A Green Greedy Process Scheduler for Cloud Data Centers

C Karthik
Department of Computer Science
NITK Surathkal
India
karthikiyer2000@gmail.com

Aayush Gupta
Department of Computer Science
NITK Surathkal
India
aayushgupta.nitk11@gmail.com

K Chandrasekaran
Department of Computer Science
NITK Surathkal
India
kchnitk@ieee.org

***Abstract*—In this paper we have addressed a major problem in current day data centers- power consumption. Power consumption in data centers has become a major problem these days, both from economic and environmental perspective. Various factors affect the power consumption, one of them being the scheduling of tasks on the data center servers. Basically we achieved a real-time simulation of two cloud scheduling algorithms and compared the power efficiency of the two algorithms in terms of two main temperature parameters of the servers-idle temperature and critical temperature. We assumed that we were given all the task parameters such as running time etc. and then we calculated a final temperature that a system will reach on running that particular task. Then we decided which system could accommodate that task based on that systems critical temperature and chose the best system among those based on a score proposed in the paper.**

I. Introduction

Cloud Computing is Internet-based development and use of computer technology. It is a technology which provides IT-based services over the internet. Cloud computing mainly relies on resource sharing. This in turn helps achieve huge scalability over large scale distributed systems. Cloud computing saves the cost of large infrastructural setup and their operational costs. As a result over the years cloud computing has emerged as a widely used technology with large enterprises shifting their computational operations to cloud.

One of the major problems with any IT based system has been the huge energy consumption which account for a significant share of the worlds electricity consumption [1] [2]. Cloud computing has been no exception. With large scale distributed systems getting deployed on cloud the need to for huge data centers has rapidly increased over the past few years. Operational costs of data centers account for almost 50% of total cost and the energy related costs alone cost about 12 % of the net cost [2]. Extensive studies have been carried out in data center energy consumptions. All this energy and environmental awareness surrounding data centers and cloud in general brought into picture a new field of research- Green Computing.

The major areas of research in Green Computing are:

1) Green Algorithm: It basically deals with task consolidation i.e. assigning n tasks to r different resources without violating the various constraints of efficiency.
2) Data Center management: The power consumption by data centers has become a major concern for cloud service providers. Various techniques are used to reduce the overall energy consumption over the various servers of the data centers. Few of them are sleeping method, system components turning on/off etc.
3) Efficient VM Management: The basic idea is to deploy various VMs on a single server to give the effect of large number of servers being run using a few number of physical servers. This method leads to the lesser power consumption due to reduced number of hardware.
4) Product Longevity: Product longevity helps in ensuring the intelligent utilization of resources in the manufacturing of products and solutions. The general trend these days is to dispose off dysfunctional servers and hard disks. Due to increasing demand of data centres, the companies providing cloud services should ensure that the hardware in their data centers should have higher longevity thus reducing e-waste.

The objective of this paper is to propose an efficient Green Algorithm. Green algorithms basically constitute of scheduling algorithms that allocate tasks in a way such as to save power consumption as much as possible.

II. Related Work

Green Computing has become a parallel area of research with cloud computing. The increasing energy and environmental crisis has led to governments and large companies shift their focus from efficient to energy efficient technologies. Extensive research has been going on in all areas of green computing. CompatibleOne is one such open source cloud management software [3]. It acts as a cloud service broker and is used for energy monitoring and energy efficient management of cloud systems. VM monitoring is another major area of research. VM monitoring algorithms such as in [4] talk about techniques to monitor the power consumption using indirect methods and schedule tasks accordingly. In [5],a model has been proposed to decide whether a VM should be shutdown, hibernated or kept on standby based on various energy parameters. The model when implemented could save upto 46% of energy.

Classification of the tasks on the basis of being compute intensive, storage intensive, I/O intensive, Compute storage intensive, and Compute I/O intensive is another useful model [6]. The model after classifying the tasks on the above mentioned parameters, considers parameters like number of CPU cycles being used, modes of storage devices (Idle, Standby, Sleep) and speed of storage devices etc for allocating the tasks to the systems ensuring minimum energy consumption and the model concludes that there is a turning point for CPUs frequency and storage devices speed and mode which ensures minimum energy consumption.

Another major area of research is the cooling and temperature monitoring strategies used in data centers based on the amount of load being handled [7]. Cooling accounts for a huge share of data center costs. Moreover efficient cooling can reduce the power consumption in data centers.

Machine Learning is a major field of study these days. People have tried to come up with intelligent systems that could reduce the energy consumption on clouds. Using Artificial neural networks(ANN) [8] is a way out for efficient task scheduling in cloud. ANN satisfy many metrices such as throughput, fault tolerance, response time and resource utilization. It can work efficiently with noise and incomplete information. ANN further ensures to distribute dynamic workload across multiple nodes to ensure no single node is overwhelmed, while others are idle or doing little work. Our paper mainly focuses on energy efficient resource allocation strategies in data center servers. Resource allocation and scheduling in a cloud can be generalized into 3 main steps [9]:

1) Resource discovering and filtering - Data Center Broker discovers the resources present in the network system and collects status information related to them.
2) Resource selection - Target resource is selected based on certain parameters of task and resource. This is deciding stage.
3) Task submission - Task is submitted to resource selected

Many scheduling strategies have been proposed for minimizing the energy usage of the systems. One such system proposed is the e-STAB [10] which takes into account traffic requirements of cloud applications providing energy efficient job allocation and traffic load balancing in data center networks. Another such algorithm proposed is the RASA [11] which takes advantage of the Max-Min and Min-Min scheduling algorithms and has been seen outperforming existing algorithms in large scale distributed systems. In [12] ,the authors tested 6 different scheduling algorithms and finally concluded that heuristically assigning to computer with lower energy is better than assigning more tasks to randomly selected computers.

In [13] a priority is assigned to each admitted queue. Admission of each queue is decided by calculating tolerable delay and service cost. Advantage of this algorithm is that this policy with the proposed cloud architecture has achieved very high (99%) service completion rate with guaranteed QoS. As this policy provides the highest precedence for highly paid user service-requests, overall servicing cost for the cloud also increases.

III. CONCEPTUAL MODEL

In each data center there is a constant process of heating and cooling going on. The heating of the systems happens due to the task running on the servers. The cooling is done by using cooling towers and other cooling mechanisms. The way tasks are assigned to different servers is a major deciding factor for the amount of power consumption. The most important parameter to consider for reducing power consumption is heating which directly leads to temperature considerations of the systems. Each server in a data center is associated with a number of temperature parameters [14]:

1) Idle Temperature : This is the minimum temperature that a running server will attain when it is left unused for a very long time.
2) Critical Temperature : This is the maximum temperature that a server can handle. If the temperature of the server goes beyond this temperature then the server will shut down automatically.
3) Rate of Cooling : This is basically the rate at which the server will be cooled in a data center when it is idle. The cooling stops when the temperature of the server reaches the idle temperature.

Whenever a task arrives at a data center it is also associated with a number of parameters:

1) Running Time : This is the amount of time that the task will run on a particular task will take for its completion when it has been assigned to a system.
2) Rate of Temperature Increase : This is a parameter that is specific to the task. Each task has its own resource needs such as memory needs etc. Hence based on the resource needs of each task it will have a Rate of Temperature Increase. component; formatting; style; styling;

We can have two logical relations between temperature and power consumption :

- With the increase in temperature of the system the power consumption of the system will increase.
- The power consumption of a system is also decided by the relative temperature load that a system has to bear while running a task. The temperature load is basically how close the current temperature of the system is to its critical temperature.

Our model is based on the second relation i.e. temperature load of a system affects the power consumption of the system. In our model we have proposed that for a given number of systems and a given number of tasks, it is better to assign each task to a system that has to bear the least load on running that task. The following steps describe the implementation of our proposed algorithm.

A. Steps of our green greedy algorithm

- All the processes with arrival time $\leq$ the current time in the environment will be added to queue.
- Take a process present in the queue.
- For each of the system which is free find the final temperature of the system after the process is executed on it and calculate the load factor.
- After finding the load factor for every idle system assign the task to the system with minimum load factor value and remove the process from the queue. The load factor is incremented by the value equal to the minimum load factor we found.
- Repeat step 3 and step 4 till the queue becomes empty.

B. Proof of Correctness

let the number of tasks in our queue = M
let the number of servers in our data center = N
let at any time t the number of tasks scheuled by m
let G be the currently chosen configuration for the m tasks
G=[g_1, g_2, g_3......g_m]
let O be some optimal configuration such that O!=G
let O=[g_1, g_2, g_3...g_{i1}, o_i, o_{i+1}.....o_m]
the configuration O is same as G for the first i-1 tasks.
If we replace o_i by g_i then
new_ netload=netload-load(o_i)+load(g_i)
As we know that our algorithm chooses the least load at every step hence load(o_i) $\geq$ load(g_i)
therefore new_ netload$\leq$ netload
As O is an optimal solution and each system o_i can be replaced by an equivalent system g_i therefore G is also optimal, thus completing the proof.
The given proof is a typical greedy algorithm proof which shows that a greedy algorithm at any moment will provide a configuration that is atleast as good as an hypothetical optimal solution.

IV. EXPERIMENTAL SETUP

For the experimentation purpose simulated an environment as close as possible to a real time data center environment. For that purpose we went through the white papers of the data centers [15] of a few companies and came up with the following values for our server and task parameters (all in °F) [16]:

- Idle Temperature : 68 - 72
- Critical Temperature: 77 - 82
- Rate of cooling (per second) : 0.06 - 0.3
- Rate of Increase in Temp (per second) : 0.06 - 0.55

Along with the features specified above and each process had a running time in the range 5-10 seconds and arrival time in the range of 1-15 seconds. Thus while simulating the code, we generated an environment similar to the data centers of the cloud.

We used the following temperature load score for a task T relative to a system S:
final_temp=final temperature of a system after it finishes running the task on it.
*final_temp(S,T)=current_temp(S)+running_time(T) *rate_of_heating(T)*
score=(final_temp(S,T)-idle_temp(S)) / (critical_temp(S)-idle_temp(S))
The denominator in the score is basically a normalising factor to bring the score in a range of (0,1). The larger the score, the more load the system is bearing.

A. Relation between our score and actual Power Consumption

Usually in a data center all the servers tend to have the same idle temperature and critical temperature values. In such a situation our score merely becomes a measure of temperature increase in a system on running a particular task. Introducing the denominator in our score makes it more robust and can accommodate different types of servers in the same score. The denominator is fixed for each server hence the score becomes proportional to the temperature increase in the system. Studies suggest that for every 1 °F decrease there can be upto 4-5% [17] saving in power consumption. Our score is fully dependent on the temperature rise of systems hence it is a direct proportionality measure of the power consumption between the different algorithms

We simulated a random environment by generating a particular number of tasks and systems, giving them random values for their various parameters. The random values of the parameters were within the values specified above.
We then ran the algorithm given in Algorithm 1 to schedule our tasks to the various systems. This entire simulation was done by writing a python code.

We tested the results of the simulation of Algorithm 1 against a **Basic Algorithm** [14]. The difference between our algorithm and the basic algorithm is that the basic algorithm chose the first system that could accommodate a particular task and assigned the task to that system.

Algorithm 1 Greedy Scheduling System Allocation Algorithm

```
$queue \leftarrow list\ of\ tasks$
$system \leftarrow list\ of\ servers$
$system \leftarrow 0$
**for** $S = system[0]$ **to** $S = system[len(system)]$ **do**
  **if** $S\ is\ free\ AND\ current\_temp(S) > idle\_temp(S)$ **then**
    $current\_temp(S) \leftarrow current\_temp(S) - rate\_of\_colling(S)$
  **end if**
**end for**
**while** $queue \neq NULL$ **do**
  **for** $T = queue[0]$ **to** $T = queue[len(queue)]$ **do**
    $best\_sys \leftarrow -1$
    $min\_score \leftarrow 1$
    **for** $S = system[0]$ **to** $S = system[len(system)]$ **do**
      **if** $S\ is\ free\ AND\ final\_temp(S,T) < critical\_temp$ **then**
        $score \leftarrow calculate\_score(S,T)$
        **if** $score < min\_score$ **then**
          $best\_sys \leftarrow S$
          $min\_score \leftarrow score$
        **end if**
      **end if**
    **end for**
    **if** $best\_sys \neq -1$ **then**
      $queue.remove(T)$
      $best\_sys \leftarrow busy$
      $current\_temp(best\_sys) \leftarrow final\_temp(S,T)$
      $netload \leftarrow netload + min\_score$
    **end if**
  **end for**
**end while**
```

V. Results

We ran both our algorithm and the basic algorithm for 5 different scenarios having 10, 20, 30, 40 and 50 servers. Each scenario had the number of tasking varying from 1 all the way to 500. The following images show the results that we got. In the images

- TSR stands for Task to System Ratio
- PA stands for Proposed Algorithm
- BA stands for Basic algorithm
- Cycle Ratio means the number of cycles taken to schedule all the tasks i.e. assign each task a particular system
- Load Ratio is the ratio of the score that we have proposed above for the two algorithms i.e. PA to BA.

A. Graphs showing the Load Ratio of PA to BA

The PA to BA Load Ratio is the ratio of the total loads in the data center when the tasks were assigned to the systems using the Proposed Algorithm(algorithm given by us) and the Basic Algorithm.

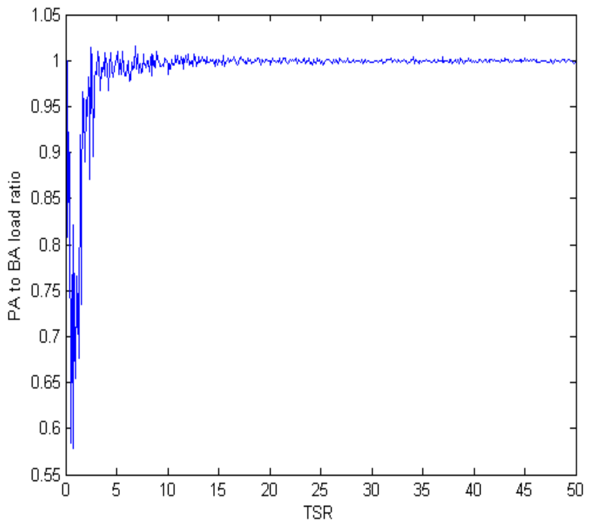

Figure 1. load graph-number of systems=10

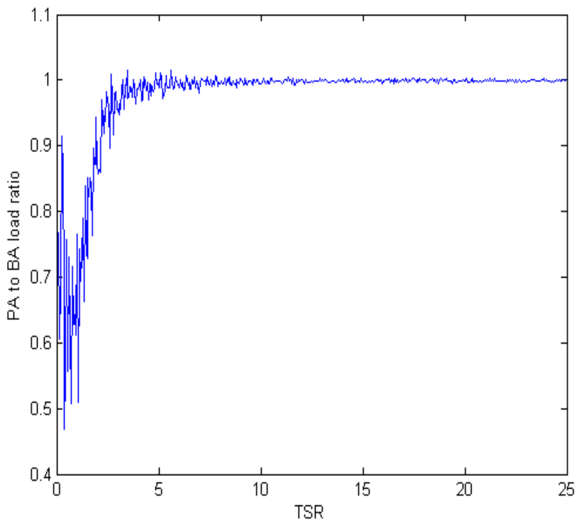

Figure 2. load graph-number of systems=20

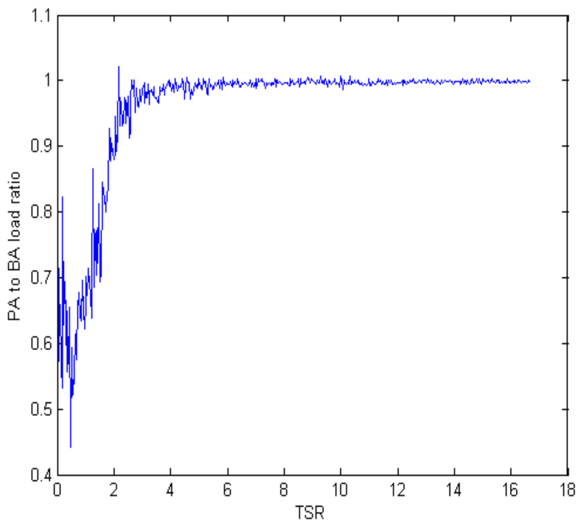

Figure 3. load graph-number of systems=30

It can be seen that the initial load ratio started from somewhere around 0.5 and kept on increasing to finally stabilize around 1. This is a direct consequence of proof of correctness of the given algorithm.

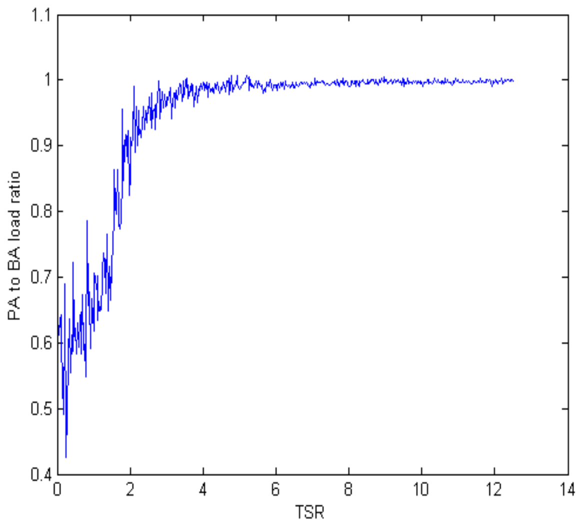

Figure 4. load graph-number of systems=40

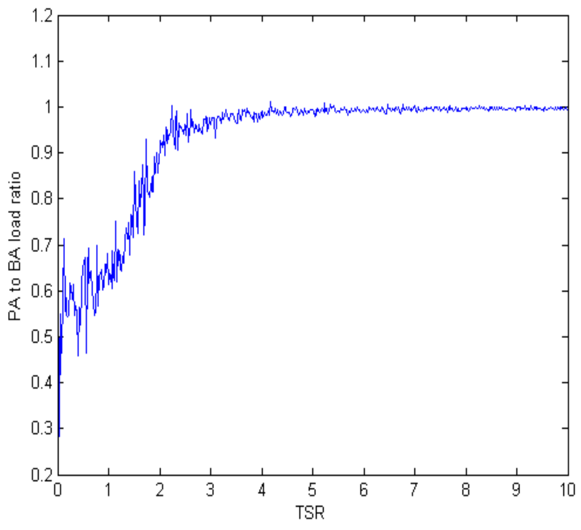

Figure 5. load graph-number of systems=50

A Load Ratio value of

- 1 means that there is no difference in the final loads obtained by the two algorithms i.e. there is no load relaxation obtained by the Proposed Algorithm in comparison to the Basic Algorithm.
- <1 means that the Proposed Algorithm gives a load relaxation compared to the Basic Algorithm.
- >1 means that the Proposed Algorithm provides no load relaxation. instead it increases the load on the data center.

As it can be seen in each of the Graphs shown in Figure 1. to Figure 5., the load ratio value converges to a stable vaalue of 1 when the TSR value is approximately 4. Beyond this point there is hardly any fluctuation in the Load Ratio value.

B. Graphs showing the Cycle Ratio of PA to BA

The PA to BA Computation Cycle Ratio means the ratio of the number of cycles taken to assign all the tasks that came in the data center to their respective system as per the two algorithms. In our simulation this cycle is the number of iterations for the entire simulation. As it is known that in algorithms the number of cycles is proportional to the time taken by that algorithm, hence the Cycle Ratio gives a clear idea of how time efficient are the two algorithms.

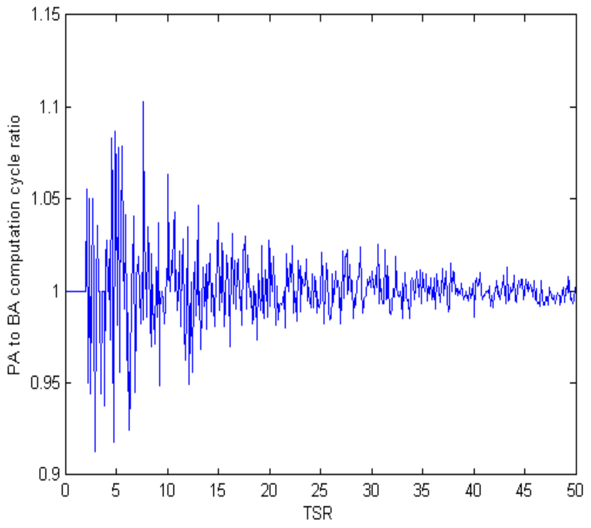

Figure 6. cycle graph-number of systems=10

A less than 1 value means that the time taken to schedule all the incoming tasks using the Proposed Algorithm (PA) is less than the time taken to schedule the tasks using the Basic Algorithm (BA). A greater than 1 value means than the time taken by PA is more the the time taken by BA.

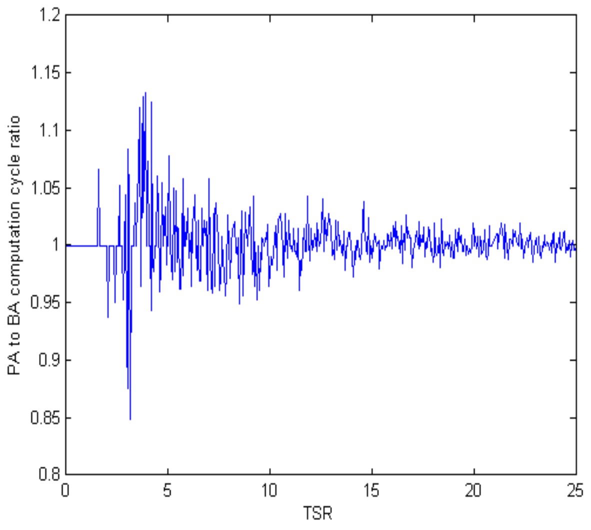

Figure 7. cycle graph-number of systems=20

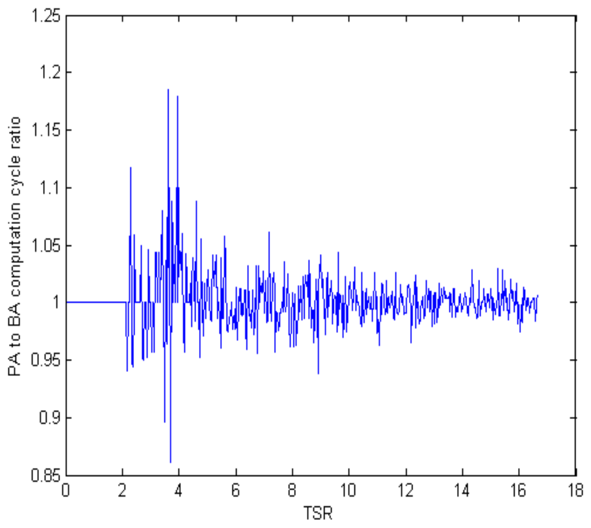

Figure 8. cycle graph-number of systems=30

In the cycle graphs of the scenarios shown above it can be seen that initially there was a huge fluctuation in the ratio of number of cycles. This value got stabilised to a value around 1 with the increase in the number of systems

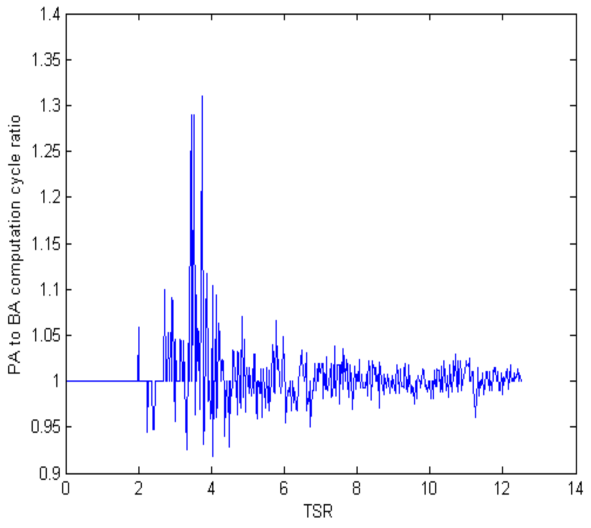

Figure 9. cycle graph-number of systems=40

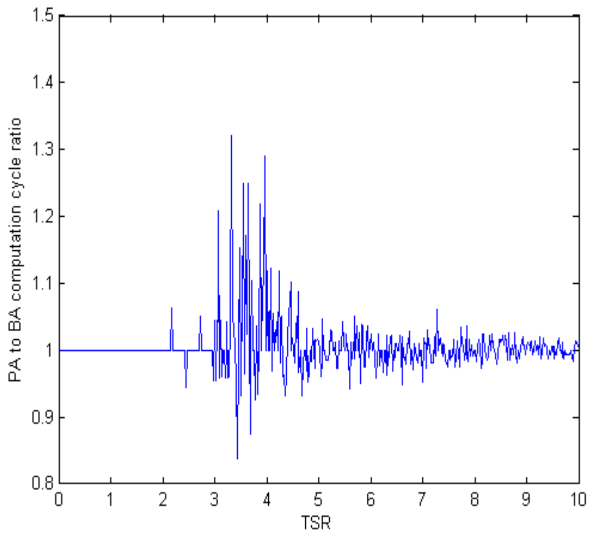

Figure 10. cycle graph-number of systems=50

The graphs in Figure 6. to Figure 10. show results similar to the Load Ratio graphs. Even over here the ratio converges to a range of 0.95 to 1.05 when the TSR value is approximately 4. Beyond this point there is very little fluctuation in the Cycle Ratio value.

VI. Conclusions

Seeing the Load Ratio and the Cycle Ratio graphs few interesting conclusions can be drawn. We proposed an algorithm that ran better than the basic algorithm that we used. As shown in the graphs, for TSR value ≤ 4, the load factor came out to be less for the Proposed Algorithm than the Basic Algorithm but there was a small trade off with the number of cycles used for scheduling the tasks, i.e. the number of cycles used to choose system for a task was comparatively more in proposed algorithm than the basic algorithm. However, for TSR value >4 both the algorithms showed the same result both in terms of temperature load and number of cycles used for choosing systems for the tasks. Every data center knows the amount of traffic it is receiving, if a data center is having the TSR value ≤ 4 then it can use our algorithm as have a considerable save in the amount of power consumption in comparison to the Basic Algorithm.

VII. Future Work

In this paper we measured a temperature load factor of the data center servers. Even though we gave a logical relation between this factor and the power consumption we could not have a direct comparison with the existing systems. Hence our future work will be converting the load factor to a direct power measure so that a better comparison can be established. We will try to implement this algorithm on a real data center to get real time performance results.

References

[1] "report to congress on server and data center energy efficiency," *U. E. P. A. E. S. Program*, Aug 2007.

[2] "Gartner says energy-related costs account for approximately 12 percent of overall data center expenditures." Mar 2011. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1442113

[3] J. Carpentier, J.-P. Gelas, L. Lefevre, M. Morel, O. Mornard, and J.-P. Laisne, "Compatibleone: Designing an energy efficient open source cloud broker," in *Proceedings of the 2012 Second International Conference on Cloud and Green Computing*, ser. CGC '12. IEEE Computer Society, 2012, pp. 199–205.

[4] Y. Li, Y. Wang, B. Yin, and L. Guan, "An online power metering model for cloud environment," in *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*, Aug 2012, pp. 175–180.

[5] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, April 2010, pp. 1–8.

[6] L. Luo, W. Wu, D. Di, F. Zhang, Y. Yan, and Y. Mao, "A resource scheduling algorithm of cloud computing based on energy efficient optimization methods," in *Green Computing Conference (IGCC), 2012 International*, June 2012, pp. 1–6.

[7] H. Shamalizadeh, L. Almeida, S. Wan, P. Amaral, S. Fu, and S. Prabh, "Optimized thermal-aware workload distribution considering allocation constraints in data centers," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 208–214.

[8] N. M. A. Sallami, "Load balancing in green cloud computation," in *World Congress on Engineering*, July 2013, pp. 7–13.

[9] P. A. Salot, "A survey of various scheduling algorithm in cloud computing environment."

[10] D. Kliazovich, S. Arzo, F. Granelli, P. Bouvry, and S. Khan, "e-stab: Energy-efficient scheduling for cloud computing applications with traffic load balancing," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 7–13.

[11] S. Parsa and R. Entezari-Maleki, "Rasa: A new task scheduling algorithm in grid environment," *World Applied Sciences Journal*, vol. 7, pp. 152–160, 2009.

[12] L. M. Zhang, K. Li, and Y.-Q. Zhang, "Green task scheduling algorithms with speeds optimization on heterogeneous cloud servers," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 76–80.

[13] D. M. Dakshayini and D. H. S. Guruprasad, "Article:an optimal model for priority based service scheduling policy for cloud computing environment," *International Journal of Computer Applications*, vol. 32, no. 9, pp. 23–29, October 2011, published by Foundation of Computer Science, New York, USA.

[14] S. Arora and V. Chopra, "A predictive energy aware hybrid resource scheduler for green cloud computing," *International Journal of Applied Research in Computing*, vol. 1, no. 3, October 2013.

[15] S. Z. P. Lin and J. VanGilder, "Data center temperature rise during a cooling system outage," Schneider Electric, Tech. Rep.

[16] "Energy efficiency guide data center temperature." Mar 2011. [Online]. Available: http://www.datacenterknowledge.com/archives/2011/03/10/energyefficiency-guide-data-center-temperature/

[17] "Server inlet temperature and humidity adjustments." [Online]. Available: http://www.energystar.gov/index.cfm?c=power_mgt.datacenter_efficiency_inlet_temp

[18] M. J. M. B. S. Arash Ghorbannia Delavar and M. K. Talebi, "Rsdc (reliable scheduling distributed in cloud computing)," *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, vol. 2, no. 3, June 2012.

[19] J. A. A. Beloglazov and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012.

[20] F. Cao and M. M. Zhu, "Energy efficient workflow job scheduling for green cloud," *IPDPS Workshops*, pp. 2218–2221, 2013.