Comparative Study of Workflow Scheduling Algorithms in Cloud Computing

Santhosh B¹, Harshitha², Prachi Kaneria A³, Dr D.H Manjaiah⁴
Assistant Professor, Department of MCA, AIMIT, St Aloysius College, Mangalore, India¹
Msc 3rd semester, Department of Software Technology, AIMIT, St Aloysius College, Mangalore, India², ³
Professor, Department of Computer Science, Mangalore University, Mangalore, India⁴

ABSTRACT: Cloud computing is a new terminology which is achieved by Distributed, Parallel and Grid computing. Cloud computing provides different types of resources like hardware and software as services via internet. Efficient task scheduling mechanism can meet users’ requirements, and improve the resource utilization, thereby enhancing the overall performance of the cloud computing environment. In this paper we have considered workflow scheduling and conducted comparative analysis of workflow scheduling algorithms. Here Tasks require minimum completion time, better performance and total cost of execution is considered.

I. INTRODUCTION

Cloud computing now is known as a provider of dynamic, scalable on demand, virtualized resources over the internet. It also makes it possible to access applications and associated data from anywhere. Companies are able to rent resources from cloud for storage and other computational purposes so that their infrastructure cost can be reduced significantly. They can also make use of software, platforms and infrastructures as a services, based on pay-as-you-go model.

INTRODUCTION TO WORKFLOW MANAGEMENT SYSTEM

Workflow Management System (WMS) is used for management of executing the workflow tasks on the computing resources. The major components of WMS are shown in Figure 1.

Workflow Management System

Workflow Design Information Retrieval Workflow Scheduling Fault Tolerance Data Movement

Fig 1: Elements of a WMS [1]

Workflow design describes how components of workflow can be defined and composed. The elements of workflow design as shown in Figure 2.

Workflow Design

Workflow Structure

Non-DAG

Abstract

Concrete

User-directed

Language-based Modeling

Graph-based Modeling

DAG

Sequence

Parallelism

Choice

Sequence

Parallelism

Iteration

Workflow Model/Specification

Workflow Composition System

Automatic

Copyright to IJIRCCE www.ijircce.com 31
A workflow structure describes the relationship between the tasks of a workflow. A workflow structure can be of two types: Directed Acyclic Graph (DAG) and Non-DAG. Workflow structure can further be categorized as Sequence, Parallelism, and Choice in the DAG-based structure. In the sequence structure, the tasks execute in a series. A new task can only be executed if the previous task has been successfully executed. In parallelism structure, tasks of workflow can execute concurrently. In choice structure, the workflow tasks can be executed in series as well as concurrently.

The Non-DAG-based structure contains all the DAG-based patterns along with Iteration pattern. In Iteration pattern, the task can be executed in an iterative manner. A complex workflow can be formed with the combination of these four types of patterns in multiple ways.

Workflow model defines the workflow, its tasks, and its structure. Workflow models are of two types; abstract and concrete. Abstract workflow describes the workflow in an abstract form which means that specific resources are not referred for task execution. Concrete workflow describes the workflow tasks bound to specific resources.

Workflow composition system allows users to accumulate components to the workflow. In the user-directed system, users modify the workflow structure directly by editing it in the workflow. But in an automatic system, workflows are generated automatically. User-directed systems are further categorized into language-based modeling and graph-based modeling. In former, users describe workflows using some markup languages such as XML. It is a difficult approach because users have to remember a lot of language syntax. Graph-based modeling is simple and easy to modify the workflow with the help of basic graph elements [1].

1.1 WORKFLOW SCHEDULING

Mapping and management of workflow tasks' execution on shared resources is done with the help of workflow scheduling. So workflow scheduling finds a correct sequence of task execution which obeys the business constraint. The elements of workflow scheduling are shown in the Figure 3.

Scheduling Architecture is very important in case of quality, performance, and scalability of the system. In the centralized workflow environment, scheduling decisions for all the tasks in the workflow are done by a single central scheduler. There is no central controller for multiple schedulers in decentralized approach but communication between schedulers is possible. Each scheduler schedules the workflow to the resource having less load. On the other hand in hierarchical scheduling, there is a central manager. Not only workflow execution is controlled by this central manager but also the sub-workflows are assigned to the lower-level schedulers [1]. Scheduling decisions which are taken on the basis of task or sub-workflow are known as local decisions and which are taken by keeping in mind the whole workflow are called global decisions. Global decisions based scheduling gives better overall results because only one task or sub-workflow is considered in local decision scheduling.

Transformation of abstract models to concrete models is done in two ways; static and dynamic. Concrete models are generated before the execution with the help of current available information about the execution environment and the
dynamic change of resource is not considered in the static scheme. Static schemes further categorized in two types; user-directed and simulation-based. In the user directed planning scheme, decision about resource mapping and scheduling is done on the basis of user’s knowledge, preferences and performance criteria. But in case of simulation based, a best schedule can be achieved by simulating task execution on resources before the workflow execution starts. Both static and dynamic information about resources used for making scheduling decisions at run-time are considered in the dynamic scheme. For dynamic schemes, there are two approaches: prediction-based and just-in-time scheduling. In prediction-based, dynamic information is used along with some results on the basis of prediction and in simulation-based static scheduling, in which prediction of performance of execution of tasks using resources is done along with generation of optimal schedule before the execution of task is done. But due to this, the initial schedule changed during the execution. Just in-time scheduling makes a decision at execution time. Scheduling strategies are categorized into performance driven, market driven, and trust driven. In case of performance driven strategies, focus is to achieve the highest performance for the user defined QoS parameter. So the workflow’s tasks mapped with resources gives the optimal performance. Most of the performance driven scheduling strategies focus to maximize the makespan of the workflow. Market driven strategies focus on resource availability, allocation cost and quality for the budget and deadline. A market model is used to schedule the workflow tasks to the available resources dynamically which results in less cost. Trust driven focuses on security and reputation of the resources and tasks are scheduled based on the trust by considering these two parameters [1].

II. WORKFLOW TASK CLUSTERING

Task clustering is basically used to combine the small sized tasks of the workflow into a comparatively large sized task to minimize the makespan of the workflow by reducing the impact of queue wait time. So task clustering restructures the workflow by combining the small sized tasks into a cluster and execute this cluster as a single task. Due to clustering, the number of tasks in the workflow is reduced and there is less queue wait time as compared to small sized tasks. Task clustering is categorized as level- and label based clustering [3], vertical, blocked and balanced clustering [4]. Figure 4 represents a simple workflow. Figure 5 describes horizontal clustering in level 2 and level 4 with each cluster containing two tasks. Figure 6 shows vertical clustering and Figure 7 shows blocked clustering.
Level-based clustering is also called horizontal clustering. In this type of clustering the tasks are independent and the tasks of the same level of the workflow can be combined together. In vertical clustering, tasks of the same pipeline can be combined together. Blocked clustering represents the combination of both horizontal and vertical clustering in the same workflow. Figure 6 represents vertical clustering, in which tasks in the same pipeline can be combined together to form a cluster. Each cluster of vertical clustering contains three tasks. Figure 7 shows the blocked clustering, in which horizontal and vertical clustering can be combined together. Tasks in the same level of workflow can have different execution time and whenever these tasks are combined without the consideration of their runtime variance then it causes the problem of load imbalance, i.e., some cluster may contain the smaller tasks and other may have larger tasks. Due to this inappropriate clustering, workflow overhead is produced. Another problem is data dependency between the tasks of workflow in case of level-based clustering which is called dependency imbalance.

III. SCHEDULING ALGORITHMS

The main objective of scheduling algorithm is to achieve the best system throughput with proper resource utilization and high performance by obeying the user's specified QoS parameter. There are various time comparison based scheduling heuristics exist in the grid and Cloud computing environment, which schedules the tasks by comparing the arrival time or execution time of the tasks. In the following section these scheduling heuristics are described.

4.1 First Come First Serve (FCFS)
In this algorithm, tasks are compared on the basis of their arrival time and the task which comes first in the ready queue is served first. Advantage of this algorithm is its simplicity and fast execution behavior. But the main disadvantage of this algorithm is that sometimes due to the execution of a longer job, which comes in the queue first, small jobs have to wait for its completion. Due to this problem the waiting time of tasks increased and overall performance of the workflow execution decreases.

4.2 Min-min
In this algorithm, small task is executed first so that large task delays for long time. Algorithm begins with by sorting the set of all unmapped tasks in increasing order of their completion time. Then the tasks having the minimum completion is scheduled from the unmapped task set and the mapped task has been removed from unmapped task list, and the process repeats until all the tasks of unmapped list is mapped [5].

4.3 Max-min
In this algorithm, large task is executed first so that small task delays for long time. This algorithm is very similar to Min-min algorithm, instead of sorting the task in the increasing order of completion time. This algorithm sorts the tasks in decreasing order of their completion time. Then the task with the overall maximum completion time is selected from this task list and scheduled to the corresponding available resource. Then the scheduled task has been removed from unmapped task set and the process repeats until all tasks of unmapped list is mapped [5].

4.4 Minimum Completion Time (MCT)
In this algorithm, task that takes least time to complete is allocated to a machine randomly. So MCT behaves somewhat like Min-min. However, Min-min algorithm considers all the unmapped tasks during each mapping decision but on the other hand MCT considers only one task at a time [6].

IV. EXPERIMENTAL RESULTS

The paper uses simulation to test and verify the efficiency and correctness of the scheduling algorithm presented in this paper.

5.1 Simulation setup
The proposed algorithm is simulated in a simulation toolkit workflowsim[7]. We have created a datacenter with the following properties and created 20 virtual machines.
5.2 Analysis of results
The fig 8 shows workflow of task dependency in CyberShake_50 and fig 9 shows simulation results when MINMIN simulation algorithm is used. Here the total cost is calculated using the formula:

\[
\text{cost} = \text{(communication cost + computation cost)} = (\text{data (both input and output)} \times \text{unit cost of data} + \text{runtime} \times \text{cpu cost}) \text{ for each task}
\]

The fig 8 shows workflow of task dependency graph. The fig 9 shows simulation results when MINMIN simulation algorithm is used. Here the total cost is calculated using the formula:

\[
\text{cost} = \text{(communication cost + computation cost)} = (\text{data (both input and output)} \times \text{unit cost of data} + \text{runtime} \times \text{cpu cost}) \text{ for each task}
\]
The table shows the makespan and total cost of scheduling of tasks using the algorithms FCFS,MCT,MAXMIN and MINMIN with Montage_100 and CyberShake_50 datasets

<table>
<thead>
<tr>
<th>Scheduling Algorithms</th>
<th>Data set</th>
<th>Make span time</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS</td>
<td>Montage_100</td>
<td>114.2</td>
<td>533.58</td>
</tr>
<tr>
<td>MCT</td>
<td>Montage_100</td>
<td>114.2</td>
<td>533.58</td>
</tr>
<tr>
<td>MINMIN</td>
<td>Montage_100</td>
<td>115.19</td>
<td>534.56</td>
</tr>
<tr>
<td>MAXMIN</td>
<td>Montage_100</td>
<td>113.55</td>
<td>534.7</td>
</tr>
<tr>
<td>FCFS</td>
<td>CyberShake_50</td>
<td>851.2</td>
<td>5996.73</td>
</tr>
<tr>
<td>MCT</td>
<td>CyberShake_50</td>
<td>851.2</td>
<td>5996.73</td>
</tr>
<tr>
<td>MINMIN</td>
<td>CyberShake_50</td>
<td>851.2</td>
<td>5996.73</td>
</tr>
<tr>
<td>MAXMIN</td>
<td>CyberShake_50</td>
<td>846.95</td>
<td>5996.69</td>
</tr>
</tbody>
</table>

Table 1: Results of various scheduling algorithms with different datasets

Our experiment shows that MAXMIN is the most efficient with respect to the makespan and total cost in comparison to the other algorithms FCFS,MCT,MINMIN.

V. CONCLUSIONS

Minimizing makespan is one of the objectives of scheduling algorithm. The experiments shown MAXMIN is the most efficient with respect to the makespan and total cost in comparison to the other algorithms FCFS,MCT,MINMIN.

ACKNOWLEDGEMENT

The authors would like to thank the dedicated research group in the area of Cloud & Grid Computing, wireless networks at the Dept of Computer Science, AIMIT, St Aloysius College, Mangalore, India, for many stimulating discussions for further improvement and future aspects of the Paper. Lastly but not least, the author would like to thank everyone, including the anonymous reviewers.

REFERENCES

