Geometric Mean based Algorithm for Prioritizing Processors on Cloud Environment

Khushboo R Parmar
Parul Institute of Technology, Vadodara, Gujarat, INDIA-39001

Upendra Bhoi
Assi.Prof. Parul Institute of Technology, Vadodara, Gujarat, INDIA-39001

ABSTRACT
Cloud computing relates to the bunch of services that are provided to the customers on lease, by the servers located at different sites over the internet. The servers have pool of resources that can be scaled up and down on the basis of requirement. This results into communication and computation over the network. Divisible load theory has become popular during the past two decades. Based on divisible load theory the computations and communications can be divided into some arbitrarily independent parts and each part can be processed independently by a processor. The fraction of load must be allocated the processors based on some priorities. Analytical Hierarchy Process(AHP) is a multi-criteria based technique used for assigning priorities to the processors. Existing approach can handle the priority of processors using Eigen Value method of Analytical Hierarchy Process. The proposed model works on Geometric mean method of Analytical Hierarchy Process in order to improve parameters such as makespan, average response time and average waiting time.

Keywords: Cloud Computing, Divisible load theory, Analytical Hierarchy process, Geometric Mean.

1. INTRODUCTION OF CLOUD COMPUTING
Cloud computing is a new technological trade which provide trustworthy services through cloud centres. We can say Cloud as a type of distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned, and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and end user or customers.

Figure:1- Cloud Computing Network

One property of Cloud platforms is the ability to dynamically adapt (scale-up or scale-down) the amount of resources provided to an application in order to attend the variations in demand that are either predictable, and occur due to access patterns observed during the day and during the night; or unexpected, and occurring due to a subtle increase in the popularity of the application service. Such capability of clouds is especially useful for elastic (automatically scaling of) applications, such as web hosting, content delivery, and social networks that are susceptible to such behaviour.

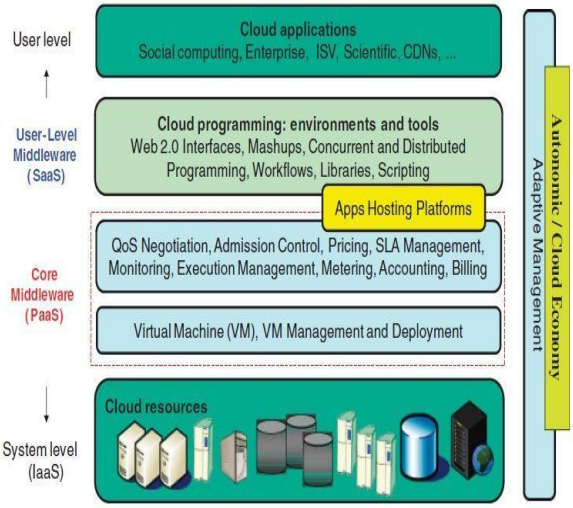

Figure-2 Layered design of Cloud Computing

Figure 2 depicts the layered design of Cloud computing architecture. Physical Cloud resources along with core middleware capabilities form the platform for delivering IaaS and PaaS. At user-level middleware, different SaaS capabilities are provided. The top layer focuses on application services (SaaS) by getting services that are provided by the lower-layer services. PaaS/SaaS services are often developed and provided by third-party service providers that are different from the IaaS providers .[1]

1.1 Cloud applications [3]:
This layer includes applications that are directly accessible by end-users. End-users are defined as the active entity that utilizes the SaaS applications over the Internet. These applications may be supplied by the Cloud provider (SaaS providers) and accessed by end-users either by a subscription model or on a pay-per-use basis. User are also allowed to deploy their applications in this model as well. In the earlier

case, there are applications such as Salesforce.com that supply business process models on clouds (namely, customer relationship management software) and social networks. In the latter cases, there are e-Science and e-Research applications, and Content-Delivery Networks.

User-Level middleware (SaaS):

Under this layer we can be aware of the software frameworks, such as Web 2.0 Interfaces (Ajax, IBM Workplace), that support developers in creating rich, cost-effective user-interfaces for browser-based applications. This layer also provides the programming environments and composition tools that makes creation, deployment, and execution of applications in clouds easier. Finally, in this layer many frameworks that support multi-layer applications development, as in Spring and Hibernate, can be deployed to support applications running in the upper level.

Core middleware (PaaS):

The PaaS Layer implements the platform-level services that provide run-time environment for hosting and managing User-Level application servicesas well. The basic services at this layer include Dynamic SLA Management, Accounting, Billing, Execution monitoring and management, and Pricing (are all the services to be capitalized?). The very known examples of services operating at this layer are Amazon EC2, Google App Engine, and Aneka. The functionalities that are carried out by this layer are accessed by both SaaS (the services represented at the top-most layer in Figure 2) and IaaS (services shown at the bottom-most layer in Figure 2) services. The sensitive functionalities that need to be realized at this layer include messaging, service discovery, and load-balancing. These critical functionalities are usually been implemented by Cloud providers and offered to application developers at an additional premium. For instance, Amazon offers a load-balancer and a monitoring service (Cloud watch) for the Amazon EC2 developers as well as consumers. Similarly, developers that are building applications on Microsoft Azure clouds can make use of the .NET Service Bus for implementing message passing mechanism.

System Level (IaaS):

The computing power in Cloud environments is supplied by a number of data centers that are typically installed with hundreds to thousands of hosts. At the System-Level layer, there exist massive physical resources (storage servers and application servers) that provide power to the data centers. These servers are transparently managed by the higher-level virtualization services and toolkits that allow them to share their capacity among virtual instances of servers. These Virtual machines are isolated from each other, therefore making fault tolerant behaviour and isolated security context possible.

2. INTRODUCTION TO DIVISIBLE LOAD THEORY

The first article was published in 1988 about divisible load theory(DLT) [4]. Based on DLT, it is assumed that the computation can be partitioned into some arbitrary sizes, and each partition can be processed independently by one processor. In the past years, DLT has found a wide variety of applications in parallel processing area such as data intensive applications[5], data grid application[6], image and vision processing[7] and so on. DLT was applied for various network topologies including chain, star, bus, tree , three-dimensional mesh.

2.1 Divisible Load Scheduling

Basically, DLT assumes that the computation and communication can be divided into parts of arbitrary sizes and these parts can be independently processed in parallel by processors(see Fig.3). It is assumed that initially amount of load is

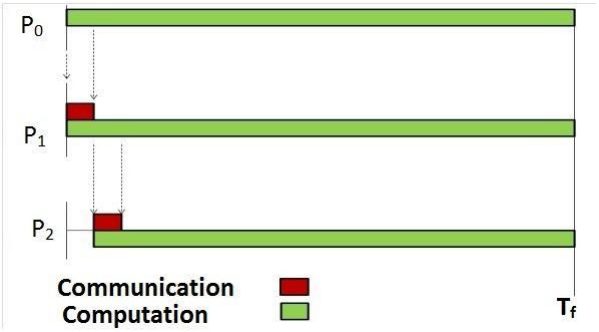

Figure:3- Gantt chart-like timing diagrams for divisible load[1]

Of load held by the originator p_0. The originator p_0 does not do any computation. It only distributes $\alpha_1,\alpha_2,...,\alpha_m$ fractions of load on worker processors p_1, $p_2,...,p_m$. Condition for optimal solution is that all processors should stop processing at the same time. This fraction of loads must be allocated to processors based on criteria and priorities.

3. INTRODUCTION TO ANALYTICAL HIERARCHY PROCESS (AHP)[8]

The Analytic Hierarchy Process (AHP) is a multi-criteria decision making method developed by Thomas Saaty that considers criteria's. This process helps decision makers to model a complex problem in a hierarchical structure, showing the relationships of the goal, objectives (criteria), and alternatives. AHP is made up of several components such as hierarchical structuring of complexity, pairwise comparisons, judgments, an eigenvector method for deriving weights, and consistency considerations that is helpful in decision making[13].

In the case of making a decision, the best alternative can be easily determined in accordance with the preference of the decision-maker present. When the decision is to be decided by a group of people, it is very common that convicting preferences complicate the evaluation processes leading to an unending conclusion. Therefore, it is necessary to aggregate the individual preferences objectively in order to optimize the decision outcomes.

Many decision problems cannot be structured hierarchically as always. The importance of the criteria can be derived by the importance of the alternatives. The Analytic Network Process (ANP) provides a solution for problems which can be modelled using a diagram called a network.

The Analytical Hierarchy Process:

1) **Hierarchical Decomposition of the Decision**
   Goal The problem statement.
   Objectives The criteria's available.
   Alternatives The alternatives with respect to each criteria.

2) Pairwise Comparisons
The first is between pairs of objectives and is used to show the priorities. The second type of pairwise comparison is between pairs of alternatives and is used to determine their relative merits.

3) Importance of Objectives
When a we judges A to be strongly more important than B we know that A is more important than B, but we do not know the interval between A and B or the ratio of A to B.

4) Preference of Alternatives with respect to Objectives
Before evaluating the importance of the objectives,we usually evaluate the preference for the alternatives with respect to the objectives.

5) Pairwise Matrix Evaluation[2]
Suppose we already know the relative weights of criteria: *w1;w2; : : : ;wn*. Then they can be expressed in pairwise comparison matrix.

6) Eigenvector Method
After finding Eigen Values of the Comparison Matrices we can find the priorities.

4. APPROACHES OF DIVISIBLE LOAD SCHEDULING

1. Minimize the overall processing time for scheduling jobs[9]
In this paper, authors attempt to investigate the use of a Divisible Load Theory (DLT) to design efficient strategies to minimize the overall processing time for scheduling jobs in compute cloud environments. They considered homogeneous processors in analysis and derived a closed-form solution for the load fractions to be assigned to each processors. The analysis also attempts to schedule the jobs such a way that cloud provider can gain maximum benefit for his service and Quality of Service (QoS) requirement user's job. Finally, they quantify the performance of the strategies via rigorous simulation studies.

2. Iterative Divisible Load Theory (IDLT)[10]
The IDLT model proved to be an effective model for optimal workload allocation**.** This model initially divides the load using Adaptive DLT, then Makespan would be calculated using cost model. If the result is optimal then that would be final makespan orelse it will calculate iteratively makespan using new average. Experimental results showed that the IDLT model was capable of producing an almost optimal solution for single source scheduling. Hence, it can balance the processing loads efficiently.

3. Fixing the processor sequence in the non-decreasing order[11]
In this approach a systematic analysis of Pareto optimal solution is performed. The author had performed a systematic analysis of the problem of scheduling divisible load on processor in order to minimize the computation time and cost. This model with a single divisible load provides a foundation for more advanced models which better describe real world scenario. This model was efficient for single divisible load, could be further used for advanced models.

4. Analytic Hierarchy Process for prioritizing processors (Target Technique) [1,14]
In this paper the authors initially divided the total load into independent parts that could be further processed by different processors. They proposed a multi criteria based AHP technique for prioritizing the processors, considering several criteria of processors and then assigned fractions of load to processors based on the priority. In AHP, they used Eigen Value method which gives efficient makespan but if the order of the processors are changed then the makespan is not definite. The fluctuation of makespan was due to different comparison matrices that were not consistent.

5. EXISTING ALGORITH- MULTI-CRITERIA BASED ALGORITHM FOR SCHEDULING DIVISIBLE LOAD[1]

- Input: μ= {P1 ,P1,..,Pm} a set of processors
- Input: C={C1 ,C2,..,Cd} a set of criteria;
- For all criterias make comparison matrix "C"
- Compute priority vectors for the matrix C
- For C compute a consistent comparison matrix
- Generate comparison for set of processors considering each criteria matrices-C1,C2,C3
- Compute priority vectors for each matrices i.e, Qc1,Qc2,Qc3
- Check consistency ratio for C1,C2,C3
- Compute PVD which is a vector included value of priority of processors
- Sort processors based on their PV D value;
- Allocate fraction of load to the sorted processors and compute α1,α2,...,αm
- End

6. PROPOSED MODEL
The existing algorithm had issues related to makespan i.e, if we consider different comparison matrix then the makespan is not definite. Thus, we improved the Multi-Criteria based algorithm for scheduling divisible load by replacing the Eigen Value method of AHP with Geometric Mean Method. The reason behind using this method is that the comparison matrix is always consistent and so no need to find the consistency ratio all the time. This in return improves the makespan and make it definite for all the possible comparison matrices as shown in fig:4.

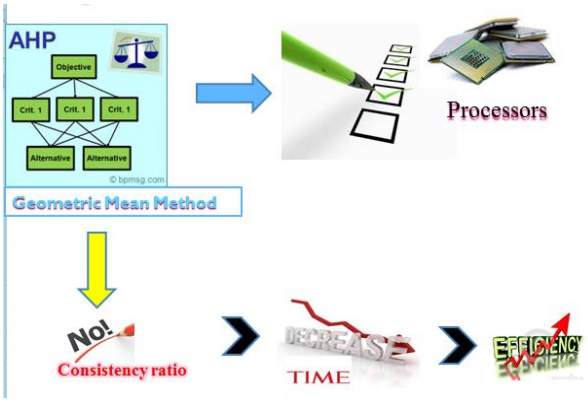

Figure:4- Proposed Model

Geometric Mean Method:

In this approach, the priorities are given by the geometric mean, which minimizes the logarithmic error (Crawford, 1985):

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\ln(a_{ij})-\ln(\frac{p_i}{p_j})\right)^2$$ where a_{ij} is the comparison between i and j

p_i is the priority of i.

This method is insensitive to an inversion of the scale: the geometric mean of the rows and the columns give the same ranking.

Saaty (1990) criticizes this method because he sees no conceptual justification for working with a logarithmic scale. He adds that the calculation is made only with a row, i.e. the indirect estimations are not considered (Saaty 1984a, 1984b).

Example:

Consider the following comparison matrix:

1	6	3
1/6	1	1/2
1/3	2	1

The priorities from the above matrix calculated with the geometric mean are:

$$p_1=\sqrt[3]{1\cdot 6\cdot 3}=2.62\,,\ p_2=\sqrt[3]{\frac{1}{6}\cdot 1\cdot\frac{1}{2}}=0.44\,,\ p_3=\sqrt[3]{\frac{1}{3}\cdot 2\cdot 1}=0.87$$

Normalizing, we obtain: p = (0.67, 0.11, 0.22)

7. PROPOSED ALGORITHM

- Input: μ= {P1 ,P1,..,Pm} a set of processors
- Input: C={C1 ,C2,..,Cd} a set of criteria;
- For all criterias make comparison matrix "C"
- Compute priority vectors for the matrix C
- Generate comparison for set of processors considering each criteria matrices-C1,C2,C3
- Compute priority vectors for each matrices i.e, Qc1,Qc2,Qc3
- Compute PVD which is a vector included value of priority of processors
- Sort processors based on their PV D value;
- Allocate fraction of load to the sorted processors and compute α1,α2,...,αm
- End

8. COMPARATIVE RESULTS

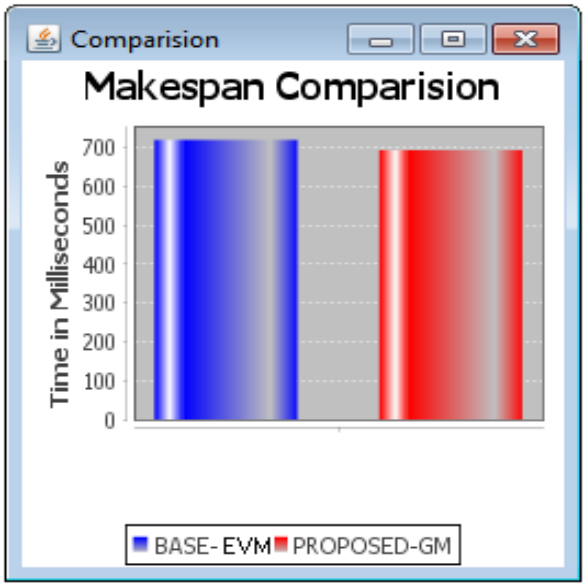

Figure: 5- Comparison of Makespan between Eigen Value Method and Geometric Mean Method

9. CONCLUSION

Divisible Load Scheduling and Allocating has become must on the Cloud environment. Divisible Load Theory divides the computational and communication load over the computer network into arbitrarily independent parts. These fractions of load are processed by the processors on the basis of the priorities. The focus is on the prioritizing technique. As such there are many techniques; the existing technique is Eigen Value Approach of Analytical Hierarchy Process.

The proposed Algorithm uses Geometric Mean method of Analytical Hierarchy Process to prioritize the processors and then allocate load respectively. The parameters improved is the makespan of the Algorithm that in turn increases the efficiency of the Algorithm. The implementation is done on cloud environment in CloudSim.

In future enhancement, other parameters may also be considered and improved.

10. REFERENCES

[1] Shamsollah Ghanbari, Mohamed Othman ,Wah June Leong, and Mohd Rizam Abu Bakar "Multi-Criteria Based Algorithm for Scheduling Divisible Load" Springer Science+Business Media Singapore 2014

[2] Saaty, Thomas L. "How to make a decision: the analytic hierarchy process." European journal of operational research 48(1), 9-26 (1990).

[3] Mohsin Nazir."Cloud Computing: Overview & Current Research Challenges". IOSR Journal of Computer Engineering (IOSR-JCE) ISSN: 2278-0661, ISBN: 2278-8727Volume 8, Issue 1 (Nov. - Dec. 2012), PP 14-22

[4] Cheng, Yuan-Chieh, and Thomas G. Robertazzi. "Distributed computation with communication delay [distributed intelligent sensor networks]." Aerospace and

Electronic Systems, IEEE Transactions on 24(6), 700-712 (1988).

[5] Ko, Kwangil, and Thomas G. Robertazzi. ”Equal allocation scheduling for data intensive applications.” Aerospace and Electronic Systems, IEEE Transactions on 40(2), 695-705 (2004).

[6] Abdullah, Monir, Mohamed Othman, Hamidah Ibrahim, and Shamala Subramaniam. ”Optimal workload allocation model for scheduling divisible data grid applications.” Future Generation Computer Systems 26(7), 971-978 (2010).

[7] Li, Ping, Bharadwaj Veeravalli, and Ashraf A. Kassim. ”Design and implementation of parallel video encoding strategies using divisible load analysis.” Circuits and Systems for Video Technology, IEEE Transactions on 15(9), 1098-1112 (2005).

[8] The Analytic Hierarchy Process and its Generalizations Thesis by Edit Adamcsek

[9] Monir Abdullah, Mohamed Othman."Cost-Based Multi-QoS Job Scheduling using Divisible Load Theory inCloud Computing."International Conference on Computational Science doi:10.1016/j.procs.2013.05.258

[10] Abdullah, Monir, Mohamed Othman, Hamidah Ibrahim, and Shamala Subramaniam. ”Optimal workload allocation model for scheduling divisible data grid applications.” Future Generation Computer Systems 26(7), 971-978 (2010).

[11] Natalia V. Shakhlevich “Scheduling Divisible Loads to Optimize the Computation Time and Cost” School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

[12] Natalia V. Shakhlevich “Scheduling Divisible Loads to Optimize the Computation Time and Cost” School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

[13] THE ANALYTIC HIERARCHY PROCESS (AHP) Geoff Coyle: Practical Strategy. © Pearson Education Limited 2004

[14] Analytic Hierarchy Process Dr. Rainer Haas Dr. Oliver Meixner Institute of Marketing & Innovation University of Natural Resources and Applied Life Sciences, Vienna

[15] Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, and Rajkumar Buyya."CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services ." Grid Computing and Distributed Systems

[16] Alessio Ishizaka, Markus Lusti. “How to derive priorities in AHP: a comparative study”.