International Journal of Advance Research in Science and Engineering
Vol. No.4, Issue 10, October 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

APPLICATION OF CLOUD DISTRIBUTED COMPUTING ON AUTONOMIC RESOURCE ALLOCATION

Kalpana Tiwari[1], Er. Pushpneel Verma[2], Er. Ankur Bhardwaj[3]

[1,2,3]*Department of Computer Science and Engineering*

Bhagwant Institute of Technology, Muzaffarnagar, Uttar Pradesh (India)

ABSTRACT

Cloud computing is the latest trend in computing service where computing is leased out as a service on a pay-per-use basis. This has a huge potential in providing scientificcomputing to those users who do not have access to traditional resources like grids, clusters etc. We study the cloud framework Eucalyptus and set it up for the purpose for scientific computing. The chosen problem - Matrix Multiplication is set up usingthe Message Passing Interface, a leading standard for message passing libraries. The experiments are done on and the resulting response times are observed [1]. These response times are modeled as functions of the size of the jobs and the resources allocated. Usingthese models, a resource allocation algorithm has been proposed. The algorithm is based on the optimization problems that reduce the overall delay over the deadline for the batch of jobs submitted every triggering interval while maximizing resource utilization. The proposed algorithm has been simulated and shown to be superior to a simple sharing algorithm in terms of delay minimization [1].

I. INTRODUCTION

Cloud Computing is defined as the model for delivering computing resources as a serviceover the Internet. Cloud computing has several advantages. It eliminates the need for the clients to set up and maintain their own physical servers, thus reducing expenses. The clientsare billed only as per their usage. Dynamic reallocation of resources ensures that the servers are utilized more efficiently. Also network access ensures that the client can access these services from anywhere. Because of these factors, cloud computing is a fast growing field. Currently Amazon, Google and Microsoft are some of the big names in this field.

Depending on the service models, clouds are classified as [1][7]

1. Software as a Service (SaaS): In this model the user purchases the ability to use a software application or service on the cloud. Eg: Google Docs
2. Platform as a Service (PaaS): In this model the user purchases access to platforms, enabling them to deploy their own applications on the cloud. Eg: Google App Engine
3. Infrastructure as a Service (IaaS): In this model, the user is delivered infrastructure, namely servers, networks and storage. The user can deploy several Virtual Machines and run specific Operating Systems on them. e.g.: Amazon EC2, Windows Azure etc.

II. EUCALYPTUS - OPEN SOURCE CLOUD COMPUTING

Eucalyptus stands for Elastic Utility Computing Architecture for Linking Your Programsto useful Systems. It is a Linux based open source architecture that can be usedto implement scalable private and hybrid clouds. The cloud will be deployed across theenterprise's on-premise IT infrastructure and can be accessed over its intranet [3][4].Eucalyptus also supports Amazon Web Service-compatibility allowing our on-premiseclouds to interact with these public clouds using a common programming interface.

This API-compatibility with Amazon's EC2, S3, ELB, Auto Scaling, and CloudWatch services offers the capability of deploying hybrid clouds. It has support for multiple virtualization platforms like Xen, KVM and VMWare. It is also packaged and supported for multiple distributions like Debian, Ubuntu, Cent-OS, SuSE etc. Eucalyptus Systems (a) gives details about Eucalyptus 3.1.2 architecture and setup.

III. A. SCIENTIFIC COMPUTING ON THE CLOUD - A DISTRIBUTED COMPUTING APPROACH

Scientific Computing is one of the leading disciplines in information technology withvaried application in fields such as economics, science and engineering. It is the practice of aggregating the computing resources in such a way that it delivers much higher performance than computations on a personal desktop or workstation. Due to specific performance requirements, it is common to operate high performance computing resource in private and thus the access to these are often restricted. Also jobs have to wait in a queue for resource allocation and execution. Also it may happen that these physical machines are underutilized because of the fluctuating demands within the particular organization [5][6].

III B. DISTRIBUTED COMPUTING - DEFINITION AND FRAMEWORKS

Distributed computing refers to the parallelization of a large computational job into several smaller computational tasks and executing these tasks on different nodes. [3]Nodes are autonomous computational resources with its own local memory that communicatewith each other by passing messages on the network. First a MATLAB DistributingComputing Server was considered but was found to be inadequate for the work due to system constraints. Then a Python based approach was considered. For communication purposes the two most prevalent approaches have been MPI and MapReduce. Finally MPI was chosen for the simplicity and flexibility provided [6].

IV. MATLAB DISTRIBUTED COMPUTING SERVER

This framework can run computationally intensive MATLAB programs and Simulinkmodels on computer clusters, clouds and grids. The program or model is first developed on a multicore desktop environment using the Parallel Computing Toolbox and then scaled up to many computers by running it on this framework. This framework can support batch jobs, parallel communication and distributed large data.

4.1 Mapreduce

In the Reduce() phase, the master node then collects the answers to all the sub problems and combines them in some way to form the output which is the answer Map Reduce is a programming framework that lets the user process large data sets with a parallel, distributed algorithm on a distributed system. A Map Reduce implementation consists of 2 main phases: Map() phase and Reduce() phase.In the Map() phase, the master node takes the input, divides it into smaller sub problems, and distributes them to worker nodes. Each mapping operation is independent and thus can be performed in parallel. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master nodeto the problem it was originally trying to solve. We can have a set of reducers that perform the reduction phase provided that all outputs of the map phase which share the same key are given to the reducers at the same time.

4.2 Message Passing Interface

The Message Passing Interface is a standardized and portable message passing system designed to function on a wide variety of parallel systems. The standard itself is not a library, but defines the syntax and semantics of the library routines for a language independent communications protocol. MPI primary addresses the message passing parallel programming model in which data is moved across the address spaces of processes through cooperative operations. Since the release of MPI, it has become the leading standard for message passing libraries for parallel systems and has achieved widespread implementation [2].

V. PYTHON

Python is a widely used general-purpose, high-level programming language. It hasan efficient high-level data structures and a simple but effective approach to object oriented programming with dynamic typing and dynamic binding. It has a holistic [2]language design with emphasis on readability and concise coding. It has the perfect balance of high level and low level programming. It is an open source programming languages with an impressive standard library and external libraries being developed by the enthusiastic Python community. Python has good language interoperability.

Python has an impressive support for scientific computing. SciPy is a computing environment and open source ecosystem of software for Python programming language that is used by scientists, analysts and engineers doing scientific computing and technical computing. SciPy also refers to the open source Python library of algorithms and mathematical tools that are at the crux of the SciPyenvironment [2].

VI. APPLICATION PARALLELIZATION AND MODELING

For the experiments we chose the Dense Matrix Multiplication Problem, a commonlyresource intensive problem. The main motivation was to be able to setup parallelized application, so that the user can run the algorithm on their datasets within the time constraints.

International Journal of Advance Research in Science and Engineering
Vol. No.4, Issue 10, October 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

VII. DENSE MATRIX MULTIPLICATION PROBLEM

Matrix multiplication, from an academic viewpoint has been one of the first and most widely studied application of parallelization. Its importance as a problem to be solved and relevance in several fields has pushed research constantly. Some specific characteristics of this problem with regards to its design and implementation of a parallel algorithm are as follows.

1. Computational Independence: Each element of the output matrix is independent from all the other elements. This allows for a wide flexibility in terms of parallelization.
2. Data Independence: In the case of dense matrix multiplication, the number and type of operations to be run are independent of the data.
3. Data Organization: Since the data is organized in two-dimensional structures,it allows for flexibility in algorithm as well as creating suitable topologies of processes for efficient performance.

7.1 Problem Description

Given two square matrices A_{nxn} and B_{nxn} of dimension n where each of its elementsare denoted as a_{ij} and b_{ij} with $1 \leq i, j \leq n$, the matrix C resulting from the operationof multiplication of matrices A and B, $C = A \times B$, is such that each of its elements$c_{ij} = \sum_{k=1}^{n} a_{ij}\, b_{ij}$

7.2 Experiment

The experiment was done using randomly generated dense matrices of sizes varyingfrom 1000*1000 to 3000*3000. Each run of the matrix multiplication was distributedover a set of VMs. The number of VMs were varied from 1 to 10 VMs for each matrix multiplication. It is assumed that no other job is running on the VMs and that there is no waiting time for any job. The response times of each job were measured and plotted.

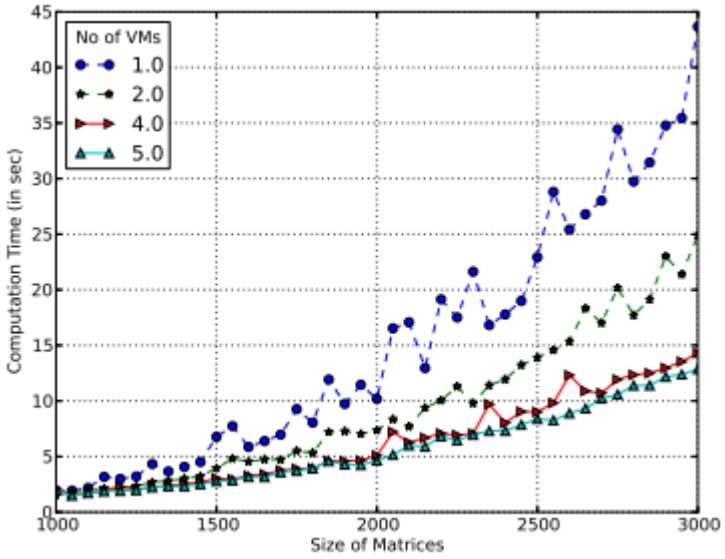

Fig.1: Response time vs size of matrices for different number of VMs

7.3 Virtual Machine Allocation - Delay Optimization And Algorithm Optimization Problem Formulation

In our analysis we consider the cloud model wherein parallelizable applications like the one described in the previous chapter arrive at the load balancer. The load balancer creates batches of several jobs and allocates the resources at regular intervals. At every resource allocation trigger, the load balancer computes the number of resources to be provided based upon the algorithm proposed [8]. Consider the batches as a function of time as *Batch* (*t*). These are considered atevery resource allocation trigger to be composed of several jobs(*A*1(*t*), *A*2(*t*), . . . , *Am*(*t*)). Each of these jobs specify the following parameters:

1. Job Size (*size*): This is the parameter that decides the size of the job. Matrixdimensions in the case of Matrix Multiplication.

2. Due Time (*d*): This is a parameter that specifies the due time expected by theuser. It is decidebased on the priority of the job assigned by the user.

Let (*x*1, *x2*,.....,*xm*) refer to the number of VMs to be assigned for each of thejobs. Then the expected worst case running time of each of the jobs can be computedaccording to the models described in the previous section. Let us denote the expectedworst case run time for job *Ai* with size *si*and due time *di* run on *j* VMs be givenas $ti(j) = F(si, j)$. Our aim is to minimize the total delay over the expected duetime experienced by the user. Hence the cost function to be minimized is given as$\sum_{i=1}^{m} \delta_{ij}$ where δijrepresents the excess delay over the expected due time when run on *j*resources and is given as

$$\delta ij = \begin{cases} t_i(j) - d_i : t_i(j) > d_i \\ 0 : t_i(j) < d_i \end{cases}$$

VIII. PROPOSED ALGORITHM

The jobs arriving are grouped into batches at regular trigger intervals.At each trigger instant, we have a batch of m jobs and n resources need to allocateto minimize the overall$\sum_{i=1}^{m} \delta_{ij}$. We create a matrix C where the rows corresponds to each job and the columns correspond to the number of VMs allocated. The proposed algorithm first computes the total number of resources required for the jobs n_{init} for minimizing the delay without imposing the constraint on total number of available resources. Then it iterates from n_{init} to the actual value n. In each iteration it looks for thejob that will add the least amount of penalty into the cost function$\sum_{i=1}^{m} \delta_{ij}$ and reduces 1 VM from it.

IX. RESULT

As proof of concept, we have simulated the proposed algorithm and compared it with a simple shared allocation algorithm.

In the simple shared allocation algorithm, we divide the total resources available equally among the jobs in the batch. We simulate the two algorithms using the multiprocessing package from python. This package lets us spawn multiple processes that can work on a queue structure. We create 3 processes for simulating each of the algorithm. The first process generates batches of jobs at regular interval and pushes it into the queue. The second process pops these batches from the queue, executes the algorithms, starts the jobs on the simulated VMs

and computes the delay. The third process monitors and updates the execution of the jobs on the simulatedVMs with time.

The jobs were sent in the form of 20 batches at regular intervals of 5 seconds. The no of virtual machines initiallyallotted were 5. They were scaled up to 10 whenever the no of jobs were more than available free VMs.We can compare the total delays experienced by batches of jobs from the graph 3. It is evident that the proposed minimum delay algorithm is superior to a simpleshared algorithm. Also a comparison of the VMs utilization 3 reveals that the proposed algorithm completes the jobs faster than the shared algorithm.

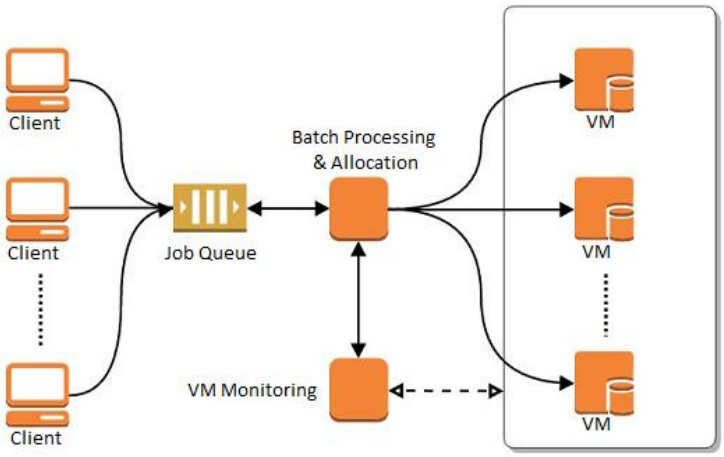

Fig.2: Job Queuing and Allocation

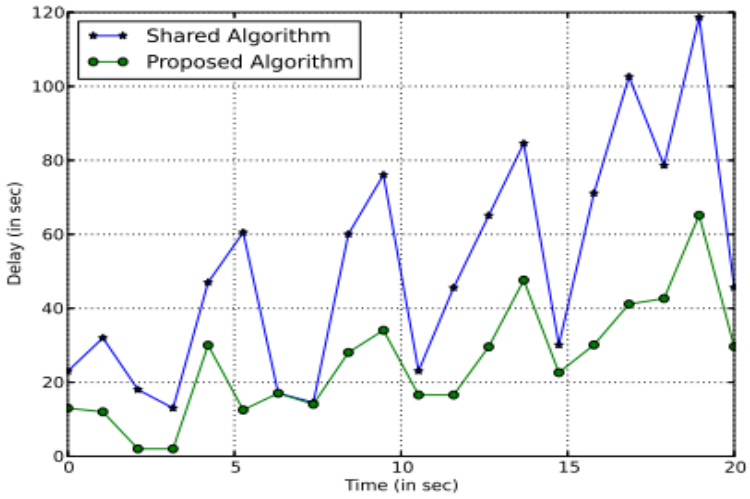

Fig.3: Comparison of Delay

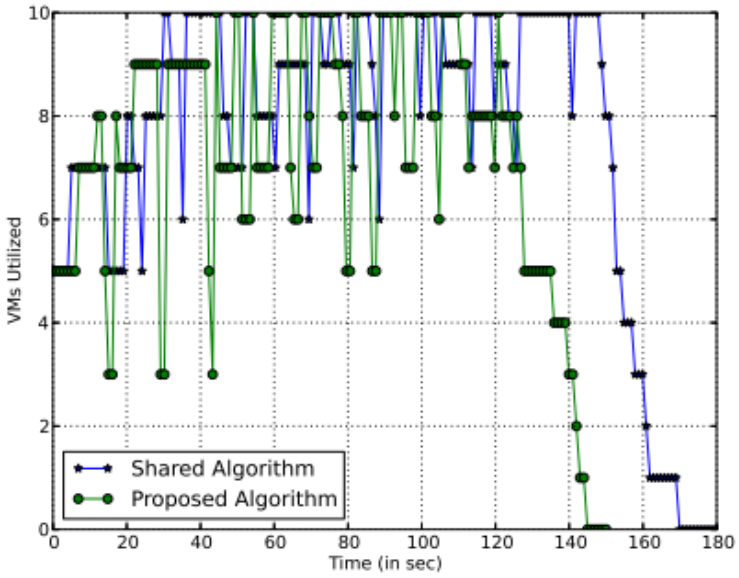

Fig.4: Comparison of VM utilization

9.1 Resource Allocation Algorithm

Require: Model equations, n number of VMs, Job numbers and details

Ensure: x_i number of virtual machines to be allocated for each job i

for I = 1 to m do

Compute j, such that$t_{ij} = d_i$

$J_i = [j_i]$

$\delta_{ij} = t_{iji} - d_i$

if $\Sigma(J_i) \leq$ n then

Allocate J_i VMs to job i

else

k = 0

fori = 1 to m do

ifJ_i =1 then

Allocate J_i = 1 VMs to job i

$extra\_delay_i = \delta_{iJi-1} - \delta_{iJi}$

m = m-k

n = n-k

iter = $\Sigma(J_i)$ – n

whileiter › 0 do

forJ_I › 1 do

$I = argmin_i extra\_delay_i$

$J_I = J_I - 1$

ifJ_I› 1 then

$extra\_delay_I = 0$

iter = iter – 1

Allocate J_I VMs to job I

Total excess delay is $\sum_{i=1}^{m} x_i$ where $x_i = \delta_{iji}$

X. CONCLUSION

In this work we have successfully set up the Eucalyptus Cloud framework on the local physical systems to function as a Private Cloud. Using the Message Passing Interface library, we were able to deploy the parallel Dense Matrix Multiplication application in the distributed environment to reduce time. This applications can thus be deployed over the cloud for use by the users in the scientific community who would like to take advantage of the benefits offered by clouds. Thus the user can get results without botheringabout the underlying resource management or implementation. The profiles of the response times for each of the applications have been modeled as a function of the job sizes and the resources allocated. Based on these benchmarking tests, a new resource allocation algorithm has been proposed to reduce overall delay over expected due time for jobs in a batch-wise fashion. The proposed algorithm has been simulated and shown to be superior to a simple sharing algorithm in terms of delay minimization.

REFERENCES

[1]. Buyya, R., C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic (2009). Cloud computingand emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6), 599 – 616. ISSN 0167- 739X. URL http://www.sciencedirect.com/science/article/pii/S0167739X08001957.

[2]. Dalcin, L. (2012). MPI for Python, Release 1.3. URL http://mpi4py.scipy.org//.

[3]. Eucalyptus Systems, I. (a). Eucalyptus 3.1.2 Installation Guide. URL https:// www.eucalyptus.com/.

[4]. Eucalyptus Systems, I. (b). Eucalyptus 3.1.2 User Guide. URL https://www.eucalyptus.com/.

[5]. Gibson, J., R. Rondeau, D. Eveleigh, and Q. Tan, Benefits and challenges of three cloud computing service models. In Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on. 2012.

[6]. Gong, C., J. Liu, Q. Zhang, H. Chen, and Z. Gong, The characteristics of cloudcomputing. In Parallel Processing Workshops (ICPPW), 2010 39th International Conferenceon. 2010. ISSN 1530-2016.

[7]. He, H., Applications deployment on the saas platform. In Pervasive Computing andApplications (ICPCA), 2010 5th International Conference on. 2010.

[8]. He, Q., S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, Case study for running hpcapplications in public clouds. In Proceedings of the 19th ACM International Symposiumon High Performance Distributed Computing, HPDC '10. ACM, New York, NY,USA, 2010. ISBN 978-1-60558-942-8. URL http://doi.acm.org/10.1145/1851476.1851535.

[9]. Jakovits, P. and S. Srirama, Adapting scientific applications to cloud by using distributedcomputing frameworks. In Cluster, Cloud and Grid Computing,2013 13th IEEE/ACM International Symposium on. 2013.