Timed Redirection of Hybrid Server Requests to Enforce Energy Saving

Pragnesh Patel
Department of Computer Science and Engineering
University of South Florida
Tampa, FL, USA

***Abstract*—This paper focuses on optimizing the energy use of servers, while considering various aspects of the performance. When it comes to large scaled data servers, where numerous requests are being processed, the task of delayed redirection can prove to be quite a challenge. In this paper, presented is the policy that focuses on maximizing the performance of such servers, while paying close attention to the idle and serving periods of server.**

Keywords-component; simulation; energy saving; timed redirection

I. INTRODUCTION

In [1], the authors introduce the issue of energy use in majority of U.S. servers and provide a solution of reducing such costs through method of defining serving and non-serving period of servers. Using Dell OptiPlex 790 MT as the Master server and Marvell SheevaPlug PC as the assistant server for redirecting timed requests, the authors demonstrate the success of their proposed timed-redirection protocol. Though, the energy efficiency in servers is a fairly new branch of research in the field of computing, various distribution schemes have been proposed to tackle such issue. The authors of [2] bring to focus the need of accelerated research in this field and also justifies some of the trade-offs in energy saving.

For many users, the idea of delaying a server request may initially seem a backwards approach in the era where faster communication speed is in high demand. However, the studies have suggested that close to none noticeable difference in response time is an acceptable sacrifice in return to the resources being conserved.

In this paper, we build on the simplified form of policy introduced in paper [1], and through simulations on a proposed model discuss the correctness of the policy. The policy that determines the on and off periods of Master Server and the delay period calculation of the assistant server focuses on maximizing the throughput of the system, while reducing

i) Energy use

ii) Request delay

iii) Number of lost requests

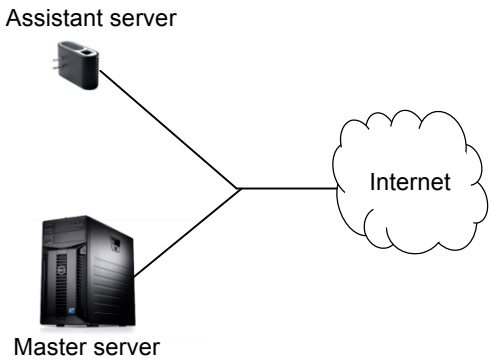

Figure 1. System configuration for hybrid web server

II. SYSTEM DESIGN

The system designed for the purpose of this paper, is based on the diagram proposed in Figure 1. The requests are initially made to the assistant server that is constantly accepting incoming requests and based on the current status of master server makes decision of redirecting the request with zero or more seconds. The master, when serving, processes requests based on First-Come-First-Serve bases and drop all the requests when sleeping. For the simplicity of the problem, this model follows M/D/1 queuing model, where requests are generated based on exponentially distributed inter-arrival periods (Poisson distribution) and requests are processes with constant serving time of 10 ms (with 1ms for assistant redirection).

It is obvious that if the master server is on 100% of the time, the requests are processed with the least delay and with 100% completion rate, but the energy saving is ignored. On the other hand, if the master server is off all the time, no requests will be served and maximum energy saving is achieved. As a middle ground, the server can be regulated with a cycle of fixed duration (called epoch), where one cycle consists of serving time + not serving time (described in Figure 2). The assistant server should then delay the request based on the serving period of master server. That is, it should not redirect any request to master server, during the time that its idle and should carefully redirect them with a delay, that wouldn't cause the service queue to exceed 50 while minimizing the average response time.

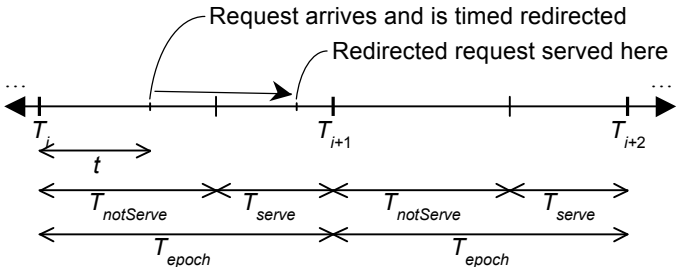

Figure 2. Arrival of requests and Master epochs

III. PROPOSED POLICY

The energy saving can be regulated in various ways. One of the solutions is to observe the mean arrival rate of the requests and based on the information, continuously varying the epoch periods of the master server. In other words, during the slow request periods the epoch cycle should consist of shorter serving periods than idle period and during high request periods, the phases should be altered.

The policy provided in this paper, keeps the epoch cycle and its phases constant, but uses Algorithm 1 to delay the request by varying periods. The algorithm is generated based on Figure 3[3], which compares the Poisson distribution for various values of Lambda. Since, the inter-arrival times in exponentially distributed generators follow a bell-shaped curve, the delayed periods of each request can be synced with its generation in order to acquire minimum response time.

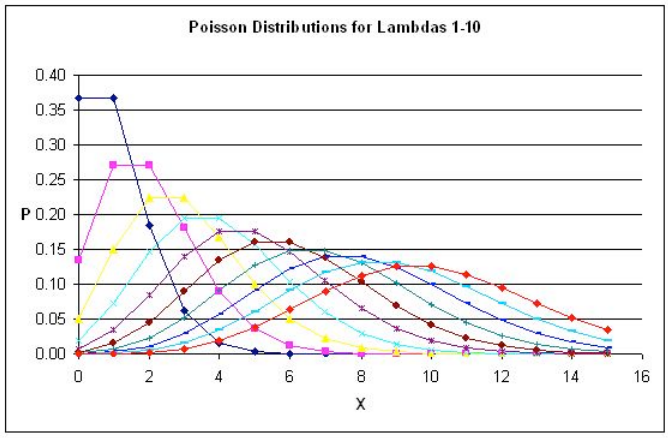

Figure 3. Poisson distribution for various values of lambda

Following is a sample algorithm that can be used to vary and synchronize the delay time with master serving time and generation. The entire simulation cycle can be divided into fixed number of *percentile* range, and using current_time/total_simulation_time, the *percentile* range of each request can be measured. Based on the phase the current request is in, the delay time of that request is incremented. Thus, for example, if the current simulation was between 20%-30% done, each request within that period is delayed by val_B before being sent to master server.

Algorithm 1:

```
if (percentile > 10 && percentile <20)
        delay_time = val_A;
else if (percentile >= 20 && percentile < 30)
        delay_time = val_B;
else if (percentile >=30 && percentile < 70)
        delay_time = val_C;
else if (percentile >=70 && percentile < 80)
        delay_time = val_B;
else if (percentile >= 80 && percentile < 90)
        delay_time = val_A;
else
        delay_time = not_serving_time;
```

Algorithm 1: The algorithm used in the model to compute the delay times

The proposed policy, pays close attention to the rate at which the request arrive at the assistant server. During the periods, when higher numbers of requests are expected the delay time is reduced and the master server is kept busy for longer periods of time. On the other hand, when a few numbers of requests are expected the server is idle for longer periods of time and thus saving more electricity than it normally would.

As a part of the experiment, the serving period of the epoch cycle was set at 30% using the algorithms and calculations discussed above and for 800 requests arriving the assistant server every second, the mean response time of 0.025 seconds with drop rate less than 0.025% was achieved. Additional results from the simulation on this model are discussed below.

IV. SIMULATION RESULTS

The model proposed in this paper was tested under various conditions and successfully achieved the expected results. The simulations were performed over varying length of simulation times with different amount of workload on the system in order to insure that the model performs equally under all the provided criteria. This proves to be an important factor of simulation, as the model can be extended to various types of data servers and thus needs to be validated for all the possible scenarios. One of the parameters that was closely observed during all the simulations is the average drop rate of the master server and it was ensured that this rate stays below 1%, since this was the acceptable trade-off for the energy saving.

Using the data from various simulations the following graphs were generated using the average value of each dependent value being observed. Figure 4 describes the average response time of the requests for different values of lambda. The proposed policy can be better evaluated to generate better response times during the times when fewer requests per seconds are made. We can see in the graph that the mean delay time of overall requests in less than 0.06 seconds, which is "close to none" difference discussed earlier. Such small delay should be welcomed in the servers that highly demand energy saving.

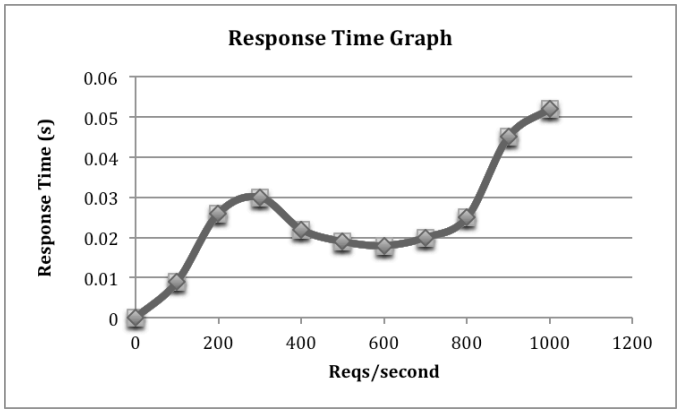

Figure 4. Arrival rate vs the response time based on presented policy

Using the same data used to generate the graph above, additional two parameters were observed during the simulation, graphs of which are presented below. In Figure 5, we can observe the increasing throughput of the system with increasing number of requests. The utilization graph follows the same curve.

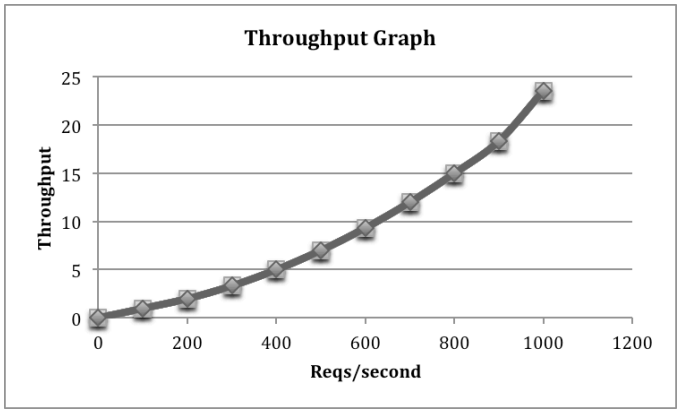

Figure 5. Throughput vs the response time based on presented policy

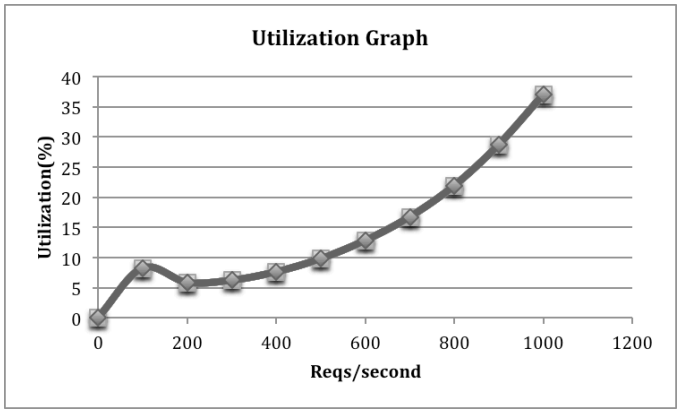

Figure 6. Utilization vs the response time based on presented policy

While conducting the experiments after implementing the initial policy, a higher number of drops in requests was observed. This problem was dealt with through method of trial-error simulations. Using different values for algorithm 1, I was able to compute values that resulted in the maximum performance of the system. The policy proposed in this paper can be further improved by synchronizing both the varying delay time of requests as well as the varying serving/non-serving periods of the master server. During higher arrival rates, the delay should be lowered and serving time should be increased. Similarly, during the slow arrival rates, only the serving period should be shortened while keeping the delay time as minimum as possible.

V. Related Work

The policy that we discussed focuses primarily on the M/D/1 queues, which makes it easier to implement and test the proposed algorithms. However, for non-constant service times of generated requests, the algorithm may not produce similar results for the chosen values. Future work on this policy can be performed where the assistant server is aware of both the expected arrival rate and expected service rate, and based on the results obtained from both it decides the delay time that would maximize the system performance, while minimizing the server use and dropped to completed ratio.

VI. Conclusion

In conclusion, the servers can be made more efficient in energy use with help of an assistant server, which based on the system characteristics handles request redirection. The low-powering assistant servers can regulate the serving periods of high-powered master server and without a noticeable difference in response period successfully assure the completion of most arriving requests.

Acknowledgment

- Mehrgan Mostowfi: Department of Computer Science and engineering, USF, FL.

References

[1] M. Mostowfi, K. Christensen, S. Lee, and J. Yun, "Timed Redirection: HTTP Request Coalescing to Reduce Energy Use of Hybrid Web Servers," submitted to IEEE Conference on Local Computer Networks in April 2012.

[2] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, “Managing energy and server resources in hosting centers,” in Proceedings of the 18th Symposium on Operating Systems Principles (SOSP),Oct.2001

[3] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In Proc. COLP, 2001.

[4] T. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for Web server end-systems: A control-theoretical approach. IEEE Transactions on Parallel and Distributed Systems, 13(1), 2002.