
[38] Akamai [n.d.]. Akamai. Web Page. https://www.akamai.com/
[39] Omar Alqaryouti and Nur Siyam. 2018. Serverless Computing and Scheduling Tasks on Cloud: A Review. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)* 40, 1 (2018), 235–247.
[40] Amazon. [n.d.]. Job Scheduling. https://docs.aws.amazon.com/batch/latest/userguide/job_scheduling.html
[41] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro, and Paulo Roisenberg. 2003. OurGrid: An approach to easily assemble grids with equitable resource sharing. In *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 61–86.
[42] Apache. [n.d.]. Hadoop Streaming. https://hadoop.apache.org/docs/r1.2.1/streaming.html
[43] Apache. [n.d.]. Spark Streaming. https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html
[44] Vahid Arabnejad and Kris Bubendorfer. 2015. Cost effective and deadline constrained scientific workflow scheduling for commercial clouds. In *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*. IEEE, 106–113. https://doi.org/10.1109/NCA.2015.33
[45] Vahid Arabnejad, Kris Bubendorfer, and Bryan Ng. 2016. A Budget-Aware Algorithm for Scheduling Scientific Workflows in Cloud. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*. IEEE, 1188–1195.
[46] Vahid Arabnejad, Kris Bubendorfer, and Bryan Ng. 2017. Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. *Future Generation Computer Systems* 75 (Jan. 2017). https://doi.org/10.1016/j.future.2017.01.002
[47] Ismail Ari and Nitel Muhtaroglu. 2013. Design and implementation of a cloud computing service for finite element analysis. *Advances in Engineering Software* 60 (2013), 122–135.
[48] Azure 2018. Azure Functions. https://azure.microsoft.com/en-in/services/functions/
[49] Enda Barrett, Enda Howley, and Jim Duggan. 2013. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience* 25, 12 (2013), 1656–1674.
[50] Anton Beloglazov and Rajkumar Buyya. 2010. Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*. IEEE Computer Society, 826–831.
[51] Nik Bessis, Stelios Sotiriadis, Florin Pop, and Valentin Cristea. 2013. Using a novel message-exchanging optimization (MEO) model to reduce energy consumption in distributed systems. *Simulation Modelling Practice and Theory* 39 (2013), 104–120.
[52] Jing Bi, Haitao Yuan, Wei Tan, and Bo Hu Li. 2016. TRS: Temporal request scheduling with bounded delay assurance in a green cloud data center. *Information Sciences* 360 (2016), 57–72.
[53] Jing Bi, Haitao Yuan, Wei Tan, MengChu Zhou, Yushun Fan, Jia Zhang, and Jianqiang Li. 2017. Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center. *IEEE Transactions on Automation Science and Engineering* 14, 2 (2017), 1172–1184.
[54] Jing Bi, Haitao Yuan, Ming Tie, and Wei Tan. 2015. SLA-based optimisation of virtualised resource for multi-tier web applications in cloud data centres. *Enterprise Information Systems* 9, 7 (2015), 743–767.
[55] Igor Bilogrevic, Murtuza Jadliwala, Praveen Kumar, Sudeep Singh Walia, Jean-Pierre Hubaux, Imad Aad, and Valtteri Niemi. 2011. Meetings through the cloud: privacy-preserving scheduling on mobile devices. *Journal of Systems and Software* 84, 11 (2011), 1910–1927.
[56] Ofer Biran, Antonio Corradi, Mario Fanelli, Luca Foschini, Alexander Nus, Danny Raz, and Ezra Silvera. 2012. A stable network-aware vm placement for cloud systems. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*. IEEE, 498–506.
[57] Khadija Bousselmi, Zaki Brahmi, and Mohamed Mohsen Gammoudi. 2016. QoS-aware scheduling of Workflows in Cloud Computing environments. In *Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on*. IEEE, 737–745.
[58] David Breitgand and Amir Epstein. 2012. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2861–2865.
[59] Dinh-Mao Bui, YongIk Yoon, Eui-Nam Huh, SungIk Jun, and Sungyoung Lee. 2017. Energy efficiency for cloud computing system based on predictive optimization. *J. Parallel and Distrib. Comput.* 102 (2017), 103–114.
[60] Rodrigo N Calheiros and Rajkumar Buyya. 2014. 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*. IEEE, 342–349.
[61] Mingxi Cheng, Ji Li, and Shahin Nazarian. 2018. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*. IEEE Press, 129–134.
[62] Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade, Lauro B Costa, Alisson Andrade, Reynaldo Novaes, and Miranda Mowbray. 2006. Labs of the world, unite!!! *Journal of Grid Computing* 4, 3 (2006), 225–246.
[63] cloudlabe 2019. Cloudlab. Web Page. https://cloudlab.us/
[64] CyVerse. 2019. Pegasus: How to Build and Deploy Containerized Workflows. Presentations. https://pegasus.isi.edu/tag/docker/
[65] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. 2015. Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE Transactions on Network and Service Management* 12, 3 (2015), 377–391.
[66] dcsim 2015. A Data Centre Simulation Tool for Evaluating Dynamic Virtualized Resource Management. GitHub. https://github.com/digs-

uwo/dcsim

[67] Marcos Dias de Assuncao, Alexandre da Silva Veith, and Rajkumar Buyya. 2018. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. *Journal of Network and Computer Applications* 103 (2018), 1–17.

[68] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. 2004. Pegasus: Mapping scientific workflows onto the grid. In *Grid Computing*. Springer, 11–20.

[69] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. 2005. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming* 13, 3 (2005), 219–237.

[70] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. 2016. Job-aware scheduling in eagle: Divide and stick to your probes. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*. ACM, 497–509. https://doi.org/10.1145/2987550.2987563

[71] Javier Diaz, Gregor von Laszewski, Fugang Wang, Andrew J Younge, and Geoffrey C. Fox. 2011. FutureGrid Image Repository: A Generic Catalog and Storage System for Heterogeneous Virtual Machine Images. In *Third IEEE International Conference on Coud Computing Technology and Science (CloudCom2011)*. Athens, Greece, 560–564. https://doi.org/10.1109/CloudCom.2011.85

[72] Youwei Ding, Xiaolin Qin, Liang Liu, and Taochun Wang. 2015. Energy efficient scheduling of virtual machines in cloud with deadline constraint. *Future Generation Computer Systems* 50 (2015), 62–74.

[73] Hancong Duan, Chao Chen, Geyong Min, and Yu Wu. 2016. Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. *Future Generation Computer Systems* 74 (March 2016). https://doi.org/10.1016/j.future.2016.02.016

[74] P. Dziurzanski and L. S. Indrusiak. 2018. Value-Based Allocation of Docker Containers. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. 358–362. https://doi.org/10.1109/PDP2018.2018.00064

[75] Alex Ellis. 2018. Jenkins cron job to run an OpenFaaS function every minute. GitHub Gist. https://gist.github.com/alexellis/dfa1b8790ac3d26614e342746c64cbc8

[76] D Cenk Erdil. 2013. Autonomic cloud resource sharing for intercloud federations. *Future Generation Computer Systems* 29, 7 (2013), 1700–1708.

[77] F. Fakhfakh, H. H. Kacem, and A. H. Kacem. 2014. Workflow Scheduling in Cloud Computing: A Survey. In *2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*. 372–378. https://doi.org/10.1109/EDOCW.2014.61

[78] Flocker 2018. Flocker. https://flocker.readthedocs.io/en/latest/

[79] Geoffrey Fox, Gregor von Laszewski, Javier Diaz-Montes, Kate Keahey, Jose Fortes, Renato Figueiredo, Shava Smallen, Warren Smith, and Andrew Grimshaw. 2012. *FutureGrid - a reconfigurable testbed for Cloud, HPC and Grid Computing*. CRC Press, Chapter 23, 603–635. https://doi.org/10.1201/9781351104005-23

[80] Geoffrey C Fox, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2017. Status of serverless computing and function-as-a-service (faas) in industry and research. *arXiv preprint arXiv:1708.08028* (2017).

[81] Marc Eduard Frîncu. 2014. Scheduling highly available applications on cloud environments. *Future Generation Computer Systems* 32 (2014), 138–153.

[82] Keke Gai, Meikang Qiu, Hui Zhao, Lixin Tao, and Ziliang Zong. 2016. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *Journal of Network and Computer Applications* 59 (2016), 46–54.

[83] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. 2011. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. *J. Parallel and Distrib. Comput.* 71, 6 (2011), 732–749.

[84] Jakub Gąsior and Franciszek Seredyński. 2016. Metaheuristic Approaches to Multiobjective Job Scheduling in Cloud Computing Systems. In *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*. IEEE, 222–229.

[85] Mohammad Hossein Ghahramani, MengChu Zhou, and Chi Tin Hon. 2017. Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services. *IEEE/CAA Journal of Automatica Sinica* 4, 1 (2017), 6–18.

[86] Bogdan Ghit, Nezih Yigitbasi, Alexandru Iosup, and Dick Epema. 2014. Balanced resource allocations across multiple dynamic MapReduce clusters. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 42. ACM, 329–341. https://doi.org/10.1145/2591971.2591998

[87] Google. [n.d.]. HPC Perfomance COmputing. Web Page. https://cloud.google.com/solutions/hpc/

[88] Google. 2018. Google Cloud Functions. https://cloud.google.com/functions/

[89] Google. 2019. Google Cluster Traces. GitHub. https://github.com/google/cluster-data

[90] Robert Grandl, Ganesh Ananthanarayanan, Srikanth Kandula, Sriram Rao, and Aditya Akella. 2015. Multi-resource packing for cluster schedulers. *ACM SIGCOMM Computer Communication Review* 44, 4 (2015), 455–466.

[91] Carlos Guerrero, Isaac Lera, and Carlos Juiz. 2018. Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. *Journal of Grid Computing* 16, 1 (2018), 113–135.

[92] Carlos Guerrero, Isaac Lera, and Carlos Juiz. 2018. Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications. *The Journal of Supercomputing* (2018), 1–28.

[93] J Octavio Gutierrez-Garcia and Kwang Mong Sim. 2013. A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling. *Future Generation Computer Systems* 29, 7 (2013), 1682–1699.

[94] Walid A Hanafy, Amr E Mohamed, and Sameh A Salem. 2017. Novel selection policies for container-based cloud deployment models. In *Computer Engineering Conference (ICENCO), 2017 13th International*. IEEE, 237–242.

[95] Hashicorp. [n.d.]. Terraform. Web Page. https://www.terraform.io/

[96] he Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. 2019. CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. Web Page. http://www.cloudbus.org/cloudsim/

[97] Robert L. Henderson. 1995. Job Scheduling Under the Portable Batch System. In *Proceedings of the 1995 Workshop on Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson and Larry Rudolph (Eds.). Springer Berlin Heidelberg, Santa Barbara, CA, USA, 279–294.

[98] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In *NSDI*, Vol. 11. 22–22.

[99] J. Hu, J. Gu, G. Sun, and T. Zhao. 2010. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. In *2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*. 89–96. https://doi.org/10.1109/PAAP.2010.65

[100] IBM. [n.d.]. LSF. Manual. https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/

[101] Weipeng Jing, Yaqiu Liu, and Hongrun Shao. 2015. Reliability-aware DAG scheduling with primary-backup in cloud computing. *International Journal of Computer Applications in Technology* 52, 1 (2015), 86–93.

[102] Rekha Kashyap and Deo Prakash Vidyarthi. 2014. Security-Aware Real-Time Scheduling for Hypervisors. In *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*. IEEE, 1520–1527. https://doi.org/10.1109/CSE.2014.282

[103] Brijesh Kashyap Chejerla and Sanjay Madria. 2017. QoS guaranteeing robust scheduling in attack resilient cloud integrated cyber physical system. *Future Generation Computer Systems* 75 (March 2017). https://doi.org/10.1016/j.future.2017.02.034

[104] Gastón Keller, Michael Tighe, Hanan Lutfiyya, and Michael Bauer. 2014. A hierarchical, topology-aware approach to dynamic data centre management. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 1–7.

[105] Nakku Kim, Jungwook Cho, and Euiseong Seo. 2014. Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems* 32 (2014), 128–137.

[106] Praveenkumar Kondikoppa, Chui-Hui Chiu, Cheng Cui, Lin Xue, and Seung-Jong Park. 2012. Network-aware scheduling of mapreduce framework ondistributed clusters over high speed networks. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*. ACM, 39–44.

[107] George Kousiouris, Tommaso Cucinotta, and Theodora Varvarigou. 2011. The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks. *Journal of Systems and Software* 84, 8 (2011), 1270–1291.

[108] Sriram Krishnan, Mahidhar Tatineni, and Chaitanya Baru. 2011. *myHadoop-Hadoop-on-Demand on traditional HPC resources*. Technical Report TR-2011-2. San Diego Supercomputer Center, University of California, San Diego.

[109] Kubernetes. [n.d.]. Argo workflow. https://argoproj.github.io/argo/

[110] Muhammad Shafie Abd Latiff, Syed Hamid Hussain Madni, Mohammed Abdullahi, et al. 2016. Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. *Neural Computing and Applications* (2016), 1–15.

[111] Dhinesh Babu LD and P Venkata Krishna. 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing* 13, 5 (2013), 2292–2303.

[112] Young Choon Lee, Chen Wang, Albert Y Zomaya, and Bing Bing Zhou. 2012. Profit-driven scheduling for cloud services with data access awareness. *J. Parallel and Distrib. Comput.* 72, 4 (2012), 591–602.

[113] Hao Li, Hai Zhu, Guoheng Ren, Hongfeng Wang, Hong Zhang, and Liyong Chen. 2016. Energy-Aware Scheduling of Workflow in Cloud Center with Deadline Constraint. In *Computational Intelligence and Security (CIS), 2016 12th International Conference on*. IEEE, 415–418.

[114] Chang Liu, Xuyun Zhang, Chi Yang, and Jinjun Chen. 2013. CCBKE Session key negotiation for fast and secure scheduling of scientific applications in cloud computing. *Future Generation Computer Systems* 29, 5 (2013), 1300–1308.

[115] Francesco Lucrezia, Guido Marchetto, Fulvio Risso, and Vinicio Vercellone. 2015. Introducing network-aware scheduling capabilities in openstack. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*. IEEE, 1–5.

[116] Sheheryar Malik, Fabrice Huet, and Denis Caromel. 2012. Reliability aware scheduling in cloud computing. In *Internet Technology And Secured Transactions, 2012 International Conference for*. IEEE, 194–200.

[117] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2018. Learning scheduling algorithms for data processing clusters. *arXiv preprint arXiv:1810.01963* (2018).

[118] Cristian Mateos, Elina Pacini, and Carlos García Garino. 2013. An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments. *Advances in Engineering Software* 56 (2013), 38–50.

[119] Víctor Medel, Carlos Tolón, Unai Arronategui, Rafael Tolosana-Calasanz, José Ángel Bañares, and Omer F Rana. 2017. Client-Side Scheduling Based on Application Characterization on Kubernetes. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*. Springer, 162–176.

[120] Peter Mell and Tim Grance. 2011. *The NIST Definition of Cloud Computing*. Special Publication 800-145. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, Gaithersburg, MD 20899-8930. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[121] Mohand Mezmaz, Nouredine Melab, Yacine Kessaci, Young Choon Lee, E-G Talbi, Albert Y Zomaya, and Daniel Tuyttens. 2011. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *J. Parallel and Distrib. Comput.* 71, 11 (2011), 1497–1508.

[122] Microsoft Azure. [n.d.]. Big Compute. https://azure.microsoft.com/en-us/solutions/big-compute/

[123] Stefan Nastic, Thomas Rausch, Ognjen Scekic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov, and Radu Prodan. 2017. A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing* 21, 4 (2017), 64–71.

[124] Nintex. [n.d.]. Workflow. https://www.nintex.com/workflow-automation/nintex-workflow-cloud/

[125] Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. 2013. Sparrow: distributed, low latency scheduling. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 69–84.

[126] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*. IEEE, 400–407.

[127] PBS. [n.d.]. PBS Pro User Guide. Misc Manual. https://www.pbsworks.com/pdfs/PBSUserGuide14.2.pdf

[128] Alfonso Quarati, Andrea Clematis, Antonella Galizia, and Daniele D'Agostino. 2013. Hybrid clouds brokering business opportunities, QoS and energy-saving issues. *Simulation Modelling Practice and Theory* 39 (2013), 121–134.

[129] S. Rampersaud and D. Grosu. 2017. Sharing-Aware Online Virtual Machine Packing in Heterogeneous Resource Clouds. *IEEE Transactions on Parallel and Distributed Systems* 28, 7 (July 2017), 2046–2059. https://doi.org/10.1109/TPDS.2016.2641937

[130] Xiaoqi Ren, Ganesh Ananthanarayanan, Adam Wierman, and Minlan Yu. 2015. Hopper: Decentralized Speculation-aware Cluster Scheduling at Scale. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. ACM, New York, NY, USA, 379–392. https://doi.org/10.1145/2785956.2787481

[131] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. 2013. Omega: flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 351–364.

[132] Sachin Shetty, Xuebiao Yuchi, and Min Song. 2016. Security-Aware Virtual Machine Placement in Cloud Data Center. In *Moving Target Defense for Distributed Systems*. Springer, 13–24. https://doi.org/10.1007/978-3-319-31032-9_2

[133] Shawn M. Strande, Haisong Cai, Trevor Cooper, Karen Flammer, Christopher Irving, Gregor von Laszewski, Amit Majumdar, Dmistry Mishin, Philip Papadopoulos, Wayne Pfeiffer, Robert S. Sinkovits, Mahidhar Tatineni, Rick Wagner, Fugang Wang, Nancy Wilkins-Diehr, Nicole Wolter, and Michael L. Norman. 2017. Comet: Tales from the Long Tail: Two Years In and 10,000 Users Later. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact* (New Orleans, LA, USA) *(PEARC17)*. ACM, New York, NY, USA, Article 38, 7 pages. https://doi.org/10.1145/3093338.3093383

[134] Sen Su, Jian Li, Qingjia Huang, Xiao Huang, Kai Shuang, and Jie Wang. 2013. Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Comput.* 39, 4 (2013), 177–188.

[135] Gang Sun, Dan Liao, Dongcheng Zhao, Zichuan Xu, and Hongfang Yu. 2015. Live migration for multiple correlated virtual machines in cloud-based data centers. *IEEE Transactions on Services Computing* 11, 2 (2015), 279–291.

[136] Xiaoyong Tang and Weizhen Tan. 2016. Energy-efficient reliability-aware scheduling algorithm on heterogeneous systems. *Scientific Programming* 2016 (2016), 14.

[137] Jie Tao, Joanna Kolodziej, Rajiv Ranjan, Prem Prakash Jayaraman, and Rajkumar Buyya. 2015. A note on new trends in data-aware scheduling and resource provisioning in modern HPC systems. *Future Generation Computer Systems* 51 (2015), 45 – 46. https://doi.org/10.1016/j.future.2015.04.016 Special Section: A Note on New Trends in Data-Aware Scheduling and Resource Provisioning in Modern HPC Systems.

[138] Douglas Thain, Todd Tannenbaum, and Miron Livny. 2005. Distributed computing in practice: the Condor experience. *Concurrency and computation: practice and experience* 17, 2-4 (2005), 323–356.

[139] The Chameleon Cloud Team and Gregor von Laszewski. 2019. Cloud Computing. https://laszewski.github.io/book/chameleon/

[140] Antony Thomas, G Krishnalal, and VP Jagathy Raj. 2015. Credit based scheduling algorithm in cloud computing environment. *Procedia Computer Science* 46 (2015), 913–920.

[141] Michael Tighe and Michael Bauer. 2014. Integrating cloud application autoscaling with dynamic vm allocation. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 1–9.

[142] Michael Tighe, Gastón Keller, Michael Bauer, and Hanan Lutfiyya. 2013. A distributed approach to dynamic VM management. In *Network and Service Management (CNSM), 2013 9th International Conference on*. IEEE, 166–170.

[143] E Torabzadeh and M Zandieh. 2010. Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. *Advances in Engineering Software* 41, 10 (2010), 1238–1243.

[144] Johan Tordsson, Rubén S Montero, Rafael Moreno-Vozmediano, and Ignacio M Llorente. 2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems* 28, 2 (2012), 358–367.

[145] tpc 2019. TPC-H: a Decision Support Benchmark. Web Page. http://www.tpc.org/tpch/

[146] twister 2019. Twister2: Flexible, High performance data processing. Web Page. https://twister2.org/

[147] Univa. [n.d.]. Grid Engine Manual. Misc Manual. http://www.univa.com/resources/files/univa_user_guide_univa__grid_engine_854.pdf

[148] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. 2013. Misc cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. *Future Generation Computer Systems* 29, 4 (2013), 973–985.

[149] Tien Van Do and Csaba Rotter. 2012. Comparison of scheduling schemes for on-demand IaaS requests. *Journal of Systems and Software* 85, 6 (2012), 1400–1408.

[150] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. 2013. Apache Hadoop YARN: Yet Another Resource Negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing* (Santa Clara, California) *(SOCC '13)*. ACM, New York, NY, USA, Article 5, 16 pages. https://doi.org/10.1145/2523616.2523633

[151] Gregor von Laszewski. 2019. Cloud Computing. https://cloudmesh-community.github.io/book/vonLaszewski-cloud.pdf

[152] Gregor von Laszewski. 2019. Cloudmesh Cloud Manual. Web Page. https://github.com/cloudmesh/cloudmesh-manual

[153] Gregor von Laszewski and Wo L. Chang. 2018. NIST Big Data Interoperability Framework: Volume 8, Reference Architecture Interfaces. https://github.com/cloudmesh-community/nist/blob/master/docs/nistvol8-2.pdf

[154] Gregor von Laszewski, Ian Foster, and Jarek Gawor. 2000. CoG Kits: A Bridge Between Commodity Distributed Computing and High-performance Grids. In *Proceedings of the ACM 2000 Conference on Java*

Grande (San Francisco, California, USA) *(JAVA '00)*. ACM, New York, NY, USA, 97–106. https://doi.org/10.1145/337449.337491

[155] Gregor von Laszewski, Geoffrey C. Fox, Gregor von Laszewski, Geoffrey C. Fox, and FutureGrid Team. 2010. Dynamic Provisioned Experiments in FutureGrid. 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom2010), Indianapolis, IN. http://grids.ucs.indiana.edu/ptliupages/presentations/vonLaszewski-10-FG-proj-management.pdf

[156] Gregor von Laszewski, Jarek Gawor, Carlos J. Peña, and Ian Foster. 2002. InfoGram: A Grid Service That Supports Both Information Queries and Job Execution. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC '02)*. IEEE Computer Society, Washington, DC, USA, 333–. http://dl.acm.org/citation.cfm?id=822086.823347

[157] Gregor von Laszewski and Mihael Hategan. 2005. Workflow Concepts of the Java CoG Kit. *J. Grid Comput.* 3, 3-4 (2005), 239–258. https://doi.org/10.1007/s10723-005-9013-5

[158] Gregor von Laszewski, Mihael Hategan, and Depti Kodeboyina. 2007. Grid Workflow with the Java CoG Kit. In *Workflows for E-science: Scientific Workflows for Grids*, Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields (Eds.). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 16. https://laszewski.github.io/papers/vonLaszewski-workflow-book.pdf

[159] Gregor von Laszewski, Hyungro Lee, Javier Diaz, Fugang Wang, Koji Tanaka, Shubhada Karavinkoppa, Geoffrey C. Fox, and Tom Furlani. 2012. Design of an Accounting and Metric-basedcloud-shifting and Cloud-seeding Framework for Federatedclouds and Bare-metal Environments. In *Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit* (San Jose, California, USA) *(FederatedClouds '12)*. ACM, New York, NY, USA, 25–32. https://doi.org/10.1145/2378975.2378982

[160] Gregor Von Laszewski, Fugang Wang, Hyungro Lee, Heng Chen, and Geoffrey C Fox. 2014. Accessing multiple clouds with cloudmesh. In *Proceedings of the 2014 ACM international workshop on Software-defined ecosystems*. ACM, 21–28.

[161] G. von Laszewski, L. Wang, A. J. Younge, and X. He. 2009. Power-aware scheduling of virtual machines in DVFS-enabled clusters. In *2009 IEEE International Conference on Cluster Computing and Workshops*. 1–10. https://doi.org/10.1109/CLUSTR.2009.5289182

[162] Lizhe Wang, Gregor von Laszewski, Jay Dayal, and Fugang Wang. 2010. Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID '10)*. IEEE Computer Society, Washington, DC, USA, 368–377. https://doi.org/10.1109/CCGRID.2010.19

[163] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu. 2014. BigDataBench: A big data benchmark suite from internet services. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. 488–499. https://doi.org/10.1109/HPCA.2014.6835958

[164] Wei Wang, Guosun Zeng, Daizhong Tang, and Jing Yao. 2012. Cloud-DLS: Dynamic trusted scheduling for Cloud computing. *Expert Systems with Applications* 39, 3 (2012), 2321–2329.

[165] Yuandou Wang, Hang Liu, Wanbo Zheng, Yunni Xia, Yawen Li, Peng Chen, Kunyin Guo, and Hong Xie. 2019. Multi-objective workflow scheduling with Deep-Q-network-based Multi-agent Reinforcement Learning. *IEEE Access* 7 (2019), 39974–39982.

[166] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. 2012. SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments. *J. Comput. System Sci.* 78, 5 (2012), 1280–1299.

[167] Baomin Xu, Chunyan Zhao, Enzhao Hu, and Bin Hu. 2011. Job scheduling algorithm based on Berger model in cloud environment. *Advances in Engineering Software* 42, 7 (2011), 419–425.

[168] A. J. Younge, G. von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers. 2010. Efficient resource management for Cloud computing environments. In *International Conference on Green Computing*. 357–364. https://doi.org/10.1109/GREENCOMP.2010.5598294

[169] R. Yu, G. Xue, X. Zhang, and D. Li. 2017. Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2017.8056945

[170] Haitao Yuan, Jing Bi, Wei Tan, and Bo Hu Li. 2016. CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers. *IEEE Transactions on Automation Science and Engineering* 13, 2 (2016), 976–985.

[171] Haitao Yuan, Jing Bi, Wei Tan, and Bo Hu Li. 2017. Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds. *IEEE Trans. Automation Science and Engineering* 14, 1 (2017), 337–348.

[172] Haitao Yuan, Jing Bi, Wei Tan, MengChu Zhou, Bo Hu Li, and Jianqiang Li. 2017. TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. *IEEE transactions on cybernetics* 47, 11 (2017), 3658–3668.

[173] Haitao Yuan, Jing Bi, MengChu Zhou, and Ahmed Chiheb Ammari. 2017. Time-aware multi-application task scheduling with guaranteed delay constraints in green data center. *IEEE Transactions on Automation Science and Engineering* 15, 3 (2017), 1138–1151.

[174] Haitao Yuan, Jing Bi, Mengchu Zhou, and Khaled Sedraoui. 2018. WARM: Workload-Aware Multi-Application Task Scheduling for Revenue Maximization in SDN-Based Cloud Data Center. *IEEE Access* 6 (2018), 645–657.

[175] Lingfang Zeng, Bharadwaj Veeravalli, and Xiaorong Li. 2015. SABA: A security-aware and budget-aware workflow scheduling strategy in clouds. *Journal of parallel and Distributed computing* 75 (2015), 141–151.

[176] PeiYun Zhang and MengChu Zhou. 2018. Dynamic cloud task scheduling based on a two-stage strategy. *IEEE Transactions on Automation Science and Engineering* 15, 2 (2018), 772–783.

[177] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li. 2019. Energy-Efficient Scheduling for Real-Time Systems Based on Deep Q-Learning Model. *IEEE Transactions on Sustainable Computing* 4, 1 (Jan. 2019), 132–141. https://doi.org/10.1109/TSUSC.2017.2743704

[178] Wei Zhu, Yi Zhuang, and Long Zhang. 2017. A three-dimensional virtual resource scheduling method for energy saving in cloud computing. *Future Generation Computer Systems* 69 (2017), 66–74.

[179] Liyun Zuo, Lei Shu, Shoubin Dong, Chunsheng Zhu, and Takahiro Hara. 2015. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access* 3 (2015), 2687–2699.

List of Tables

Table 1: Comparison of Dynamic Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Energy-aware	Keller et al. [104]	Management of data center resources to reduce the management scope	Reducing the overhead in the data center management network	High complexity	Greedy algorithm	DCSim	Power, number of migrations, average number of racks, active hosts
Energy-aware, VM-based (migration), PUE	Tighe et al. [142]	Trade-off among number of migrations, SLA violation and power	Consider energy and SLA	more bandwidth usage	First fit algorithm	DCSim	Power consumption, number of migrations, SLA violations
Energy-aware, VM-based (migration)	Tighe et al. [141]	Auto scaling algorithm alongside a dynamic VM allocation algorithm	Reduce a number of migrations	No optimization criteria	Rule-based heuristic	DCSim	Power, SLA, Migrations
Energy-aware, VM-based (migration), Energy source	Sun et al. [135]	Introduction of a virtual data center to solve VM migration issues	Low complexity	Fixed band-with	Heuristic algorithm	Simulated environment	VM migration cost and time
Compute resource-based, VM-based Cost	Tordsson et al. [144]	Optimized placement of applications in multi-cloud environments	Emphasized on Price and performance in terms of hardware configuration, load balancing	Ignored security and energy efficiency at time of scheduling	Integer programming formulations	Amazon EC2	Throughput, number of jobs
Energy-aware, VM-based	Younge et al. [168]	Power aware	reduces cost	slight reduction in performance	heuristic	on premise cloud	power consumption

Table 2: Comparison of Energy-aware VM Placement-based Scheduling Algorithms (A)

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Energy source, Compute resource-based, Job-based	Calheiros et al. [60]	Intelligent scheduling combined DVFS capability	Improved energy efficiency	Ignored Network and Storage energy consumption	Rank method	Cloud Sim	Energy consumption
Energy-aware, PUE	Bessis et al. [51]	Improving communication for Distributed systems at the time of scheduling	Improved system performance	High complexity	Graph theory concepts	SIMIC	makespan, latency times
Energy-aware, Energy source, Cost, VM-based	Bi et al. [53]	Dynamic Scheduling algorithm for reducing energy consumption	Focused on performance and energy cost	High complexity due to virtualized Data centers	meta heuristic methods	Simulated environment	Profit, CPU utilization
Energy-aware, Compute resource-based, VM-based	Duan et al. [73]	Scheduling of VM machines	Improved the CPU load prediction	No optimization	Ant colony optimization	CloudSim	Energy Consumption
Energy-aware, Energy source, Cost, VM-based	Bui et al. [59]	Balance between energy efficiency and quality of service	Low complexity	Ignored cost, scalability	Greedy first fit algorithm	Simulated Environment	Energy, Memory, CPU
Energy-aware, Energy source, Cost, VM-based	Zhu et al. [178]	Data center balance while saving power consumption	Improved management of VM resource	High complexity	Multi-dimensional vector bin packing problem-based heuristic	CloudSim	SLA violations, Resource utilization
Energy-aware, Energy source, VM-based	Beloglazov et al. [50]	Enhancement of resource utilization by re-allocation of the resources.	Considered different types of workloads, No prior information about applications	Ignored cost and time	Heuristic algorithm	CloudSim	Energy, Average SLA, migrations
Energy source, Compute resource-based , VM-based	Quarati et al. [128]	Reservation of a quota of private resources	Reduced energy consumption and carbon emission	Lacks implementation on a real-world cloud platform	Round robin algorithm	Discrete Event Simulator	User satisfaction, energy saving, energy consumption
PUE, Job-based	Garg et al. [83]	Optimal scheduling policies	Reduced energy cost, energy consumption	Ignored security	Meta-scheduling policies	Simulated environment	Average energy consumption, average carbon emission, arrival rate of application

Table 3: Comparison of Energy-aware VM Placement-based Scheduling Algorithms (B)

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Energy-aware, VM-based	Keke et al. [82]	Cloudlets for energy reduction	Reduced energy consumption	No time consideration	FCFS scheduling policy	DECM-Sim	Energy consumption
Energy source, VM-based	Dabbagh et al. [65]	Energy-aware resource management decisions	improved performance	No optimization criteria, high complexity	K-means clustering	Testbed	Average CPU and Network utilization
Energy source, VM-based	Kim et al. [105]	VM energy consumption estimation model	Reduced cost, power consumption	More complex to implement, Ignored time	Power aware scheduling algorithm	Xen 4.0 hypervisor	Energy consumption, error rate
Compute resource-based, VM-based	Van Do et al. in [149]	Interaction aspects between on-demand requests and the allocation of virtual machines	Reduced energy consumption	No cost and time optimization	Power aware scheduling algorithm	Numerical Simulation	Average Energy consumption, average heat emission
Energy-aware, Energy source, Compute resource-based, VM-based (migration), Workflow	Li et al. [113]	Scheduling algorithm to reduce energy consumption while meeting the deadline constraint	Focused on energy consumption	Ignored processing power energy consumption, VM migration	Heuristic method	Simulated environment	Energy consumption
Energy source, PUE, VM-based	Ding et al. [72]	Dynamic VMs scheduling	Increased Processing Capacity	Ignored VM migration, Power penalties of status transitions of processor	FCFS	Simulated environment	Deadline, Energy consumption

Table 4: Comparison of Network-aware VM Placement-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, VM-based (placement)	Rampersaud et al. [129]	Used page-sharing concept to handle VM Packing problem	Improvement of memory sharing during allocation decisions	High complexity	Linear programming technique	Simulated environment	Memory reduction, number of excess servers
Compute resource-based, VM-based (placement)	Biran et al. [56]	Consideration of traffic bursts in deployed services	Minimizing the maximum load ratio over all the network	Ignored energy consumption	Greedy heuristic algorithm	Testbed	Average packet delivery delay , placement solving time
VM-based (placement)	Yu et al. [169]	Service provisioning on IaaS platform while focusing on the inter-connected VMs.	High availability	High complexity	Heuristic algorithm	Simulator	Average VM consumption ratio, average running time
Compute resource-based, VM-based (placement)	Bi et al. [54] [52]	Architecture for self management of data centers	Considered temporal request of multi-tier web applications	does not consider security parameters	Queuing approach	trace-driven simulation	Cost
Compute resource-based, Workflow, Network-based	Kondikoppa et al. [106]	To make Hadoop scheduler aware of network topology	Improved data locality	Ignored cost, energy, security	FIFO	Eucalyptus-based testbed	Execution time, delay for scheduling task
Compute resource-based, VM-based (placement)	Lucrezia et al. [115]	Investigated OpenStack for the deployment of network service graphs	Increased throughput	Analyzing time is more, Ignored policy-constraints in order to define administration rules	Brute force algorithm	KVM hypervisors	VM locations, traffic throughput and latency

Table 5: Comparison of Cost-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Task-based, Cost-based	Bi et al. [54] [52]	Architecture for self management of data centers	Considered temporal request of multi-tier web applications	does not consider security parameters	Queuing approach	trace-driven simulation	Cost
Compute resource-based, data-based, task, Latency, VM-based, Cost-based	Yuan et al. [171, 172]	Emphasizing profit maximization	handles service delay bound	High complexity	PSO and SA	simulation environment	Revenue
Compute resource-based, Task-based, Cost-based	Zuo et al. [179]	Multi-objective Task Scheduling	Improved performance	Ignored energy consumption	Ant colony optimization	CloudSim	Cost, makespan, deadline violation rate
Compute resource-based, Workflow-based, Cost-based	Arabnejad et al. [44]	Re-use of pre-provisioned instances for scheduling	Less complexity	Ignored security and energy efficiency	Deadline early Tree algorithm	CloudSim	Cost and deadline
Compute resource-based, VM-based, Cost-based	Wu et al. [166]	VM usage efficiency designed utility function by considering dynamic VM deploying time, processing time and data transfer time	Improved cost saving	Does not support security and energy efficient	Admission control and scheduling algorithm	CloudSim	Average response time, total profit
Data-based, Cost-based	Lee et al. [112]	Personalized features of the user request and the elasticity of SLA properties	Reduced operational costs and increase profits	Objectives conflict with each other	binary integer programming	CloudSim	Average utilization, average net profit rate, average response time
Compute resource-based, VM-based, Cost-based	Ari et al. [47]	Finite Element Analysis cloud service with a focus on mechanical structural analysis, performance characterization and job scheduling issues	Throughput improvement and resource utilization	Ignored cost	Adaptive algorithm	Testbed	Throughput and time

Table 6: Comparison of Time-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Workflow-based, Time-based	Arabnejad et al. [46]	Dynamically provisioned commercial cloud environments	Evaluation of task selection algorithms reveals impact of workflow symmetry	High complexity	Rank method	CloudSim	Response time, Cost
Compute resource-based, Data-based, Task-based, Time-based	Van den Bossche et al. in [148]	Deadline-based workloads in a hybrid cloud setting	Minimize cost and time	does not handle multiple workflows	hybrid scheduling approach	Simulator	Total Cost, application deadline met, turnaround time, data transferred
Compute resource-based, Workflow-based, Time-based	Thomas et al. [140]	Task length aware scheduling	Lesser makespan and increased resource utilization	No comparison with existing algorithm	Min-min	CloudSim	Makespan
Compute resource-based, VM-based	Erdil [76]	Disseminated information as agents of dissemination sources for resource scheduling	Availability of resource state, reduces dissemination overhead	Ignored cost as parameters	Adaptive proxy algorithm	Scalable simulation network framework	Query satisfaction rates, random walk hop count limit
Compute resource-based, Task-based, Time-based	Xu et al. [167]	Berger model and assign tasks on optimal resources to meet user's QoS requirements	Optimal completion time	Ignored cost and energy efficiency, security	Resource allocation algorithm and then followed by job scheduling	CloudSim	Time, bandwidth
Compute resource-based, VM-based, Time-based	Frincu [81]	Priority scheduling and searching for optimal allocation of components on nodes to ensure a homogeneous spread of component types on nodes	Minimizing the application cost	Centralized approach represents a single point of failure	Nonlinear-programming	Simulator platform	Average load per node, optimal allocation, reliability
Compute resource-based, Task-based, Time-based	Yuan et al. [173]	Task scheduling in green data centers	Investigated temporal variations	Ignored energy consumption and cost	PSO and SA	Simulated Environment	Delay bound and time

Table 7: Comparison of Reliability-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Job-based, Reliability-based	Malik et al. [116]	Reliability assessment mechanism for scheduling resources	Reliability assessment algorithms for general applications and real time applications	No security and energy parameters consideration	Max -min	Amazon EC2 cloud	Fault tolerance, time
Compute resource-based, Job-based, Reliability-based	Jing et al. [101]	Model for fault-tolerant aware scheduling	Low complexity	No cost, time optimization	Adaptive secure scheduling algorithm	Simulated environment	Reliability
Compute resource-based, Task-based, Reliability-based	Abdulhamid et al. [110]	Uncountable numeric nodes for resource in clouds	Lower makespan	No optimization	League championship algorithm	CloudSim	Failure ratio, the failure slowdown and the performance improvement rate
Energy-aware , Energy source, Reliability-based	Tang et al. [136]	Reliability and energy-aware task scheduling architecture	To get good trade off among performance, reliability, and energy consumption	No support for cost optimization	Heuristic method	Discrete event simulation environment	Schedule length, Energy consumption, Application reliability

Table 8: Comparison of Security-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, VM-based, Security-based	Afoulki et al. [37]	Security risk management in a cloud	Less complexity	Consolidation issues while implementing policies	Greedy Algorithm	Simulated environment	VM placement time
Compute resource-based, Data-based, Security-based	Chejerla et al. [103]	Scheduling of resources in cloud integrated Cyber-physical Systems	Consideration of security, time	High complexity	Heuristic algorithm	Simulated environment	Speed up, resource utilization, makespan
Compute resource-based, VM-based, Security-based	Kashyap et al. [102]	Secure aware scheduling of real time-based applications	Improved response time and overall security	High complexity	Priority Algorithm	Hypervisor	Deadline, Security
Compute resource-based, VM-based, Security-based	Shetty et al. [132]	VM placement techniques to reduce security risks	Reduced computing costs and deployment costs	No optimization criteria	Heuristic algorithm	Simulated environment	Cost, security risks
Compute resource-based, Workflow-based, Security-based	Liu et al. [114]	Scheme for security aware scheduling	Reduced the computational load and execution time	No cost optimization involved	Adaptive secure scheduling algorithm	KVM hyper-visor	Time unit consumed per computational load
Compute resource-based, Workflow-based, Security-based	Zeng et al. [175]	Scheduling algorithm for resource utilization	Low complexity	Ignored energy consumption	Clustering and prioritization algorithm	Simulated environment	Makespan and speed up
Compute resource-based, Task-based, Security-based	Wang et al. [164]	Uncountable numeric nodes for resource in clouds	Provided scheduling of resources in secure way	Ignored cost	Bayesian algorithm	CloudSim	Trust value, average schedule length
Compute resource-based, VM-based, Security-based	Bilogrevic et al. [55]	Scheduling services on the cloud for mobile devices	Enhanced Performance	No support cost optimization, Ignores power consumption by the network	Privacy aware scheduling schema	Testbed	Time, Data exchanged, privacy in approach

Table 9: Comparison of Heuristic-based Scheduling Algorithms (A)

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Job-based	Mezmaz et al. [121]	Addressed the precedence-constrained parallel applications for cloud computing	Reduced energy consumption	High complexity of implementation and operation	Genetic algorithm	Simulated environment	Energy, speed up
Compute resource-based, Job-based	Gasior et al. [84]	Parallel and distributed scheme for scheduling jobs	Multi-objective optimization, consideration of security risks also	No cost consideration	Genetic algorithm	Simulation Testbed	Flow time, makespan, turnaround time
Compute resource-based, Job-based	Cristian et al. [118]	Scheduler for job scheduling, consider static cloud	Minimize weighted flowtime and makespan	does not handle energy consumption	Ant colony optimization and swarm intelligence approach	CloudSim	makepan
Compute resource-based, Task-based, Energy-based	Babu [111]	Based priority of tasks, designed load balancing algorithm	Maximize throughput	High operational complexity	Honey Bee algorithm	CloudSim	Makespan, Number of task migrations
Compute resource-based, Task-based	Kousiouris et al. [107]	Virtual machines affect the performance of other VMs executing on the same node	Reduce performance overhead	Lacks implementation on a real-world cloud platform	Genetic algorithm	Simulated environment	Degradation, test score delay
Compute resource-based, VM-based	Torabzadeh et al. [143]	Flowshow job problem	Minimized makespan and mean completion time	Not considered cost	Simulated annealing	Simulated environment	Computation time
Compute resource-based, Task-based	Sen et al. [134]	Cost-efficient task-scheduling algorithm using two heuristic strategies	Reduced monetary costs	Ignored security	Heuristic strategies	Numerical experiments	Makespan

Table 10: Comparison of Heuristic-based Scheduling Algorithms (B)

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Workflow-based	Abrishami et al. [35]	Cost-optimized, deadline-constrained execution of workflows in cloud. considered required run-time and data estimates in order to optimize workflow execution	Minimize execution cost with in deadline	Ignored data transfer time, security	PCP algorithm	Simulated environment	Normalized cost
Compute resource-based, Workflow-based	Bousselm et al. [57]	QoS-based	Consideration of QoS parameters	High complexity	Parallel Cat Swarm Optimization	Simulated environment	Execution time, execution and storage cost, availability of resources and data transmission time
Compute resource-based, Job-based	Gutierrez-Garcia et al. [93]	Scheduling of Bag-of-tasks-based on allocation times of virtualized cloud resources	Makespan	Ignored cost	Heuristic algorithm	Testbed	Makepan, overhead time

Table 11: Comparison of ML-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Job-based, ML-based	Mingxi Cheng et al. [61]	Two stage resource provisioning and/or task scheduling processor	Consideration of energy consumption aspects	No security consideration	Reinforcement approach	Simulation Testbed	Energy cost and runtime
Compute resource-based, Workflow-based, ML-based	Hongzi Mao et al. [117]	Presented Decima scheduler	Consideration of low latency scheduling decisions	High complexity	Reinforcement learning method	Simulated environment	Execution time
Compute resource-based, Job-based, ML-based	Qingchen Zhang et al. [177]	A hybrid dynamic voltage and frequency scaling (DVFS) scheduling algorithm	Minimize energy consumption	does not handle security	Deep Q-learning model	Simulation environment	energy consumption
Compute resource-based, Workflow-based, ML-based	Yuandou Wang et al. [165]	Focused on workflow scheduling	Minimize energy consumption	Ignored time, security	Markov method	Simulated environment	Energy efficiency
Compute resource-based, Task-based, ML-based	Enda Barret et al. [49]	Guided an optimal decision in the process of resource allocation	Reduced Time	Ignored security	Deep Q learning method	Simulation	Makespan

Table 12: Comparison of Different Batch Resource Management Systems

Classifaction	Framework	Batch Features	Cloud Bursting	Containers	Comment
Cloud bursting, Cluster model	Slurm [18]	Policy driven, backfill, exclusive and non-exclusive access to compute nodes	AWS, Azure, Google, Oracle	Yes	Open Source, popular
Cloud bursting, Cluster model	Univa Grid Engine [19]	Policy driven, backfill, exclusive and non-exclusive access to compute nodes, fault tolerant master	AWS, Azure, Google	Yes	previously SUN Grid Engine, Genias Codine
Cloud bursting, Cluster model	Load Sharing Facility (LSF) [100]	Policy driven, backfill, exclusive, non-exclusive access to compute nodes	IBM Cloud, AWS, Google and Azure	Yes	Previously OpenLava, IBM Open Source
Cloud bursting, Cluster model	Moab [36]	Fairness policies, dynamic priorities, and extensive reservations	AWS, Azure, Oracle, Google	Yes	Open Source
Cloud bursting, Cluster model	Open Portable Batch System (OpenPBS) [15, 97]	Policy driven, backfill, exclusive and non-exclusive access to compute nodes, fault tolerant master	AWS, Azure, Google, Oracle	Yes	Open Source

Table 13: Comparison of Different IaaS Models

Classification	Provider or Framework	Pricing	Database RDS	Reliability	Monitoring	Base OS	Programming Framework
VM-based, Data-based	Future Grid (discontinued) [79]	Free Academic	User Choice	Good	Good	Linux	Openstack, Open-Cirrus, Eucalyptus, Cloudmesh
VM-based, Data-based	Rackspace [5]	Pay-as-you-go	MySQL	Good	Extensive	Ubuntu	Java, Python
VM-based, Data-based, Container-based	Google App Engine [10]	Pay as you go	Cloud SQL	Extensive	good	linux, free BSD, windows	Python, Java, PHP and Go, Node.js
VM-based, Data-based, Container-based	Amazon WS [3]	Pay-as-you-go or Yearly, reserved, spot	My SQL, Ms SQL, Oracle	Good	Good	Linux and windows	Python, Java, PHP, Ruby
VM-based, Data-based, Container-based	Microsoft Window Azure [2]	Pay-as-you-go, semester, year	Microsoft SQL Database	Average	Average	Windows and linux	Java, Php, .net
Data-based	Cloud Sigma [4]	Pay-as-you-go	SQL	Good	Good	Average	Python, Java, PHP, Python, Ruby, Clojour
VM-based, Data-based	Chameleon Cloud [139]	Free Academic	User Choice	Moderate	Moderate	Linux	Openstack

Table 14: Comparison of Container-based Scheduling Algorithms

Classification	Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters
Compute resource-based, Container-based	Guerrero et al. [91]	Optimize physical machine utilization	Increase resource utilization	High complexity	Genetic Algorithm	Simulation environment	Resource Utilization , Performance
Compute resource-based, Energy-aware, Container-based	Hanaf et al. [94]	Container and host selection policies	Improved SLA	Highly complex	Pre-Selection method	Simulated environment	Energy Consumption
Compute resource-based, Container-based	Medel et al. [119]	Scheduler for minimizing resource contentions	Reduce resource contention	Ignored Time optimization	Priority algorithm	Kubernetes	Time
Compute resource-based, Container-based	Dziurzanski et al. [74]	Optimization of the container allocation	Easy to implement	Ignored network optimization	Heuristic method	Simulated environment	Performance
Compute resource-based, Container-based	Guerrero et al. [92]	Optimized the deployment of micro services-based applications	Improved security	High complexity	Genetic algorithm	Simulation environment	Resource utilization

Table 15: Comparison of other Resource Management Systems

Classification	Framework	Architecture	Usage	Open source	Support	Applications	Programming Framework
Datacenter-based	Eagle [70]	Hybrid	Differentiates short and long jobs	EPFL IC IINFCOM LABOS, Switzerland	Spark	Different workloads and Parallel jobs	Python, Java, PHP, Python, Ruby
Cluster-based	Hopper [130]	Decentralized	Speculation-aware job scheduler	Microsoft Research	Spark	CPU intensive	Java, Php, .net
Cluster-based	Tetris [90]	Centralized	Multi-resource bin-packing	Microsoft	Generic applications	CPU intensive	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala
PaaS-based, Data-based	Fawkes [86]	Centralized	Dynamic resource balancing	TU Delft	Mapreduce frameworks	Data intensive	Python, Java, PHP, Python, Ruby
Cluster-based	Omega [131]	Decentralized	Shared state abstraction	University of Cambridge	Custom applications	Parallel applications	Java, Php, .net
Peer-to-Peer	OurGrid [41]	Centralized	Equitable Resource Sharing	Universidade Federal de CampinaGrand, Brazil	Generic applications	Bag of Tasks	Java, Python
PaaS-based, Data-based	Sparrow [125]	Decentralized	Randomized sampling approach	U.C. Berkeley AMPLab	Spark	CPU intensive	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala, Clojure, .Net
PaaS-based, Data-based	Yarn [150]	Monolithic	Resource requests with containers	Hadoop	Spark	Data intensive	Python, Java, PHP, Python, Ruby, Clojour