Performance Evaluation of the Weighted Least Connection Scheduling for Datacenters with BigHouse Simulator

Timothy Radtke
Electrical and Computer Engineering Dept.
Marquette University
Milwaukee, WI, USA
timothy.radtke@marquette.edu

Cristinel Ababei
Electrical and Computer Engineering Dept.
Marquette Universty
Milwaukee, WI, USA
cristinel.ababei@marquette.edu

***Abstract*—In this paper, we investigate the performance of a weighted least connection algorithm for scheduling jobs in datacenters. The novelty of the proposed algorithm is that the weights for the compute units in the datacenter are determined based on their current dynamic power consumption. The algorithm is implemented inside the BigHouse simulation framework and compared against the default least utilized scheduling approach of the framework. Simulation experiments show that the proposed algorithm provides significantly better performance for large number of large queries per second (QPS) values as well as lower power consumption. In addition, the computational runtime is linear with respect to the increase in queries per second. However, while linear, the computational runtime is longer than that of the default scheduler due to the increased computational complexity required to determine where a job should be placed. These results indicate a tradeoff between performance (i.e., latency of all scheduled jobs) and computational runtime of the scheduling algorithm.**

***Index Terms*—datacenter scheduling, weighted least-connection algorithm, datacenter simulator**

I. Introduction

Within the last two decades, we have witnessed dramatic development in datacenters and cloud computing. Developing efficient scheduling algorithms for datacenters is important because the primary goal of cloud computing is to use the resources of the datacenter in an efficient way so users can experience maximum throughput and speed while maintaining the datacenter's cost position [1]. The objective of this paper is to use the BigHouse datacenter simulator to investigate the weighted least connection algorithm for scheduling jobs in datacenters. The algorithm is implemented inside the BigHouse tool and compared against the default least utilized scheduling approach of the framework. The performance is evaluated as a function of the inter-arrival time between service requests.

The remainder of this paper is organized as follows. Section II provides a discussion of previous work. Section III discusses the BigHouse simulation tool used in this study. Section IV describes the proposed scheduling algorithm, which is implemented inside the BigHouse tool. Section V presents and discusses the results obtained. The paper is concluded in section VI.

II. Literature Review

At the datacenter level, job scheduling - in addition to workload placement, load balancing, power budgeting, and server farm management policies - represents one of the most popular methods to improve performance and reduce energy usage. As such, one can find many previous studies that focused on performance and power optimization; with some relevant examples including [2]–[6].

One common recent optimization approach in datacenters is the exploitation of hardware heterogeneity [7] introduced by continuous server upgrades and by multicore processors that integrate high-performance "big" and energy efficient "little" cores [8], [9]. Another trend is the combined management of several geographically distributed datacenters to save electricity costs while providing customers a uniform experience [10]. Hence, one can find previous studies that focus on scheduling jobs across these geo-distributed datacenters. One way to accomplish this is through the application of principles of locality to traditional scheduling algorithms [11]. In this approach, the jobs are scheduled to the datacenter with the most capacity that also has the data needed for the job. Researchers have found that when the data is geographically far away from the server assigned to the job, the speed of getting the data to the server outweighs any savings from a less busy faster server [11].

Many scheduling algorithms in previous studies have as an underlying approach one of the following strategies: round-robin, weighted round-robin, least-connection, and weighted least-connection [12]–[14]. For example, the highest density first (HDF) algorithm is similar to weighted round-robin scheduling, but performs the selection in a greedy manner [12]. The standard round-robin and least connection schedulers make the assumption that all servers are equal in computing capacity and throughput; however, this is almost never the case in practice. Therefore, the weighted variants are the most commonly used scheduling algorithms in practice due to their ability to adjust to various parameters across datacenter servers.

III. BigHouse Simulator

BigHouse is a methodology for system characterization and discrete-event simulation developed by the Advanced Computer Architecture Lab at the University of Michigan. It is a datacenter simulation framework that models datacenter workloads at scale. As such, it can be used to perform quantitative exploration of performance optimization, distributed data placement, power provisioning and management, and fault-tolerant design [15], [16].

The simulator is based on the stochastic queuing simulation methodology. This is different from traditional simulation tools, because workloads are not simulated at the granularity of an instruction, memory, or disk access; instead, the simulator exploits the theoretical framework of queuing theory, and uses as fundamental unit of work the job (i.e., task). It has proven itself to have a small error rate of 9.2% when compared to real web search data. Also, the power capping model is useful for real-world implementation of datacenter equipment. Overall, BigHouse provides an excellent platform for the exploration and comparison of various scheduling algorithms as applied to datacenter computing.

The BigHouse simulator comes with a default scheduling algorithm that is designed to find the least utilized socket on a server and assign the job to it [16]. The simulator can measure performance by generating curves of latency vs. utilization as a percentage of maximum queries per second (QPS) for several different types of workloads including DNS, Mail, Shell, Google, and Web. This curve is generated by running the statistical model built within the BigHouse simulator for different values of the QPS parameter.

When considering job placement within the datacenter, BigHouse abstracts the various pieces of the datacenter infrastructure. At the top level is the datacenter structure which holds various server classes within it. The server class contains cores, and each of the cores contain sockets. The socket is the base unit on which a job runs. For this study, a datacenter will be composed of 100 servers each with 4 cores and a variable number of sockets.

The source of BigHouse is build in an abstracted way such that it is easy to modify various parts of the datacenter without affecting the simulator as a whole [16]. Additionally, the flexibility in experimentation provides the capability to statically control the QPS value, thus reliably reproducing latency information that is comparable across schedulers. Hence, the server, socket, and accuracy sensitivity experiment source files are the only ones that require modification for integrating a new scheduling algorithm.

IV. Proposed Scheduling Algorithm

The least connection algorithm underlines many previous greedy or heuristic algorithms. Fig. 1 is a graphical description of this algorithm, where various servers, each with its own weight, gets assigned different jobs based on its weight and the number of jobs already being served. The variant implemented in this study is a weighted least connection (WLC) algorithm where the weight is based on the dynamic power of the socket being considered for the job. By ranking each of the sockets on a core within a server by their dynamic power consumption and then considering the one with the least connections we maximize both power efficiency and the performance of the overall datacenter.

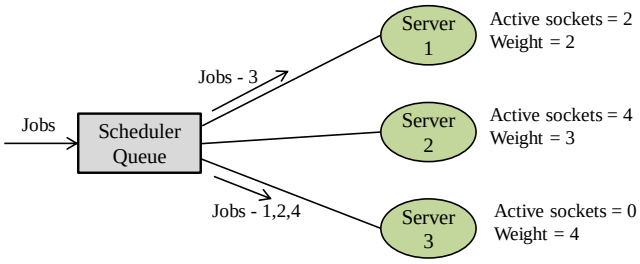

Fig. 1. Graphical representation of the least connection algorithm determining the proper socket to deliver a job based on the number of current jobs and weight of each server.

Algorithm 1 describes the scheduling algorithm, which selects for a given job the appropriate socket based on the highest weight and least number of connections. This happens during the start of a job's service routine within the server class. Prior to assigning a job to a socket, all the socket's weights are updated using an efficient routine described later in Algorithm 2. Once the weights are recalculated, each socket is examined to determine which socket is the least utilized (i.e., the one with the least connections). During the search for the socket with the fewest connections, we also consider the weight of that socket. Thus, we end up choosing a socket that is the highest weight of the least utilized sockets. In the event of a tie, the socket chosen is the first one to be examined.

Since each socket will need to be assigned a weight, an efficient method for (re)calculating socket weights during each pass of the scheduler needs to be implemented. Dynamic power is chosen as the metric for the weight calculation because by considering how much power each socket is consuming and favoring those consuming less, we can increase the energy efficiency of the overall datacenter without degrading performance. When the datacenter is first initialized, each socket receives a random integer weight between 0 and 1000. Since each socket will have the same dynamic power on initialization, this randomization helps smooth out the initial job loading. After initialization, Algorithm 2 describes how the weights for each socket are calculated during the insertion of jobs to the datacenter. The weight is increased or decreased based on specific power value thresholds. These thresholds were determined empirically by surveying the dynamic power values during the use of the default scheduler for a variety of workloads. During this empirical experimentation, it was found that $Threshold\_1 = 17$ - $Threshold\_2 = 20$ was the average dynamic power for most of the sockets at an average load. Hence, those sockets consuming more than $Threshold\_2$ would have their weight decreased, while those sockets consuming less than $Threshold\_1$ would have their weight increased during the recalculation process. Addition-

Algorithm 1 Creation of Job Service for Insertion into Datacenter

```
procedure StartJobService(time, job)
    targetSocket ← null
    leastUitilizedSocket ← null
    lowestUitilization ← MAX VALUE
    highestWeight ← MIN VALUE
    WLCSocket ← null
    for j ← 0, j < num sockets, j+ = 1 do
        ReCalculateWeight(socket)
    end for
    for i ← 0, i < num sockets, i+ = 1 do
        currentSocket ← sockets[i]
        currentUtilization ← Utilization(sockets[i])
        currentWeight ← GetWeight(sockets[i])
        if currentUtilization < lowestUitilization then
            lowestUitilization ← currentUtilization
            leastUitilizedSocket ← currentSocket
            if currentWeight > highestWeight then
                highestWeight ← currentWeight
                WLCSocket ← currentSocket
            end if
        end if
    end for
    targetSocket ← WLCSocket
    InsertJob(time, job, targetSocket)
end procedure
```

ally, to prevent the weights from crashing, if a weight ever reduced to below 0 a new random weight would be assigned to that socket. This helps to provide a more balanced utilization of all sockets as those with large negative weights would end up not being used, which effectively decreases the number of available resources.

Algorithm 2 Recalculation of Weights During Datacenter Operation

```
Ensure: num sockets ≥ 2
procedure ReCalculateWeight(socket)
    socketPower ← dynamicPower
    if socketPower ≥ Threshold_2 then
        weight ← weight − 1
    else if socketPower ≤ Threshold_1 then
        weight ← weight + 1
    else
        Weight stays the same
    end if
    if weight < 0 then
        weight = RandomWeight(void)
    end if
end procedure
```

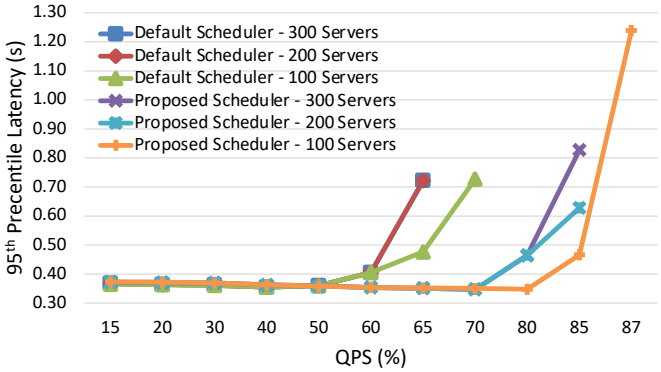

Fig. 2. Comparison of performance achieved with the proposed scheduler against the default scheduler for different number of servers.

V. Simulation Results

In this section, we present simulation results obtained with the proposed scheduling algorithm and compare them against those obtained with original default scheduler of the BigHouse datacenter simulator. These simulations are conducted for different numbers of servers in order to study the scalability of both schedulers for an increasing number of servers. Specifically, we simulated datacenters with 100, 200, and 300 servers.

A. Simulations

The comparison in terms of performance (as the 95^{th} percentile latency) versus queries per second (QPS) for each of three different datacenter sizes is shown in Fig. 2. We observe that the proposed weighted least connection algorithm outperforms the default scheduler in that the latency remains low for larger values of QPS when the proposed scheduling algorithm is used. This can be observed in Fig. 2, where the saturation point (i.e., the point where the latency curve goes up abruptly) is located at larger values of QPS (about $QPS = 80 - 85\%$) when the proposed scheduling algorithm is used compared to where it is located when the default scheduler is used (about $QPS = 60 - 65\%$). We also observe that both schedulers show good scalability with the number of servers; although, when the number of servers increases from 100 to 200 performance degrades slightly, whereas when the number of servers changes from 200 to 300, performance is less affected.

The comparison in terms of socket power consumption versus QPS for each of three different datacenter sizes is shown in Fig. 3. We observe that the power consumption is consistently smaller when the proposed scheduling algorithm is used. That is, for a given value of QPS, the power consumption when the proposed scheduler is used is less than the power consumed when the default scheduler is used. For example, at $QPS = 60\%$, the power consumed when the proposed scheduling algorithm is used is about 10 Watts while the power consumed when the default scheduler is used is 13 Watts. On average, across all QPS values for which both schedulers were

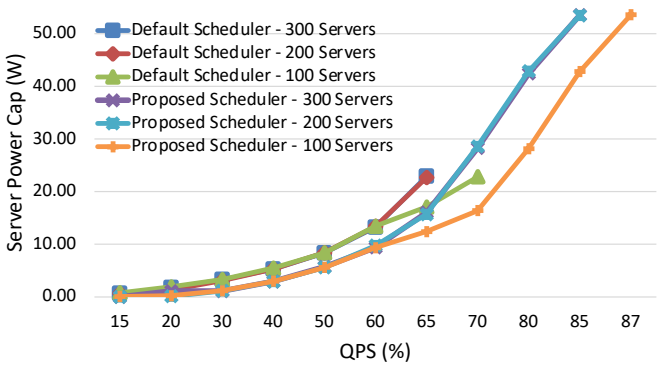

Fig. 3. Comparison of power consumption achieved with the proposed scheduler against the default scheduler for different number of servers.

simulated, the power consumed is less with about 50% for all three cases of server count when the proposed scheduler is used, as shown in Table I.

TABLE I
SOCKET POWER CONSUMPTION REDUCTION WHEN PROPOSED SCHEDULER IS USED VS. DEFAULT SCHEDULER.

	100 servers	**200 servers**	**300 servers**
QPS (%)	**Power Reduction (%)**	**Power Reduction (%)**	**Power Reduction (%)**
0.1	89.16	96.16	98.60
0.2	87.24	90.25	34.92
0.3	64.49	63.67	65.45
0.4	46.08	44.28	44.19
0.5	33.34	33.57	31.60
0.6	30.52	27.09	29.07
0.7	28.14	30.71	29.89
AVG.:	**54.14**	**55.11**	**47.67**

B. Discussion

These results demonstrate the importance of choosing an efficient scheduling algorithm to maximize datacenter performance. Additionally, they provide insights into the tradeoff between a computationally fast scheduling algorithm and overall datacenter efficiency. While the least utilized core scheduler used as a baseline (i.e., the default scheduler) is easy to compute and simulate, datacenter performance degrades as the QPS value increases. On the other hand, using a weighted least connection scheduling algorithm (i.e., the proposed scheduler) increases the initial computation since all weights need to be updated before a job is assigned, it provides an improved and more consistent performance across a larger range of QPS values. The proposed scheduling algorithm has a higher computational complexity, which in our simulations translated into longer execution times of the scheduling algorithm by 40% on average. However, the datacenter latency improves significantly at large QPS values (Fig. 2) and power consumption is reduced as well (Fig. 3).

VI. Conclusion

With the increasing reliance on cloud computing, it is imperative to ensure the performance of datacenters through the use of job scheduling algorithms. This study aimed to use the BigHouse simulator to verify the performance of a weighted least connection based scheduling algorithm and compare it against the default algorithm included with BigHouse. Simulation experiments showed that the proposed weighted least connection scheduling algorithm provided significantly better performance for many large queries per second values as well as lower power consumption. However, that comes at the expense of an increase in the execution time of the proposed scheduling algorithm due to its higher computational complexity. The results indicated a tradeoff between performance, power and computational runtime of scheduling algorithms.

Acknowledgment

The authors wishes to thank Marquette University for supporting courses such as Computer Architecture. Additionally, the authors wish to thank D. Meisner and the University of Michigan for creating and providing as an open source project the BigHouse simulation tool.

References

[1] T.J. Nirubah and R.R. John, "Energy-efficient task scheduling algorithms for Cloud Data Centers," *Int. Journal of Research in Engineering and Technology*, vol. 3, no. 3, pp. 322-326, 2014.
[2] J.Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": temperature-aware workload placement in data centers," *Annual Technical Conference on USENIX (ATEC)*, 2005.
[3] N. Liu, Z. Dong, and R. Rojas-Cessa, "Task and server assignment for reduction of energy consumption in datacenters," *IEEE Int. Symposium on Network Computing and Applications (NCA)*, 2012.
[4] M. Lin, Z. Liu, A. Wierman, and L.L. Andrew, "Online algorithms for geographical load balancing," *Int. Green Computing Conference (IGCC)*, 2012.
[5] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," *Int. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2013.
[6] V. Petrucci, M.A. Laurenzano, J. Doherty, Y. Zhang, D. Mosse, J. Mars, and L. Tang, "Octopus-man: QoS-driven task management for heterogeneous multicore in warehouse scale computers," *IEEE Int. Symposium on High Performance Computer Architecture (HPCA)*, 2015.
[7] J. Mars and L. Tang, "Whare-map: heterogeneity in "homogeneous" warehouse-scale computers, "*Int. Symposium on Computer Architecture (ISCA)*, 2013.
[8] ARM big.LITTLE technology. [Online]. Available: http://www.arm.com/products/processors/technologies/biglittleprocessing.php
[9] X. Liang, M. Nguyen, and H. Che, "Wimpy or brawny cores: a throughput perspective," *Journal of Parallel and Distributed Computing archive*, vol. 73, no. 10, pp. 1351-1361, Oct. 2013.
[10] N. Hogade, S. Pasricha, H.J.Siegel, A.A. Maciejewski, M.A. Oxley, and E. Jonardi, "Minimizing Energy Costs for Geographically Distributed Heterogeneous Data Centers," *T-SUSC*, vol. 3, no. 4, pp. 318-331, 2018.
[11] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," *ACM Symposium on Cloud Computing (SoCC)*, 2015.
[12] C. Bussema and E. Torng, "Greedy Multiprocessor Server Scheduling," *Operations Research Letters*, vol. 34, no 4, pp. 451-458, 2006.
[13] R. Kumari and V.K. Jha, "Performance Analysis of Load Balancing Algorithms in Amazon Cloud," *Int. Conference on Microelectronics, Computing and Communication Systems*, 2021.
[14] Least connection method, 2021. [Online]. Available: https://docs.citrix.com/en-us/citrix-adc/current-release/load-balancing/load-balancing-customizing-algorithms/leastconnection-method.html
[15] D. Meisner, J. Wu and T.F. Wenisch, "BigHouse: A simulation infrastructure for data center systems," *IEEE Int. Symposium on Performance Analysis of Systems & Software*, 2012.
[16] D. Meisner, BigHouse - Stochastic Queuing Simulation, 2021. [Online]. Available: https://github.com/meisner/BigHouse