Applied Mathematical Sciences, Vol. 9, 2015, no. 46, 2273 - 2280
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2015.53208

Scheduling of Scientific Workflows in Cloud with Replication

J. Angela Jennifa Sujana

Department of Information Technology
Mepco Schlenk Engineering College, Sivakasi, TamilNadu, India

T. Revathi

Mepco Schlenk Engineering College, Sivakasi, TamilNadu, India

M. Malarvizhi

Department of Information Technology
Mepco Schlenk Engineering College, Sivakasi, TamilNadu, India

Abstract

Cloud infrastructure is rapidly scalable, accessible and also it is a pay as we use model. Due to the elastic nature of cloud, the resource provisioning can be done dynamically according to the requirements of the workflow application. We aim at executing the scientific workflows within the user-defined deadline, with the smaller possible cost by reducing the total execution time of applications. The execution time of the task is reduced by proper provisioning and scheduling of all tasks in the scientific workflows. The proposed algorithm RIPCP (Replication based IC-PCP) uses the principle of IC-PCP algorithm for finding the Partial Critical Path (PCP) of the tasks and includes the task replication for better make span. Three different Scientific Workflows are simulated in different characteristics of virtual machines to demonstrate that the proposed algorithm reduces the total execution time of the applications.

Keywords: Scientific Workflow, Cloud Computing, replication

1 Introduction

In the development of scientific applications, workflow model is applied in many areas such as astronomy, bioinformatics, and physics. Each workflow can consist of hundreds or thousands of tasks. So, these type of workflow based applications can be benefited by large scale infrastructures, like Cloud computing. Cloud computing provides pay-per-use system and also poses dynamic scaling in response to the needs of the application. So, the resources for executing the workflow can be provisioned on demand and also its number can be increased until it contains enough budget to support it [3]. In cloud utilization model users obtain virtual machines as resources for deploying the applications, which is called Infrastructure as a Service (Iaas) [2]. So, Clouds provide suitable platform for deadline constrained scientific workflows [5].
The budget allotted for the execution of the workflow is the key constrain, which may limit the number of resources we hire. It is because that the cloud providers charge resource utilization by integer time interval. Our work aims at scheduling the workflow, such that the execution is completed before the deadline and within the budget constraint.

2 Related Works

Shi and Dongarra proposed an algorithm for efficient scheduling of workflow applications represented by weighted directed acyclic graphs (DAG [4]. This algorithm works based on priority and it is named as SDC which is a new static list Scheduling algorithm for heterogeneous processors. However, they are not able to decide the optimal number of resources for reducing the cost of the infrastructure used. So, it is not suitable for elastic cloud using a pay-per-use system.

In the work done by Abrishami et al [1] proposed two workflow scheduling algorithms for the Cloud environment by adapting the PCP(Partial Critical Path) algorithm for utility of Grids, evaluate their performance on some well-known scientific workflows. These algorithms are used for cost optimized and deadline constrained execution of workflows in clouds. But these algorithms do not consider data transfer times during provisioning and scheduling and also increases the execution budget.

Lin and Lu [2] propose SCPOR, a scientific workflow scheduling algorithm that is able to schedule workflows in need of elastically changing compute resources. But it misses the capability of considering the cost for utilization of resources.

Buyya et al describes an algorithm for scheduling of workflows in Grid computing environments [5]. It assumes cooperation model free of financial cost or based on economic model. But the disadvantage is that the budget calculated by this algorithm is differing from the amount charged by the cloud providers.

In order to overcome the limitations, proposed algorithm applies replication of tasks, based on budget constrains such that the tasks are completed before deadline.

3 System Model

The Workflow application is modeled by a Direct Acyclic Graph (DAG) G= (T, V) where T is a set of tasks and V is the set of dependencies between the tasks. Each dependency are in the form of edges $v_{i,j}=(t_i,t_j)$ which indicates that task t_i should complete executing before task t_j can start. We consider the task t_{entry} represents the common entry task for the workflow and the task t_{exit} represents the common exit task for the workflow. If there is no common entry task or exit task for the workflow, we insert dummy tasks t_{entry} and t_{exit} and the execution time for these tasks are 0. The set of ancestor and descendant tasks are represented by *ancestor* (t_j) and *descendant* (t_j) and each workflow has a deadline *D (G).* Deadline determines the time for completing the execution of workflows.
The cloud provider provides the resources as virtual machines (VM) and the VMs are of different types like the Amazon AWS EC2 Instance types. Let VM= {vm_1, vm_2,…,vm_n} represents the set of n and each is of different type. VMs are charged per integer amount of units and there is no limit in the number of VMs for the execution of the workflow. The *Execution time matrix (E)* is generated which contains execution time of each task in each virtual machine. In this matrix an element e_{ij} denotes the execution time of a *task i* in the *VM type* vm_j. E_{min} is the minimum execution time of a task in the matrix E. *Data Transfer Time (DTT)* is the time required for transferring the data of a task which is running in one VM required by another task which is running in another VM. If an ancestor and its immediate descendant are allocated in the same VM, then *DTT=0.* The parameter *EST and LFT denotes the Earliest Start Time and Latest Finish Time* of an task in the workflow respectively.
EST and LFT are defined as,

$$EST(t_j)=\begin{cases} 0, & if\ t_j = t_{entry} \\ \max_{t_a \epsilon ancestor(t_j)} \left(EST(t_a) + E_{min}(t_a) + DTT(v_{a,j})\right), & otherwise \end{cases} \quad (1)$$

$$LFT(t_j)=\begin{cases} D(G), & if\ t_j = t_{exit} \\ \max_{t_a \epsilon descendant(t_j)} \left(LFT(t_a) - E_{min}(t_a) - DTT(v_{a,j})\right), & otherwise \end{cases} \quad (2)$$

The *Schedule time (ST)* is the time at which the task has been scheduled for execution and it can assume any value between EST and LFT. The aim of the paper is to execute the workflow G in the cloud within the deadline D. This is achieved by proper allocation and scheduling of tasks in the workflow. It uses the principle of IC-PCP algorithm for finding the Partial Critical Path (PCP) of the tasks

and includes the task replication for better make span, so the proposed algorithm is named as RIPCP (Replication based IC-PCP). If both the ancestor and descendant task gets executed in the same virtual machine then the data transfer time between these tasks becomes zero.

3.1 System Architecture

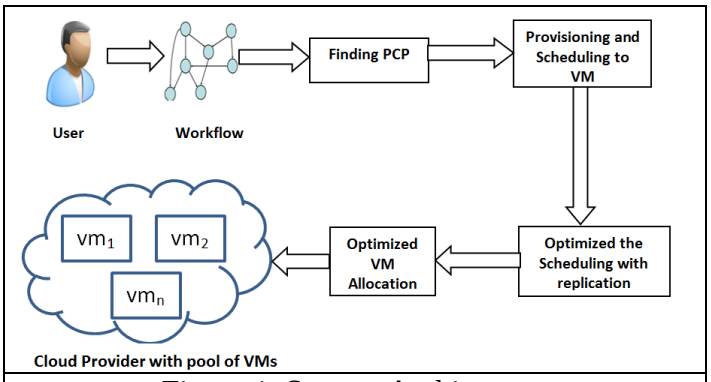

Figure 1. System Architecture

Figure 1 represents the system architecture of the proposed system. The user will submit the workflow, which has to be executed in the cloud. The Partial Critical Path (PCP) of the workflow will be first found. We do this to allocate all the tasks in the PCP to the same VM in order to reduce the data transfer cost. We find the optimized schedule and then try to further optimize it by further replicating the tasks to additional VMs. The number of additional VMs will be decided by the budget limit that is available for executing the workflow. Then based on the generated schedule the VM allocation is done.

ALGORITHM 1-Finding the partial critical path of a task t

S: Unscheduled tasks of the workflow G
Initialize PCP=null;
While (!isEmpty(S))
{ for each task (t_a) ϵ to ancestor (t)
{ Set readytime = -1
if ($EST(t_a)+E_{min}(t_a)+DTT(t_a,t)$ > readytime)
{ readytime= $EST(t_a)+E_{min}(ta)+DTT(ta,t)$;
ancestor =ta; }
}
PCP= ancestor $\cup$ PCP;
Remove ancestor from the unscheduled list S.
Set task t_a = ancestor }
return PCP for the particular task t_a .

ALGORITHM 2: Scheduling and allocation of the task in Virtual Machines

Input : PCP of a task from algorithm 1
for each allocated VM arranged in ascending order of cost
Assign $S_{beg} = PCP \cup S$
Calculate $LFT_{beg}(t)$ for the new schedule $S_{beg.}$
If (execution order or $LFT_{beg}(t)$ is not satisfied) then
Assign $S_{end} = S \cup PCP$
Calculate $LFT_{end}(t)$ for the new schedule $S_{end.}$
If (execution order or $LFT_{end}(t)$ is not satisfied) then
Create the cheapest VM that is able to execute each task t in PCP
before its LFT and allocate the PCP in the created VM
end if
else
Put the PCP at the end of the schedule
end if
Put the PCP at the beginning of the schedule
Allocate the PCP to the VM in the particular chosen position
end for
for each task t from PCP
Update the EST and LFT of each task
Recursively Apply algorithm 2 on t
end for

3.2. Task Replication

In order to complete the task within the stipulated deadline, the scheduling process must select the appropriate virtual machines. The VMs must be ready to receive the data and execute the tasks in the moment they are received and complete it within the deadline. RIPCP(Replication based IC-PCP) tries to mitigate such effects with the utilization of task replication in idle slots of provisioned VMs or on new VMs allocated for enabling extra replication (if the replication budget allows). The goal of this replication is increasing performance rather than fault tolerance. Hence space replication is the goal of RIPCP. Therefore, tasks are only replicated on different VMs, rather than time replication approach where the same task could be scheduled multiple times in a single VM to increase fault-tolerance. The Replication algorithm is explained in the algorithm 3. Initially, the algorithm generates new VMs based on the available replication budget.

Algorithm 3: Task Replication RIPCP

```
rb: Replication budget.                VM: List of provisioned VMs.
 T: Set of tasks from G.                RS: List of Reserved idle slots.
 US: List of unused idle slots.
 V = reverseSort( VM, Schedule order );
Stack st= Createstack();
 while (rb !=0)
{   for all VMs in V
       vm = choose the cheapest VM from V ;
        st.push(vm);
       while(!st.isEmpty())
        {       rm=st.pop();
                vm=  Replicate   rm;
                V=V+vm;  RS=RS+RS_vm;
                rb =rb - cost of vm;         }}
Sort (RS);  Sort(US);  RS = RS ∪ US U;
 Sort (t, replication precedence);
for each slot s in RS
 {   for each task t in T {
         if t fits s then
              Create replica t' of t;
              Schedule t' in s, respecting EST(t);
              Update slots information;
              Move t to the end of T;
             if s is unpaid then
             { Update budget information;
               Update Reservation status of slots deriving from s;
               Update provisioning information of VM that contains s; }}}}
```

4 Results and Discussion

We have simulated the proposed algorithm RIPCP using cloudsim by using three realtime workflows [6] namely Montage (generation of sky mosaics), CyberShake (earthquake risk Characterization risk characterization) and LIGO (detection of gravitational waves).

4.1 Comparison between IC-PCP and RIPCP

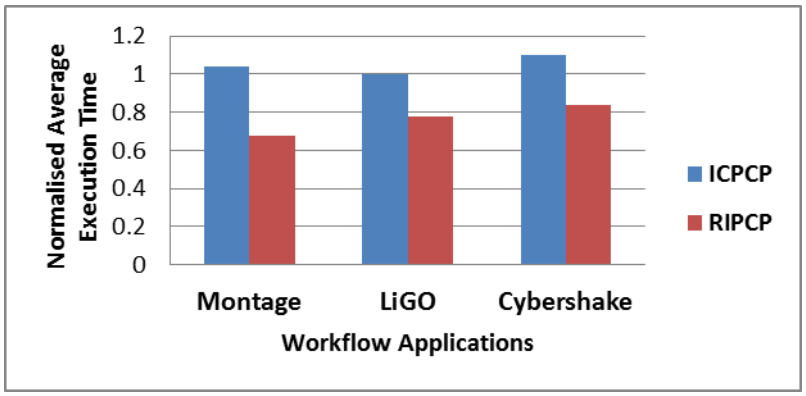

Figure 2. Average execution time using ICPCP and RIPCP

We have used six types of VM which is based on Amazon AWS EC2. The billing time is considered for 1 hour. The deadline is user defined and it is based on some criteria, typically the base run time. The base runtime is the execution time of each workflow task to the most powerful virtual machine (m3.xxlarge). Then the deadline is by multiplying the base value with the number of tasks in the workflow.

5 Conclusion

Scientific Workflows provide a set of characteristics that make suitable for execution in Cloud infrastructures, which offer on-demand scalability so that the resources are increased and decreased based on the demand of applications. Experiments using three well-known scientific workflow applications showed that the RIPCP algorithm increases the chance of deadlines being met and reduces the total execution time of workflows. This is implemented in single cloud. As future work, we will increase the capabilities of the RIPCP algorithm across multiple Clouds.

References

[1] S. Abrishami, M. Naghibzadeh, and D. Epema, Deadline constrained workflow scheduling algorithms for IaaS clouds, *Future Generation Computer Systems*, **29-1** (2013), 158–169. http://dx.doi.org/10.1016/j.future.2012.05.004

[2] C. Lin and S. Lu, SCPOR: An elastic workflow scheduling algorithm for services computing, *Proceedings of the International Conference on Service-Oriented Computing and Applications (SOCA)*, **1 (**2011). http://dx.doi.org/10.1109/soca.2011.6166213

[3] Rodrigo N. Calheiros and Rajkumar Buyya, Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication, *IEEE transactions on parallel and distributed systems*, **25-7** (2013), 1787 – 1796. http://dx.doi.org/10.1109/tpds.2013.238

[4] Z. Shi and J. J. Dongarra, Scheduling workflow applications on processors with different capabilities, *Future Generation Computer Systems*, **22-6** (2006), 665–675. http://dx.doi.org/10.1016/j.future.2005.11.002

[5] J. Yu, R. Buyya, and K. Ramamohanarao, Workflow scheduling algorithms for grid computing in *Metaheuristics for Scheduling in Distributed Computing Environments, F. Xhafa and A. Abraham , Eds. Springer*, 2008. http://dx.doi.org/10.1007/978-3-540-69277-5_7

[6] The XML files describing the applications are available via the Pegasus project: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

Received: December 15, 2014; Published: March 21, 2015