Draft:

Cloud Resource Scheduling Taxonomy

Rajni Aron[2], Gregor von Laszewski[1*], Geoffrey C. Fox[1]

Contents

Cloud Resource Scheduling Taxonomy

Rajni Aron[2], Gregor von Laszewski[1*], Geoffrey C. Fox[1]

[*] Corresponding Author: laszewski@gmail.com
[1] Intelligent Systems Engineering Dep., Indiana University, Bloomington, IN 47408, USA.
[2] School of Computer Science and Engineering, Lovely Professional University, Punjab, India

Abstract

The growth and development of scientific applications in the cloud demands the creation of efficient resource management systems. Due to the scale of resources, the heterogeneity of services, their interdependencies and unpredictability of load this is a complex problem. We present a resource scheduling taxonomy that is originating from our experience in utilizing and managing multi-cloud environments. Our study is backed up by a literature review that targets not only virtual machine, but also container and Function as a Service frameworks. It justifies our model and provides a an overview of existing scheduling techniques in cloud computing. As a result this work can lead a better understanding of scheduling for clouds in general. The study promotes 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.

1 Introduction

COLOR IN BLUE = modified by Gregor

Cloud computing has emerged as a computing paradigm to solve large-scale application in many domains including science, e-commerce, lifestyle, and many other areas. 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 [1]. 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-scalable usage patterns we see in large scale cloud applications. Hence, scheduling plays an important role in improving resource utilization providing the necessary guidance to optimize the allocation of resource. 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.

We showcase that scheduling in the cloud requires a multi-layered approach that not only schedules tasks and jobs, but also integrates provisioning of resources and dynamic adaptation of loads during the run-time of applications. Information has to be passed between the various layers that comprise our scheduling architecture for clouds to guide the optimal placement onto resources. This cloud-based scheduling models is more comprehensive than previous classical scheduling approaches as it is conducted on scales and resource pools that were previously not considered. The 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. Our study identifies services that assist in the formulation of the scheduling needs and interfaces for the NIST Big Data Reference Architecture (NIST-BDRA) [2] and it's interface definitions currently under development [3].

The paper is structured as follows. We present first in Section 2 our general terminology we use throughout the paper. Next we present in Section **??** our architecture view and taxonomy that we have derived from our practical experience with FutureGrid [4, 5], FutureSystem, and Software such as Cloudmesh [6], Virtual Clusters [7], and Rain [8, 9, 10] while working on multi-hosted heterogeneous cloud frameworks. This view is backed up by our extensive literature research in Sections 4 and our classification based on the taxonomy we introduced in the previous sections. Lastly, we summarize some open issues in Section 5 that this study revealed and some concluding remarks in Section **??**.

2 General Scheduling Terminology for Clouds

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

Cloud Computing: 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 [1].

Cloud Resource: Is a resource offered by a cloud provider 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 to target an application providing concrete implementations for its instantiation and execution.

Resource Provisioning in the Cloud: The allocation of the resources demanded by users to specific applications is called provisioning.

Resource Scheduling in the Cloud: Resource scheduling in the cloud refers to the mapping of resources to fulfill the jobs resource requirements.

3 Scheduling Taxonomy for Clouds

In this section we introduce our scheduling taxonomy for clouds. Our taxonomy integrates the classical cloud architecture as defined by NIST [1]. However, as scheduling is conducted with resources in mind we also focus on aspects to deal with cloud resources, their physical instantiation and their connectivity while showcasing their relationship in our taxonomy.

3.1 Layered scheduling

The NIST cloud model promotes an easy to understand separation between infrastructure, platforms and services. 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 while enhancing it with Function as a Service (FaaS).

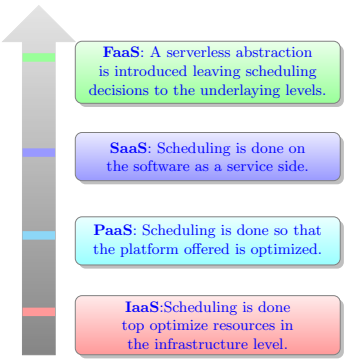


Figure 1: Multi-phase scheduling in a hierarchical resource model with less scheduling control and needs in the higher service levels by the user [11].

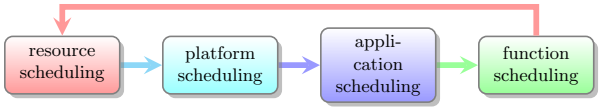


Figure 2: Multi-layered scheduling in a hierarchical resource model motivated by the NIST cloud architecture [11].

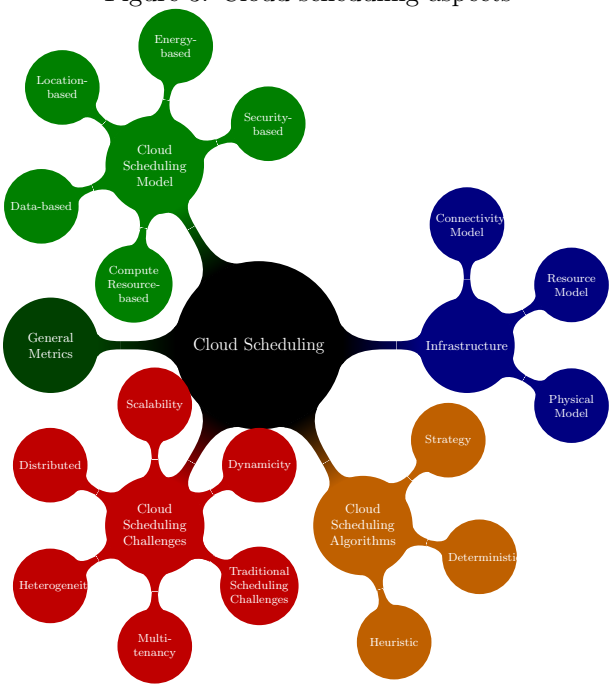
A platform provider may utilize insights of the infrastructure to offer to the users an optimized platform placement, while a software provider or application user may utilize information form the platform and or the infrastructure to offer scheduling on levels accessible to them. To facilitate the scheduling 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 the strategy of conducting a multi-layered scheduling approach in which we separate concerns as showcased in Figure 2. As most recently the FaaS model is introduced by the community we added it to the figure to indicate that although they are defined by the user, the functions have well defined resource constraints that make scheduling decisions on the infrastructure

provider level easier. Hence to optimize usage of the infrastructure, providers have come up with an easy to use extension to the original NIST model allowing for 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, but enough information between the layers needs to be exchanged to facilitate good scheduling decisions on each of the layers.

When put together, we distinguish several aspects that comprise Cloud scheduling. This includes metrics, cloud scheduling models, cloud scheduling challenges, the cloud infrastructure, and algorithms specifically designed to address clouds as seen in FIgure F:mindmap. These aspects are elaborated in more detail next.

Figure 3: Cloud scheduling aspects



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 [11].

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 integrated in the Taxonomy. It includes:

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

Resource Model representing models that the scheduling algorithm addresses including containers and functions, virtual machines and jobs, virtual clusters, provider managed resources, and multi-region provider managed resources.

Connectivity Model introduces a 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 hypergraphs. The connectivity model allows us to leverage classical scheduling algorithms while applying such models and leveraging established or new scheduling algorithms for these models.

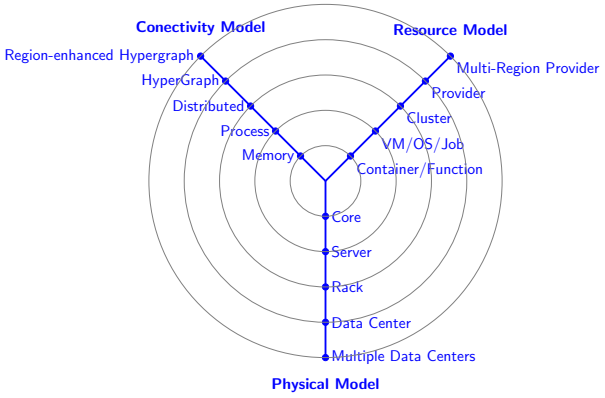


Figure 4: Resource Provider Focused Y-Cloud Taxonomy [11].

3.3 Cloud Scheduling Model

Now that we have identified the resource provider focused Y-Cloud taxonomy we can identify some important aspects 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.

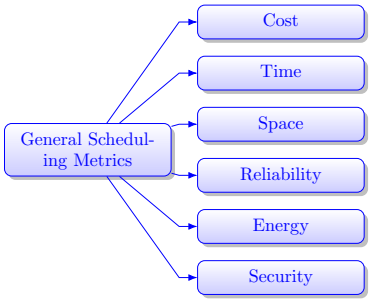


Figure 5: Traditional Compute Scheduling Metrics

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 in order to identify scheduling model based on virtual machines, attributes such as the once 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 Taxonomy of Challenges in Cloud Scheduling

Some of the obvious characteristics and challenges that are specifically related to clouds are listed next and are summarized in Figure 7 while enhancing the traditional scheduling challenges we introduced earlier.

Large scale: Clouds offer large number of resources to its users that need to be optimally utilized under 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 get catered at any point of time too can be in large scale. Handling them in such scale requires efficient

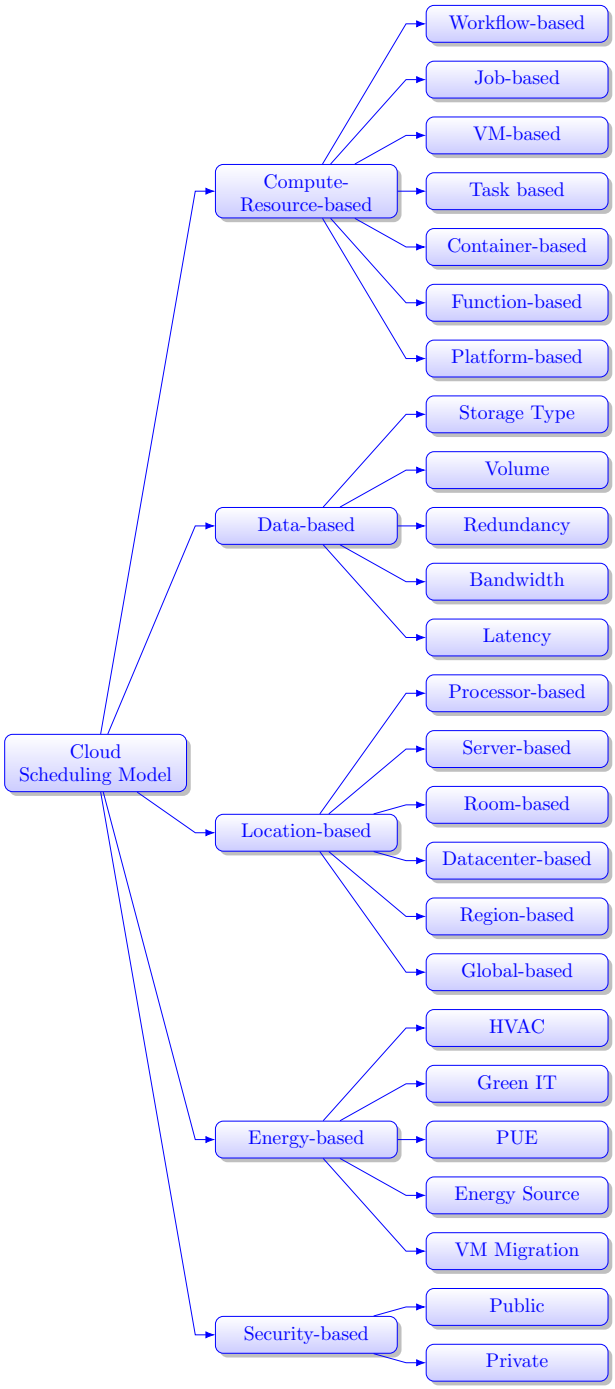


Figure 6: Cloud Scheduling Model Metrics

resource management. As such, scheduling becomes a complex endeavour. Rather a complex, dynamic and multi-faceted scheduling is necessary.

Dynamic nature of clouds: Due to the dynamic nature of the physical cloud infrastructure in which resources belonging to different administrative domains keep on joining and leaving the system scheduling must be adaptive.

Heterogeneous providers and services: There is no single cloud, but we have to recognize that the competitive nature in the cloud market promotes not only heterogeneous cloud providers, but also heterogeneous cloud services that may compete with each other and either offer the same or customized services targeting a particular user community. Resources in cloud environment are highly diversified in nature, capacity, working style and administrative domains. Being owned by different organizations, the resources offer minimal control over them making multi-layered scheduling necessary,

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 if 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 are 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 care 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 a number of scheduling challenges one of which is governed by on differentiating users and providers. Here, on the one hand, we focus on cloud providers that try to utilize in the best possible way 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 these quality assurances, but also have own constraints such as deadlines, cost, and implicit requirements from their applications such as data placement and management.

In both cases we need to address the challenge of provisioning resources and also the challenge of scheduling services onto these resources. Although these steps 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 function more, thereby, sometimes, degrading the node's performance.

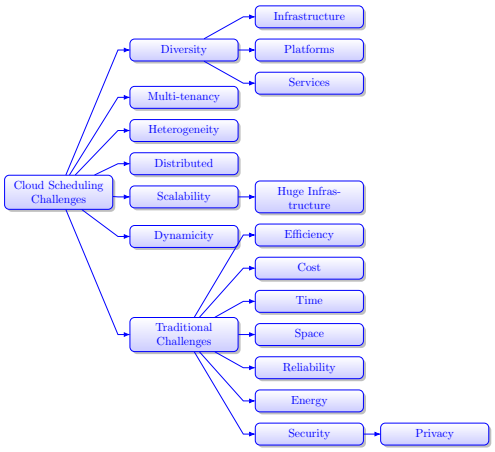


Figure 7: Scheduling challenges applied to all levels

3.5 Scheduling Challenges Arising form use of Containers

By using virtualization technologies such as virtual machines, they help to provide the illusion of a hardware resources but introduce 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 then the resources required to boot a virtual machine with its virtualized OS. As example we point out Kubernetes [12] where containers within a pod [13] share an IP address and find each other via local host. 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 a number of 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 issues. 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.

3.6 Challenges in Function as a Service

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

3.7 Taxonomy of Scheduling Units

The traditional units for scheduling include, processes, tasks, and jobs. However in the cloud we have an enhanced model that needs in addition to this address scheduling of virtual machines, containers, functions, platforms, clusters, 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 workflows. Hence, we distinguish as part of this model the following units that typically define work units:

Task: represents an abstract unit to be run on a cloud that may have complex resource association attached with it and may itself be build from other tasks with dependencies. It is not yet mapped onto a resource.

Job: A job is a computational activity made up of several tasks that may require different processing capabilities.

Function: represents a small computational unit with precisely specified resource requirements to run on a cloud.

Application: An application is a software for solving a (large) problem in a computational infrastructure; it may require splitting the computation into many jobs or it may be a monolithic application. In the later case, the whole application is allocated in a computational node and is usually referred to as application deployment.

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

On the other hand we find resources units that are typically associated with the provisioning step:

Resource: A resource is a basic computational entity that can be used to fulfill the requirements of application's execution. Resources have their own characteristics such as CPU characteristics, memory, software, etc. It can be a source of information and expertise. Resource is phenomenon that enhances the quality of application. Various parameters are associated with a resource, among them, the data speed, the processing speed, space and workload, which change over time.

Deployment: A deployment is a series of jobs that deploy a service onto the cloud that can be used for subsequent use.

Containers: can be defined as a any service related to scheduling, made available to users on demand from a cloud computing provider's servers.

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

Virtual clusters: An agglomeration of virtual services that build the core of a computational resource hosted in the cloud. They 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 is typically associated with an application and utilized for it. Just as containers or virtual machines it can be created, suspended, resumed, or terminated

Schedulers: Schedulers are processes that decide which task and process should be accessed and run at what time by the available resources. It helps to keep the performance of cloud at the highest level by scheduling in optimized way. Based on the frequency of schedulers operations, categorization is done: local scheduler, global scheduler and enterprise scheduler etc.

3.8 Taxonomy Classification of Resource Scheduling Algorithms

Next we present a short sample on resource scheduling algorithms that we found in literature related to cloud computing scheduling as showcased in Figure 8. We present in the figure only a relevant subset of algorithm classifications including the distinction between VM placement and QoS parameters based algorithms. Dependent on the locality and scope of the scheduling task often a deterministic approach is not suitable. When looking at heuristics [14] we find traditional algorithms such as hill-climbing but also a variety of nature inspired algorithms. The detail description of existing work in the field of resource scheduling algorithm is done in the next section.

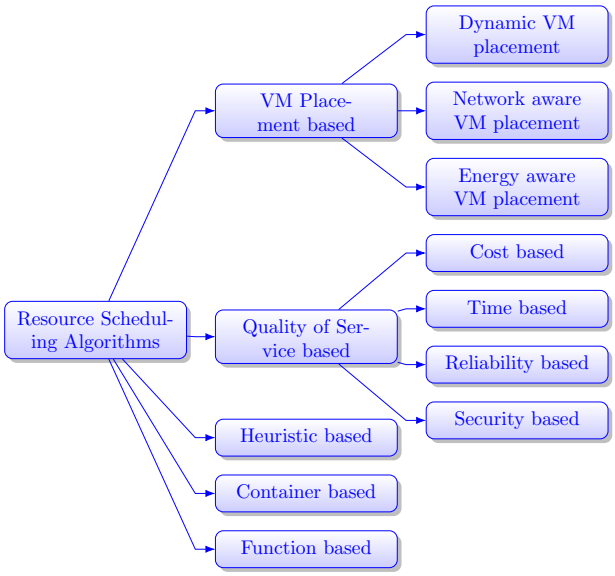


Figure 8: Classification of Resource Scheduling Algorithms

4 Literature review of Cloud Resource Scheduling Algorithms

In this section we conduct an exemplary but extensive literature review of cloud scheduling in order to confirm our taxonomy. As part of this review, we present a number of tables to compare the reviewed research and frameworks related to cloud scheduling. We organize this section by scheduling aspects 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).
We review scheduling needs for HPC in the cloud (Section 4.4) and workflows (Section 4.5).
We mention scheduling in public clouds (Section 4.6) while also looking at containers (Section 4.7), func-

tion as a service (Section 4.8) as well as distributed resource providers (Section 4.9) which can utilize service meshs (Section 4.10).

4.1 Dynamic Scheduling

As part of this scheduling task we often also find the distinction between static scheduling, where resources are scheduled once, [15] and dynamic scheduling, which is constantly updated during execution to find better resource utilization over time. The later is often motivated by need for scalability [16] across and within data centers or increased fault tolerance [17]. Association of other metrics into the dynamic scheduling approach is common to for example integrate power, reduce network bandwidth and enable more sophisticated Service level agreements [17]. In many cases not only the cloud user, but obviously also the cloud provider can benefit from dynamic scheduling [18]. We find that it can be beneficial to separate the scheduling task in multiple steps such as shown in [19]. Here live migration for correlated VMs is optimizing on data, compute, and bandwidth. Other cloud metrics such as price [20] are also common and will be in more detail addressed in Section 4.2.3. Obviously a rich number of algorithm can be applied such as shown in Section 4.3.
Table 1 lists a number of 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. Figure 8 showcases many different metrics that influence their schedule.

4.2.1 Energy Aware Scheduling

Energy consumption is a key issue for cloud providers due to the enormous cost associated with operating large cloud data center. By using server consolidation, optimizing operation on physical machines and using dynamic voltage scaling processors, energy consumption can be reduced as shown in [21, 22, 23].
Various scheduling methods such as minimize the total makespan [24], dynamic meta-heuristic [25], fractal mathematics [26], and machine learning clustering and stochastic [27] have been utilized.
It is obvious that multiple metrics must be included the correlate for example CPU, RAM and bandwidth [28]. Dynamicity, for example, while addressing peak loads [26] or migration [29] has naturally also an impact on the energy cost. Energy in hybrid and multiple data centers in the clouds is used in [30, 31, 32] while at the same time increasing the cloud provider brokers revenue. Energy consumption in heterogeneous clouds has also been considered [33]. Others create models to predict the energy consumption of each virtual machine [34] this requires certainly proper monitoring of the underlying server farms in the cloud in [35]. Integration of historical or previous program executions while recording their energy consumption can also be utilized [36]. Others focus on predicting future resource consumption needs [37].
A comparison of energy aware scheduling algorithm in cloud computing is shown in Table 2.

4.2.2 Network Aware Scheduling

As clouds project remote large scale resources network traffic to, from, and within must be considered for scheduling. This not only contains moving data in and out of the compute center or storage, but also may contain message exchange between to sometime complex distributed applications that run in these cloud data centers. Minimizing the distance between data providers and data consumers while for example replicating data [38] can save significant amount of traffic and has long been applied in the internet as one of its beneficial strategies. Service level agreements (SLA) [39] are playing an important role to achieve proper utilization as part of the scheduling effort. Treating the network as shared scarce resource [40] motivates the development of network-based scheduling algorithms. Just as in other metric-based scheduling models, we find the distingtion between static [41] and dynamic scheduling during runtime so we can deal with traffic bursts.
A variety of resource abstractions (see Figure 5) are applicable also to scheduling as part of the network traffic, such as demonstrated by [42] to optimize traffic in virtual clusters. Scheduling across on multiple layers is especially of benefit for networking [43] to minimize across different tiers. Scheduling of platforms such as Hadoop offers naturally advantages when networking is integrated [44]. Having access to lower level infrastructure such as offered by OpenStack presents opportunities to include Network Function Virtualization (NVF) [45]

Table 3 shows examples for network aware scheduling algorithms in cloud computing.

4.2.3 Cost Aware Scheduling

Cost in clouds arise for 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 in regards to on-premise compute and data centers. Costs for such centers include hardware operation cost 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 need for data, speed, and reliability as part of their cost. Other options such as renewable energy use of the data center in case of energy aware customers may also play a role.

Cost aware scheduling has been applied to virtual machines [46], tasks [47, 48], workflows [49, 50], as well as high-throughput [51] computing. Revenue maximization [52] has not only been applied to metrics such as latency [53], but is also useful via advanced dynamic Voltage and Frequency Scaling (DVFS) [54, 23] due do reducing the high energy costs with little performance reduction. This also could be achieved through delayed execution [55] or relaxation of deadlines [56]. Other strategies include the introduction of penalties as part of SLA [57]. Typical resource utilization such as optimizing processor sharing [58] data placements [58], have known to decrease cost. Obviously also dynamic dynamic adaptations at runtime allow reduction of cost [59]

Table 4 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. This is often motivated to fulfill deadlines [60].

Besides virtual machine and task scheduling it is also important to integrate data-aware scheduling to reduce access time to the data [61]. As already mentioned previously, historical data [62] or proxies [63] about execution times help designing time-aware scheduling algorithms.

We find algorithms that integrate deadline constraint [64], completion time [65] with fairness [66], low downtime to improve time for execution [67], and delay bounds [68].

Table 5 presents a comparison of time aware scheduling algorithms.

4.2.5 Reliability Aware Scheduling

Users and providers need the guarantee of reliability. Thus many scheduling efforts integrate how to increase reliability. Strategies such as replication of data and compute services are common practice. Obviously this comes often at a price and increased cost may occur when reliability is concerned. The distributed nature of clouds make it a formidable challenge to offer reliability. However at the same time while providing large scale data centers to offer cloud services with highly specialized operating staff and abilities to replicate and migrate workloads to other services increases reliability when compared to on-premise data centers due to larger efficiency of the cloud data centers in regards to overall cost for its users.

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

This includes reliability assessment models [69], integration of communication and networks [70], increase of resource availability [71]. Trade offs between different scheduling metrics such as energy and reliability have also been studied [72].

A comparison of reliability and scheduling is given in Table 6.

4.2.6 Security based Scheduling

As mentioned earlier security as a key aspect cloud users and providers require in order for cloud infrastructure to be useful for many applications.

Virtual machine scheduling requires the need for isolation, that can be controlled by security policies [73]. Isolation can also apply to the incoming and outgoing data [74, 75]. Risks occurring by inspecting the connections among VMs [76] can be analyzed and integrated in scheduling strategies. To enable trust between components in the cloud background key exchanges have been proposed [77] Multi-objectives (possibly contradictory) need to be also considered. Most often it includes cost vs. security scheduling frameworks [75, 78, 79]. 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 devices have been considered [80].

Security based scheduling algorithms are presented (see Table 7).

4.3 Heuristic based Scheduling

Heuristic methods help to design efficient algorithm to fulfill the users application requirements. We provide here a small sample of different heuristics as found in literature. This includes particle swarm optimization [81], multi-objective genetic algorithm-based [82, 83], colony optimization with swarm intelligence [84], bee colony [85], artificial neural networks [86], simulated annealing [**?**], game-theory [83], and Game theory by minimizing the Pareto dominance and makespan [87]. Other heuristics utilize classical models such as using the critical path in multi-phase scheduling algorithms [88]. Besides virtual machines we often also find workflows to be the scheduling unit in heuristics [89].
A comparison of heuristic methods based scheduling algorithm is done in 8.

4.4 HPC and Cloud Computing Scheduling

Next we review scheduling aspects 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 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. These are specialized high-performance super-computing systems that are offered 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 industry we for example find computational fluid dynamics, and modeling of biochemical processes as one of its user communities. Example offerings for HPC in AWS [90], Azure [91], Google [92], but also other less prominent clouds such as Penguin Computing HPC in the cloud [93], and SabalCore [94].

Cloud Bursting of On Premise HPC tasks. The HPC systems are often over-utilized and thus the situation of resource starvation is to be considered. For this reason many batch queuing system allow 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 [**?**, 95]. Example for the integration in prominent HPC scheduling includes Slurm [96], Univa Grid Engine [97], PBSpro [98], LSF [99], Moab [100].

Container Isolation. Due to the usage of queuing systems it is also possible to provide in part an improved container security 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 [**?**].

On Demand Platforms. Resource starvation in academic cloud and super computing centers motivate also the ability to run platforms that would typically run also in the cloud but provide a cheaper alternative if run locally in the existing HPC centers. A good example is Hadoop that can be run through myhadoop [101] in HPC centers [102].

Bare Metal Provisioning. In other cases it may be better to provide bare metal provisioning capabilities in case the existing platform or cloud abstraction may not be sufficient. Academic efforts such as Futuregrid [5] now followed by Comet [7] and Chaleleoncloud [103] are good examples for it. Commercial efforts in this regard include OpenStack Ironic [104], IBM [105], AWS [106] and Rackspace [107].

4.5 Workflow Scheduling Frameworks

In the previous sections we already pointed out several workflow related scheduling algorithms while using specific metrics to conduct the scheduling. In addition we can integrate virtual machines, containers, and tasks.
Workflow schedulers are often distinguished by DAG [108, 109, 110] and non-DAG scheduling while some support both [111, 112, 113]. Even prior to

the official FaaS frameworks existed that focused on the execution of functions [114]. Scientific applications such as bio-informatics have introduced not only workflow systems, but also promoted graphical workflow design tools to create dependency graphs the are executed by workflow schedulers [115, 116].
In addition to our findings, in [117] workflows are also organized by design, scheduling and data movement abilities.
It is often an overlooked fact that existing HPC batch queuing systems contain features for job dependency management. In many cases these features can be used to accommodate the users needs for scheduling workflows onto the same hardware or in some cases clusters that are managed through the same queuing system [99, 100, 118, 98].

4.6 Scheduling in Public Cloud Providers

Next, we compare scheduling methods and needs offered in public cloud service providers are presented. 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 aspects naturally influence the scheduling efforts which need to be addressed by the provider. In many cases we do not find some of the information on how such scheduling is conducted due to security and company secrets.
However we find metrics that users can utilize to formulate their own 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 [119]. Cost also motivates users to suspend 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 shutdown 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 [120, 121, 122, 123]. Such schedulers can integrate functions, data and compute instances. More sophisticated schedulers can switch workloads between cloud data centers [124].
In [122] 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 the scale up and down the services based on such loads and allocate dynamically through a scheduler resources to fulfill this demand.
Other providers have focused on making use of multicloud virtual machine placement possible while offering optimization strategies for workflows [125] including detailed analysis of cost metrics [126]
Other efforts such as [4, 5] 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 [7] the re-provisioning is even done with the help of a traditional batch queuing system.
Table 10 depicts examples as used in public cloud providers.

4.7 Scheduling in Container Frameworks

Container schedulers provide mechanisms to fine-tune the selection processes of containers onto distributed resources [127, 128]. Typically a default default scheduling policy is provided. Policies might place new services on hosts with the fewest currently active services.
Based on our Y-diagram we need to distinguish 2 different services. First, scheduling on the same server and second scheduling on a number of servers that are treated typically as one abstract 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 aspects. We focus next on the distributed frameworks in more detail we

focus on Docker Swarm, Kubernetes, Singularity, and Mesos.

4.7.0.1 Docker Swarm. Docker Swarm is a clustering and scheduling tool for Docker containers [129] 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 [130].

4.7.0.2 Kubernetes. Kubernetes is an open-source orchestrator developed by Google for automating container management and deployment [13]. The basic deploy-able object is a Pod which 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 policy constraints to expose availability, performance and capacity. Auto-scaling, load balancing and secrets managements are also provided by Kubernetes.

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

4.7.0.4 Mesos. Mesos [131, 132] 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. It's goal is improved utilization, respond quickly to workload changes, by maintaining system's capability in terms of scalability and robustness.

4.7.0.5 Community Efforts. Many community efforts to improve container scheduling are conducted. This includes for example the use of genetic algorithm [133], container and host selection policies for cloud deployment models [134] with SLA's, the characterization of applications [135] scheduling of virtual clusters [136], and migration [137], 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 [138]. .
Table 11 shows the comparison of existing work related to scheduling.

4.8 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 [139]. For example a minimum execution time per request is five minutes provided by AWS lambda and azure functions [140]. It supports manage user defined functions on highly available infrastructure in an unified manner [141]. This also allows the scheduling of workflows comprised out of functions [142]. In [143] 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 has 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.
A number of 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 [144], Google cloud functions [145], Azure Function [146]
Other open source frameworks such as Apache OpenWhisk [147] allow users to install FaaS services on their own infrastructure.

4.9 Scheduling among Distributed Resources and Providers

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

Such efforts contain Eagle, a hybrid data center scheduler for data-parallel programs [148]; Hopper [149], a job scheduler that trades off existing and speculated job scheduling decisions; Tetris [150], a cluster scheduler that aims to match multi-resource task requirements with resource availability; Fawkes [151] a multi-cluster systems for map-reduce; Omega [152] with optimistic concurrency control; OurGrid [153, 154] for worldwide computing platform with isolated environments; Sparrow [155] and fine-grained task scheduling scheduler.
We can also find more prominent schedulers such a contains Apache's Hadoop YARN [156] which acts as a resource management systems 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)

4.10 Service Meshups

To support scheduling across clouds and services, service mashups can be used. This includes long standing efforts such as Cloudmesh, which targets the creation of reproducable 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 [157] which focuses on reproducible environments.

5 Conclusion and Lessons Learned

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

More than VM scheduling. Due to the shift and enhancement of clouds from VM to containers and FaaS, we must consider also new scheduling strategies as motivated by thes new cloud compute offerings.

Integration for data. Big Data and management of data in general motivate the integration of data resources as firs class activity within scheduling.

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

Y-Diagram. Our Y-diagram promotes scheduling across scale and models. This allows to create 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-diagarm, a layered scheduling approach seems appropriate.

Analytics Services. While FaaS provide 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 as analytical calculations.

Security and Privacy. We need to deal more stringently with security and privacy as part of our scheduling needs which contrasts traditional HPC scheduling efforts.

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

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

Edge Computing and Fog computing. Due to the increased edge and Fog computing computational powers available. significant activities can also be conducted on edge devices. Billions of cellphones today already conduct a significant amount of computation thus scheduling must balance between activities that can take place on the edge (Fog) or needs to be conducted in the cloud.

In this paper, we have surveyed the many important aspects of scheduling problems in cloud computing. After introducing the needed terminology, we presented a comprehensive taxonomy of the different cloud scheduling approaches and issues. A layered and phased scheduling model is presented that differentiates the concerns between infrastructure, platform, servers and function as a service models. 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 available. Please inform us 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 our taxonomy. Please send us your reference in format while adding in the abstract what your paper provides and how it fits in our scheduling taxonomy. The contribution can be send either to `laszewski@gmail.com`, or via a github pull request at `https://github.com/cloudmesh/bib/blob/master/cloud-scheduling.bib`.

References

[1] P. Mell, T. Grance, et al., The nist definition of cloud computing.

[2] nist big data interoperability framework: Volume 6, reference architecture.

[3] G. von Laszewski, W. L. Chang, NIST Big Data Interoperability Framework: Volume 8, Reference Architecture Interfaces (Dec. 2018).
URL `https://github.com/cloudmesh-community/nist/blob/master/docs/nistvol8-2.pdf`

[4] G. C. Fox, G. von Laszewski, J. Diaz, K. Keahey, J. Fortes, R. Figueiredo, S. Smallen, W. Smith, A. Grimshaw, FutureGrid - a reconfigurable testbed for Cloud, HPC and Grid Computing, in: Contemporary HPC Architectures, draft Edition, 2012.
URL `https://laszewski.github.io/papers/vonLaszewski-12-fg-bookchapter.pdf`

[5] G. C. Fox, G. von Laszewski, J. Diaz, K. Keahey, J. Fortes, R. Figueiredo, S. Smallen, W. Smith, A. Grimshaw, Futuregrid: a reconfigurable testbed for cloud, hpc, and grid computing, in: Contemporary High Performance Computing, Chapman and Hall/CRC, 2013, pp. 603–635.

[6] G. Von Laszewski, F. Wang, H. Lee, H. Chen, G. C. Fox, Accessing multiple clouds with cloudmesh, in: Proceedings of the 2014 ACM international workshop on Software-defined ecosystems, ACM, 2014, pp. 21–28.

[7] S. M. Strande, H. Cai, T. Cooper, K. Flammer, C. Irving, G. von Laszewski, A. Majumdar, D. Mishin, P. Papadopoulos, W. Pfeiffer, R. S. Sinkovits, M. Tatineni, R. Wagner, F. Wang, N. Wilkins-Diehr, N. Wolter, M. L. Norman, 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, PEARC17, ACM, New York, NY, USA, 2017, pp. 38:1–38:7. `doi:10.1145/3093338.3093383`.
URL `http://doi.acm.org/10.1145/3093338.3093383`

[8] J. Diaz, G. von Laszewski, F. Wang, A. J. Younge, G. C. Fox, 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, 2011, pp. 560–564. `doi:10.1109/CloudCom.2011.85`.
URL `https://laszewski.github.io/papers/vonLaszewski-draft-11-imagerepo.pdf`

[9] G. von Laszewski, G. C. Fox, G. von Laszewski, G. C. Fox, F. Team, Dynamic Provisioned

Experiments in FutureGrid, 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom2010), Indianapolis, IN (12/1/2010 2010).
URL http://grids.ucs.indiana.edu/ptliupages/presentations/vonLaszewski-10-FG-proj-management.pdf

[10] G. von Laszewski, H. Lee, J. Diaz, F. Wang, K. Tanaka, S. Karavinkoppa, G. C. Fox, T. Furlani, 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, FederatedClouds '12, ACM, New York, NY, USA, 2012, pp. 25–32. doi:10.1145/2378975.2378982.
URL http://doi.acm.org/10.1145/2378975.2378982

[11] G. von Laszewski, Towards a cloud scheduling taxonomy (Dec. 2018).
URL https://github.com/cyberaide/paper-cloud-scheduling-whitepaper/tree/master

[12] Kubernates (2018).
URL https://kubernetes.io/docs/concepts/

[13] Kubernates (2018).
URL https://github.com/kubernetes/kubernetes

[14] K. Vivekanandan, et al., A study on scheduling in grid environment, in: International Journal on Computer Science and Engineering (IJCSE), Citeseer, 2011.

[15] B. Jennings, R. Stadler, Resource management in clouds: Survey and research challenges, Journal of Network and Systems Management 23 (3) (2015) 567–619.

[16] G. Keller, M. Tighe, H. Lutfiyya, M. Bauer, A hierarchical, topology-aware approach to dynamic data centre management, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, IEEE, 2014, pp. 1–7.

[17] M. Tighe, G. Keller, M. Bauer, H. Lutfiyya, A distributed approach to dynamic vm management, in: Network and Service Management (CNSM), 2013 9th International Conference on, IEEE, 2013, pp. 166–170.

[18] M. Tighe, M. Bauer, Integrating cloud application autoscaling with dynamic vm allocation, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, IEEE, 2014, pp. 1–9.

[19] G. Sun, D. Liao, D. Zhao, Z. Xu, H. Yu, Live migration for multiple correlated virtual machines in cloud-based data centers, IEEE Transactions on Services Computing.

[20] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, I. M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Generation Computer Systems 28 (2) (2012) 358–367.

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

[22] L. Wang, G. von Laszewski, J. Dayal, F. Wang, 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, 2010, pp. 368–377. doi:10.1109/CCGRID.2010.19.
URL https://doi.org/10.1109/CCGRID.2010.19

[23] R. N. Calheiros, R. Buyya, Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through dvfs, in: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, IEEE, 2014, pp. 342–349.

[24] N. Bessis, S. Sotiriadis, F. Pop, V. Cristea, Using a novel message-exchanging optimization (meo) model to reduce energy consumption in distributed systems, Simulation Modelling Practice and Theory 39 (2013) 104–120.

[25] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, J. Li, 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.

[26] H. Duan, C. Chen, G. Min, Y. Wu, Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems, Future Generation Computer Systems.

[27] D.-M. Bui, Y. Yoon, E.-N. Huh, S. Jun, S. Lee, Energy efficiency for cloud computing system based on predictive optimization, Journal of Parallel and Distributed Computing.

[28] W. Zhu, Y. Zhuang, L. Zhang, A three-dimensional virtual resource scheduling method for energy saving in cloud computing, Future Generation Computer Systems 69 (2017) 66–74.

[29] A. Beloglazov, R. Buyya, 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, 2010, pp. 826–831.

[30] A. Quarati, A. Clematis, A. Galizia, D. D Agostino, Hybrid clouds brokering: business opportunities, qos and energy-saving issues, Simulation Modelling Practice and Theory 39 (2013) 121–134.

[31] S. K. Garg, C. S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers, Journal of Parallel and Distributed Computing 71 (6) (2011) 732–749.

[32] K. Gai, M. Qiu, H. Zhao, L. Tao, Z. Zong, Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing, Journal of Network and Computer Applications 59 (2016) 46–54.

[33] Y. Ding, X. Qin, L. Liu, T. Wang, Energy efficient scheduling of virtual machines in cloud with deadline constraint, Future Generation Computer Systems 50 (2015) 62–74.

[34] N. Kim, J. Cho, E. Seo, Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems, Future Generation Computer Systems 32 (2014) 128–137.

[35] T. Van Do, C. Rotter, Comparison of scheduling schemes for on-demand iaas requests, Journal of Systems and Software 85 (6) (2012) 1400–1408.

[36] J. Hu, J. Gu, G. Sun, T. Zhao, A scheduling strategy on load balancing of virtual machine resources in cloud computing environment, in: Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, IEEE, 2010, pp. 89–96.

[37] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, Energy-efficient resource allocation and provisioning framework for cloud data centers, IEEE Transactions on Network and Service Management 12 (3) (2015) 377–391.

[38] Akamai, Web Page.
URL `https://www.akamai.com/`

[39] D. Breitgand, A. Epstein, Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 2861–2865.

[40] S. Rampersaud, D. Grosu, Sharing-aware online virtual machine packing in heterogeneous resource clouds, IEEE Transactions on Parallel and Distributed Systems.

[41] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, E. Silvera, A stable network-aware vm placement for cloud systems, in: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, IEEE, 2012, pp. 498–506.

[42] R. Yu, G. Xue, X. Zhang, D. Li, Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers, in: IEEE INFOCOM, 2017.

[43] J. Bi, H. Yuan, M. Tie, W. Tan, Sla-based optimisation of virtualised resource for multi-tier web applications in cloud data centres, Enterprise Information Systems 9 (7) (2015) 743–767.

[44] P. Kondikoppa, C.-H. Chiu, C. Cui, L. Xue, S.-J. Park, 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, 2012, pp. 39–44.

[45] F. Lucrezia, G. Marchetto, F. Risso, V. Vercellone, Introducing network-aware scheduling capabilities in openstack, in: Network Softwarization (NetSoft), 2015 1st IEEE Conference on, IEEE, 2015, pp. 1–5.

[46] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, J. Li, Ttsa: An effective scheduling approach for delay bounded tasks in hybrid clouds, IEEE transactions on cybernetics 47 (11) (2017) 3658–3668.

[47] H. Yuan, J. Bi, W. Tan, B. H. Li, Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds., IEEE Trans. Automation Science and Engineering 14 (1) (2017) 337–348.

[48] L. Zuo, L. Shu, S. Dong, C. Zhu, T. Hara, A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing, IEEE Access 3 (2015) 2687–2699.

[49] V. Arabnejad, K. Bubendorfer, Cost effective and deadline constrained scientific workflow scheduling for commercial clouds, in: Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on, IEEE, 2015, pp. 106–113.

[50] V. Arabnejad, K. Bubendorfer, B. Ng, 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, 2016, pp. 1188–1195.

[51] H. Yuan, J. Bi, W. Tan, B. H. Li, 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.

[52] H. Yuan, J. Bi, M. Zhou, K. Sedraoui, Warm: Workload-aware multi-application task scheduling for revenue maximization in sdn-based cloud data center, IEEE Access 6 (2018) 645–657.

[53] M. H. Ghahramani, M. Zhou, C. T. Hon, Toward cloud computing qos architecture: Analysis of cloud systems and cloud services, IEEE/CAA Journal of Automatica Sinica 4 (1) (2017) 6–18.

[54] A. J. Younge, G. von Laszewski, L. Wang, S. Lopez-Alarcon, W. Carithers, Efficient resource management for cloud computing environments, in: International Conference on Green Computing, 2010, pp. 357–364. doi:10.1109/GREENCOMP.2010.5598294.

[55] J. Bi, H. Yuan, W. Tan, B. H. Li, Trs: Temporal request scheduling with bounded delay assurance in a green cloud data center, Information Sciences 360 (2016) 57–72.

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

[57] L. Wu, S. K. Garg, R. Buyya, Sla-based admission control for a software-as-a-service provider in cloud computing environments, Journal of Computer and System Sciences 78 (5) (2012) 1280–1299.

[58] Y. C. Lee, C. Wang, A. Y. Zomaya, B. B. Zhou, Profit-driven scheduling for cloud services with data access awareness, Journal of Parallel and Distributed Computing 72 (4) (2012) 591–602.

[59] I. Ari, N. Muhtaroglu, Design and implementation of a cloud computing service for finite element analysis, Advances in Engineering Software 60 (2013) 122–135.

[60] V. Arabnejad, K. Bubendorfer, B. Ng, Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources, Future Generation Computer Systems.

[61] R. Van den Bossche, K. Vanmechelen, J. Broeckhove, Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds, Future Generation Computer Systems 29 (4) (2013) 973–985.

[62] A. Thomas, G. Krishnalal, V. J. Raj, Credit based scheduling algorithm in cloud computing environment, Procedia Computer Science 46 (2015) 913–920.

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

[64] H. Li, H. Zhu, G. Ren, H. Wang, H. Zhang, L. Chen, Energy-aware scheduling of workflow in cloud center with deadline constraint, in: Computational Intelligence and Security (CIS), 2016 12th International Conference on, IEEE, 2016, pp. 415–418.

[65] B. Xu, C. Zhao, E. Hu, B. Hu, Job scheduling algorithm based on berger model in cloud environment, Advances in Engineering Software 42 (7) (2011) 419–425.

[66] G. Jasso, The theory of the distributive-justice force in human affairs: Analyzing the three central questions, in: Sociological theories in progress: New formulations, Sage, 1989.

[67] M. E. Frîncu, Scheduling highly available applications on cloud environments, Future Generation Computer Systems 32 (2014) 138–153.

[68] H. Yuan, J. Bi, M. Zhou, A. C. Ammari, Time-aware multi-application task scheduling with guaranteed delay constraints in green data center, IEEE Transactions on Automation Science and Engineering 15 (3).

[69] S. Malik, F. Huet, D. Caromel, Reliability aware scheduling in cloud computing, in: Internet Technology And Secured Transactions, 2012 International Conference for, IEEE, 2012, pp. 194–200.

[70] W. Jing, Y. Liu, H. Shao, Reliability-aware dag scheduling with primary-backup in cloud computing, International Journal of Computer Applications in Technology 52 (1) (2015) 86–93.

[71] M. S. A. Latiff, S. H. H. Madni, M. Abdullahi, et al., Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm, Neural Computing and Applications (2016) 1–15.

[72] X. Tang, W. Tan, Energy-efficient reliability-aware scheduling algorithm on heterogeneous systems, Scientific Programming 2016 (2016) 14.

[73] Z. Afoulki, A. Bousquet, J. Rouzaud-Cornabas, A security-aware scheduler for virtual machines on iaas clouds, Report 2011.

[74] B. K. Chejerla, S. K. Madria, Qos guaranteeing robust scheduling in attack resilient cloud integrated cyber physical system, Future Generation Computer Systems.

[75] R. Kashyap, D. P. Vidyarthi, Security-aware real-time scheduling for hypervisors, in: Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on, IEEE, 2014, pp. 1520–1527.

[76] S. Shetty, X. Yuchi, M. Song, Security-aware virtual machine placement in cloud data center, in: Moving Target Defense for Distributed Systems, Springer, 2016, pp. 13–24.

[77] C. Liu, X. Zhang, C. Yang, J. Chen, Ccbke session key negotiation for fast and secure scheduling of scientific applications in cloud computing, Future Generation Computer Systems 29 (5) (2013) 1300–1308.

[78] L. Zeng, B. Veeravalli, X. Li, Saba: A security-aware and budget-aware workflow scheduling strategy in clouds, Journal of parallel and Distributed computing 75 (2015) 141–151.

[79] W. Wang, G. Zeng, D. Tang, J. Yao, Cloud-dls: Dynamic trusted scheduling for cloud computing, Expert Systems with Applications 39 (3) (2012) 2321–2329.

[80] I. Bilogrevic, M. Jadliwala, P. Kumar, S. S. Walia, J.-P. Hubaux, I. Aad, V. Niemi, Meetings through the cloud: privacy-preserving scheduling on mobile devices, Journal of Systems and Software 84 (11) (2011) 1910–1927.

[81] S. Pandey, L. Wu, S. M. Guru, R. Buyya, 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, 2010, pp. 400–407.

[82] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, A. Y. Zomaya, D. Tuyttens, A

parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, Journal of Parallel and Distributed Computing 71 (11) (2011) 1497–1508.

[83] J. Gasior, F. Seredyński, Metaheuristic approaches to multiobjective job scheduling in cloud computing systems, in: Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on, IEEE, 2016, pp. 222–229.

[84] C. Mateos, E. Pacini, C. G. Garino, An aco-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments, Advances in Engineering Software 56 (2013) 38–50.

[85] D. B. LD, P. V. Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments, Applied Soft Computing 13 (5) (2013) 2292–2303.

[86] G. Kousiouris, T. Cucinotta, T. Varvarigou, 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.

[87] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, J. Wang, Cost-efficient task scheduling for executing large programs in the cloud, Parallel Computing 39 (4) (2013) 177–188.

[88] S. Abrishami, M. Naghibzadeh, D. H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Generation Computer Systems 29 (1) (2013) 158–169.

[89] K. Bousselmi, Z. Brahmi, M. M. Gammoudi, Qos-aware scheduling of workflows in cloud computing environments, in: Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on, IEEE, 2016, pp. 737–745.

[90] Amazon, Job scheduling.
URL https://docs.aws.amazon.com/batch/latest/userguide/job_scheduling.html

[91] Microsoft Azure, Big compute.
URL https://azure.microsoft.com/en-us/solutions/big-compute/

[92] Google, Hpc perfomance computing, Web Page.
URL https://cloud.google.com/solutions/hpc/

[93] Pod hpc cloud (2019).
URL https://www.penguincomputing.com/pod-hpc-cloud

[94] Sabalcore (2019).
URL http://www.sabalcore.com/

[95] Bursting hpc (2019).
URL https://insidehpc.com/2018/04/universities-step-cloud-bursting/

[96] Slurm workload manager (2018).
URL https://slurm.schedmd.com/

[97] Univa (2018).
URL http://www.univa.com/

[98] PBS, Pbs pro user guide, Online Manual.
URL https://www.pbsworks.com/pdfs/PBSUserGuide14.2.pdf

[99] IBM, Lsf job dependencies, Manual.
URL https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/job_dep_sched.html

[100] Adaptive Computing, Moab job dependencies.
URL http://docs.adaptivecomputing.com/mwm/7-2-8/Content/topics/jobAdministration/jobdependencies.html

[101] S. Krishnan, M. Tatineni, C. Baru, myhadoop-hadoop-on-demand on traditional hpc resources, San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego.

[102] San diego supercomputer center (2019).
URL https://www.sdsc.edu/support/user_guides/tutorials/hadoop.html

[103] Chameleoncloud (2019).
URL https://www.chameleoncloud.org/

[104] Openstack ironic (2019).
URL https://docs.openstack.org/ironic/latest/

[105] Ibm bare metal (2019).
URL https://www.ibm.com/cloud/bare-metal-servers

[106] Aws (2019).
URL https://aws.amazon.com/about-aws/whats-new/2019/02/introducing-five-new-amazon-ec2-bare-metal-instances/

[107] Rackspace (2019).
URL https://www.rackspace.com/en-us/cloud/servers/onmetal

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

[109] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny, Pegasus: Mapping scientific workflows onto the grid, in: Grid Computing, Springer, 2004, pp. 11–20.

[110] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the condor experience, Concurrency and computation: practice and experience 17 (2-4) (2005) 323–356.

[111] G. von Laszewski, M. Hategan, Workflow concepts of the java cog kit, J. Grid Comput. 3 (3-4) (2005) 239–258. doi:10.1007/s10723-005-9013-5.
URL https://doi.org/10.1007/s10723-005-9013-5

[112] G. von Laszewski, I. Foster, J. Gawor, Cog kits: A bridge between commodity distributed computing and high-performance grids, in: Proceedings of the ACM 2000 Conference on Java Grande, JAVA '00, ACM, New York, NY, USA, 2000, pp. 97–106. doi:10.1145/337449.337491.
URL http://doi.acm.org/10.1145/337449.337491

[113] G. von Laszewski, M. Hategan, D. Kodeboyina, Grid workflow with the java cog kit, in: I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields (Eds.), Workflows for E-science: Scientific Workflows for Grids, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
URL https://laszewski.github.io/papers/vonLaszewski-workflow-book.pdf

[114] G. von Laszewski, J. Gawor, C. J. Peña, I. Foster, 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, 2002, pp. 333–.
URL http://dl.acm.org/citation.cfm?id=822086.823347

[115] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, et al., Taverna: a tool for the composition and enactment of bioinformatics workflows, Bioinformatics 20 (17) (2004) 3045–3054.

[116] W. Tan, P. Missier, I. Foster, R. Madduri, D. De Roure, C. Goble, A comparison of using taverna and bpel in building scientific workflows: the case of cagrid, Concurrency and Computation: Practice and Experience 22 (9) (2010) 1098–1117.

[117] J. Yu, R. Buyya, A taxonomy of scientific workflow systems for grid computing, ACM Sigmod Record 34 (3) (2005) 44–49.

[118] Univa, Grid engine manual, Online Manual.
URL http://www.univa.com/resources/files/univa_user_guide_univa__grid_engine_854.pdf

[119] Amazon ec2 (2015).
URL https://aws.amazon.com/ec2/

[120] Aws instance scheduler (2019).
URL https://aws.amazon.com/solutions/instance-scheduler/

[121] Azure scheduler (2019).
URL https://azure.microsoft.com/en-in/services/scheduler/

[122] Rackspace (2016).
URL https://www.rackspace.com/en-in/why-rackspace

[123] Google app engine (2018).
URL https://cloud.google.com/appengine/docs/

[124] Microsoft azure (2014).
URL https://azure.microsoft.com/en-in/services/scheduler/

[125] Cloud sigma (2016).
URL https://www.cloudsigma.com/features/

[126] Cloud metrics (2019).
URL https://www.rightscale.com/about-cloud-management/cloud-cost-optimization/cloud-pricing-comparison

[127] Containers (2018).
URL https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-scheduling-and-orchestration

[128] M. D. de Assuncao, A. da Silva Veith, R. Buyya, 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.

[129] Docker swarm engine (2018).
URL https://docs.docker.com/engine/swarm/manage-nodes

[130] Docker swarm (2018).
URL https://semaphoreci.com/community/tutorials/scheduling-services-on-a-docker-swarm-mode-cluster

[131] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, I. Stoica, Mesos: A platform for fine-grained resource sharing in the data center., in: NSDI, Vol. 11, 2011, pp. 22–22.

[132] Mesos (2018).
URL http://mesos.apache.org/getting-started/

[133] C. Guerrero, I. Lera, C. Juiz, Genetic algorithm for multi-objective optimization of container allocation in cloud architecture, Journal of Grid Computing 16 (1) (2018) 113–135.

[134] W. A. Hanafy, A. E. Mohamed, S. A. Salem, Novel selection policies for container-based cloud deployment models, in: Computer Engineering Conference (ICENCO), 2017 13th International, IEEE, 2017, pp. 237–242.

[135] V. Medel, C. Tolón, U. Arronategui, R. Tolosana-Calasanz, J. Á. Bañares, O. F. Rana, Client-side scheduling based on application characterization on kubernetes, in: International Conference on the Economics of Grids, Clouds, Systems, and Services, Springer, 2017, pp. 162–176.

[136] P. Dziurzanski, L. Indrusiak, Value-based allocation of docker containers, in: Proceedings of the 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, York.

[137] Flocker, Flocker (2018).
URL https://docs.clusterhq.com/en/1.0.3/

[138] Nomad (2018).
URL https://www.nomadproject.io/docs/internals/scheduling.html

[139] G. von Laszewski, G. C. Fox, F. Wang, Cloud computing (Apr. 2019).
URL https://github.com/cloudmesh-community/book/blob/master/vonLaszewski-cloud.epub

[140] Serverlesscomputing (2018).
URL https://www.apriorit.com/dev-blog/551-serverless-computing

[141] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, R. Prodan, A serverless real-time data analytics platform for edge computing, IEEE Internet Computing 21 (4) (2017) 64–71.

[142] O. Alqaryouti, N. Siyam, Serverless computing and scheduling tasks on cloud: A review, American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS) 40 (1) (2018) 235–247.

[143] G. C. Fox, V. Ishakian, V. Muthusamy, A. Slominski, Status of serverless computing and function-as-a-service (faas) in industry and research, arXiv preprint arXiv:1708.08028.

[144] Aws lambda (2018).
URL https://aws.amazon.com/lambda/

[145] Google, Google cloud functions (2018).
URL https://cloud.google.com/functions/

[146] Azure functions (Nov. 2018).
URL https://azure.microsoft.com/en-in/services/functions/

[147] Openwhisk (2018).
URL https://openwhisk.apache.org/

[148] P. Delgado, D. Didona, F. Dinu, W. Zwaenepoel, Job-aware scheduling in eagle: Divide and stick to your probes, in: Proceedings of the Seventh ACM Symposium on Cloud Computing, ACM, 2016, pp. 497–509.

[149] X. Ren, G. Ananthanarayanan, A. Wierman, M. Yu, Hopper: Decentralized speculation-aware cluster scheduling at scale, in: ACM SIGCOMM Computer Communication Review, Vol. 45, ACM, 2015, pp. 379–392.

[150] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, A. Akella, Multi-resource packing for cluster schedulers, ACM SIGCOMM Computer Communication Review 44 (4) (2015) 455–466.

[151] B. Ghit, N. Yigitbasi, A. Iosup, D. Epema, Balanced resource allocations across multiple dynamic mapreduce clusters, in: ACM SIGMETRICS Performance Evaluation Review, Vol. 42, ACM, 2014, pp. 329–341.

[152] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, J. Wilkes, Omega: flexible, scalable schedulers for large compute clusters, in: Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 351–364.

[153] N. Andrade, W. Cirne, F. Brasileiro, P. Roisenberg, Ourgrid: An approach to easily assemble grids with equitable resource sharing, in: Workshop on Job Scheduling Strategies for Parallel Processing, Springer, 2003, pp. 61–86.

[154] W. Cirne, F. Brasileiro, N. Andrade, L. B. Costa, A. Andrade, R. Novaes, M. Mowbray, Labs of the world, unite!!!, Journal of Grid Computing 4 (3) (2006) 225–246.

[155] K. Ousterhout, P. Wendell, M. Zaharia, I. Stoica, Sparrow: distributed, low latency scheduling, in: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, ACM, 2013, pp. 69–84.

[156] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al., Apache hadoop yarn: Yet another resource negotiator, in: Proceedings of the 4th annual Symposium on Cloud Computing, ACM, 2013, p. 5.

[157] Hashicorp, Terraform, Web Page.
URL https://www.terraform.io/

[158] J. O. Gutierrez-Garcia, K. M. Sim, A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling, Future Generation Computer Systems 29 (7) (2013) 1682–1699.

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

[160] LSF (2018). [link].
URL https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_foundations/lsf_introduction_to.html

[161] OpenPBS: The Portable Batch System Software (2018).
URL https://www.pbspro.org/

[162] R. Henderson, Job Scheduling Under the Portable Batch System, in: Proceedings of the 1995 Workshop on Job Scheduling Strategies for Parallel Processing, Santa Barbara, CA, USA, 1995.

[163] Moab (2019).
URL https://www.adaptivecomputing.com/moab-hpc-basic-edition/

[164] Openstack (2018).
URL https://www.openstack.org/

[165] OpenNebula, Opennebula (2018).
URL https://opennebula.org/

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

List of Tables

Table 1: Comparison of Dynamic Scheduling Algorithms

Author	**Basis**	**Advantages**	**Disadvantages**	**Scheduling techniques**	**Experimental Scale**	**Experimental Parameters**	**Taxonomy Classification**
Sun et al. [19]	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	Energy based, energy source
Tighe et al. [18]	Auto scaling algorithm alongside a dynamic VM allocation algorithm	Reduce a number of migrations	No optimization criteria	Rule based heuristic	DC Sim	Power, SLA, Migrations	Energy based, VM migration
Keller et al. [16]	Management of data center resources to reduce the management scope	Reducing the overhead in the data centre management network	High complexity	Greedy algorithm	DCSim	Power, number of migrations, average number of racks, active hosts	Energy based
Tighe et al. [17]	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-based, PUE
Tordsson et al. [20]	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	Compute-resource based, VM based
Younge et al. [54]	Power aware	reduces cost	slight reduction in performance	heuristic	on premise cloud	power consumption	Energy-based, VM-based

Table 2: Comparison of Energy aware VM Placement based Scheduling Algorithms

Author	Basis	Advantages	Disadvantages	Scheduling techniques	Experimental Scale	Experimental Parameters	Taxonomy Classification
Younge et al. [54]	Power aware	reduces cost	slight reduction in performance	heuristic	on premise cloud	power consumption	Energy-based, VM-based
Bi et al. [25]	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 based, Energy source
Zhu et al. [28]	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 based, Energy source
Duan et al. [26]	Scheduling of VM machines	Improved the CPU load prediction	No optimization	Ant colony optimization	CloudSim	Energy Consumption	Energy based, Compute resource based, VM
Li et al. [64]	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 based, Energy source, Compute resource based, workflow
Keke et al. [32]	Cloudlets for energy reduction	Reduced energy consumption	No time consideration	FCFS scheduling policy	DECM-Sim	Energy consumption	Energy based Green IT
Bui et al. [27]	Balance between energy efficiency and quality of service	Low complexity	Ignored cost, scalability	Greedy first fit algorithm	Simulated Environment	Energy, Memory, CPU	Energy based - Energy source
Dabbagh et al. [37]	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
Ding et al. [33]	Dynamic VMs scheduling	Increased Processing Capacity	Ignored VM migration, Power penalties of status transitions of processor	FCFS	Simulated environment	Deadline, Energy consumption	Energy source, PUE
Calheiros et al. [23]	Intelligent scheduling combined DVFS capability	Improved energy efficiency	Ignored Network and Storage energy consumption	Rank method	Cloud Sim	Energy consumption	Energy source, Compute resource based, Job based
Kim et al. [34]	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	energy source, VM based
Quarati et al. [30]	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	Energy source and Compute resource based ,VM
Bessis et al. [24]	Improving communication for Distributed systems at the time of scheduling	Improved system performance	High complexity	Graph theory concepts	SIMIC	makespan, latency times	Energy based, PUE
Van Do et al. in [35]	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	Computer resource based, VM
Garg et al. [31]	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	PUE, Job based
Beloglazov et al. [29]	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	Energ based Energy source

Table 3: Comparison of Network aware VM Placement based Scheduling Algorithms

Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters	Taxonomy Classification
Yu et al. [42]	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	VM based
Bi et al. [43] [55]	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, VM based
Rampersaud et al. [40]	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
Lucrezia et al. [45]	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	Compute resource-VM based
Biran et al. [41]	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	Compute resource VM based
Kondikoppa et al. [44]	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 workflow based

Table 4: Comparison of Cost based Scheduling Algorithms

Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters	Taxonomy Classification
Yuan et al. [47] [46]	Emphasizing profit maximization	handles service delay bound	High complexity	PSO and SA	simulation environment	Revenue	Compute resource, Data based, task, latency, VM
Bi et al. [43] [55]	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, task based
Arabnejad et al. arabnejad2015cost	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, workflow based
Zuo et al. [48]	Multi-objective Task Scheduling	Improved performance	Ignored energy consumption	Ant colony optimization	CloudSim	Cost, makespan, deadline violation rate	compute resource based, task based
Ari et al. [59]	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	compute-resource based, VM based
Wu et al. [57]	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	compute resource based, VM
Lee et al. [58]	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	Data based , access type

Table 5: Comparison of Time based Scheduling Algorithms

Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters	Taxonomy Classification
Yuan et al. [68]	Task scheduling in green data centers	Investigated temporal variations	Ignored energy consumption and cost	PSO and SA	Simulated Environment	Delay bound and time	Compute-resource based, task
Arabnejad et al. [60]	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 workflow
Thomas et al. [62]	Task length aware scheduling	Lesser makespan and increased resource utilization	No comparison with existing algorithm	Min-min	CloudSim	Makespan	Compute-resource based, Workflow
Frincu [67]	A priory scheduling and searching for an optimal allocation of components on nodes in order to ensure a homogeneous spread of component types on every node.	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, VM
Erdil [63]	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, VM
van den Bossche et al. in [61]	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, Data based, Task-based
Xu et al. [65]	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 , task

Table 6: Comparison of Reliability based Scheduling Algorithms

Author	**Basis**	**Advantages**	**Disadvantages**	**scheduling techniques**	**Experimental Environments**	**Performance matrices**	**Taxonomy Classification**
Abdulhamid et al. [71]	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	Compute - resource based, Task based
Tang et al. [72]	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	Energy based , energy source
Jing et al. [70]	Model for fault-tolerant aware scheduling	Low complexity	No cost, time optimization	Adaptive secure scheduling algorithm	Simulated environment	Reliability	Compute-resource based, Job based
Malik et al. [69]	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

Table 7: Comparison of Security based Scheduling Algorithms

Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters	Taxonomy Classification
Chejerla et al. [74]	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 and Data based
Shetty et al. [76]	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, VM based
Zeng et al. [78]	Scheduling algorithm for resource utilization	Low complexity	Ignored energy consumption	Clustering and prioritization algorithm	Simulated environment	Makespan and speed up	Compute resource based-workflow
Kashyap et al. [75]	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
Liu et al. [77]	Scheme for security aware scheduling	Reduced the computational load and execution time	No cost optimization involved	Adaptive secure scheduling algorithm	KVM hypervisor	Time unit consumed per computational load	Compute-resource based, Workflow
Wang et al. [79]	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, Task based
Afoulki et al. [73]	Security risk management in a cloud	Less complexity	Consolidation issues while implementing policies	Greedy Algorithm	Simulated environment	VM placement time	Compute-resource based, VM based
Bilogrevic et al. [80]	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	Compute-resource based, VM based

Table 8: Comparison of Heuristic based Scheduling Algorithms

Author	Basis	Advantages	Disadvantages	scheduling techniques	Experimental Scale	Experimental Parameters	Taxonomy Classification
Gasior et al. [83]	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
Bousselm et al. [89]	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, workflow based
Cristian et al. [84]	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, job based
Abrishami et al. [88]	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
Sen et al. [87]	Cost-efficient task-scheduling algorithm using two heuristic strategies	Reduced monetary costs	Ignored security	Heuristic strategies	Numerical experiments	Makespan	Compute resource based, task based
Gutierrez-Garcia et al. [158]	Scheduling of Bag-of-tasks based on allocation times of virtualized cloud resources	Makespan	Ignored cost	Heuristic algorithm	Testbed	Makepan, overhead time	Compute-resource based, Job based
Babu [85]	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, energy based
Kousiouris et al. [86]	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, task based
Mezmaz et al. [82]	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
Torabzadeh et al. [159]	Flowshow job problem	Minimized makespan and mean completion time	Not considered cost	Simulated annealing	Simulated environment	Computation time	Compute resource based, VM based

Table 9: Comparison of Different Batch Resource Management Systems

Framework	**Batch Features**	**Bursting**	**Containers**	**Bare Metal**	**Cloud Platforms**	**Comment**
Load Sharing Facility (LSF) [160]	policy driven, backfill, exclusive and non-exclusive access to compute nodes	IBM Cloud, AWS, Google and Azure	Yes	Yes	? cloud platform	previously OpenLava, IBM Open Source ?
Slurm [**?**]	policy driven, backfill, exclusive and non-exclusive access to compute nodes	Amazon EC2, Google Cloud Platform	Yes	? Bare metal	? cloud platfrom	Open Source, popular
Open Portable Batch System (OpenPBS) [161, 162]	policy driven, backfill, exclusive and non-exclusive access to compute nodes, fault tolerant master	AWS, Azure, Google Cloud Platform, Oracle Cloud Platform	Yes	? Bare metal	? cloud platfrom	Open Source
Moab [163]	fairness policies, dynamic priorities, and extensive reservations	AWS, Azure, Telecom, Oracle, Google Cloud	Yes	Yes	? cloud platfrom	? Open Source
Univa Grid Engine [**?**]	policy driven, backfill, exclusive and non-exclusive access to compute nodes, fault tolerant master	AWS, Azure, Google	yes	? Bare metal	? cloud platform	previously SUN Grid Engine, Genias Codine

Table 10: Comparison of Different IaaS Models

Provider or Framework	**Pricing**	**Database RDS**	**Reliability**	**Monitoring**	**Base OS**	**Programming Framework**
Amazon EC2 [119]	Pay-as-you-go or Yearly, reserved, spot	My SQL, Ms SQL, Oracle	Good	Good	Linux and windows	Python, Java, PHP, Ruby
Microsoft Window Azure [124]	Pay-as-you-go, semester, year	Microsoft SQL Database	Average	Average	Windows and linux	Java, Php, .net
Rackspace [122]	Pay-as-you-go	MySQL	Good	Extensive	Ubuntu	Java, Python
Google App Engine [123]	Pay as you go	Cloud SQL	Extensive	good	linux, free BSD, windows	Python, Java, PHP and Go, Node.js
Cloud Sigma [125]	Pay-as-you-go	SQL	Good	Good	Average	Python, Java, PHP, Python, Ruby, Clojour
Openstack [164]	Pay-as you go, monthly	My SQL	Good	Extensive	Linux, windows	Python, Perl, PHP,
Open Nebula [165]	Subscription	My SQL	High	Good	Linux	C,C++, Ruby, java,
Future Grid (discontinued) [4, 5]	Free Academic	User Choice	Good	Good	Linux	Openstack, OpenCirrus, Eucalyptus, Cloudmesh

Table 11: Comparison of Container based Scheduling Algorithms

Author	**Basis**	**Advantages**	**Disadvantages**	**Scheduling techniques**	**Experimental Scale**	**Experimental Parameters**	**Taxonomy Classifications**
Guerrero et al. [166]	Optimized the deployment of micro services-based applications	Improved security	High complexity	Genetic algorithm	Simulation environment	Resource utilization	Compute resource based, Container
Guerrero et al. [133]	Optimize physical machine utilization	Increase resource utilization	High complexity	Genetic Algorithm	Simulation environment	Resource Utilization , Performance	Compute-resource based, Container
Dziurzanski et al. [136]	Optimization of the container allocation	Easy to implement	Ignored network optimization	Heuristic method	Simulated environment	Performance	Compute-resource based, Container
Medel et al. [135]	Scheduler for minimizing resource contentions	Reduce resource contention	Ignored Time optimization	Priority algorithm	Kubernetes	Time	Compute-resource based, Container
Hanaf et al. [134]	Container and host selection policies	Improved SLA	Highly complex	Pre-Selection method	Simulated environment	Energy Consumption	Compute-resource based, Energy based

Table 12: Comparison of Different Resource Management Systems

Framework	**Architecture**	**Usage**	**Open source**	**Support**	**Applications**	**Programming Framework**
Eagle [148]	Hybrid	Differentiates short and long jobs	EPFL IC IINFCOM LABOS, Switzerland	Spark	Different workloads and Parallel jobs	Python, Java, PHP, Python, Ruby
Hopper [149]	Decentralized	Speculation-aware job scheduler	Microsoft Research	Spark	CPU intensive	Java, Php, .net
Tetris [150]	Centralized	Multi-resource bin-packing	Microsoft	Generic applications	CPU intensive	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala,
Fawkes [151]	Centralized	Dynamic resource balancing	TU Delft	Mapreduce frameworks	Data intensive	Python, Java, PHP, Python, Ruby
Omega [152]	Decentralized	Shared state abstraction	University of Cambridge	Custom applications	Parallel applications	Java, Php, .net
OurGrid [153]	Centralized	Equitable Resource Sharing	Universidade Federal de CampinaGrand, Brazil	Generic applications	Bag of Tasks	Java, Python
Sparrow [155]	Decentralized	Randomized sampling approach	U.C. Berkeley AMPLab	Spark	CPU intensive	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala, Clojure, .Net
Yarn [156]	Monolithic	Resource requests with containers	Hadoop	Spark	Data intensive	Python, Java, PHP, Python, Ruby, Clojour
Mesos [132]	Two way protocol	Pessimistic resource offers	University of California, Berkeley	Spark	CPU and Data Intensive	Python, Perl, PHP, Rest, Ruby, .net, C#