Co-operative Scheduled Energy Aware Load-Balancing technique for an Efficient Computational Cloud

T.R.V. Anandharajan[1], Dr. M.A. Bhagyaveni[2]

[1] Research Scholar, Department of Electronics and Communication, Anna University, College of Engineering Chennai – 600 025, Tamilnadu, India

[2] Asst. Professor, Department of Electronics and Communication, Anna University, College of Engineering Chennai – 600 025, Tamilnadu, India

Abstract
Cloud Computing in the recent years has been taking its evolution from the scientific to the non scientific and commercial applications. Power consumption and Load balancing are very important and complex problem in computational Cloud. A computational Cloud differs from traditional high-performance computing systems in the heterogeneity of the computing nodes, as well as the communication links that connect the different nodes together. Load Balancing is a very important component in the commodity services based cloud computing. There is a need to develop algorithms that can capture this complexity yet can be easily implemented and used to solve a wide range of load-balancing scenarios in a Data and Computing intensive applications. In this paper, we propose to find the best EFFICIENT cloud resource by Co-operative Power aware Scheduled Load Balancing solution to the Cloud load-balancing problem. The algorithm developed combines the inherent efficiency of the centralized approach, energy efficient and the fault-tolerant nature of the distributed environment like Cloud.
Keywords: *Cloud Computing, Load Balancing, Green computing, Boundary scheduling approach.*

1. Introduction

Cloud Computing is a computing model, where resources such as computing power, storage, network and software are abstracted and provided as services on the Internet in a distributed environment. Billing models for these services are generally similar to the ones adopted for public utilities.

On-demand availability, ease of provisioning, dynamic and virtually infinite scalability are some of the key attributes of cloud computing.

An infrastructure setup using the Cloud Computing model is generally referred to as the “Cloud”. The following are the broad categories of services available on the Cloud [8]:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

This Cloud is generally available as a service to anyone on the Internet. However, a variant called “Private Cloud” is increasingly becoming popular for private infrastructure that has some of the attributes of the Cloud as mentioned above.

Amazon Web Services (AWS) is one of the major players providing IaaS. They have two popular services – Elastic Compute Cloud (EC2) and Simple Storage Service (S3). These services are available through web services. The client tools can use EC2 and S3 APIs to communicate with these services. The popularity of these APIs has encouraged other Cloud products to provide support for them as well.

Using the Cloud Computing paradigm, a host of companies promise to make huge compute resources available to users on a pay-as-you-go basis. These resources can be configured on-the-fly to provide the hardware and operating system of choice to the customer on a large scale. Given the current infrastructure bandwidth and topologies utilized in these commercial offerings, however, the only current feasible market would be small energy and memory foot-marks embarrassingly parallel or loosely coupled applications, which inherently require little to no inter-processor communication. While providing the infrastructure (bandwidth, latency, memory, etc.) the numbers of physical resources, the resources are distributed and shared among many users, and the resources may be heterogeneous and highly active. Advanced resource monitoring, analysis, and configuration tools can help address these issues, since they bring the ability to efficiently and dynamically provide and respond to information and communication about the platform and application state and would enable more appropriate, efficient, and flexible use of the resources. Additionally such tools could also be of benefit to cloud providers, users, and applications by providing more efficient resource deployment in general.

2. Preliminaries

Cloud and other distributed systems rely on bulk processing and in turn run lot of discrepancies in the implementation. Bulk processing parallel programs have the property that the problem can be divided into sub problems or jobs, each of which can be solved or executed in roughly the same way. Each run consists of I iterations of P jobs, which are distributed on P processors: each processor receives one job per iteration. The processor receives a job, and the IT is equal to the maximum of the individual job times plus the synchronization time. In this section, we briefly discuss the two main methods to cope with the dynamics of our Cloud environment: DS and EA.

2.1 Dynamic Scheduling

Dynamic Scheduling deals with the scheduling of the Cloud jobs from the sensors located globally and available on the internet. Cloud Computing shares its major base on the Internet. HTTP protocol our protocol here. Computational Cloud is the paradigm where we are going to show our focus Dynamic Scheduling starts with the execution of an iteration of the jobs from a Cloud User 'C_N '. However, at the end of each iteration, the processors predict their memory and the processing time for the next iteration. We select one processor to be the DS scheduler. After every N iteration, the processors send their prediction to this scheduler. Subsequently, this scheduler calculates the various job required parameters suitable for the resource to process the job and schedules the load distribution given those predictions and sends relevant information to each Cloud client. The load distribution is optimal when all cloud resource finish their execution exactly at the same time. Finally, all resources redistribute the load. The effectiveness of DS partly relies on the dividing possibilities of the job. We introduce the concept of a method that selects and matches jobs to the cloud resources. To this end, it measures and defines a threshold Z during the run. After each iteration, the method gives a preference for a given type of implementation, based on a comparison of Z with the threshold value Z. In this method, the processor that redistributes the load in DS. The steps to be taken are the same as in the DS phase:

1) The cloud resource send their prediction to the scheduler,
2) The cloud scheduler computes the load distribution, and
3) The scheduler informs redistribution of load.

At the end of each iteration, the processors predict their processing speed for the next iteration. Our proposed algorithm takes threshold value into account and the load is bounded accordingly.

3. Dynamically Scheduled Load balancing algorithm using Boundary Value

The load balancing algorithm is done before it reaches the processing servers the job is scheduled based on various parameters and scheduled through a scheduler which implements the load balancing algorithm. The job will then be scheduled to the processing severs and the processed jobs will be sent to the Client interface for the knowledge of the data processed.

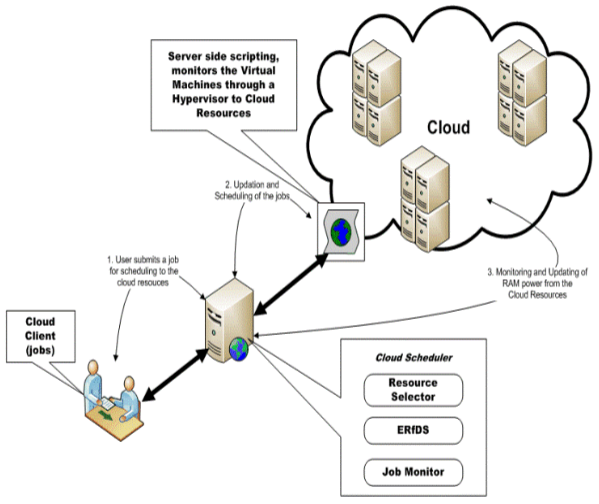
Server side scripting, monitors the Virtual Machines through a Hypervisor to Cloud Resources
Cloud
2. Updation and Scheduling of the jobs
1. User submits a job for scheduling to the cloud resouces
3. Monitoring and Updating of RAM power from the Cloud Resources
Cloud Client (jobs)
Cloud Scheduler
Resource Selector
ERfDS
Job Monitor

Fig. 1 Cloud Scheduling Architecture.

Many cloud vendors like Amazon EC2 instance can handle at most 400 Mbps combined ingress and egress traffic. For some application Google App Engine is only able to handle 10 Mbps in/out or less traffic because of its quota mechanism. HTTP protocol can instruct the client to send the request to another location instead of returning the requested page. Hence a front-end server can load balance traffic to a number of back-end servers.

A standard way to scale web applications is by using a hardware-based load balancer [3]. The load balancer assumes the IP address of the web application, so all communication with the web application hits the load balancer first. The load balancer is connected to one or more identical web servers in the back-end. Depending on the user session and the load on each web server, the load balancer forwards packets to different web servers for processing.

In this paper, LB algorithm is proposed which performs the job to resource matching in the dynamically varying Cloud environment. The job information is placed in the job pool and the resource information is placed in the resource pool. In the job pool, the job information such as job id, required free memory, required CPU speed, required node count and execution time of job is present. Similarly in the resource pool, the resource and topology related information such as resource id, available free memory, CPU speed, delay to reach the resource, number of hop count to reach the resource and the number of task remaining in resource or cluster queue are present. First this LB algorithm calculates the total load of the resource C by adding all the resource load information at time T as follows,

$$\sum C_{TN} = C_1 + C_2 + ... + C_N \qquad (1)$$

where C1, C2, …, CN are the Cloud heterogeneous resources in the Computational Cloud Environment. Then we compute the mean of Cloud resource M_{CN} by taking the ratio of the $\sum C_{TN}$ to the total number of resource available 'C'.

$$M_{CN} = \sum C_{TN} \ / \ C \qquad (2)$$

Then set the upper and lower boundary by using the equation and the parameter X. The Maximum and Minimum value of the of the resource C is expressed as,

$$C\alpha_X = M_{CN} + X \qquad (3)$$

$$C\alpha_Y = M_{CN} - X \qquad (4)$$

where X is the Boundary value.

Resource which has the load greater then $C\alpha_X$ is said to be overloaded and the resource which have the load lesser then $C\alpha_Y$ is said to least loaded by exploiting equation. A resource is Bearable load then resource load is in the state between $C\alpha_X$ and $C\alpha_Y$. Suppose, if the load information exceeds the boundary value, then the load balancing policies such as selection, information and location policy are considered to migrate the job to other resources which are below the boundary value. If the resource is overloaded then we find the

The Dynamic Scheduling (DS) algorithm using Boundary Value approach is as shown in Algorithm 1.

Algorithm 1
Dynamic Scheduling using boundary value approach

```
Algorithm LoadBalancing;
begin
    get information of all Cloud resource
    set the boundary value of the resource as Z
    if ( $\xi C_L < Z$ ) then
        set $C_N$ as minimum load resource $C_{min}$
    else if ( $\xi C_L > Z$ ) then
        set $C_N$ as maximum load resource $C_{max}$
    else
        set $C_N$ as moderate load resource $C_{bearable}$;
    for ( all the resources ) do
        if ( the load $\xi C_L$ of the resource $C_N = C_{max}$ ) then
            node is overloaded
            select (node)
        else if ( the load $\xi C_L$ of the resource $C_N = C_{min}$) then
                node is least loaded
                get the job for execution
            else
                resource is moderately load
                no need for job selection here
end;
```

This algorithm will get the resource load information from the Cloud Resources, which is present in the scheduler. First this DS algorithm assigns ξC_L , is the estimated load of Cloud Resource C. If the load ξC_L of the resource C_N falls below the Threshold load Z, then the resource load is assigned as,

$$C_{MIN} = C_N \qquad (5)$$

Suppose, if the load ξC_L of the resource C_N is greater than the threshold load Z, then the resource load is assigned as,

$$C_{MAX} = C_N \qquad (6)$$

In case, if the load ξC_L of the resource C_N is equal to the Threshold load Z, then the resource load is assigned as,

$$C_{bearable} = C_{N,} \qquad (7)$$

where ξC_L denotes the load of the resource C_N.

Using equations identify the overloaded resource and then finally use load balancing policies such as selection, information and location policy to migrate the job to other least loaded resources which are below the Boundary value. If the resource is overloaded means Job Selection (JS) algorithm is used which works as shown.

Job Selection (JS) algorithm

```
Procedure select (node);
begin
    for (all the overloaded resource)
     while ( $C_{Max}$ )
            find one job from resource queue and
            compute the completion time of job $FC_N$ in
each
            resource.
    for ( all least loaded resource )
            choose the best resource $*C_N$ having minimum
            completion time of job
    then migrate the job to the resource
end;
```

In some cases, the completion time of the job get doubled when it is migrated to the remote resource. So in-order to provide the better solution to load balancing, it is necessary to estimate the Finish time or Completion time of the job in remote resource before it is being migrated to the remote location. The Completion Time of Job (F) in each least loaded resource is calculated as follows,

$$FC_N = JT + F_W \qquad (8)$$

where the FC_N represents the completion time of job J_N at the resource C_N, JT denotes the latency of the job J_N to reach the resource C_N and F_W denotes the completion time of waiting job at resource C_N.

In the Job Selection algorithm for each job in the overloaded resource, it calculates the completion time of job in the remote resource and chooses the resource having minimum completion time as the best resource for job transfer to the Cloud resource. Here choosing the best resource for J_N is expressed as follows,

$$*C_N = minof\left(FC_{NI},\ldots,FC_{NN}\right) \qquad (9)$$

where $*C_N$ represent the appropriate Cloud resource for migrating job J_N. Where the FC_N represents the completion time of job J_N at the resource C_1 ,C_2 ,.., C_N. Once the Job Selection algorithm is executed successfully, we can realise the load balancing and also the work load can be distributed among all the Cloud resources. As a result of load balancing and job selection, waiting time and latency of the job is reduced, and increases the overall performance of the Cloud resource. Since Cloud Computing deals with the various on-the-fly adaptations of servers and processors available at that point of time that a client access with his set of jobs.

4. Energy Aware balancing using ErfDVS

Energy is the integral over time in power [2]. Energy consumption depends on the power potentially available in a device and since the resources in the cloud are all commodity based and needs to analyze the available power and use the energy to the maximum possible. Speed is cube root of Power and hence we need to scale speed in order to find the energy and a good energy aware scheduling can be done. Normally the processors run at their maximum frequency and they need to be scaled for a distributed application like a cloud environment.

Energy consumption can be reduced by reducing the supply voltage and it depends on the devices status and in a standby position the device consumes less power and our approach will first find the least loaded processor and the find the power consumed and in turn will give us the EFFICIENT cloud resource available. By varying the frequency of operation we can reduce the power consumption. We take the RAM frequency of operation into account ERfDVS (estimated ram frequency dynamic voltage scaling) since we are going to deal with the thin clients and the processor as a commodity in Cloud environment. According to the model in [7], total CPU power changes only when a job begins or ends its execution. The energy is equal to the product of total CPU power during the interval between the previous begin or end event and the current one and the interval duration I_i:

$$Energy = \sum P_i * T_i \qquad (10)$$

where i = 1 to 2 * Tot Jobs – 1 and

$$P_i = N_{fram} * P_{fram} + N_{idlefram} * P_{idlefram} \quad (11)$$

where I = 1 to n

n is the number of frequencies available in the used frequency set. P_{fram} is sum of dynamic and static components of CPU ram power running at the fth supported frequency. $P_{idlefram}$ can have two values as we already described. N_{fram} and $N_{idlefram}$ are numbers of processors running at the fth supported frequency and idling, respectively.

5. Experimental analysis and performance evaluation

In the result phase the main focus is to show the result, as the proposed Load Balancer performs well, when comparing to the Normal Load Balancer by considering all the challenging issues. We have simulated the result by exploiting ten resources and hundreds of jobs. The simulated values from parallel performance loads and the parameters for resource and job are taken for simulated cloud environment [5],[6]. The load is calculated by the arrival and service rate are calculated and is scheduled based on the load and scheduled by our load balancer.

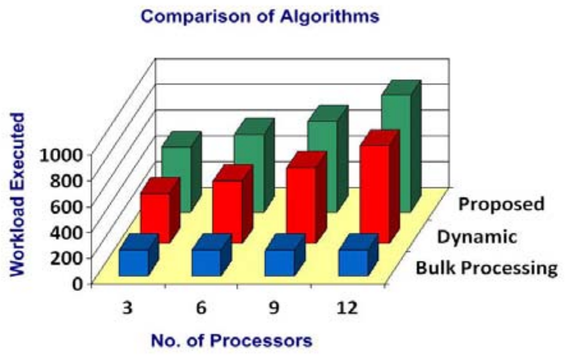

Fig. 2

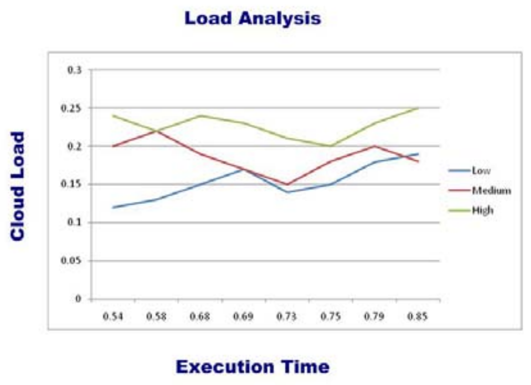

Fig. 3

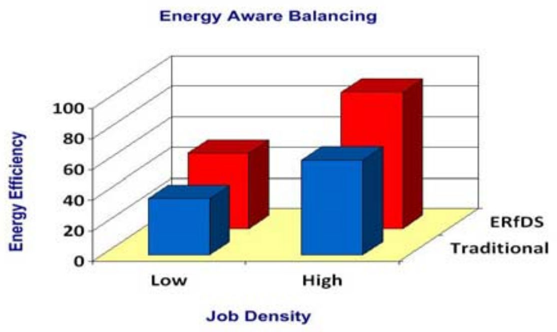

Fig. 4

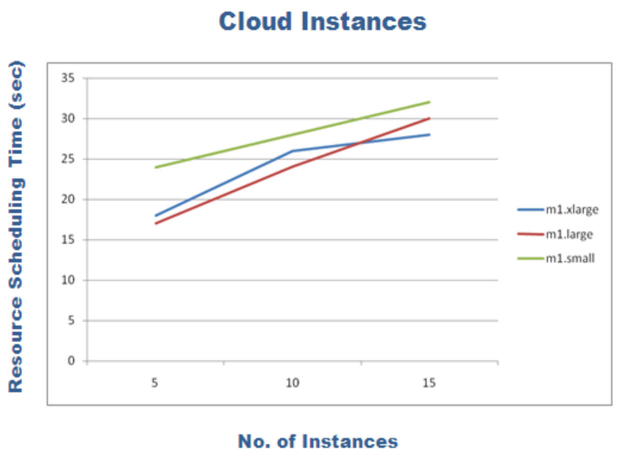

Fig. 5

6. Conclusion and future work

Our proposed Dynamic Scheduled Load Balancer model (DS) in the scheduler is developed with all the features. The result obtained with performance evaluation, can balance the load, decreases the elapse time and increase the utilization of the Cloud resource, which are idle or least loaded. So the obtained result shows the proposed Load Balancing algorithms perform better than normal Load balancer with respect to delay and load. The analysis is shown in the figure 2, 3, 4 and 5. These analysis shows that cloud computing environment can help in analyzing loads in various heterogeneous loads and the efficiency of the cloud resource can be harvested based on these analysis.

Our future work would be in working towards load balancing and Job Selection for the available appropriate Cloud resource between the scheduler in the real Cloud environment which is highly dynamic and distributed in nature, with everything available as a service it is very important for going for an appropriate Cloud resource which can offer the best Quality of Service for Cloud Customer at last it is the Customer Satisfaction which can lead a Good Technology as Cloud Computing.

References

[1] Menno Dobber, Rob van der Mei, and Ger Koole "Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison" *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 20, no.2, pp 207, Feb 2009.
[2] Encyclopedia of algorithms, Springer.
[3] F5 Networks. http://www.f5.com.
[4] Samee Ullah Khan, Ishfaq Ahmad, "A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids" *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 20, no. 3, pp 346, Mar 2009.
[5] D.Feitelson, Parallel workloads http://www.cs.huji.ac.il/labs/parallel/workload.
[6] U.S. Department of Energy, US Energy Information Administration (EIA) report, www.eia.doe.gov/cneaf/electricity/epm/table5 6 a.html.
[7] C. Hsing Hsu and W. Chun Feng. A power-aware run-time system for high-performance computing. Sc, 0:1, 2005.
[8] http://en.wikipedia.org/wiki/Cloud_computing.

Anandharajan TRV received his B.E. degree in Electronics and Communication Engineering from Anna University, Chennai, India in 2005 and M.E. degree in Applied Electronics from Anna University, Chennai, India in 2007. His current area of research is cloud computing and green IT. His current area of interest includes communication systems, operating systems and distributed computing. He is a Life member Computer Society of India, student member in IEEE and member in IACSIT and IAENG.

Dr. M.A. Bhagyaveni received her B.E. degree in Electronics and Communication Engineering from GCT, Coimbatore, India in 1997 and M.E. degree in Optical Communication from CEG, Gunidy, India in 1999 and Ph.D. degree from CEG, Guindy, India in 2006. She is currently working as Assistant Professor in the Department of Electronics and Communication Engineering, CEG Campus, Anna University, Chennai, India. Her present research interests include Wireless communication, Digital communication, MIMO systems, Ad hoc networks, Sensor networks, Cloud computing, Cognitive radio technologies. She has published more than 20 papers in National/International Conferences and Journals. She is a member of IEEE and several international association bodies.