International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Implementation of Energy Saving Mechanism for multi core Architecture in Cloud Computing

Kinnari Solanki[1], Prof. Tushar A. Champaneria[2]

[1]Department of Information Technology, L. D. College of Engineering, Gujarat Technological University, Ahmedabad, India

[2]Assistant Professor, L. D. College of Engineering, Gujarat Technological University, Ahmedabad, India

Abstract: ***Cloud computing denotes to Application and Services that run on a Distributed network using virtualized resources and accessed by Internet protocols and networking standards. Cloud computing makes dreams of utility computing with a pay-as-you-go, cloud is scalable, universally available system. The fast growing demand for computational power utilized by modern applications with rapidly changing Cloud computing technology has directed to the foundation of large-scale virtualized data centers. In order to integrate the system resource, utilize the resource flexibly, save the energy consumption, and meet the requirements of users in the cloud computing environment. From load balancing algorithm we can get dynamic queue of available server with their maximum free memory and response time. So in this research work we can allocate best server for upcoming request and stop that server which infrequently used. So we can stop unused server and this way we can try to save energy.***

Keywords: Cloud computing, Virtualization, Load Balancing

1. Introduction

Modern resource-intensive enterprise and scientific applications create growing demand for high performance computing infrastructures. This has led to the construction of large-scale computing data centers consuming enormous amounts of electrical power. Despite of the improvements in energy efficiency of the hardware, overall energy consumption continues to grow due to increasing requirements for computing resources. For example, in 2006 the cost of energy consumption by IT infrastructures in US was estimated as 4.5 billion dollars and it is likely to double by 2011 [1]. Apart from the overwhelming an operational cost, building a data center leads to excessive establishment expenses as data centres are usually built to serve infrequent peak loads resulting in low average utilization of the resources. Moreover, there are other crucial problems that arise from high power consumption. Insufficient or malfunctioning cooling system can lead to overheating of the resources reducing system reliability and devices lifetime. In addition, high power consumption by the infrastructure leads to substantial carbon dioxide (CO2) emissions contributing to the greenhouse effect.

A number of practices can be applied to achieve energy efficiency, such as improvement of applications' algorithms, energy efficient hardware, Dynamic Voltage and Frequency Scaling (DVFS) [2], terminal servers and thin clients, and virtualization of computer resources [3]. Virtualization technology allows one to create several Virtual Machines (VMs) on a physical server and, therefore, reduces amount of hardware in use and improves the utilization of resources. Among the benefits of virtualization are improved fault and performance isolation between applications sharing the same 0resource (a VM is viewed as a dedicated resource to the customer); the ability to relatively easy move VMs from one physical host to another using live or off-line migration; and support for hardware and software heterogeneity. Cloud computing naturally leads to energy-efficiency by providing the following characteristics:

- Economy of scale due to elimination of redundancies.
- Improved utilization of the resources.
- Location independence – VMs can be moved to a place where energy is cheaper.
- Scaling up and down – resource usage can be adjusted to current requirements.
- Efficient resource management by the Cloud provider

The energy consumption is not only determined by the efficiency of the physical resources, but it is also dependent on the resource management system deployed in the infrastructure and efficiency of applications running in the system. This interconnection of the energy consumption and different levels of computing systems can be seen from Figure 1.

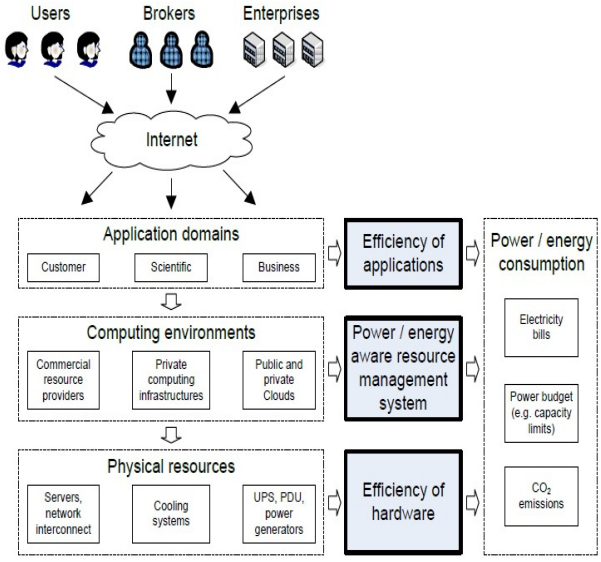

Figure 1: Energy consumption at different levels in computing systems.

2. Static and Dynamic Power Consumption

Multi-core consideration

- P-State (Performance State)
  No of P-state is processor specific. Higher P- state number represents **slower processor speed.**
- C-State (CPU Operating State)
  Higher the c-state number the deeper the CPU sleep mode. More components are shut down to save power.
  C0 – CPU fully turned on.
  C1 – First Idle State, Stops CPU main internal clock via software.
  C2 – Stop CPU main internal Clocks via hardware.

3. Proposed Algorithm

The pseudo code of the algorithm is as under

Pseudo Code

Step-1: **[**Calculate load factor x]

X<- (Total_Resourses – Used_ Resources)
//where x is free memory in terms of percentage.

Step-2: [Calculate Performance Factory]:
Y1<- average(current_response_time)
Y <- Y1 – (previously calculated Y1)
Y <- Y1(previous Y1) * 100
//counting Y in terms of previously counted Y.

Step-3: [finding Z]
Z <- X – Y
If(Z < 0)
Z=0

Step-4: [find minimum of all Z except the nodes with Z value 0]
Min_Z = min(all Z's)

Step-5: [find min_factor and divide all Z by the factor]
Min_factor <- Min_Z
Z <- Z(Min_factor)

Step-6: [Generate Dynamic queue on base of Z]
Step-7:[check number of request per second]
Step-8:[find the probability of Z]
Step-9: [stop the infrequently used server]
Step-10: [Start the server]
if number of request increases then start the server dynamically or based on probability of Z
End

In the above algorithm x is considered as a free load on server, y for the performance on the server and y1 is the current response time.

4. Explanation

Step-1:
The value of x is calculated by considering the total available resources and allocated resources on the server. The available resources would be calculated using the equation x = (Total _Resources – Used _Resources). Once x value is calculated for all nodes servers we get the available free load on the servers.

Step 2:
Here the performance factor calculates the increase or decrease in performance on the server and the calculated value is stored y. Now for calculating y a request is send to all the nodes at regular interval of time and the response time(total of both request time + response time) is calculated. So every hour sever would have various values of y and averaging all the value of y1 would be calculated to generate a queue. Now, the previously calculated y1 will be deducted from current values y1 which was currently used to calculate the performance (i.e. Response time increases or decreases). The same way the increased or decreased performance is calculated and the value of y is calculated as the percentage of previously counted y1. That is y/(previous y1)*100.

Step 3:
Counting z = x - y; Here Y value is subtracted from x value to count the z value Here we are interested in the node with the lowest response time hence we subtract the y value from x. i.e. nodes having more response time will contain less z value and they will
get less number of requests to handle. Now suppose in the worst case node have very less memory available and very large response time than z = x – y may get negative value so we need to remove that node from queue (think that it is temporary unavailable) and it will not consider in any step of the algorithm and in this iteration of algorithm it will not get any request to handle. If any node is temporarily unavailable the response time will be infinite (or very large) of that specific node. So the y value also becomes too much large for that node which will leads to minus z value. And in step 4 of algorithm that node will discarded for future process in this irritation of algorithm. So with this step we can also detect the unavailable node (with value of y as infinite or to large).

Step 4:
Once the z is calculated then the minimum of all z which we calculated are stored in min _z. Here we will not consider node with 0 z value so, this node will be remove from any further process and for this iteration of the algorithm. In the next iteration of the algorithm the z value of node will count again so it will get the next chance to join cloud environment if it is ready for it i.e. available.

Step 5:
Here minimum factor is calculated and divided by all the factors of z. Suppose z values for node 1 to 5 are given below:

Node z-value
N1 22.30
N2 12.23
N3 43.00
N4 14.36
N5 28.29

Now N2 has a list z value and so every nodes z value are divided by N2's z value and stores a float or cell value of it to make it easy and less complex. So now the z values are 2, 1, 4, 1, and 2. Node with z value 0 will get 0 as z / min_fact so they can't get any request to handle.

Step 6:
From the above value N1 has the capacity to handle two requests, node N2 can handle only one while N3 will handle 4 to keep load balanced on each node. So the temporary queue will look be prepared as follows:

N3	N3	N3	N3	N1	N1	N5	N5	N4	N2

Figure 2: Dynamic Queue

So, once the temporary queue and permanent queue will be changed and accordingly, first 4 requests will go to node 3 than 2 will go to n1 and so on until the end of queue. Once the queue is over it will assign next 4 to N3 and so on.

Step 7:
Now we check the number of request on server.

Step 8:
Find the probability of z factor means how many time the particular server is fulfil the user request. And find how many server are ideal bashed on this z.

Step 9:
As we discuss in step 8 we have list of ideal server so now we stop the ideal server.

Step 10:
If number of request are increased then start the server dynamically.

5. Experimental Result

Here we use server with 2 GHz Processor, 20 GB HDD and 1 GB RAM. If we take 5 server and per server it will handle 1000 user request. First when 1000 request arrives at that time all 5 server are working and power consumption increase.

Table 1: without applying any algorithm

user Request	Total Server	In use Server	Imp_cpu	Imp_eng	sml_cpu	sml_eng
1000	5	1	0.601	62.1	1.821	910.5
2000	5	2	0.816	167.2	1.836	918
3000	5	2	1.803	546.9	2.423	1211.5
4000	5	3	2.404	969.6	2.724	1362

As shown in table 1 when any number of user request arrives all server are working so power consumption is increased bashed on following energy consumption formula:

Power Consumption = [(CPU + MEM_OPERATION) * 100] * Number of used server

So after applying over proposed algorithm we can got the result as shown in table 2. Because each server handle 1000 user request so no need to use all server just start the server as per user request other server are stoped at that time and this way we can save the energy. Following graph shows the comparison of energy consumption with applying algorithm and without applying algorithm.

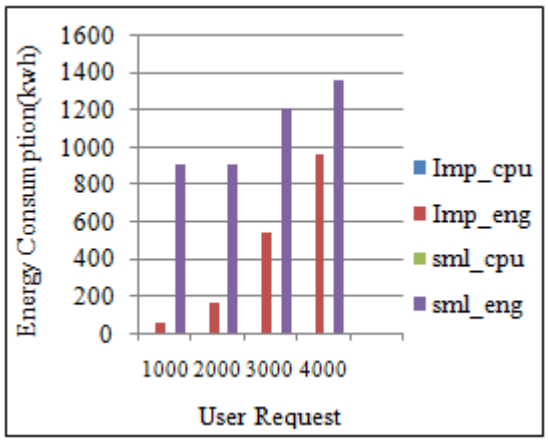

Figure 2: Energy Consumption Comparison

6. Conclusion

This Paper Proposed Dynamic available server queue generated with Energy Saving Mechanism in the cloud computing environment to reduce the energy consumption. By applying the proposed algorithm we can stop ideal server and this way we can save energy.The experimental result shows that the energy consumption can be saved by applying the proposed algorithm then without applying algorithm because when we not apply algorithm at that time all servers are running and its waste energy.

References

[1] R. Brown *et al.*, "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.

[2] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, 2002, pp. 29–42.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the 19th ACM symposium on Operating systems principles*, 2003, p. 177.

[4] A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems Anton Beloglazov1, Rajkumar Buyya1, Young Choon Lee2, and Albert Zomaya2 .

[5] N. Rodrigo, Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", ISSN:0038-0644, Wiley Press, New York, USA **(2011)**, pp. 23-50.

[6] Ratan Mishra1 and Anant Jaiswal,"Ant colony Optimization: A Solution of Load balancing in Cloud",*International Journal of Web & Semantic Technology (IJWesT) Vol.3, No.2, April 2012*

[7] Liang-Teh Lee, Kang-Yuan Liu, Hui-Yang Huang and Chia-Ying Tseng ," A Dynamic Resource Management with Energy Saving Mechanism for Supporting Cloud

Computing", International Journal of Grid and Distributed Computing Vol. 6, No. 1, February, 2013

[8] G. Praveen, "Analysis of Performance in the Virtual Machines Environment", International Journal of Advanced Science and Technology (IJAST), vol. 32, SERSC, **(2011)**, pp. 53-64.

[9] Pragati Priyadarshinee, Pragya Jain,"Load Balancing and Parallelism in Cloud Computing",International Journal of Engineering and Advanced Technology (IJEAT) Volume-1, Issue-5, June 2012