Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 50 (2015) 353 – 356

Procedia Computer Science

2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

Efficient Scheduling Algorithm for Cloud

Ellendula Madhukara, Thirumalaisamy Ragunathanb

aSreenidhi Institute of Science and Technology,Hyderabad,India
bACE Engineering College,Hyderabad, India

Abstract

In the emerging e-commerce scenario, information captured by enterprises through their web-based applications has been exploding. Extensive use of applications based on multimedia and social media effects in exponential growth of data. Cloud computing system (or simply cloud) provides the required infrastructure for storing the large data generated by these applications. Efficient task scheduling in the cloud will reduce the task completion time and hence improve the performance of the cloud. In this paper, we have proposed an efficient cloud scheduling algorithm based on the results of the analysis performed on the log which consists of previous scheduling information for improving the performance of the cloud.

Keywords: Distributed System; Speculation; Performance;

1. Introduction

In the emerging e-commerce scenario, information captured by enterprises through their web-based applications has been exploding. Extensive use of applications based on multimedia and social media effects in exponential growth of data. Cloud computing system (or simply cloud) provides the required infrastructure for storing the large data generated by these applications. According to [3], a cloud is defined as ”A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort of service provider interaction”. Efficient task scheduling in the cloud will reduce the task completion time so that efficient services can be provided to the users. Ant colony- and simulated annealing-based algorithms, genetic algorithms and hybrid algorithms are the ones proposed in the literature for improving the performance of the cloud.

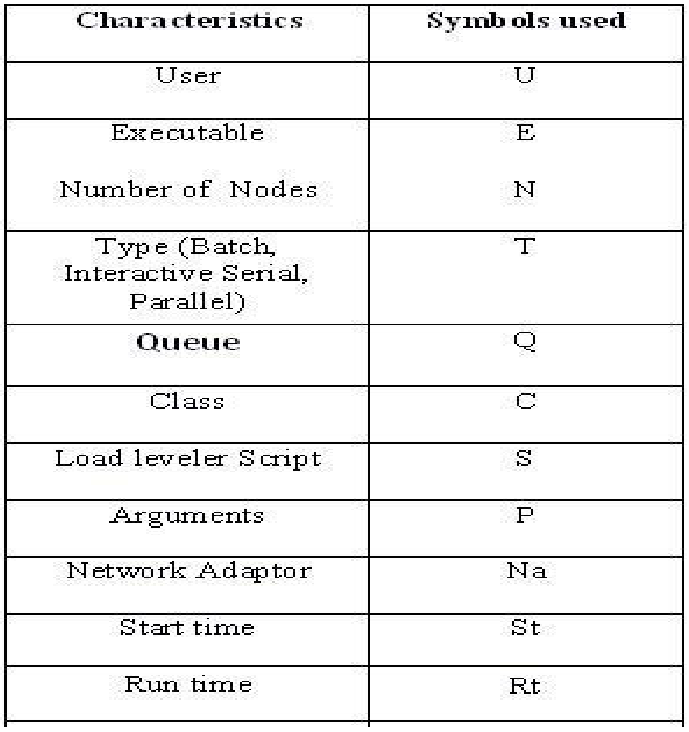
In this paper, we have assumed that a cloud consists of a controller node and one or more compute nodes. The controller node does the control of compute nodes present in the cloud including task scheduling. The compute node store the data and carry out the execution of tasks. Virtual machines (VMs) are created in the compute nodes to execute the user tasks. We have assumed that a fixed number of VMs are available (or created) in the compute nodes and different execution environments are maintained in these VMs. Note that, in a cloud environment VMs carry out the execution of tasks. In this paper, we are proposing that a log

Email addresses: emadhukar@gmail.com (Ellendula Madhukar), ragu_savi@yahoo.com (Thirumalaisamy Ragunathan)

doi:10.1016/j.procs.2015.04.036

is maintained in each compute node and all scheduling information relevant to individual VMs are captured in the log.

In this paper, we have proposed an efficient cloud scheduling algorithm which can improve the performance of the cloud. The proposed algorithm mines the log maintained in the compute nodes to come out with the some scheduling patterns for VMs present in the compute nodes of the cloud. The incoming task to the cloud (controller node) will be scheduled to a particular compute node if the task requirements matches one of scheduling patterns of VMs present in those nodes.



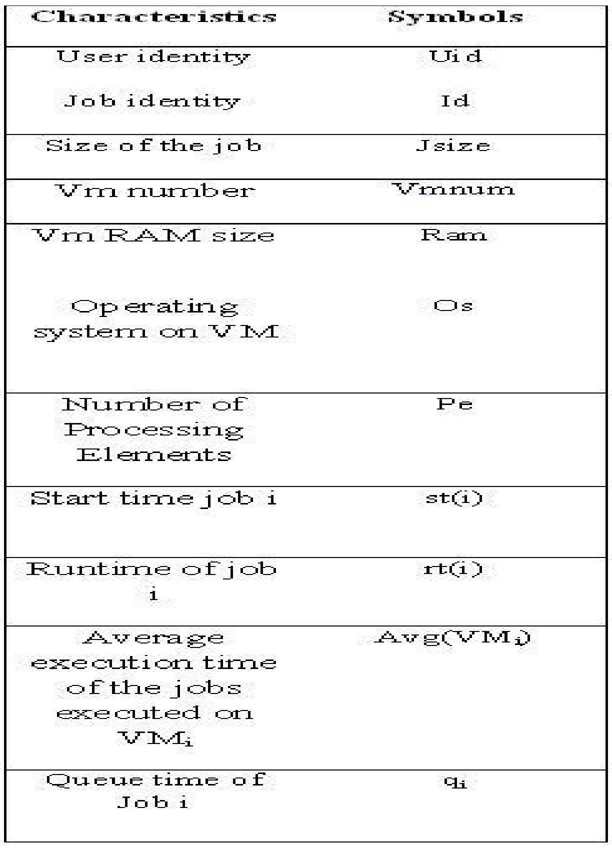
Characteristics	Symbols used
User	U
Executable Number of Nodes	E N
Type (Batch, Interactive Serial, Parallel)	T
Queue	Q
Class	C
Load leveler Script	S
Arguments	P
Network Adaptor	Na
Start time	St
Run time	Rt

Fig. 1: Job Characteristics Proposed by Smith

2. Related Work

In this section, we discuss how job characterization is done in the cloud. Gibson [2] proposes a template model and that has been modified by Smith [4]. The job (or task) characteristics used by Smith is shown in the Figure 1.

Ariel Goyeneche [1] proves that using dynamic templates without normalized data will lead to inaccurate predictions. The authors have used weighted templates. In this method, characteristics of jobs will be considered based on the different weights allocated. Important characteristics will have higher priority and they are mandatory for job performance estimations. The prediction accuracy is based on the weights allocated to characteristics. Autopsy of all templates can be used to fix the weights of templates. In this method, characteristics are divided into two sets. First set comprises of User Id, Job Id, VMnum , Job size, VM ram, VMos(VM Operating system). Second set consists of all the remaining characteristics. The characteristics specified in the first set will have more weights than characteristics specified in the secondset



Characteristics	Symbols
User identity	Uid
Job identity	Id
Size of the job	Jsize
Vm number	Vmnum
Vm RAM size	Ram
Operating system on VM	Os
Number of Processing Elements	Pe
Start time job i	st(i)
Runtime of job i	rt(i)
Average execution time of the jobs executed on VM_i	$Avg(VM_i)$
Queue time of Job i	q_i

Fig. 2: Job characteristics captured in the log

3. Proposed Prediction-Based Scheduling Algorithm

We have assumed that logs are present in all the compute nodes. The format of the log is shown in Figure 2. For a new task (T) submitted in the cloud the following characteristics are considered: User identity, Job identity, Size of the job, Ram requirements and Operating system required.

First, we identify VMs which are matching the requirements of T based on the matching of scheduler patterns. Let us call this list as L. The VMs in L are arranged according to the average execution time of those VMs. Then T will be allocated to the first VM in the list provided the load of that VM is not above the threshold (th) specified. Otherwise, the second VM in the list is considered or the third VM is considered and so on.

We have assumed that scheduler patterns of VMs present in the compute nodes are obtained by mining the log and stored in the list SP.

Proposed Algorithm:

User submits a task (T1) (Characteristics specified are User identity, Job identity, Size of the job, Ram requirements and Operating system)
If T1 characteristics are matching with any one scheduler pattern specified in SP of VMs *then*
Store the VM numbers in the list LVM. Let us assume that *n* umber of VMs are specified in the
list. LVM is sorted according to average response time of VMs
For each VMs specified in the LVM *do*

Allocate T1 to that VM if work load (number of tasks currently getting executed in that VM) is less than
th break.
If no VM in LVM is satisfying the conditions of threshold *then* allocate one VM from SPM randomly.

4. Conclusion and Future Works

In this paper, we have proposed an efficient cloud scheduling algorithm for improving the performance of the cloud. The proposed algorithm mines the log maintained in the compute nodes of a cloud to come out with the some scheduling patterns for VMs present in those compute nodes and the incoming task is allocated to a suitable VM of a compute node. As a part of future work, we wish to implement this algorithm in CloudSim (a tool used for simulating the cloud) and compare its performance with other ant colony-based and simulated annealing-based algorithms.

References

[1] A. G. et.al. Improving grid computing performance prediction using weighted templates.

[2] R. Gibson. A historical application profiler for use by parallel scheduler.

[3] P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing, 2011.

[4] W. Smith, I. T. Foster, and V. E. Taylor. Predicting application run times using historical information. In *Proceedings of the Work-shop on Job Scheduling Strategies for Parallel Processing*, IPPS/SPDP '98, pages 122–142, London, UK, 1998. Springer-Verlag.