TAXONOMY OF SCHEDULING OF WORK-FLOWS IN INFRASTRUCTURE-AS-A-SERVICE(IaaS) CLOUDS

S.R. Anver, N. Mohanasundaram, M. Abdul Rahiman*

Abstract

Cloud Computing offers cost savings for new and existing ventures during Infrastructure setup and expansion, and in acquiring trained hardware maintenance by providing a competitive edge. Scheduling is the art of allocation of various resources of computation to the application components, with the objective of meeting the resource conservation goals and Service Quality (QoS) goals. Scheduling with respect to Cloud is mainly concerned with the Infrastructure as a Service model. This paper survey Scheduling algorithms for work-flows on clouds. This paper is offering a survey of Cloud system architecture, work-flow application models and Quality of Service (QoS) goals, which are to be considered while designing a scheduling algorithm, and providing a taxonomy of various algorithms on Cloud scheduling taken from the literature and doing their evaluation.

Keywords : Cloud Computing, Infrastructure as a Service Clouds, Scheduling, Taxonomy.

I. INTRODUCTION

TThe computing term Cloud Computing evolved in the late 2000s which consists of an infinite pool of remote resources such as servers, networks and software hosted over the Internet, which allow users for the storage, management and processing of data, which provide computational resources with features of scalability, elasticity, redundancy and security on-demand. Cloud computing is also defined as a pay-as-you-go computing model, in which the services provided by Cloud Computing are metered services, which are payable for the consumption. The various services of Cloud Computing are mainly available in three categories, Infrastructure-as-a-Service(IaaS), Platform-as-a-Service(PaaS) and Software-as- a-Service(SaaS).

Software-as-a-Service (SaaS) is usually a software distribution model, where a third-party provider is hosting computing applications over the Internet, and which are available to the users through the Internet. SaaS provides on-demand delivery for computing software in which the same copy of the computing application is available to all the customers. Microsoft Office 365 and Google Apps are two examples of SaaS. SaaS eliminates the expenses in the provisioning as well as maintenance of hardware acquisition, also eliminates the expenses in software installation, licensing and supporting for new software. The software updates are easier, since a single copy needs to be updated, and it can be accessible from anywhere through the Internet. Even though there are many advantages, there are certain potential disadvantages. In SaaS, customers have to depend on outside vendors for their software to be up and executing, tracking as well as reporting the accurate billing. SaaS also facilitates secure environment for the business data of its customers.

In Cloud, Platform-as-a-Service (PaaS) is a Service offering, where software and hardware tools required for the application developments are delivered to customers through the Internet, which are hosted by third-party service providers on the Internet. With the help of Platform-as-a-Service offering, users need not worry about the costly hardware and complex installation of new software to develop or run their applications. Google App Engine is an example of PaaS. A businesses' entire IT infrastructure is not

Department of CSE,
Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India
*Corresponding Author

replaced by PaaS, but the users of PaaS relies on their providers for the key services like, a particular application development environment, which can run on top of the users current infrastructure. Hence users can concentrate more on creation and running of new applications rather than acquiring and maintaining new infrastructure for the new application development.

With the uses of PaaS, many users can collaborate on software development, regardless of their physical location. Hence it provides convenience to the users and reduces the capital expenses in building up the hardware as well as software on their premises. The main concern of PaaS is the service availability. Any infrastructure disruption from service provider can seriously affect its customers, which may result in financial lapses in their production. Another concern of PaaS is that the customers using a PaaS service cannot easily migrate to a competitor's product or service, and this phenomenon is known as vendor lock-in.

The term, Infrastructure-as-a-Service (IaaS) is an offer- ing, where virtualized computing resources are provided to the customers through the Internet on demand. Amazon-Web- Services (AWS), Google-Compute-Engine (GCE), Cisco Metapod, Microsoft-Azure are some popular examples of IaaS. The various Computing resources in IaaS include Servers, Storage, network, virtualized software etc. Several factors are associated with each of the above resources. For example, the various factors of computing servers are CPU cores, memory, Network I/O, System I/O, Secu- rity, Graphics etc. and a network capacity is measured in terms of bandwidth. There may be various categories of the above resources available in a cloud infrastructure, with different prices.

An abstraction layer (virtualization layer) is also included in the Cloud infrastructure, which virtualizes all those resources and provides them to the customers via application programming in- terfaces or graphical interfaces. IaaS providers give opportunities to Organizations to take those components of Infrastructure such as storage, compute and networking over the Internet on rent. The IaaS providers also provide a variety of services like monitoring, detailed billing, security [1], log access, load balancing, storage resiliency and clustering, that are needed to accompany those infrastructure components and these services are usually policy driven [2].

With IaaS, businesses can rent the Infrastructure from the providers; hence, it offers faster, cost-effective and easier meth- ods to handle temporary, experimental or unexpected workloads without building up the complex infrastructure for those workloads. IaaS is a pay-as-you-go model in which each resource in IaaS is associated with a billing interval, which may be several minutes, an hour, a week or a month. As IaaS is a pay-as-you-go model, the users are always trying to minimize the cost by optimizing (minimizing) the resource usage from the IaaS providers [3]. Hence cost-efficient resource acquisition from IaaS providers is one of the important issues in IaaS Clouds.

Another concern of IaaS Clouds is that, the third party providers owns the infrastructure, hence the performance and con- figuration of infrastructure are rarely transparent to the users. Yet another concern is the service resilience, mainly due to the bottle- neck in networks, and/or in any form of external or internal down- time. Once the resources are allotted for their customers for their workloads, reliable, flexible, secure and cost-efficient utilization of these resources is a challenging task for the customers. Hence, it is required for scheduling the workloads for the available resources for meeting the required service level agreements.
In computing, the method of scheduling is to assign the given work to the available resources to complete the work. The term work may be referred to as any virtual computational elements like processes, threads [4], or data flows that are to be scheduled accordingly to the hardware resources like

processors, expansion cards or network links. Obtaining a correct schedule, which meet all the specified objectives falls into class NP. Hence it is trying to obtain approximate schedules, which optimize the specified objectives.

The term Scheduling is associated with many areas, where many workers (Computing resources) are available and many pro- cesses or tasks to be completed. For example, Process scheduling in Operating Systems and Project Scheduling in a manufacturing Organization. The term scheduling is significant in Cloud computing. It is mainly used with IaaS clouds to optimize the VM usage to the submitted applications; hence it reduces the cost of cloud usage.

In the Cloud Computing environment, VM Scheduling algo- rithms can be categorized as static scheduling as well as dynamic scheduling. In static scheduling [5], the input workload is submitted to the system in the beginning, and VMs are scheduled to various tasks. In Dynamic Scheduling algorithms, the input workloads are submitted to the system on the run [6]. It may be submitted periodically, or it may be arriving as streaming data. In this broad context, we are presenting taxonomy of work-flow scheduling onto Cloud Computing Infrastructure, from a survey of available literature in Scheduling algorithms in Cloud. Since, Scheduling in Cloud Computing is evolved from many factors. This review is essential for several reasons as discussed below.

I.1 Scheduling on High Performance Computing (HPC) Clusters and Clouds.

Traditionally, Scheduling is an important aspect in operating systems, Supercomputing clusters, HPC clusters as well as Com- puting Grids [7] [8]. While operating systems scheduling is dis- tinct in considering scheduling within a single host, the latter examines the scheduling over distributed resources connected over networks. Of these, Grid computing is offering shared distributed resources where the strategies of scheduling are crucial, in contrast, Cloud computing has emerged as a unique capability of distributed computing which is offered by commercial data center providers. Amazon web services (AWS) introduced Infrastructure as a Service (IaaS) computing in 2006. Since then, IaaS clouds have grown exceedingly popular for business and academic use, with several other Cloud providers like Microsoft-Azure, Rackspace and Google-Compute-Cloud [9]. Cloud resources have key distinctions from HPC Clusters and Grid Computing that affects on the method of scheduling the applications upon them [10], which are enumerated below.

I.1.1 Captive resources Vs. On-demand scaling.

High Performance Clustering centers have captive cluster in- frastructures which are often used in full capacity by users for ex- tend periods. Applications that access such clusters are given to a queue of batches, which schedules the jobs according to their time of arrival and waits till adequate hosts/cores for the available jobs. Jobs themselves may request a few or thousands of cores. Here the emphasis is given on how best is in allocating those available resources to the jobs waiting in the batch queue from many of users. However, the public Cloud Infrastructure offers access to the Virtual Machines (VMs) immediately without requiring a queue and gives the feeling of “infinite” pool of resources available on-demand [11]. Cloud infrastructure also allows the requested Virtual Machines to be elastically scaled up or down, based on the require- ments rather than retain VMs for the entire duration of the application. In Cloud Infrastructure, applications request one to tens of Virtual machines preferably than many cores usually in High performance computing clusters. The modest sized resource requests spread across thousands of users who acquire and release these re- sources at different times from a Cloud data enter makes it possible for providers to seemingly offer on-demand access to unlimited resources [9].

I.1.2 High performance vs. Commodity machines.

HPC clusters and Grids are often using high-performance processors and networks where, the servers are robust and have high hardware fault-tolerance. In contrast, Clouds use commodity hardware, running on Ethernet, which are not as resilient to the failure of hardware. As a consequence, the types of jobs that they are suitable for are different. HPC clusters host tightly-coupled applications which can run for minutes to hours, whereas, Clouds are favoured for independent applications or applications which are not tight and they run for seconds, minutes, hours and days each.

I.1.3 Non-vitualized vs. Virtualized environments.

HPC applications run directly on top of the base Operating Systems, but have limited control over the execution environment, which tends to be homogeneous to all applications [12]. IaaS Clouds give access to VMs that run within a hypervisor, and al- low VMs of different sizes having variable capacities of CPU, RAM, disk and network. They also allow users to define guest OS and execution environment on each VM. This virtualization en- ables multi tenancy on the same physical node, thus allowing the Cloud provider to use partial hardware resources at units smaller than a whole node. This also imposes overheads such as perfor- mance degradation, boot time for the guest OS and VM acquisition lag [13].

I.1.4 Quota vs. Cost model.

HPC clusters and Grids encourage their users for the full uti- lization of the high-end infrastructure, where the capital cost is fully paid for, and they do not leverage any monetary value to their users, also they are best meant for their fairness [12]. Public clouds are following pay-on-the-go model [9]. Users are charged for the resource usage and billing is in increments of fixed quanta. The users have to pay to acquire resources even if it is not fully utilized. Different pricing schemes are followed by different providers based on size, region and reliability. These schemes along with the focus on conserving real monetary costs, which offers an entirely different dimension in the scheduling algorithms, that is to optimize the cost. Issues related to resource provisioning to determine the number and type of VMs to be secured, hot to scale up and down the resources as the application progresses, and how to leverage the billing scheme while meeting the application's QoS are all important.

I.1.5 Wide Area Architecture.

Clients access HPC clusters or clouds remotely using a high- speed network connecting their local machine to institution with HPC center to launch their application [14]. Clouds have a narrower pipe from the client to the Cloud and there are monetary charges for the bandwidth associated with moving data in and out of the Cloud. As a result, data is often staged within the Cloud data center application to access locally. These affect the application's QoS while scheduling in a wide area network.

This survey paper is considering Infrastructure as a Service Clouds, for scheduling applications with VMs being the primary computing resource being scheduled on.

I.2 Work-flows as a programming abstraction for Cloud Scheduling.

Application is being scheduled on the Cloud may be imple- mented in a variety of manners, but for the purpose scheduling are abstracted and provided with a generic programming model. This allows the scheduling algorithm to operate on the generic appli- cation model along with its properties to provision resources and map application components into them, to meet the QoS and con- servation goals. Work-flow is a popular abstraction which allows the flexibility to compose complete, yet common, dependencies and execution patterns are between the schedulable components of the application [15]. Work-flows are often represented in the form of Directed Acyclic Graphs (DAGs),

which exhibit control dependencies among tasks that can be scheduled discretely [16]. Work-flows offer the malleability to represent both compute and data intensive applications, including Big Data applications [17]. Mappings from data flows, which capture data dependencies among tasks, to workflows that model control dependencies also exists [18]. This paper focuses the survey on scheduling applications provided as work- flows onto Cloud resources.

This taxonomy presents the specifications of the work-flows and IaaS Cloud systems relevant to application scheduling on Clouds and techniques used to schedule the work-flows on the Clouds.

I.3 Contributions.

We provide the following specific contributions in this survey.

1. We provide taxonomy for the various approaches and concepts used in the scheduling of workloads in Cloud resources, subjected to the data available in the literature.

2. Then we provide the characteristics of the key scheduling al- gorithms in the literature using this taxonomy.

II TAXONOMY

Our Taxonomy classifies the work-flow-based application models, public IaaS Cloud infrastructure model and the goals re- quired for the scheduling algorithms. We are also classifying the design of scheduling algorithms as well as the approaches to evaluate them. Fig. 1 shows the main classification.

II.1 System Design

In Scheduling Algorithms, System Design refers to the vari- ous parameters associated with the Cloud Service Providers. This may include, the types of resources such as computing power, network bandwidth, storage capacity, pricing details etc. That is, the computing resources are available with different capabilities along with different pricing schemes. The different classifications in System design is shown in the Fig. 2, which includes, VM size, Pricing Model, Pricing Characteristics and System Characteristics. This taxonomy is considering various capabilities of the above said items from the popular IaaS Clouds, viz Amazon-Web-Services(AWS), Google-Cloud-Engine and Mirosoft-Azure. These features are sub- sumed by smaller providers as well.

II.1.1 VM Size

Cloud Computing resources are available with different capabilities based on a payable-on-the-go pricing scheme. The computing resources offered as Virtual Machines (VMs) by the IaaS Clouds characterized by the sizes as well as the types. Hypervisors running on host machines along with hardware virtualization support allows the providers to list VMs with the various compute capabilities like clock speed, CPU cores, architecture etc., network bandwidth and physical memory along the corresponding pricing. The VM sizes play an important role in the context of scheduling Cloud applications.

(i) Heterogeneous VM Sizes:

Schedulers used to leverage the capability of acquiring heterogeneous VMs [19], which have the capacity of allowing the best-fit among the VM resources and the application requirements. Cloud providers usually offers VMs of different sizes and configurations. For example, AWS offers more than 275 EC2 instance types under the heads such as Storage optimized, memory optimized, Compute Optimized, general purpose, accelerated computing etc. Apart from the capacities of the resources, these may vary in disk types such as HDD or SSD, ex- istence of GPUs like accelerators as well as architectures like

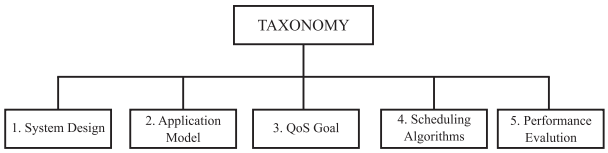


Fig. 1: Taxonomy of concepts and approaches for scheduling workflows on Cloud Resources

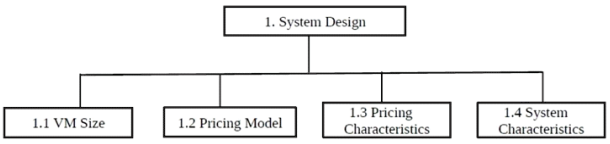


Fig. 2: Taxonomy of System Design

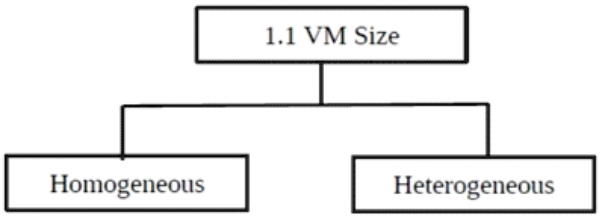


Fig. 3: Taxonomy of VM sizes of IaaS Clouds

latest CPU generation or DDR4 memory etc. However, such diversity makes scheduling decisions more complex and some features are not vendor-neutral. Hence, most Cloud schedul- ing algorithms limit themselves to VM diversity in terms of CPU cores, memory size and network bandwidth. For exam- ple, [20] offer scheduling algorithms that can utilize different VM sizes.

(ii) Homogeneous VM sizes:

Each additional dimension in the resource capabilities adds to search space for the scheduling algorithms. In order to sim- plify this, many algorithms assume a uniform capacity to all the resources, and call it as homogeneous VMs [14].

II.1.2 Pricing Model

The different types of VMs imply that the pricing of VMs are also different. But this goes further and different pricing models themselves are used by the Cloud providers. The different pric- ing schemes allow those scheduling algorithms in making finer op- tions for compromising price to the performance for their applica- tion (Fig. 4). Also, Cloud VMs are paid for when acquired and until they are released, irrespective of whether the end-user uses them or not. So having high resource utilization for acquired VMs is essential to get value for the money they paid. While we try to generalize the pricing models, not all Cloud providers offer all these different models, and there may be provider-specific features that may need to be addressed.

(i) Fixed Price VMs:

This pricing model assigns a static price per unit of time to each VM size, and these prices do not change with time. Within this fixed price model, there are further distinctions. On-demand VMs can be acquired for shorter units of time, ranging between 1 min and 1 hour, depending on the Cloud service provider. This offers a flexible payable-on-the-go model where users are payable for whole or partial billing intervals for which the VMs are acquired for, and the billing stops once the users release the resources. These types of VMs are suitable for irregular or adaptive workloads that are active for short durations, and VM acquisition and release can match the application demand. Several scheduling algorithms are de- signed for on-demand Vms [8] [21] [11] [20].

Alternatively, users can commit for acquiring VMs for ex- tended intervals of time using Reserved VMs (for AWS) or pre-purchased VMs (for azure). Here users can reserve com- puting capacity for long time periods of days or months, but get discounts of 50% - 75% off the hourly rate compared to use on-demand VMs for the same period. These types of VMs are better choices for those, who are having a long-term work-load, which is predictable, that makes full use of the reserved instances. These VMs are similar to have captive resources, and prior research of scheduling for HPC clusters are applicable to reserved VMs since the goal for such algorithms is to maximize the resource utilization given the upfront costs paid.

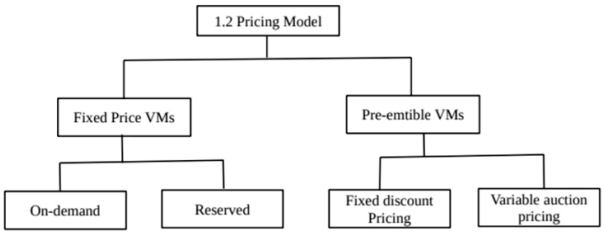


Fig. 4: Taxonomy of Pricing Model of VMs in IaaS Clouds

(ii) Spot and Pre-emptible VMs:

There may be having spare capacities in the Data Centers pro- vided by the Cloud Providers, which may be sold to the users at smaller charges than their original on-demand peers, some- times, ten times cheaper [22]. Aforementioned spot VMs al- lows the Cloud Providers to raise their revenue as well as utilization of owned Data Centers. Even though the spot priced VMs offers a similar performance of the same sized on- demand VM, the main difference is that, it doesn't guaranteed to be present at all times, whenever the user wants. Also, the Cloud Providers have the right to revoke the VMs, whenever certain conditions are met. Hence it is required to checkpoint actively to get the progress of the applications running in spot priced VMs, in which to avoid the loss of data during revoke. The spot priced VMs in Amazon uses a model based on auc- tions, which consider the highest cost of bids over billing pe- riod for each VM size, and the lowest successful bid until ca- pacity for that VM size is exhausted, forms the spot price that all bidders will pay per time interval for that type. The spot prices of Amazon vary with time, often within minutes, de- pending on their space capacity and the bids that are received. Once the spot VMs are acquired, it can be used till the spot price of the acquired VMs becomes lower that their user's bid price are revoked from the user when the price goes out of bid. Partially used billing intervals are not charged for revoked VMs. It is required that the schedulers who acquired spot VMs should be aware of its current spot price relative to their bid price, as it affects both the longevity of their application and the price paid. In Amazon, it is started to give a warning time of 2 minutes whenever the spot are going to be revoked, eas- ing the task of schedulers in deciding when to checkpoint and migrate.

Another type of VMs, which is offered by Google is pre-emptible VMs. Pre-emptible VMs have fixed prices, but usually they are about a maximum of 80% cheaper than its on-demand counterparts. But the difference is that, it can be pre-empted at any time. These VMs can be used for a maximum period of twenty four hours only, and Google used to give a warning time of 30 seconds, before that VMs are pre-empted.

Several scheduling and bidding strategies have been proposed for spot VMs to balance monetary costs paid against reliability of applications running on them [23]. In [12] spot VMs are exploited to fasten Map-Reduce jobs where the expected lifetime of spot VMs are calculated using Markov Chain to model the price transitions. Expected lifetime is used to calculate the amount of work a VM will be able to accomplish, for data back-up in case of failure and during bidding. Others have evaluated the reliability of applications when using spot instances. These show that even simple bidding strategies based on 70% - 110% of the on-demand VM price give a high reliability of > 98% with a significant cost savings of > 80% for several applications, making them a promising option.

(iii) Hybrid:

All the Scheduling methods make use of a collection of dif- ferent kinds of resources and platforms to obtain the required Quality of Service for their application. Systems may have captive off-cloud computing resources that do not incur additional monetary expense for their use, as part of a private data center. These local resources may be individual servers, clusters, or private Clouds that are virtualized and host VMs, or host containers such as Docker.

The different possible scheduling approaches used here are "Cloud-bursting" and "Cloud-firsting". In Cloud-bursting, the applications start on the local off-Cloud resource and the priority is given by the scheduler to maximize the usage of its free hostage resources prior to it is moved to the on-demand counterpart when local capacity is exhausted [13]. In the second approach, there is a default usage of public Cloud VMs and a small number of captive private servers offer instantaneous capacity, say to avoid VM boot up time for latency sensitive applications, as a fall back if budget constraints [24] are violated, or even for security reasons [25]. Yet another hybrid approach uses a blend of pre-emptible and on-demand instances that reduces the execution cost [26] uses on-demand. Virtual machines together with spot VM instances do the fast execu- tion of Map Reduce jobs and [27] propose different strategies used for the management of the life cycle of jobs in on-demand, spot and off-cloud local servers.

II.1.3 Pricing Characteristics

The next characteristic to be considered along with the pricing scheme is pricing characteristics that are elaborated in Fig. 5.

(i) Billing Interval:

Since, it is a payable-on-the-go model; the users have to pay per unit time for their VM usage. The billing increment forms the smaller whole unit of time for which users are billed, whether used partially or fully. Different Cloud Providers give different billing intervals. A 60 minutes granularity is given by Amazon, where the last partially consumed instance hour is billed full. Others offer shorter time intervals, with 5 mins from Cloud Sigma, 1 min from Azure, and 1 min from Google with a minimum charge for 10 minutes. The billing interval has the consequences on many factors, such as, it depends on the granularity of applications that are to be scheduled on Virtual Machines, Scheduler's temporal granularity controls, and its amount paid. As an example, in [28], there are different time periods of 5 minutes and 1 hour, which are to be consid- ered for evaluating the scheduling cost, and it seems that the normalized cost of shorter billing intervals are lower, which is as expected.

(ii) Network Pricing:

Since the pricing schemes discussed in previous sections focus on VM prices, the network bandwidth cost, in and out of the Cloud data center is charged as well, measured as data transferred in Giga Bytes. The data transfer may be between servers or off-cloud clients and VMs in the Cloud, or between VMs in different data centers. Typically, there is no charge for transfers within the same data center, and sometimes, there are additional charges per transaction as well.

Different schemes are following for pricing the network bandwidth. The first scheme is symmetric pricing, where, the same price is charged for moving data in and out of the cloud. The second scheme is asymmetric pricing, where, different prices are charged depending on the direction of data transfer. Usually the data-in bandwidth is kept lower or free for encouraging the data movement into the Cloud. This impacts the cost of moving input to/output from those applications between user's machine and Cloud, also the decisions concerned with respect to the hostage off-Cloud resources which called for relocating application status above the network.

(iii) Price to performance ratio:

While selecting a Virtual machine, there exists a trade-off connecting the rated VM performance and the fixed prices. The elastic nature of Virtual machines indicates that, the Cloud providers are attempting to raise the prices of VMs linearly with their size of VM, even though their performances are super-linear because of the reduced resource contention on account of VMs accumulated in the same owner, else better collocation of application tasks on the same host and [29] investigates how much quicker a task

executes when it is moved to a VM with a higher cost. Algorithms like LOSS-GAIN [8] attempt to normalize VM cost against the performance as well. In spot VMs, there can be cases where the current bid price for a larger VM is lower than a smaller VM, offering arbitrage opportunities.

II.1.4 System Characteristics:

This section describes the other system characteristics to be considered while designing a scheduling algorithm which is de- picted in the Fig. 6.

(i) Acquisition lag:

Once the VMs are requested, it takes some time to provision the requested VMs, booting up the same, and make it ready to be used by the requested-user. This is known as acquisi- tion lag or provisioning delay, which is not under the user's control, and may vary with time and the number of VMs re- quested [30]. The acquisition delay will influence their per- formance unfriendly when the algorithm regularly instantiate and shift among VMs. In [14], the effect of provisioning de- lay on several algorithms for scheduling ensembles has been shown. Other Cloud scheduling algorithms consider the acquisition delay when they plan the algorithm itself, and it forms a major factor in the effectiveness of the schedule [31].

(ii) Performance Variations:

In Cloud, the resources which are virtualized may not be offering predetermined performance in their data transferring and execution. The differences are due to the multi-user environment, where Virtual machines gathered on the single host may be competing for its resources / due to the occasional over- heads in Cloud fabric managements. These may cause the divergence in expected as well as the observed makespan of workloads, and hence it may be affecting the performances of static schedules. According to [32] performance variations are up to 30% in execution time and 65% in data transfer time. Many scheduling algorithms consider the performance changes as a main characteristic while allocating the cloud resources [7] models the performance variation by adjusting the processing capacity of each VM and introduces a performance degradation percentage.

(iii) Failures:

Besides variable performance, the computational resources are vulnerable to infrequent failures, which happen because of the failures in memory modules and disks, network transitory errors, hypervisor and Cloud fabric failures, and issues in power supplies. The above failures happen very rarely, where most of the cloud providers are promising an uptime of at least 99.95% per month. Still, the above said infrequent failures can adversely affect the run time of applications which are mission-critical, and there are algorithms where such situations are addressed. In addition, out-of-bid failures are common for pre-emptible VMs as well. For example, [26] proposes a scheduling algorithm which is said to be just-in-time that uses both on-demand and spot instances to lessen the cost and provide fault tolerance against failures and performance deviations of VMs.

(iv) Availability:

A key feature required in Clouds is the instant availability of the computational resources whenever the user requests. Even though the clouds are assumed to be available with infinite pool of on-demand VMs, but in practice the number of VMs available to a single customer is limited to 1000. But, during high demand periods, there is a possibility for the unavailability of a particular sort of VM instance in a particular data center. The accessibility would be restored as soon as the demand becomes stable. While availability for pre-emptible and spot VM instances are naturally irregular, but the reserved instances provide guaranteed availability.

II.2 Application Model:

This paper define scheduling unit as an application unit agreed by the user for the allocation of resources in the Cloud, together with the specified QoS requirements thrusted on it. This section

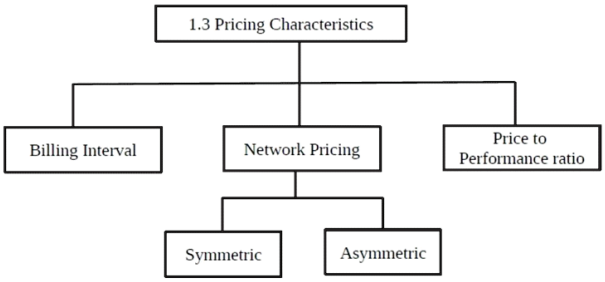


Fig. 5: Taxonomy of pricing Characteristics of VMs in IaaS Clouds

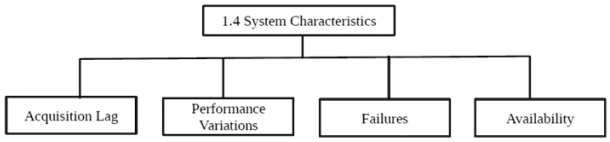


Fig. 6: Taxonomy of Cloud resources Characteristics

discusses the various factors of application model (Fig. 7). The scheduling algorithms discussed in this taxonomy may be using a finer or coarser granularity in scheduling. While the taxonomy consider work-flows as the primary application composition model, there are several variations are discussed. Furthermore, the unit of scheduling can have some particular meaning which may be helping in deciding the mapping of resources as well as the possible approaches for defect tolerance.

II.2.1 Scheduling Unit Structure:

In this taxonomy, the scheduling unit may be Directed Acyclic Graph (DAG) or a task as described in Fig. 8. Here a DAG is used to represent a work-flow, which is the scheduling unit given by the user. A work-flow graph is defined as a di-graph G = (V, E), where V is the set of nodes which represents the tasks, that is the basic unit in scheduling, and E, defined as the set of edges between the nodes (tasks), which represents control or data dependency of the sink task on the source task. The existence of an edge implies that a sink task cannot start executing till the source task completes (control flow), or the output from the source task arrives at the sink task as input (data flow).

Work-flows are also annotated with additional information that guides the scheduling. One common attribute that is provided in the work-flow is the execution time for each task in the work-flow. This can be expressed differently.

1. May be in terms of the absolute or normalized execution time of a particular task in a standard Virtual Machine
2. In terms of the execution time of the task in each of the differ- ent VMs available [7] [28], or
3. The number of million instructions available(required) in the task.

Correspondingly, the VM sizes in the Cloud system model can be denoted based on this normalized factor. The edges may also optionally be labeled using the amount of data that is transferred among the nodes to identify the data moving time in the network, hence the cost incurred during the transfer [12].

A particular work-flow may contain many tasks, even a few to thousands. The Fig. 9 shows an example work-flow which con- sists of 7 tasks and 9 dependency edges. The label given to each of the edge indicates the amount of data to be transferred between the tasks, which are the endpoints of the edge, say in Megabytes. The dependencies also express the ordering among the execution of tasks. For example, the task labeled 5 is dependent on tasks labeled 1, 2 and 3, and therefore, can only start executing after those three finish their execution. However, 1, 2 and 3 themselves can execute concurrently, as soon as their Homogenous dependency task 0 completes which itself can start executing right after the DAG is started as it has no prior dependency.

Besides work-flows, its simpler variant is considered, a Task, which can also be considered as a DAG with a single task. Several Cloud scheduling algorithms available in the literature deals with the topic of scheduling just tasks rather than DAGs. Here we can say a task as, it is the basic unit to be scheduled in the Cloud. It has no dependencies on other tasks.

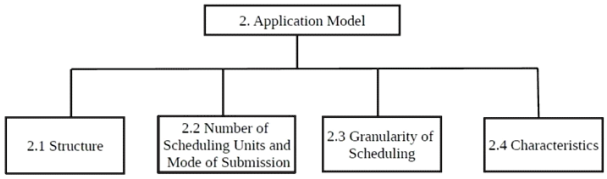


Fig. 7: Taxonomy of Application Models

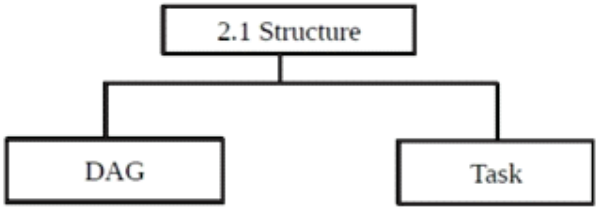


Fig. 8: Taxonomy of Structure of Scheduling Unit

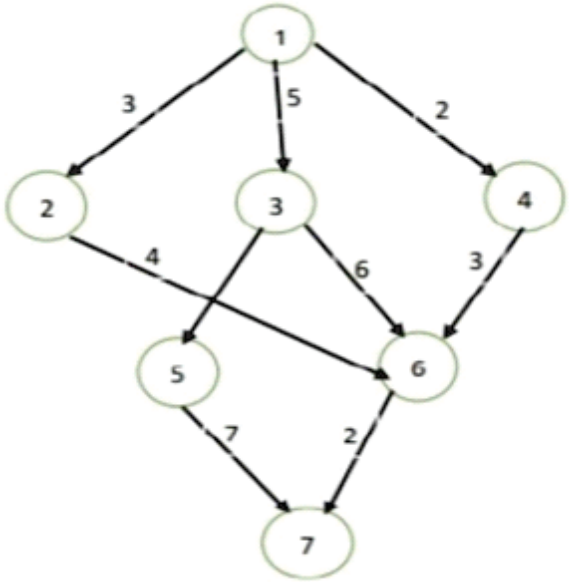


Fig. 9: A Sample Workflow

II.2.2 Mode of Submission:

While work-flows or tasks can be the scheduling unit given to the system by the end-users, the scheduler may be responsible for scheduling just individual units or multiple units. In the first case, a user may be providing a single unit, which can be either a DAG or a single task, whose Quality of Service requirements are specified separately and the scheduler is only responsible for scheduling this single execution unit. In the second case, many users may be providing a group or set of units, with separate requirements associated with each unit, then the scheduler may need to obtain the QoS requirements among all these scheduling units. In case of multiple units, all units may be homogenous - that is, all units follows similar structure, or heterogeneous - that is, each unit have a different structure, it may be a combination of DAG or tasks.

This variation helps in producing competent schedules. While intuitive, the existing literature does not explicitly tackle the term heterogeneous scheduling-units, which instead allow tasks to be defined as singleton DAGs. These categories are shown in the Fig.10 .

Moreover, when many users submit multiple units to the sys- tem, it may be either provided as batch of all units at once or may be submitted at different times. The later method of submission is called transactional model of submission. The time gap between the submissions determines the information provided to the algo- rithms while making decisions. For eg., in batch submission, one could consider the resource and QoS requirements for all schedul- ing units to decide upon a schedule which should be more “globally optimal” across all. However, when the scheduling units come up continuously, the scheduler should make independent decisions that will affect the effective scheduling of following units. These arrival modes are illustrated in the Fig. 10.

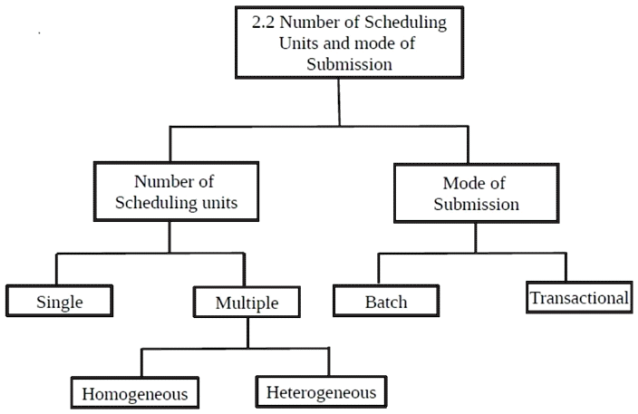
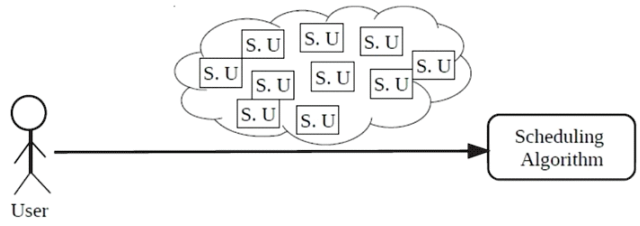
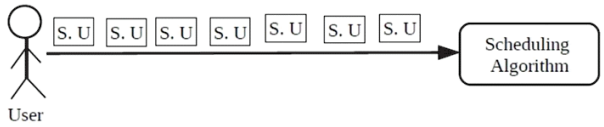


Fig. 10: Taxonomy of number of Scheduling units and their mode of submission

Moreover, when many users submit multiple units to the system, it may be either provided as batch of all units at once or may be submitted at different times. The later method of submission is called transactional model of submission. The time gap between the submissions determines the information provided to the algorithms while making decisions. For eg., in batch submission, one could consider the resource and QoS requirements for all scheduling units to decide upon a schedule which should be more "globally optimal" across all. However, when the scheduling units come up continuously, the scheduler should make independent decisions that will affect the effective scheduling of following units. These arrival modes are illustrated in the Fig. 11.



(a)Batch Mode of Submission



(b)Transactional Mode of Submission

Fig. 11: Submission of Scheduling Units to the Scheduling algorithm

II.2.2 Granularity of Scheduling:

Once the units are submitted to the system by the user, it has to consider, how the algorithm processes the submitted units. When the granularity is considered to be a single unit, which indicates that obtain a schedule for the particular unit while not considering the further units in the system. When the granularity is a collection granularity, it is required to generate a schedule for the collection as a whole, which observe the influence of everyone inside the col- lection.

Mode of submission of the units and scheduling granularity are interrelated. When the submission mode is transactional, then the choice of granularity is a single unit. When the submission mode is batch, then the choice of granularity is a collection. Several scheduling algorithms do just that [14][31].

That said, it is possible to translate from one mode of submis- sion to a different granularity of scheduling. For e.g. In a trans- actional mode of submission, a scheduler may schedule each unit upon arrival to start it with low latency, but without regard for future units. Alternatively, it can buffer the arrived units for sometime and produce a schedule for this collection of units, hence increases the ability to pack units onto VMs more efficiently, albeit at a higher la- tency for starting the units. Similarly, if in batch submission mode, the units don't have a collaborative QoS mentioned, the scheduler "flattens" those units and examines them independently for scheduling. Furthermore, within a collection, the scheduling units may be stored in an ordered queue or as an unordered bag. Ordering can be based on arrival time or the priorities of scheduling units assigned by the user. Fig. 12 depicts the Taxonomy in scheduling granularity.

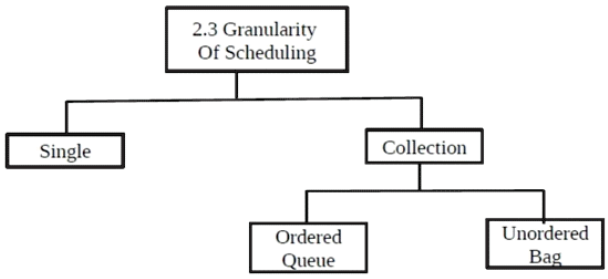


Fig. 12: Taxonomy of Granularity of scheduling

II.2.3 Scheduling unit characteristics:

Besides the above common features, there are some specific characteristics to the units which determine the strategies that can be employed by the scheduling algorithm in meeting its QoS goals on Clouds. As mentioned before, Cloud resources have perfor- mance variations, outright faults in on-demand VMs, or out-of-bid failures due to dynamic pricing in spot VMs. The algorithms may need to guarantee completion of the scheduling units within spe- cific deadlines under such unreliable conditions. Therefore, some scheduling algorithms incorporate fault tolerance techniques to mit- igate or resolve the effects of these problems. There may also be variability in the applications themselves which make reliable scheduling challenging. Below, we discuss some of the charac- teristics of units which affect its scheduling strategies that can be utilized to handle such dynamism (Fig. 13).

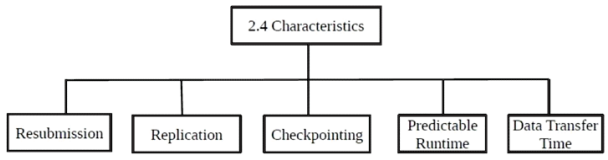


Fig. 13: Taxonomy of Characteristics of Scheduling Unit

(I) Resubmission:

Resubmission allows a task, which has been failed has to be executed again absoluted starting from the beginning without any aftereffects. The task may be part of the DAG or an in- dividual task. This importance or stateless property is typi- cally the lowest common denominator that ensures the fault- tolerance of tasks in unreliable VMs.

(ii) Replication:

Replication allows several of the same job to be executed concurrently using various resources, without having after- effects. This improves robustness of the task to guarantee timely completion even if one of the tasks fail due to VM loss. This feature opportunistically replicates the task on spare Virtual Machines, that are freely available, so that the first one to complete wins.

(iii) Checkpointing:

Here tasks can be check pointed. In check pointing, the sched- ules saves its state once it has executed partially, later it can resume the task from the last checkpoint. Unlike grid and cluster scheduling where checkpoints performed on a node could be resumed from the same node after it is recovered, VMs are transient and hence checkpointing implicitly also requires the ability to migrate the state and resume on a different VM. This is useful in two situations.

(a) When migrating from VMs to improve the time or cost of execution,
(b) To improve the resiliency of the task when running on unreliable VMs to meet deadline constraints.

In the former case of migrating to improve the cost or time, one could migrate and restart without checkpointing, but the progress of the task made thus far would be lost.

Cloud VMs experience failures due to various reasons. There- fore, checkpointing is employed to minimize the impact of such failures on the applications. [18] discuss a strategy for determining the number of checkpoints and intervals between checkpoints while trying to minimize the checkpointing over- head.

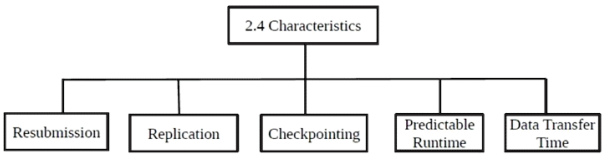


Fig. 13: Taxonomy of Characteristics of Scheduling Unit

(iv) Predictable Runtime:

Running time of the workload is said to be deterministic when expected execution time of tasks in different Virtual machines are predetermined, that means there are no unpredictability. In reality, the execution times for many applications cannot be categorically stated (even if the VMs themselves have de- terministic performance), and the uncertainties in application execution times may adversely affect the algorithm's perfor- mance. Several algorithms rely on accurate specification of runtimes of scheduling units, while some others may use an initial estimate of the tasks in the workload but do not assume they are perfect [22]. The algorithms may use a real time mon- itoring information at running time to upgrade the estimated runtime and revise the schedule on-the-go [20] analyze how their auto-scaling mechanism is affected due to imprecise esti- mation of task execution time.

(v) Data Transfer Time:

Workloads may take large inputs and produce large outputs. The same holds for tasks also. This large volumes of data are to be transmitted between the nodes (tasks) within the DAG or may be send from/to a persistent storage location. These data transfer takes time which put a price on when shifted among the tasks across different VMs, or between desktop and Cloud. The applications need to specify the sizes of data to be trans- ferred along with the definition of the application in case these have a non-trivial impact on the scheduling performance, and the algorithms considers them when provided [32] considers the case in which the allocated VMs are started before the start time of its initial task to include the data transfer time to its input. Similarly, the time to transfer out data generated from the last task of the VM is also provisioned [33] also considers input data size for the task as a factor to decide whether to mi- grate the task within a VM in Cloud, or from the Cloud to an off-Cloud server. In other works, the earliest-start-time(EST) as well as latest-finish-time (LFT) are calculated by including the data transmission time between tasks in a work-flow. The critical path length of a DAG, the width of the longest path in the graph from its start node to the end node, is also calcu- lated by including the same [11]. Since data transfer does not benefit from more CPU cores in VMs, [30] use Amdahl's law to estimate the execution time of each task where the I/O and intermediate data transfer time is considered as the sequential part of total execution time of the task.

There are other scheduling algorithms, however, that do not considered the input, output and intermediate data transfer between the tasks and assume either that the data is already present in the Cloud location, the task running time specified includes the intermediate data transfer time, or the data sizes are too small to matter [20] [22].

II.3 Purpose of Quality-of-Service (QoS):

Getting a schedule that satisfies the required Quality-of-Service of the submitted units is the purpose of every algorithm (Fig. 14). Different QoS goals are identified as QoS Contraint, Constraint guarantees, Optimization goals and Granularity.

II.3.1 Constraints of QoS:

Constraints, the Fig.15 shows that the generated schedule should meet the constraint specification, in addition to other QoS requirements that may be specified [34]. Typically, temporal con- straints used to be defined on makespan of those scheduling units, where users need to complete the execution of the scheduling unit within the specified deadline

from submitted time. Important or urgent tasks or DAGs can have a shorter deadline to finish quicker.

However, in Clouds, the monetary value is the main measure of success, and in such cases, a budget is specified as constraint by the users to bound the amount paid for executing those scheduling unit. Generally, faster resources are costlier than slower ones and thus the user faces a cost-time trade-off [11]. Hence, as a substitute of specifying either of these constraints, or both separately, users may also incorporate an objective function which combine both time as well as cost for their application to one dimension. So, their specification for QoS requires that the prescribed objective function and inside certain bound [35].

II.3.2 Guarantee of Constraints:

The user specified constraints may be hard constraints or soft constraints (Fig.16). An absolute constraint whose violation indi- cates a catastrophic failure or other kind of natural disasters for its end-user is termed as Hard Constraints. Hard time constraints impose a maximum time limit, and require guaranteed completion before the given deadline; hard cost limits impose a maximum monetary budget. At the same time, the constraint which permits relaxation in achieving the goals and a maximum effort is adequate [20] is termed as soft constraints. If the algorithm did not meet the specified constraints in such situations, then penalty functions are used to analyze the same. In other words, if a soft deadline is missed by small margins, the application is not deemed to have failed. For example, the auto-scaling discussed in [20] uses soft constraints for its work-flows while some others implement hard deadlines. The GAIN and LOSS algorithms discussed in [36] considers a hard constraint on budget, with an approach to get a shortest makespan.

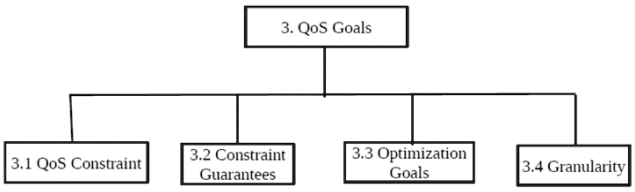


Fig. 14: Taxonomy of Quality of Sevice (DoS) Goals

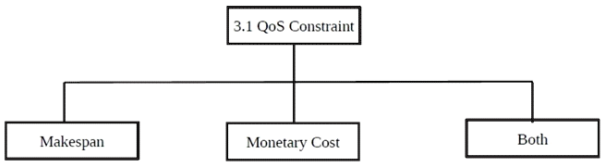


Fig. 15: Taxonomy of Quality of Sevice (DoS) Constraints

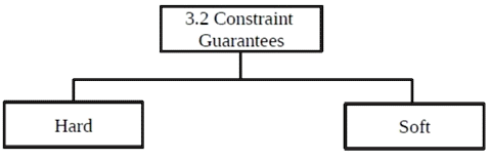


Fig. 16: Taxonomy of Constraints Guarantees

II.3.3 Goal of Optimization:

The goal of optimization (Fig. 17) analyses the scheduling al- gorithm performance, with the condition of meeting the constraints. These goals typically reduce to an optimization problem that mini- mizes or maximizes an objective function.

(i) Minimization:

The goal can minimize an objective function, which could be the makespan of the application or the cost associated with the execution. Generally, under a deadline constraint, the goal for the algorithm is to try and minimize the cost, and there are several examples of such deadline constrained work-flow scheduling algorithms [11] [20] [23] [37]. Similarly, under a budget constraint, the performance of the algorithm is based on its ability to minimize the total time [36]. [38] which con- siders both alternatives: given a budget constraint for a BoT while minimizing its overall execution

time, and given a dead- line constraint on the BoT while minimizing the total cost [39]. Also, in some there is no QoS constraint and the objective function is to determine the assignment of tasks of a DAG to processors such that makespan is minimized [12].

(ii) Maximization:

The optimization goal may be to maximize the objective func- tion, such as the throughput, the number of tasks executed per unit time, or the utilization of resources. Priorities provided to different scheduling units in a collection can also be used to define user-specific maximization goals. For example, in [22], the given constraints have a fixed deadline and a fixed budget. The outcome is to maximize the completion of a variety of high-priority work-flows in the given ensemble.

(iii) Not both:

In some Scheduling algorithms, neither of the optimization goals is specified and [40] analyzes a trade-off among the cost savings above the fixed price Virtual machines and flexibility when tasks got executed on spot Virtual machines. Thus they try to obtain soft deadlines of tasks when using spot VMs on a best effort and report the makespan, budget and successful completion ratio for the tasks. In [32], the user specifies a deadline constraint, also a flexible budget, and the objective is to obtain the soft dead-lines of work-flows.

Fig. 17: Taxonomy of Optimization Goals

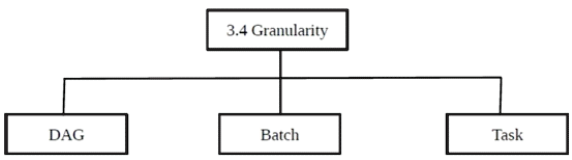


Fig. 18: Taxonomy of Granularity of QoS

II.3.4 Granularity:

The above mentioned constraints and goals can be described at different granularities (Fig. 18). The default granularity may be a single unit, if not, a batch, that purely based on its submission mode. Even so, other alteration in describing the granularity also exists.

(I) DAG:

Placing the constraint(s) on the Directed-acyclic-graphs indi- cates that they have to be obtained for the graph altogether re- gardless of QoS for specific tasks. In those situations, usually the optimization objectives are also placed at the equal gran- ularity. After all, in some cases, these granularities may be distinct. Here, the constraint on makespan may be described at DAG level, and the optimization goal to minimize the cost may be described for collection of DAGs.

(ii) Batch:

A batch may be a collection of work-flows or BoT (Bag-of-Tasks) and Constraints may be specified on these batches also [41]. Constraints on batches indicate that, it should meet the constraint for the whole group, regardless of independent units. In such cases, the QoS objectives are also described at the equal granularity.

(iii) Task:

The users can specify constraints on individual tasks also, that may be the tasks of a DAG or tasks inside a bag (BoT) or unique tasks. Several scheduling algorithms target the opti- mization goal at the task level as well. The users can also describe the optimization objectives and constraints for dis- tinct phases of the graph [42]. In these situations, distinct al- gorithms are applied to schedule the tasks of distinct phases based on the objectives and constraints at every step.

II.4 Scheduling Algorithm

Specified the categorization of the system model, application model as well as requirements of QoS, the goal of the algorithm is to schedule those applications onto the system which meet its QoS. Finding the solution for the scheduling problem tends to be com- putationally infeasible for non-trivial dimensions. As an outcome, there exits many algorithms and techniques in literature to face this situation, usually to obtain an approximate solution in preference to optimal solutions. Surveys on existing literature categorize the al- gorithms (Fig.19) depending on the techniques that they are incor- porated, and those techniques determine the grade of the schedule- plan obtained as well as the necessary time to get the same.

II.4.1 Scheduling Techniques:

Getting an optimal schedule falls into the traditional combina- torial optimization problem which is an alternative of classic prob- lem of job shop problem (JSP), which is NP Hard. The method of Brute force tries all possible combinations of mappings from VM resources to the scheduling units to reach a solution that is globally optimum while meeting its constraints.

There are certain dynamic programming approaches, where the given problem is divided into a number of intersecting sub- problems, and optimal solution to each of them can be obtained in manageable time and the results are memorized. The memorized results are reused while exploring the solution for the original prob- lem within these sub-problems.

In contrast to the above, certain algorithms use Divide- and- conquer technique, which follows three steps, divide, conquer and combine. In Divide step, the original problem is divided into num- ber of sub-problems, which may be non-overlapping, Conquer step finds the solution to each sub-problem recursively, and in the com- bine step combines all these solutions to obtain the solution to given problem(DAG).

Moreover, the solution for scheduling problems can be ob- tained through sub-optimal algorithms which use some heuristics come near to the optimal solutions within affordable time. The Greedy method chooses the best option from the solution space at any given stage. It makes use of a sequence of local optimum choices with an assumption of getting a global optimum.

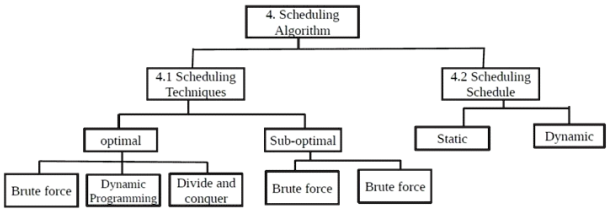


Fig. 19: Taxonomy of the techniques used and types of schedules generated

I.1.1 Scheduling Schedule:

There may be static scheduling algorithms or dynamic scheduling algorithms. A static scheduling algorithm runs in an offline manner, it execute once as the scheduling unit arrive. In Dynamic scheduling, the algorithm continuously determines the schedule as and when new units are entering the system.

Static scheduling makes use of the information of the schedul- ing units and the resources available in the system before the start- ing of the scheduling algorithm. With the help of the above in- formation, it maps the available scheduling units to those available resources once in offline manner. Once the allocation is done, it is hold on to the lifespan of the DAG or tasks. This approach does not allow the tasks to change or resources during execution and con- clude that the information available apriority is perfect. Also, it does not leverage features of the application such as check pointing and resubmission.

If we know the workload in advance, also using a cloud, whose resource performances are predetermined, and then static schedule is preferred. However, static planning, exclusively, may not work in Cloud environments due to the dynamism in resource performance, data transfer times, boot time, spot prices, and non-deterministic estimates of the execution time of tasks on VM. It may cause the performance of sub-optimal QoS and in worst-case, the breach of hard constraints.

Apart from the static knowledge, Dynamic techniques make use of the runtime information of its resources and applications [25], to build scheduling decisions throughout the lifespan of the DAG or task. They use runtime information from the Cloud infrastructure and task execution times to adjust its schedule quickly to avoid deadline breaches and budget overruns. Dynamic algorithms can either be just-in-time, where the scheduling decision for a task in a work-flow is made just once before the task's execution (but after the DAG itself has started executing), or fully dynamic where running tasks can be remapped and migrated from one VM to an- other during its execution.

I.2 Performance Evaluation

It is concerned with how the performance of the proposed algorithm is evaluated (Fig. 20). This includes the type of applications and work-flows, how the cloud behaviour is simulated, what data about the prices and performance of resources has been considered.

Applications may include real world or randomly generated applications. For example, real world graphs like Cybershake, Genome, LIGO, Montage are used in [11] [22] [30] [37] while randomly generated graphs are used in evaluating the performance of HEFT [43], CPOP [12] algorithms for work-flow scheduling on heterongenous processors.

Many algorithms consider price and performance of VMs similar to that provided by major cloud providers during simulation. In [20], 4 types of on-demand VMs are considered for simulation, with price and configuration same as that of Amazon EC2. Other important characteristics of cloud resources such as VM acquisition lag has been addressed in [30], [37] where the boot time is set according to the study on the VM startup time [20].In [20] [22], lag is varied to evaluate the impact on the scheduling algorithms while [38] has set the startup time to 10 s arbitrarily. In [32] [33] variations in performance and data transfer times of VMs are considered and modeled using normal distribution.

Evaluating the performance of proposed algorithm is important to know how well the algorithm performs and to get a comparative analysis against the algorithms which solve the same problem. Evaluation techniques can be categorized as follows:

(I) Analysis:

Analysis involves working out a proof to show the performance of the proposed algorithm without actually running the algorithm. Analytical proof is better than empirical and sim- ulation based results as it gives a general description of the algorithm's performance for any value of parameters whereas experiments and simulations can't be performed for all the pa- rameters.

(ii) Empirical Evaluation:

Empirical evaluation involves actually executing the sheduling units on the target environment and collecting the results.

(iii) Simulation:

Simulation does not involve actual execution of the scheduling unit. Instead it involves representing or mimicking the execution in simulation environments which represents the tar- get environment. It is useful to simulate instead of actually testing the algorithm on cloud resources since it causes real expenditure. Using simulation, one can test and run the algo- rithm as many times without worrying about the expenditure. It is much easier and time-saving for reproducing the results. Cloudsim [33] is a simulation framework which can easily model and simulate cloud infrastructure. Cloudsim supports modelling of data centers, virtual machines and host which helps in evaluating the algorithms in much less time.

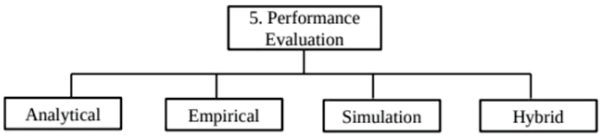


Fig. 20: Taxonomy of Performance evaluation techniques used

(iv) Hybrid:

Hybrid evaluation involves a mix of the above techniques.

III CONCLUSION

This paper investigates various algorithms evolved to schedule different kinds of work-flows in Infrastructure-as-a-Service Cloud-Computing environments. In particular, it focuses on various factors to be considered while designing a scheduling algorithm for the work-flows in IaaS Clouds. It gives a taxonomy build on a comprehensive study of existing algorithms that focuses on features like, System model, Application model, Quality of Service Goals, Scheduling algorithms and Performance Evaluation. The taxonomy also includes a deep understanding of various factors to be considered in all of the above heads.

REFERENCES

[1] Gaurav Somani, Manoj Singh Gaur and Buyya, , 2017. “Ddos attacks in cloud computing: Issues, taxon- omy, and future directions”. In Computer Communications.

[2] Kumar and Sharma, 2017. “Dynamic load bal- ancing algorithm for balancing the workload among virtual machines in cloud computing”. In International Conference on Advances in Computing and Communications.

[3] Convolbo and Chou, 2016. "Cost-aware dag scheduling algorithms for minimizing execution cost on cloud resources". In The Journal of Supercomputing.

[4] S. N. Arora and Plaxton, 2001. "Thread scheduling for multiprogrammed multiprocessors". In Theory of Computing Systems,.

[5] Daoud, and Kharma, 2008. "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems". In Journal of Parallel and Distributed Computing, Vol. 68, pp. 399–409.

[6] Muthucumaru Maheswaran, Shoukat Ali and Freund, 1999. "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems". In Heterogeneous Computing Workshop 1999 (HCW 99).

[7] Mattias Hovestadt, Odej Kao, and Streit, 2003. "Scheduling in hpc resource management systems: Queuing vs. planning". In 9th International workshop on Job Scheduling Strategies for Parallel Processing.

[8] Xhafa and Abraham, 2010. "Computational models and heuristic methods for grid scheduling problems". In Future Generation Computer Systems.

[9] Michael Armbrust, Armando Fox, 2009. "Technical report". U. of California at Berkley, ed.

[10] Wei Tan, Paolo Missier, and Goble, 2010. "A comparison of using taverna bpel in building scien- tific work-flows" the case of cagrid". C. C. P. Exper., ed.

[11] S. Abrishami, and Epena, D. H., 2013. "Deadline contrained workflow scheduling algorithms for infrastructure as a service clouds". In Future Generation Computer Systems, Vol. 29, pp. 153–169.

[12] H Topcuoglu, S. H., and Wu, M. Y., 2002. "Performance effective and low complexity task scheduling for heterogeneous computing". In IEEE Transactions on Parallel and Distributed Systems, Vol. 13 of 3, pp. 260–274.

[13] Warneke, D., and Kao, O., 2011. "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud". In IEEE Transactions on Parallel and Distributed Sys- tems, Vol. 22 of 6.

[14] Anderson, D. P., 2004. "Boinc: a system for public-resource computing and storage". In Proceedings of fifth IEEE/ACM International Workshop on Grid Computing.

[15] H. Kllapi, E. Sitaridi, and Ioannidis, Y., 2011. "Schedule optimization for data processing flows on the cloud". In ACM Special Interest Group on Management of Data (SIGMOD) Conference, pp. 12–16.

[16] Ewa Deelman, Gurmeet Singh and Katz, D. S., 2005. "Pegasus: A framework for mapping complex scientific work- flows onto distributed systems". In Scientific Programming.

[17] J. Diaz-Montes, M. Diaz-Granados, and Parashar, M., 2015. "Supporting data-intensive workflows in software-defined federated multi-clouds". In Transactions on Cloud Computing, IEEE, ed.

[18] S. Di, Y. Robert and Cappello, 2013. "Optimization of cloud task processing with checkpoint- restart mechanism". In International Conference for High Performance Computing, Networking, Storage and Analysis(SC).

[19] Richard F. Freund, Michael Gherrity and Siege, 1998. "Scheduling resources in multi-user, heteroge- neous, computing environments with smartnet". IEEE, ed.

[20] Mao, and Humphrey, 2011. "Autoscaling to minimize cost and meet application deadlines in cloud workflows". In International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1–12.

[21] Anwar, and Deng, 2018. "Elastic scheduling of scientific work-flows under deadline constraints in cloud computing environments". In future Internet, Vol. 10 of 1.

[22] Maciej Malawski, Giden Juve, and Nabrzyski, 2012. "Cost and deadline contrained provisioning for scien- tific workflow ensembles in iaas clouds". In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC).

[23] Rodriguez and Buyya, R., 2014. "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds". In IEEE Transactions on Cloud Com- puting, Vol. 2 of 2, pp. 222–235.

[24] Bhalerao, B. M., and Shende, S. W., 2015. "Weight based budget distribution (wbd), a budget constrained scheduling algorithm for workflows in cloud". In International Journal of Innovative Science, Engineering and Technology

(IJISET), Vol. 2 of 4.

[25] M.A. Rodriguez, R. Buyya., 2017. "Scheduling dynamic work- loads in multi-tenant scientific workflow as a service platforms". In Future Generation Computer Systems.

[26] Rpdriguez, M. A., and Buyya, R., 2010. "A taxonomy and sur- vey on scheduling algorithms for scientific worklows in iaas cloud computing environments". In Concurrency and Com- putation: Practice and Experience, pp. 1–32.

[27] Zhi-Hui Zhan, Xiao-Fang Liu,, and Li, Y., 2015. "Cloud computing resource scheduling and a survey of its evolutionary approaches". In ACM Computing Surveys, Vol. 47 of 4.

[28] D. Abramson, J. G., and Kotler, L., 2000. "High performance parametric modeling with nimrod/g: killer applcation for the global grid". In Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS).

[29] Deepak Poola, K. R., and Buyya, R., 2016. "Enhancing reli- ability of workflow execution using task replication and spot instances". In ACM Transactions on Autonomous and Adap- tive Systems, Vol. 10 of 4.

[30] Zhou, A. C., and He, B., 2014. "Transformation-based mone- tary cost optimizations for workflows in the cloud". In IEEE TRANSACTIONS ON CLOUD COMPUTING, Vol. 2 of 1.

[31] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, A. S., and Tsafrir, D., 2013. "Deconstructing amazon ec2 spot instance pricing". In ACM Transactios on Economics and Computation.

[32] Calheiros, R. N., and Buyya, R., 2014. "Meeting deadlines of scientific workflows in public clouds with tasks replication". In TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE, ed., Vol. 25 of 7.

[33] Rodrigo N. Calheiros, Rajiv Ranjan and Buyya, R., 2011. "Cloudsim: a toolkit for modeling and sim- ulation of cloud computing environments and evaluation of resource provisioning algorithms". In SOFTWARE – PRAC- TICE AND EXPERIENCE, Vol. 41, pp. 23–50.

[34] Meng Xu, Lizhen Cui, 2009. "A multiple qos con- strained scheduling strategy of multiple workflows for cloud computing". In IEEE International Symposium on Parallel and Distributed Processing with Applications.

[35] Haolin Feng, Guanghua Song, Y. Z., and Xia, J., 2003. "A deadline and budget constrained cost-time optimization algo- rithm for scheduling dependent tasks in grid computing". In International Conference on Grid and Cooperative Comput- ing.

[36] R. Sakellarioum H Zhao, E. T., and Dikaiakos, M. D., 2007. "Scheduling workflows with budget contraints". In Integrated Research in Grid Computing, pp. 189–202.

[37] Deepak Poola, K. R., and Buyya, R., 2014. "Fault-tolerant workflow scheduling using spot instances on clouds". In 14th International Conference on Computational Science (ICCS 2014), Vol. 29, pp. 523–533.

[38] Long Thai, B. V., and Barker, A., 2015. "Task scheduling on the cloud with hard constraints". In IEEE World Congress on Services.

[39] Grekioti, A., and Shakhlevich, N. V., 2013. "Scheduling bag- of-tasks applications to optimize computation time and cost". In International Conference on Parallel Processing and Ap- plied Mathematics.

[40] Kushwaha, V., and Simmhan, Y., 2014. "Cloudy with a spot of opportunity: Analysis of spot priced vms for practi- cal job scheduling". In Cloud Computing for Emerging Mar- kets(CCEM).

[41] Gutierrez-Garcia, J. O., and Sim, K. M., 2013. "A family of heuristics for agent-based elastic cloud bag-of-tasks concur- rent scheduling". In Future Generation Computer Systems.

[42] Frincu, M. E., and Craciun, C., 2011. "Multi-objective meta-heuristics for scheduling applications with high avail- ability requirements and cost constraints in multi-cloud envi- ronments". In Proceedings of forth International IEEE con- ference on Utility and Cloud Computing.

[43] Haluk Topcuoglu, Salim Hariri, 1999. "Task schedulig algorithms for heterogeneous processors". In IEEE Proceeding on Eighth Heterogeneous Computing Workshop (HCW'99) USA.