Comparative Analysis of Deadline Constrained Task Scheduling Algorithms for Cloud Computing under Cloudsim

Karthika C Babu and A.S. Mahesh
1Department of Computer Science and IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.
karthikacbabu693@gmail.com
2Department of Computer Science and IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham, India.
asmaheshofficial@gmail.com

Abstract

Cloud computing is an ever-growing distributed, accessible and scalable infrastructure, it offers a widespread paradigm that contributes efficient computing resources. The primary goal of the cloud computing is to provision and provide the cloud resources to the user, the user can pay only for those resources according to their usage within the lease schedule time (pay as you go, model). Efficient scheduling strategy needed to allocate the task to appropriate resources. However convoluted tasks, which contains lots of interconnected applications, called workflows, its need effective tasks scheduling to meet the user-defined QoS, like execution time (makespan) or cost. In this paper, is the comparative analysis of two Deadline Constrained Heuristic scheduling algorithm, GAIN, and IaaS_Cloud Partial Critical Paths (IC-PCP) Which assigns the user's tasks on to the resources and executes it with minimum execution time (makespan) under given constraints. The paper compares and examines the efficiency of the algorithm with diverse cloudsim parameters, such as VM (VM size, MIPS, bandwidth), cloudlet (length, file size, PE's number) and data centers. The Algorithms simulated using cloudsim 2.2.2 toolkit package with NetBeans IDE8.0.

Key Words: Cloud computing, task scheduling, Makespan, IC-PCP, GAIN,
Cloudsim.

1. Introduction

Cloud computing is an ever-growing model that offers a widespread paradigm, its grasp and attained the IT Industry rapidly. The cloud computing significance is internet based computing or it’s a pattern of server based computing.

The Instant scheduling and provisioning is the drastic feature of cloud computing environment. Generally three main classes are categorized in cloud computing based on the services: Platform as a Service (PaaS), Software as a Service (SaaS) and Infrastructure as a Service (IaaS). It is the expansion of, parallel computing, grid computing and distributed computing, or characterized as the viable application of computer science perceptions. It’s a system that provides the unified computing resources with the group of virtualized and interconnected computers that can be dynamically provisioned and leased with minimal running effort.

In the distributed computing environment, cloud computing provides the latest emerging trend, that handover services as hardware infrastructure and software applications. Based on a Service Level Agreement (SLA) the consumers can access and use this service that established between the service provider and consumers. On a pay-as-you-go basis (SLA) states a users' essential quality of service (QoS) parameters. There are diverse types of scheduling algorithm exists in a distributed computing system, task scheduling is one among them. It’s challenging key issues to fulfill user-defined QoS like execution time (makespan) or cost in order to improve the performance of the cloud services. However, convoluted tasks, which contains lots of interconnected applications, called workflows, it's designed as a DAG, in which each node shows the tasks and the edge between corresponding nodes shows data In general, mapping tasks on to a set of distributed instances belong to the class of scheduling problem is NP-complete or NP-hard. For efficient resource scheduling, meta-heuristic and heuristic, search-based are the two policies have been suggested.

The Scheduling and provisioning stages are the important phases of the majority cloud scheduling system. Depends upon the types of tasks and assigning strategy the better task scheduler should adopt it with the changing environment, some performance constraints like high throughput, low response time, minimum makespan and flow time are the conventional metric constraints used for task scheduling. Scheduling regulates the availability of CPU, memory and good scheduling policy provides maximum utilization of the resource.

In this paper, is the comparative study of two Deadline Constrained Heuristic scheduling algorithm, GAIN and IaaS-Cloud Partial Critical Paths (IC-PCP) Which assigns the user's tasks on to the resources and executes it with minimum execution time (makespan) under cloud simulation constraints. The paper compares the performance of the algorithms with different cloudsim parameter
settings, such as VM, cloudlet, and data centers (VM size, MIPS, bandwidth, cloudlet length, ram speed). As per the result, the IC-PCP has shown the better performances in most parameter settings. The Algorithms simulated using cloudsim 2.2.2 toolkit package with NetBeans.

2. Related Work

Cloud computing is the emerging technology, millions and millions of user use and shares the cloud resources. Users submit millions of tasks into the cloud resources, so handling and assigning the task to the appropriate resources is a challenging and tremendous work in a cloud computing environment that met the user-defined QoS. The Figure: 1 shows the taxonomy of the scheduling process. The cloud should take the minimal number of systems for task scheduling or optimal resource allocation, as per the result the total cost is minimized. Based on the scheduling priority methods task can select from a collection of tasks, it's a traditional hugely acceptable process. Using the QoS constraint the priority of a task can be defined dynamically at runtime. So priority consignment scheduling is a complex process, as there static and dynamic are the two types. The static priority, assignment of tasks faces many difficulties. In order to assure the certain level of data flow performance users QoS has the capability to provide different priority in diverse applications. The scheduling components or brokering components uses task scheduling algorithms, its enhance the overall performance of the cloud. However, convoluted tasks, which contains lots of interconnected applications, called workflows, it's designed as a DAG, in which each node shows the tasks and the edge between corresponding nodes shows data. In general, mapping tasks on to a set of distributed instances belong to the class of scheduling problem is NP-complete or NP-hard.

There is so many resource scheduling algorithm has for efficient scheduling, there are meta-heuristic strategies and heuristics-based. In such based system, their task considered as a dependent (as workflow) or independent (bags task).

The Scheduling and provisioning stages are the important phases of the majority cloud scheduling system, GAIN algorithm categorized as pure scheduling. There are two levels is classified at cloud service scheduling, there are system level and user level. The user level is an interactive scheduling level that negotiates between the user and the providers, its focus on the problems and suggestion raised by service providers. The system level deals with the storage details that is resource scheduling like resource management within data centers of the cloud system. Each geologically dispersed data centers are heuristic-based request scheduling and the strategy emphases to universally diminish the fine charging in cloud computing system. The best effort and QoS constraint scheduling are the two categories of the workflow. At best-effort scheduling algorithms, minimizing the makespan (overall execution time) are the main goal, but it's not considered the cost factor. Max-Min, Min-Min, and Suffrage
are simple heuristics in nature, its aim is to find the optimized execution time, and it’s a complicated process in workflow scheduling. The Minimum Completion Time (MCT) is estimated by the Min-Min algorithm for every task/jobs in entire resources. The job scheduled the resources which are optimal execution times, Max-Min and Min-Min are similar algorithms except, for a task within the overall maximum completion time which is completed. To attain the user-defined requirements, QoS parameter scheduling continuously tries to meet within budget and deadline is shared. For the completion of request the user need to wait for a specified amount of time to receive the results, the maximum amount of time its need is called Deadline. The maximum amount of money when the user is needed to use the resource is called Budget. In workflow QoS-constrained, scheduling is correlated to best effort scheduling and scientific real-world application. Many guided random searches exists, there are Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), have been used to handle the problem of workflow scheduling with multiple constraints.

Figure 1: Taxonomy of Scheduling

Usually acceptable answers provided by guided random searches in the cloud platform for the initialization phase need more time consuming algorithm to reach the final answer. The researchers surveyed different algorithm approaches, in terms of budget and makespan optimization. The algorithm LOSS and GAIN which are interrelated in the means of the budget and makespan, its find the best schedule to meet these constraints. We implement the GAIN and IC-PCP algorithm. The various constraints such as throughput scalability, resource utilization, makespan, cost, migration time, and fault tolerance are the main consideration of available task scheduling process in the cloud system.

3. Application and System Models

However, convoluted tasks, which contains lots of interconnected applications, called workflows, it’s designed as a Directed Acyclic Graph (DAG) \( D = (T, T_d) \) Tis the tasks set that encompass the workflow and, \( T_d \) tasks dependencies. The
DAG edge’s represent the dependency, where the edges \( e_{i,j} = (e_t, e_t) \), \( e_t \in E \) and \( e_t \neq \emptyset \). The tasks \( e_t \) and \( e_t \) are interrelated task, while the \( e_t \) consume the data originated from \( e_t \) for the execution. So \( e_t \) can start only after the completion of \( e_t \) and the recent data generated is moved to the location where the \( e_t \) well executed. In DAG model is consist an entry task and exit task. The entry task (\( t_{\text{entry}} \)) task which is without parent node and exit task (\( t_{\text{exit}} \)) task which without child node. They’re also possible more than one \( t_{\text{entry}} \) or \( t_{\text{exit}} \) tasks, as like Figure.2 (a). For the running of the algorithm, it added a two dummy task (\( t_{\text{entry}} \) dummy, \( t_{\text{exit}} \) dummy) which no cost and no impact, and for ensuring it has 1 input and 1 output (Figure.2 (b)). The entire DAG workflow total completion time is the schedule length or makespan. If the makespan or overall execution time is less than the user-defined deadline, then we can say that the scheduling is successful and completed. The cloud environment is a combination of different characterized instances. Similarly, our platform is a heterogeneous structure. A set of storage services and evaluating units are offered by the cloud provider, it’s have diverse parameters such as different CPU, processor, different memory size, and different charges. Each workflow Gw has a deadline \( d(Gw) \) related to it. It chooses, completion time that time to overall execution (makespan), from the initial stage of task schedule.

Then later controls and maintain the execution of the scheduling, VM task scheduling, and for dispatches and schedules the tasks for completion.

A Cloud provider (cp) provides a number of (n) virtual machine (VM) representation by (vm) \( = 1, 2, \ldots, n \). Each VM characteristic is different, in the terms of cost and the amount of resources affords. Let the cost factor, denoted by \( C = c_1, c_2, \ldots, c_n \) be the related with each VM usage. The early start time (\( est(t) \)) and latest finish time (\( eft(t) \)) are the important factors in the task scheduling process.

The early start time (\( est(t) \)) represents a task initialization point, where the task begins which occurs only after the completion of the parent task. The latest finish time (\( eft(t) \)) represents the a task finalization point, where the task ends without missing the deadline, which occurs only after the completion of child task. Formally, \( est \) and \( eft \) are defined the scheduled time \( s((t)) \) of a task \( t \) the scheduling time for completion.
Task Scheduling Based IC-PCP and GAIN

The IC-PCP and GAIN are the optimized heuristic-based algorithm. The aim of the task scheduling is within the limited resource's schedule the task with minimum makespan to reduce total execution time using limited resource or budget. IaaS cloud Partial Critical Path (IC-PCP) is a static multi-objective scheduling which considers both cost and time.

The GAIN approach main intends is to minimize the makespan, as this way all tasks are scheduled to the resources, that re-scheduled by the weight value of GAIN/LOSS. GAIN has short time complexity and easy to implement. To assess the performance IC-PCP, and GAIN was selected. As the GAIN algorithm approach invented and best suit in a grid environment, but in this paper altered the algorithm to well out fit in the cloud environment.

The Algorithm
IC-PCP (IaaS Cloud Partial Critical Path)

IC-PCP? (IaaS Cloud Partial Critical Path) is a 1 phase static algorithm which uses a comparable strategy to the deadline distribution stage of the PCP algorithm, except that it really as sign each workflow task, as an alternative of sub deadline assigning.

IC-PCP Algorithm

1: procedure SCHEDULE_WORKFLOW(G(T,E), D)
2: determine available computation services
3: add \( t_{min}, t_{max} \), and their corresponding dependencies to \( G \)
4: compute \( EST(t_i), EFT(t_i) \), and \( LFT(t_i) \) for each task in \( G \)
5: \( AST_{t_{min}} \) \( \leftarrow 0 \), \( AST_{t_{max}} \) \( \leftarrow D \)
6: mark \( t_{min} \) and \( t_{max} \) as assigned
7: call AssignParents\( (t_{min}) \)
8: end procedure

The IC-PCP algorithm initialize by discovering the tasks which are in the critical path (PCP), the critical tasks, where belonging to the critical path. It’s associated with the workflow exit node (no child tasks node is exit node). The critical path lies the task (PCP) thereby scheduling all the tasks in the PCP to the cheapest already leased applicable instance of the virtual machine, which can finish them within the deadline of the workflow. The instance, before the latest finish time (lft) all task can assign based on their path, if though any reason, if
this cannot be accomplished, the lft is leased and assigned the path to the cheapest instance.

At the end of this process, each task has been assigned to a VM and has a start and end times associated with it. Additionally, each VM has a start time determined by the start time of its first scheduled task and an end time determined by the end time of its last scheduled task. Recursively, until the execution of the all tasks, the process continued.

\[ \text{Algorithm for Parent Assignment} \]

1. Procedure ASSIGN PARENTS(a)
2. While (t has an unassigned parent) do
3. \[ \text{PCP} = \text{null, } \text{a} = \text{null} \]
4. While (unassigned parent of a exists) do
5. add CriticalParent(a) to the beginning of PCP
6. set CriticalParent(a)
7. endwhile
8. Call AssignPath(PCP)
9. for all (a, \& ppc) do
10. Update EST and EFT for all successors of a
11. Update LFT for all predecessors of a
12. Call AssignParents(ai)
13. end for
14. end while
15. end procedure

\[ \text{Algorithm for Path Assigning} \]

1. procedure AssignPath (ai)
2. ty = the cheapest applicable existing instance for a
3. if (ty is null) then
4. launch a new instance ty of the cheapest service a, which can finish each task of a before its LFT
5. end if
6. schedule a on ty and set \( \text{AST} (a) \) for each ti \& ia
7. set all tasks of P as assigned
8. end procedure

\[ \text{(AST(ti)}): \text{Actual Start Time of ti,} \]

4. Gain Approach

As originally the GAIN algorithm approach was modelled as a set in grid Environment, but we transformed the algorithm to better adapt to the Cloud Environment. The main objective of the algorithm designed to satisfy the budget constraints. As the way the algorithm is compressed in the two terms “best assignment “and “affordable assignment” concentrates on minimum execution time with minimum cost, the cost does not exceed the overall budget available. Its primary step is the initial schedule of the tasks onto machines and computers. The each task in the diverse machine, there is a weighted value linked with each rescheduling for a particular change. Those weight values are categorized and
tabulated; thus, a weight table is created for each and all tasks in the DAG and each machine also. The algorithm shows another alteration by keeping re-assigning tasks to the machine.

Where there is going to be the biggest benefit in makespan.

\[
\text{GAIN Weight}(i,n) = \frac{T_{old} - T_{new}}{C_{new} - C_{old}} 
\]

For this purpose, the algorithm reorganizes the tasks on different instances using the (1a) equation. Where T\text{old} is the makespan of t\text{j} (total execution time) and C\text{old} is cost of t\text{j} on the scheduled instance by the primary assignment, respectively. T\text{new}s the execution time of t\text{j} on resource n. The GAIN algorithm continuously attempts to assign the task in dissimilar steps, with the minimum makespan until the deadline meets. As a result, the GAIN approach uses the existing instances without the prioritization and it can use the instances without limitation. So the algorithm can acquire more instances its directly reflects the total execution time.

Algorithm 2: The Gain Algorithm

```java
1: procedure FIND INSTANCE DAG (WORKFLOW) W,
2: WITH TASK EXECUTION WITH TIME
3: Procedure FIND SET OF MAChINES WITH COST OF
4: EXECUTING JOBS
5: for all task \( i \in \text{DAG} \) do
6: create schedule \( S \) as map each task onto the
7: cheapest machine
8: Build an array \( A[f, m] \) [number of tasks \( f \)]
9: for each Task in \( W \)
10: for each Machine()
11: if, according to Schedule \( S \), Task is
12: assigned to Machine then
13: \( A[\text{Task}][\text{Machine}] = 0 \)
14: else compute the Weight for a \( \text{Task} [\text{Machine}] \)
15: end for
16: task level task classification (defined in (2))
17: for each Task in all levels in DAG \( W \) for each
18: Machine
19: if, depends to Schedule \( S \), Task \( (t) \) is assigned to
20: Machine then
21: \( A[\text{Task}][\text{Machine}] = 0 \)
22: else compute the Weight for a \( \text{Task} [\text{Machine}] \)
23: end for
24: end for
25: if (gain and cost of \( S = B \))
26: then invalidate previous reassignment of
27: Task i to Machine j;
28: end while
29: if (cost of schedule \( S > B \))
30: then use cheapest assignment for \( S \).
31: Return \( S \)
```

5. Performance Evaluation and Results

For acquire and analyze the results of the algorithm, we create a simulator called CloudSim 2.1.1 on LINUX operating system with Core X86 processor with a Xen virtual machine. To run the CloudSim 2.1.1 NetBeans IDE 8.2 is used. In the simulation modeled, using cloudsim toolkit and the heuristic based task scheduling algorithm is applied, and a comparative examine has been made
among two algorithms; IC-PCP and GAIN. We compare and evaluate the algorithm as consider the makespan constraint based on the different cloudsim parameters such as Cloudlet (cloudlet length), Virtual machine (VM size, MIPS, bandwidth), datacenter (datacenter number) with dissimilar sizes, computing speed, performance etc.

Table 1: (a) shows the parameter setting details of cloud simulator. In this, we take the different parameters and check the efficiency of both algorithms by changing these parameter settings and compares the results.

Simulation Output of IC-PCP and Gain

IC-PCP Scheduler Output

Gain Scheduler Output
Experimental Results

MAKE SPAN: The following line graph represents the comparative examine the results of the IC-PCP and GAIN assembled on the minimized makespan constraints as consider the cloud parameters such as Number of VMs, cloudlet length, Vms Mips, VMs size, bandwidth.

Figure 4(a): Makespan based on Cloudlet Length

Figure 4(b): Makespan based on VM’s Number

Figure 4(c): Makespan based on VM’s MIPS Parameter
Summary of Observations

The above simulation shows the comparative analysis of algorithm IC-PCP and GAIN, in this paper is able to find a schedule with better makespan as consider the different cloudsim parameters. The above figure 4 (a) (b) (c) (d) (e) the line graph results of the simulation. As the simulation results, figure 4 (a) shows the overall execution time, (a) based on the cloudlet length (b) based on the number of VMs (c) based on VM’s MIPS, (d) based on VM’s Size (e) based on bandwidth. As the figure (a) we consider the different cloudlet length (50000, 60000, and 70000). In these, the IC-PCP algorithm shows the better performance than GAIN. IC-PCP completes the scheduling process with minimum makespan. GAIN tries to achieve the minimum makespan ease with IC-PCP, but comparatively, IC-PCP shows better makespan. As figure (b) based on the number of virtual machines (1,2,3..), initially the IC-PCP and GAIN take consistent minimum schedule, as increase the number of VM ’s, GAIN approach fails to meet the minimum makespan. As the figure (C) based on the
VM MIPS (200, 250, 300...), as increase the MIPS, it’s consistently reflected the IC-PCP and GAIN, both algorithm shows similar performance. As the figure (d) (e) based on the VM’s size (5000, 10000,..) and bandwidth (500, 1000..), increase VM size, and bandwidth, IC-PCP takes more time to execute the schedule, and GAIN considerably takes minimum makespan than IC-PCP.

6. Conclusion

The paper compares and evaluate the performance of heuristic task scheduling algorithm GAIN and IC-PCP with different cloudsim scheduling parameters. The main objective of the two algorithms is to reduce the total execution time (MAKESPAN) using limited resource and budget. The IC-PCP shows the minimum makespan as depends on the cloudblet length parameter (figure: 4 (a)) and the number of VM’s (figure 4 (b)). The parameters such as datacenter number, RAM, CPU number in this all parameters IC-PCP shows the consistent results as better than the GAIN algorithm. GAIN shows the great impact based on the parameter such as VM's size (figure: 4 (d)) and bandwidth ((figure: 4 (e)). The GAIN algorithm schedules the task with minimum time with drastic influence on the overall makespan than IC-PCP. For an initial startup IC-PCP takes more time to schedule the task. But the end of the schedule IC-PCP track the minimum makespan as with GAIN. In VM MIPS both algorithms show similar performance. (Figure: 4 (c)). It can be concluded that on a cloud platform simulated by Cloudsim, the IC-PCP shows the comprehensive efficiency of the entire platform in almost all parameters. Although several factors considered like the size of the tasks, the processing capacity of the virtual machines, effects of bandwidth and data transmission. Processor number, transmission speed etc.

References


