Techniques for Energy-Efficient Power Budgeting in Data Centers

Xin Zhan
School of Engineering
Brown University
Providence, RI 02912
xin_zhan@brown.edu

Sherief Reda
School of Engineering
Brown University
Providence, RI 02912
sherief_reda@brown.edu

ABSTRACT

We propose techniques for power budgeting in data centers, where a large power budget is allocated among the servers and the cooling units such that the aggregate performance of the entire center is maximized. Maximizing the performance for a given power budget automatically maximizes the energy efficiency. We first propose a method to partition the total power budget among the cooling and computing units in a self-consistent way, where the cooling power is sufficient to extract the heat of the computing power. Given the computing power budget, we devise an optimal computing budgeting technique based on knapsack-solving algorithms to determine the power caps for the individual servers. The optimal computing budgeting technique leverages a proposed on-line throughput predictor based on performance counter measurements to estimate the change in throughput of heterogeneous workloads as a function of allocated server power caps. We set up a simulation environment for a data center, where we simulate the air flow and heat transfer within the center using computational fluid dynamic simulations to derive accurate cooling estimates. The power estimates for the servers are derived from measurements on a real server executing heterogeneous workload sets. Our budgeting method delivers good improvements over previous power budgeting techniques.

ACM Categories & Subject Descriptors C.5.5 [Computer System Implementation]: Servers.
General Terms: Management, Performance, Algorithms.
Keywords: Power, Budgeting, Management, Data Centers.

1. INTRODUCTION

Data center and computing clusters with hundreds or thousands of servers consume excessive amounts of power, with large facilities consuming up to 20 MW for a total cost of $12 million per year [14, 3]. As a result, the total cost of ownership of data centers is dominated by power consumption, which constrains total performance and scalability [7, 14, 11]. In many cases, the power consumption of a facility at any moment of time must be capped below a maximum limit that is specified by the electric grid operators and the electrical current carrying capacity of its power cables [7, 8].

One of the challenges in server power management is that different workloads trigger different power consumption patterns, and thus the power management settings that work for one set of workloads do not necessarily work for another set of workloads [9, 5]. As a result, one needs to find settings for each server that lead to a global optimal for the entire computing facility. Another challenge is that the total power consumption of a data center is the sum of the power consumption of its computing servers and the Computer Room Air Conditioning (CRAC) units, where the power consumption of the CRAC units depends on the power consumption of servers and the hot spots in the layout of the center [1]. The goal of this paper is to devise new power budgeting technique, where the total power budget is allocated among the servers and cooling equipment to maximize the total throughput, or equivalently minimize the average runtime. We summarize our contributions as follows.

1. We propose a novel method to partition the total power budget between the computing servers and cooling units in a *self-consistent* way, where the cooling power meets the heat removal requirements for the computing power, which is allocated using an optimal power budgeting technique.
2. We propose a novel *throughput predictor* for servers with heterogeneous workload sets, where the measurements from the performance counters are used to estimate the change in the throughput as a function of the server power cap.
3. Leveraging the throughput predictor, we propose an *optimal computing power budgeting* technique that is inspired by methods for solving the well-known knapsack problem. The budgeting technique identifies the optimal power caps for the servers, such that the total server power meets the computing budget and the total throughput is maximized.
4. We setup a realistic simulation environment for a data center with a large number of servers, where the power estimates for the servers are derived from real measurements on a server executing heterogeneous workload sets. We use Computational Fluid Dynamics (CFD) simulations to ensure accurate modeling of air flow and heat transfer within the center, and use the CFD results to estimate the cooling power. We experimentally demonstrate the advantages of our power budgeting method compared to previous approaches.

The organization of this paper is as follows. In Section 2, we describe previous related techniques in the literature. We formulate the power budgeting problem and describe our proposed framework in Section 3. Our experimental results are presented in Section 4, and Section 5 provides the conclusions of this work and directions for future work.

2. RELATED WORK

A number of models have been proposed in the literature to capture the relationship between the throughput and power of a single server. For example, Rajamani *et al.* proposed linear models [17] and Gandhi *et al.* proposed linear and cubic models [8]. The coefficients of these models are functions of the server configuration and the workload characteristics. In these previous works, fixed values for these coefficients were assumed irrespective of the workload characteristics. These values were obtained through prior characterization of standard benchmarks. As a result, these models are likely to show prediction errors for throughput and power in case heterogeneous applications with wide range of characteristics are executed on a cluster.

To enforce a required power cap, a number of previous approaches have proposed equipping each server with a feedback controller that computes the observed difference between the measured power and the power cap, and accordingly adjusts the p-state of the server using dynamic frequency and voltage scaling (DVFS) [10, 16]. If the difference is positive then DVFS is decreased, and if the difference is negative then DVFS is increased. To determine the power cap of each server, a number of power budgeting methods have been proposed [13, 8]. Ghandi *et al.* proposed power budgeting methods for servers that execute the same workload. This situation can be useful for data centers that execute transactional workloads of the same nature; however, they are not relevant for computing facilities that execute high-performance computing (HPC) applications. These later facilities typically have high utilizations where most of the servers are fully utilized executing a large range of workloads with heterogeneous characteristics. Nathuji *et al.* consider the case of power budgeting for heterogeneous workloads and servers [13]. The main proposed approach is a greedy method, where the throughput per Watt for the servers are calculated, and then servers with higher throughput per Watt are allocated more power during budgeting.

A related problem to power budgeting in data centers is the problem of power allocation in multi-core processors [9, 18]. Power budgeting for data centers is different in a number of ways: (1) unlike independent servers, multi-core processors do not offer power cap controllers for the individual cores; (2) workloads on a multi-core processor are likely to show memory interference issues, whereas workloads servers are relatively independent unless they explicitly communicate using message passing; (3) data centers feature air conditioning units that have to be considered during power budgeting; and (4) the interactions between computing and cooling power in data center are highly complex in nature.

3. PROPOSED APPROACH

We assume that a data center or a computing cluster is composed of n servers with identical hardware configuration and m CRAC units. We make the general assumption of heterogeneous workload sets, where different servers and different cores within the same server can be executing different workloads, and that the set points of the CRAC units can be controlled independently. We assume a closed-loop queueing model where all servers are fully utilized. As a result, maximizing the total throughput is equivalent to minimizing the response time [8].

Problem Formulation: Given n fully utilized servers with heterogeneous workloads, m CRAC units, and a total B power budget, the objective is to distribute the total power among the n servers and m CRACs, such that the total throughput is maximized or equivalently the average response time is minimized. That is, if $\tau_i(p_i)$ and p_i denote the throughput and allocated power for server i respectively and c_k denote the cooling power of CRAC k, then the goal is to maximize $\sum_{i=1}^{n} \tau_i(p_i)$ such that $B_s + B_{CRAC} \leq B$, where $B_s = \sum_{i=1}^{n} p_i$ is the total server computing power, and $B_{CRAC} = \sum_{k=1}^{m} c_k$ is the total cooling power. The budgeting has to be done in a *self-consistent* way, where cooling power is able to extract the heat generated from the servers.

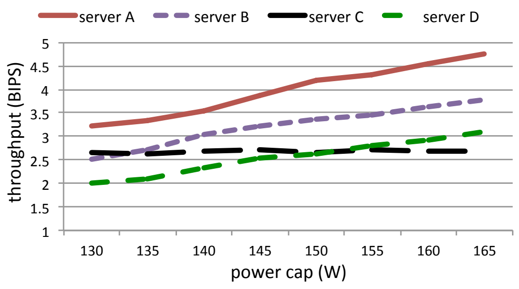

Figure 1: Impact of power cap on the throughput of a server.

Motivation. If the workloads on all servers are identical then the budgeting problem is trivial since the total power can be divided uniformly among all servers. To get a better understanding of the relationship between throughput and the allocated power cap in case of heterogeneous workloads, we equip our experimental server with a power capping controller. The capping controller executes once every second and adjusts the p-state of the server using DVFS based on the difference between the allocated power cap and actual power consumption [10]. We report the average throughput as a function of the power cap for four identical servers with different workload sets in Figure 1, where each server is executing a heterogeneous workload mix from the SPEC CPU06 benchmarks. The plot leads to a number of observations.

1. The observed throughput is highly dependent on the workload characteristics. Servers A and B show large improvement in throughput with increased power allocations. Server D shows modest improvements, while server C shows little or no improvements. Thus, some servers will not be able to leverage their allocated power caps to improve throughput.
2. The plot shows that the slope of an individual server plot changes as a function of the operating power cap. For instance, Server A shows a larger slope in the range of 140-150 W compared to other regions of operation. Thus, accurate modeling requires considering the impact of the operational power cap of the server on its slope.
3. The plots of servers C and D show that greedy allocation methods based on the throughput/Watt alone (e.g., [13]) will not give optimal results. For example, if the current power allocations for servers C and D are at the lowest cap, then Figure 1 shows that server C has higher throughout than server D, which can lead to the wrong conclusion that it is better to allocate more power to Server C. However, the plots of the two servers eventually cross over, where server D attains large throughput than server C at higher power caps.

Our power budgeting approach consists of two components: (i) a total power budgeting method that partitions the total power budget, B, into the computing budget, B_s, and the cooling budget, B_{CRAC}, in a self-consistent way; and (ii) an *optimal computing power budgeting* method that identifies the power cap for each server, such that the total computing power budget, B_s is met and the total throughput is maximized. Our optimal computing power budgeter makes use of a novel *throughput predictor* that takes as inputs the measurements, e.g., throughput, power and performance counters, of servers at the current power cap, and uses them to predict the change in throughput of each server for every possible power cap. We describe each of these components in the next subsections.

3.1 Total power budgeting

Our goal is to apply a total power budget for both computing power and cooling power in a self-consistent way, where the cooling power, $B_{CRAC} = \sum_{k=1}^{m} c_k$, extracts the heat generated from the computing power. The cooling power is a function of many factors, including the layout of the data center, the spatial allocation of the computing power, the air flow rate, and the efficiency of the CRAC units. The power consumption, c_k, of a CRAC unit k is equal to

$$c_k = \frac{\sum_i p_i}{CoP}, \tag{1}$$

where $\sum_i p_i$ is the power consumption of servers with their heat flow directed towards the CRAC unit, and CoP is the *coefficient of performance* that gives the performance of the CRAC units [12]. For example, based on physical measurements, an empirical model for CoP of a commercial water-chilled CRAC unit is equal to

$$CoP(t) = 0.0068t^2 + 0.0008t + 0.458, \tag{2}$$

where t is the supply air temperature of the CRAC unit in degrees Celsius [12]. To find the minimum sufficient cooling power for an allocation of a certain computing power, it is necessary to maximize the supply air temperature t, while ensuring that the inlet temperatures of all the servers will not exceed the manufacturer's redline temperature T_{red}. Identifying the inlet temperature for the servers requires accurate CFD models for the air flow and the heat transfer dynamics inside the data center. If the results of the CFD simulation show that the inlet temperature of any server violates T_{red}, then t should be lowered to bring the inlet temperature back under T_{red}, and if the inlet temperature has not reached T_{red}, then t should be increased without causing an inlet temperature of racks increase beyond T_{red}. Given a spatial layout of computing power consumption inside the data center, Equation (1) and Equation (2) and the associated CFD simulations enable us to compute the required cooling power $B_{CRAC} = \sum_k c_k$.

To ensure that the sum of the computing power, B_s, and the cooling power, B_{CRAC}, meets the total power budget B, we propose an algorithm, given in Figure 2, to identify a self-consistent partitioning of the total power budget. The main loop of the iterative algorithm first calculates the computing power budget B_s in step 3,

Procedure: Self-Consistent power budgeting algorithm
Input: Total power budget B; data center configuration; T_{red}.
Output: Computing power B_s and cooling power B_{CRAC}.

```
1. initialize $B_{CRAC}$ based on initial CFD simulation.
2. repeat:
3.     let $B_s = B - B_{CRAC}$.
4.     budget $B_s$ using the multi-choice knapsack algorithm.
5.     run CFD simulations to get minimum $B_{CRAC}$ given $T_{red}$.
6. until $B_{CRAC}$ is equal to $B - B_s$.
```

Figure 2: Algorithm for self-consistent total power budgeting.

and then in Step 4, the computing budget, B_s, is allocated optimally among the servers using the multi-choice knapsack algorithm described in the next subsection. Given power allocation and the data center configuration, Step 5 estimates the minimum required cooling power, B_{CRAC}, as described in the previous paragraph. If it happens that $B_{CRAC} + B_s = B$ (step 6), then the algorithm has converged to a solution; otherwise, it continues iterating. The proposed algorithm is guaranteed to converge; proof is available in the supplemental material.

3.2 Computing Power Budgeting

Our goal is to maximize the total throughput under a total computing power budget B_s. We consider a discrete set of individual server power caps with a fixed increment (e.g., 130 W, 135 W, . . ., 165 W). The choice of a discrete number of power caps is natural given that p-states are discrete and changing them does not lead to a continuous power range. Thus, the power cap of a server can be described as

$$p_i = p_0 + \sum_{j=1}^{r} w_j x_{ij}, \tag{3}$$

where p_0 is the least possible power cap, r is the number of individual server power caps, w_j is the increment power for each cap over the least possible cap, and $x_{ij} \in \{0,1\}$, where x_{ij} is only equal to 1 when server i is assigned a power cap equal to $p_0 + w_j$. For the case of power caps: 130 W, 135 W, . . ., 165 W, we have $r = 8$, $p_0 = 130W$ and $w_1 = 0$, $w_2 = 5$, $w_3 = 10, \ldots, w_8 = 35$.

A challenging aspect is that we need to estimate the impact of a change in power cap on the throughput of a server. We propose the following *throughout predictor*. Suppose that $\hat{p}_i$ denotes current allocated power cap to server i, and that the attained throughput for the server from using the power cap controller is equal to $\tau_i(\hat{p}_i)$. Given the measurements at the current power cap, the *objective* of the *throughput predictor* is to estimate the throughput of the server resulting from allocating a new power cap p_i to the server. We propose a piecewise linear model, where the predicted throughput is equal to,

$$\tau_i(p_i) = \tau_i(\hat{p}_i) + s_i(\hat{p}_i)(p_i - \hat{p}_i), \tag{4}$$

where $s_i(\hat{p}_i)$ is the *slope* of the throughput-power plot of server i at the server's current power cap. To predict the throughput, we need to identify the slope from the observations at the current operating point. To get an insight into the factors that determine the slope of the throughput-power characteristics, we analyzed a large number of performance counters from off-line characterization data. We have found that the Last Level Cache (LLC) misses is one of the most reliable predictor of the slope. Figure 3 illustrates the relationship between the slope and LLC using a large volume of characterization data collected from the SPEC CPU 2006 benchmarks. The results show a trend where workloads with larger LLC suffer throughput degradation. This trend is plausible as LLC misses show memory boundedness [2, 6], and as a result allocating more power caps to memory bound workloads give little improvements to throughput. In addition to the LLC misses, we have found that the ratio, i.e., $\tau_i(\hat{p}_i)/\hat{p}_i$, is a good predictor of the slope at the setting. Figure 4 illustrates the relationship between the two using our off-line characterization data. The results show that servers with higher throughput per Watt usually have higher slopes.

Our slope estimator makes uses of both $\tau_i(\hat{p}_i)/\hat{p}_i$ and $\hat{LLC}_i$. We experimented with a number of models for the slope, and we found the following model to give the best results:

$$s_i = \alpha_1 + \alpha_2 \frac{\tau_i(\hat{p}_i)}{\hat{p}_i} + \alpha_3 e^{\alpha_4 \cdot \hat{LLC}_i}, \tag{5}$$

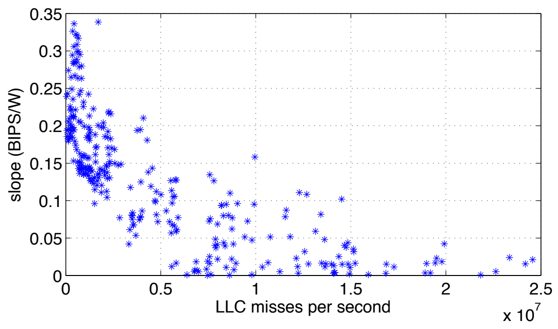

Figure 3: Relationship between LLC and the slope for a large number of heterogeneous workload sets.

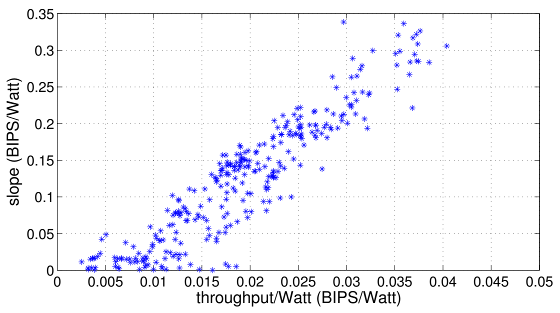

Figure 4: Relationship between current throughput/Watt and the slope for a large number of heterogeneous workload sets.

where $\alpha_1, \ldots, \alpha_4$ are the model coefficients for the current power cap. The coefficients can be easily found through off-line training on subset of workload characterization data.

Using Equations (3) and (4), and given the current $\tau_i(\hat{p}_i)$ and $\hat{p}_i$, it can be shown (see supplemental material) that the throughput objective can be recast as follows:

$$\max \sum_{i=1}^{n} \tau_i(p_i) \quad \Leftrightarrow \quad \max \sum_{i=1}^{n} \sum_{j=1}^{r} v_{ij} x_{ij}, \tag{6}$$

where $v_{ij} = s_i w_j$. Thus, the entire optimization formulation is given by

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{r} v_{ij} x_{ij} \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{r} w_j x_{ij} \leq B_s - np_0, \\ & \textstyle\sum_{j=1}^{r} x_{ij} = 1 \qquad \forall i = \{1 \ldots n\}, \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

We observe the similarity between power budgeting formulation and the *multiple-choice knapsack* problem [15]. In the multiple-choice knapsack problem, there are a number of classes, where each class has a few items, each with its own value and weight, and we have to select one item from each class to maximize the total value for the given total weight of the knapsack. Our problem naturally leads to a multiple-choice formulation, where the each server corresponds to a class, and the items within the class correspond to the power cap settings that can be applied to the server, each with its own throughput value ($v_{ij}x_{ij}$) and weight (power cap $w_j x_{ij}$). The multiple-choice knapsack problem is readily solved using dynamic programming. The supplemental material provides the details of the dynamic programming algorithm, which has a complexity of $O(nrB_s)$. In a computing cluster with hundreds or thousands of servers, it is easy to envision a server dedicated to carrying out the computations necessary for power budgeting.

4. EXPERIMENTAL RESULTS

In our data center configuration, we assume 320 servers forming 8 U40 racks with 40 servers per rack. To simulate the heat flow and air flow in the data center, we use TileFlow [20], which is a CFD software tool for simulating cooling characteristics of data centers. We provide the center's layout details in Experiment 3. The throughput and power estimates for the servers are derived from measurements on a real server executing heterogeneous workload sets. The Linux-based server has a quad-core Intel Core i7 processor and 8 GB of memory. To measure power consumption, the 120 V AC power lines to the server are intercepted and the electric current is measuring using an Agilent 34410A digital multimeter. The total power measurements are read back to the server over USB using the SCPI interface and provided as inputs to the power cap controller. The engagement period of the feedback power cap controller is 1 second.

We use the experimental server to construct a database of 320 execution traces of workload sets selected from the SPEC CPU06 [19] and PARSEC benchmarks [4]. For the SPEC CPU06 benchmarks, each workload set consists of four randomly chosen benchmarks, so that all the cores of our server are fully utilized. For the PARSEC benchmarks, all workloads are executed with four-thread configuration. We measured the number of retired instructions per second and LLC misses using the `pfmon` tool library interface. To train our predictor, we collected a large volume of characterization, where the throughput and LLC are measured for different workloads under different power caps. The database enables us to simulate the impact of different power budgets on a large number of servers in an extremely fast way that preserves the accuracy of results. In particular, each time a new power budget is applied, the power and performance outcomes are computed by reassembling the proper sections of the workload set traces of different servers from the database.

Exp 1. Throughput Predictor Accuracy. In the first experiment, we evaluate the accuracy of our throughput predictor compared to the actual throughput results. We compare our predictor in three versions: (i) `predictor` which uses the measurements of throughput, power and LLC as described in Subsection 3.2; (ii) `predictor-LLC` that just uses LLC measurements, and (iii) `predictor-TP` just uses the throughput and power. We also compare against the linear (`previous-linear`) model [17, 8] and cubic (`previous-cubic`) model proposed in previous works [8]. The average absolute error of the predictors are reported in Table 1. The results show that our predictor leads to better throughput prediction, and that combining LLC measurements together with throughput and power leads to more accurate results. Both linear and cubic models previously proposed in the literature trail our models in accuracy.

Exp 2. Computing Power Budgets. In the second experiment, we evaluate the effectiveness of the proposed knapsack-based optimal

prediction method	throughput prediction error
`predictor`	3.57%
`predictor-LLC`	7.83%
`predictor-TP`	4.89%
`previous-linear` [17, 8]	15.94%
`previous-cubic` [8]	8.16%

Table 1: Error in throughput prediction for various models.

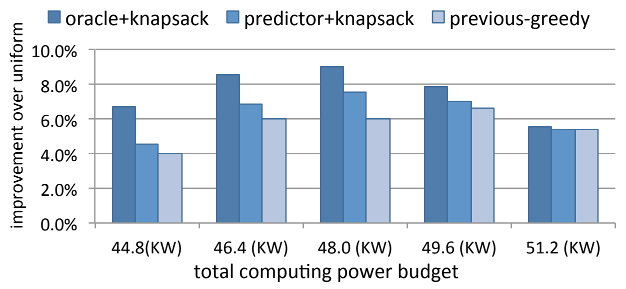

Figure 5: Throughput improvement over baseline uniform power allocation for heterogeneous workloads across servers, homogenous within server.

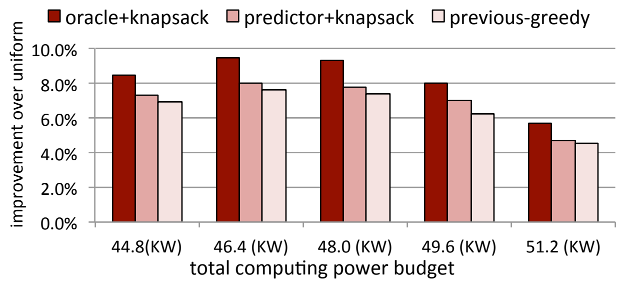

Figure 6: Throughput improvement over baseline uniform power allocation for heterogeneous workloads across servers, heterogeneous within server.

power budgeting method given a total computing power budget. We do not consider the cooling power consumption in this experiment. We refer to our technique by `predictor+knapsack`. We report the improvement in throughput over a baseline method (`uniform`), where the budget is allocated uniformly among the servers. We also compare against a previously proposed approach (`previous-greedy`) [17], which utilizes a greedy approach for power budgeting, where servers with higher throughput per Watt at the moment of re-calculating the power budget are allocated more power. Finally, we compute an upper bound on the attainable throughput by using the optimal knapsack algorithm on the true throughput and power for each server at the power cap, which are not known during runtime, but can be computed in our simulation environment. We refer to this method by `oracle+knapsack`. We consider two cases:

a) Heterogeneous across servers, homogenous within server: In the first case, the servers execute different workload sets, but the workload set assigned for each server is homogenous, e.g., a PARSEC workload with four threads or four instances of the same SPEC CPU06 benchmark. This is the most common case in modern clusters as administrators prefer to eliminate the interference between workloads arising from execution on the same server. The results are given in Figure 5 for a number of total computing power budgets. The results demonstrate that our `predictor+knapsack` method consistently outperforms other methods. For the case of 48 KW, we increase the throughput by 7.56% compared to uniform allocation case, whereas the `previous-greedy` method only increases the throughput by 5.98%. The results from our predictive method are close to the results from the oracle case.

b) Heterogeneous across servers, heterogeneous within server: In the second case, the servers execute different workload sets, and each workload set on a server consists of different benchmarks (e.g., four instances of different SPEC CPU06 applications). The results are given in Figure 6 for five total computing power budgets. The results show that our proposed method consistently outperforms other methods. For example, for total budget of 49.6 KW, our method increases throughput by 7.02% over `uniform`, whereas the previously proposed greedy method increases throughput by 6.12%. The relative improvements in this case are less than the first case, which is expected given the heterogeneity of workloads within the server. This heterogeneity causes averaging in characteristics, which leads to less differentiation among the ensemble of servers. Furthermore, the interactions between the workloads within the servers reduce the accuracy of the throughput predictor. Therefore, there might be further room for improvement through better throughput predictors.

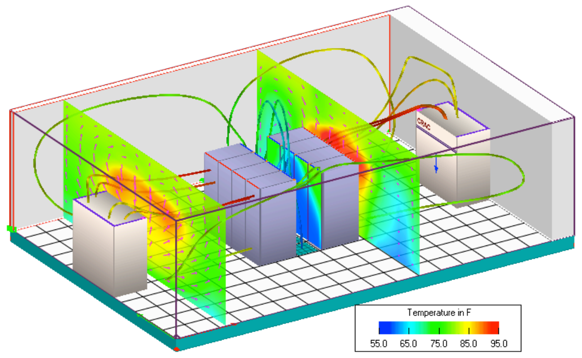

Figure 7: The inlet temperatures of racks and the spatial temperature maps in Fahrenheit.

In both cases, it is natural to expect that the relative advantages among the methods would disappear when the total power budget is too high or too low. If the total budget is too high, then all servers can afford to run at the highest power cap and throughout irrespective of the method, and similarly when the total budget is too low, then all servers will be forced to the lowest power state.

Exp 3. Total Power (Computing+Cooling) Budgeting. Our first three experiments assume that the power budget is entirely applied to the computing servers. In the fourth experiment, we include cooling power into our method and calculate the optimal partition between cooling power and computing power of a given total power budget. In our data center configuration, the 8 racks are arranged into two symmetric rows at the center of the room as illustrated in Figure 7. Two down flow CRAC units are located at two sides of the center. Cold air comes from under floor through perforated tiles between the two front side rack rows. The fans integrated with the racks draw the cold air through servers, which removes the heat generated by the operation of servers. The air heated by servers leaves the racks from the back side and is sucked into the CRAC units at the sides. The CRAC units extract the heat from the hot air and push cold air back into data center from perforated tiles on the floor at user specified temperature. We assume a redline inlet temperature of racks is 24 $^{\circ}C$.

We consider five total power budgets 62 KW, 66 KW, 70 KW, 74 KW and 78KW. We execute the self-consistent budgeting algorithm of Figure 2 to find the optimal partition of total power budget between computing power and cooling power under several total power budgets. After each simulation, TileFlow returns a report about the estimated maximum inlet temperature of racks. We can check the temperature at any specific point by the temperature mapping tool in Figure 7. The partitioning of the total power into its computing and cooling components is given in Figure 8. From Fig-

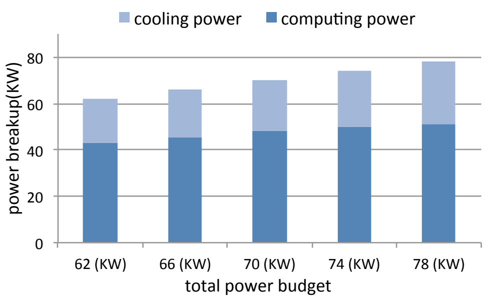

Figure 8: The breakup of cooling power and computing power under different total power budgets.

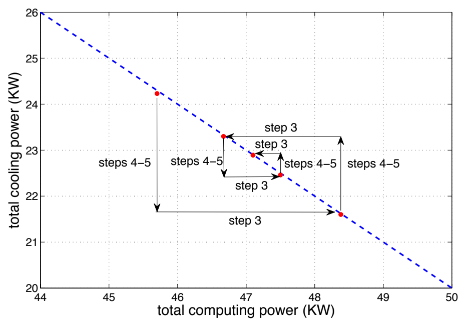

Figure 9: Illustration of the self-consistent power budgeting of the algorithm in Figure 2 for the case of 70 KW.

ure 8, we can observe that the cooling power consumption typically takes $30\% - 35\%$ of total power consumption. Another interesting observation from the results in Figure 8 is that the proportion of cooling power increases with the increase in total power budget, and that the rate of this increment also increases. Figure 9 illustrates the application of the self-consistent budgeting algorithm of Figure 2 to the case of 70 KW total power budget. The dashed blue line gives the power partitions that sum to 70 KW, and the red points give the intermediate partitions before convergence. The algorithm requires only 5 iterations to converge to a self-consistent solution.

5. CONCLUSIONS & FUTURE WORK

In this work we considered the problem of optimal power budgeting for servers with heterogeneous workloads. It is well-known that workloads exhibit different power and performance characteristics depending on their memory or processor boundedness. We leveraged this observation to devise a power budgeting method that allocates power to servers that can efficiently translate their power allocation to increases in throughput. During runtime, a power budgeting system has only one snapshot of the servers' status based on their current measurements. Thus, we devised a throughput prediction method that estimates the changes in throughput as functions of potential changes to allocated power caps. We have demonstrated that our throughput predictor is capable of providing accurate predictions under different power cap and workload characteristics. We have devised an optimal computing power budgeting method based on the multiple-choice knapsack formulation to identify the optimal power allocations for each server such that the total throughput is maximized. Furthermore, we proposed a self-consistent method to partition the total power budget between the computing and the cooling component of the data center. Our results show good improvements over previous methods.

Future work. Our future work will consider the possibility of under-utilized servers and use of other Quality of Service (QoS) metrics besides the total throughput and average response time [3].

6. REFERENCES

[1] F. Ahmad and T. Vijaykumar, "Joint Optimization of Idle and Cooling Power in Data Centers while Maintaining Response Time," in *Proceedings of Architectural Support for Programming Languages and Operating Systems*, 2010, pp. 243–256.

[2] H. Amur, K. Schwan, and M. Prvulovic, "Towards Optimal Power Management: Estimation of Performance Degradation due to DVFS on Modern Processors," Georgia Tech, Tech. Rep. GIT-CERCS-10-02, 2010.

[3] L. A. Barroso and U. Holzle, *The Datacenter as a Computer*. Morgan and Claypool Publishers, 2009.

[4] C. Bienia and K. Li, "PARSEC 2.0: A New Benchmark Suite for Chipmultiprocessors," in *In Proceedings of the Annual Workshop on Modeling, Benchmarking and Simulation*, 2009.

[5] R. Cochran, C. Hankendi, A. Coskun, and S. Reda, "Identifying the Optimal Energy-Efficient Operating Points of Parallel Workloads," in *ACM/IEEE International Conference on Computer-Aided Design*, 2011, pp. 608–615.

[6] S. Eyerman and L. Eeckhout, "A Counter Architecture for Online DVFS Profitability Estimation," *IEEE Transactions on Computers*, vol. 59, no. 11, pp. 1576–1583, 2010.

[7] X. Fan, W. Weber, and L. Barroso, "Power Provisioning for a Warehouse-sized Computer," *International Symposium on Computer Architecture*, pp. 13–23, 2007.

[8] A. Gandhi, M. Harchol-Balter, and R. Das, "Optimal Power Allocation in Server Farms," in *International Conference on Measurement and Modeling of Computer Systems*, 2009, pp. 157–168.

[9] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," in *International Symposium on Microarchitecture*, 2006, pp. 347 –358.

[10] C. Lefurgy, X. Wang, and M. Ware, "Power Capping: A Prelude to Power Shifting," *Cluster Computing*, vol. 11, pp. 183–105, 2008.

[11] J. Leverich and C. Kozyrakis, "On the Energy (In)efficiency of Hadoop Clusters," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 61–65, 2010.

[12] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers," in *Proceedings of USENIX Annual Technical Conference*, 2005, pp. 61–75.

[13] R. Nathuji, C. Isci, E. Gorbatov, and K. Schwan, "Providing Platform Heterogeneity-Awareness for Data Center Power Management," *Cluster Computing*, vol. 11, pp. 159–271, 2008.

[14] C. Patel and A. Shah, "Cost Model for Planning, Development and Operation of a Data Center," *Hewlett-Packard Laboratories Technical Report*, 2005.

[15] D. Pisinger, "Algorithms for Knapsack Problems," Ph.D. dissertation, University of Copenhagen, 1995.

[16] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "Power" Struggles: Coordinated Multi-Level Power Management for the Data Center," in *Architectural Support for Programming Languages and Operating Systems*, 2008, pp. 48–59.

[17] K. Rajamani, H. Hanson, J. Rubio, S. Ghiasi, and F. Rawson, "Application-Aware Power Management," in *International Workshop on Workload Characterization*, 2006, pp. 39–48.

[18] J. Sartori and R. Kumar, "Three Scalable Approaches to Improving Many-core Throughput for a Given Peak Power Budget," in *International Conference on High-Performance Computing*, 2009, pp. 89–98.

[19] C. D. Spradling, "SPEC 2006 Benchmark Tools," *SIGARCH Computer Architecture News,*, vol. 35, no. 1, pp. 13–134, 2007.

[20] TileFlow, "http://inres.com/products/tileflow."

7. SUPPLEMENTAL MATERIAL

7.1 Dynamic Programming Algorithm for Computing Power Budgeting

Using Equations (3) and (4), and using the fact that $\tau_i(\hat{p}_i)$ and $\hat{p}_i$ are given inputs, we can re-cast the throughput objective as follows:

$$\begin{aligned} \max \sum_{i=1}^{n} \tau_i(p_i) \quad &\Leftrightarrow \quad \max \sum_{i=1}^{n} (\tau_i(\hat{p}_i) + s_i(p_i - \hat{p}_i)) \\ &\Leftrightarrow \quad \max \sum_{i=1}^{n} s_i \sum_{j=1}^{r} w_j x_{ij} \\ &\Leftrightarrow \quad \max \sum_{i=1}^{n} \sum_{j=1}^{r} v_{ij} x_{ij}, \end{aligned} \tag{7}$$

where $v_{ij} = s_i w_j$. Thus, the entire optimization formulation is given by

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{r} v_{ij} x_{ij} \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{r} w_j x_{ij} \leq B_s - n p_0, \\ & \textstyle\sum_{j=1}^{r} x_{ij} = 1 \quad \forall i = \{1 \ldots n\}, \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

We observe the similarity between power budgeting formulation and the *multiple-choice knapsack* problem [15]. In the multiple-choice knapsack problem, there are a number of classes, where each class has a few items, each with its own value and weight, and we have to select one item from each class to maximize the total value with the knapsack total weight. Our problem naturally leads to a multiple-choice formulation, where the each server corresponds to a class, and the items within the class correspond to the power cap settings that can be applied to the server, each with its own throughput value ($v_{ij} x_{ij}$) and weight (power cap $w_j x_{ij}$). The standard dynamic programming given in Figure 10 can be used to solve the problem optimally. It has a complexity of $O(nrB_s)$.

Procedure: Optimal computing power budgeting algorithm.
Input: values and weights for the servers, n, and B_s.
Output: Power allocated for every server.

```
Let $V$ be a vector that holds the total knapsack's value for each
   possible budget. $V$ is initialized to all zero.
for $i := 1 : n$
  for $k := B_s : -1 : 1$
    for $j := 1 : r$
      $p_i := p_0 + w_j$
      if $k \geq p_i$ and $V(k) \leq v_{ij} + V(k - p_i)$
        let $V(k) = v_{ij} + V(k - p_i)$
        let $x_{ij} = 1$ and let $x_{il} = 0$ for all $l \neq j$
```

Figure 10: Algorithm for optimal power budgeting.

7.2 Proof of Convergence of the Self-Consistent Power Budgeting Algorithm

In this subsection, we will prove the convergence of the self-consistent power budgeting algorithm given in Figure 2 of Subsection 3.1. Let (B_s^*, B_{CRAC}^*) denote the self-consistent solution of a total power budget $B_s^* + B_{CRAC}^* = B$ at a maximum CRAC supply temperature of t^*. Let the computing power at iteration k of the algorithm is denoted by $B_s(k)$, and the minimum cooling power required for heat extraction is $B_{CRAC}(k)$ at a maximum CRAC supply temperature of t_k. Define $\delta_p(k) = |B_s(k) - B_s^*|$ and $\delta_c(k) = |B_{CRAC}(k) - B_{CRAC}^*|$.

When $B_s(k) > B_s^*$, the CRAC unit's supply temperature, t_k, is higher than t^*. According to the CRAC model of Equation (2), $CoP(t_k) > CoP(t^*)$, which will also hold true for any monotonically increasing CRAC model as a function of temperature. We can derive the following relationship between $\delta_p(k)$ and $\delta_c(k)$:

$$\begin{aligned} \delta_p(k) &= |B_s(k) - B_s^*| \\ \frac{\delta_p(k)}{CoP(t_k)} &= |\frac{B_s(k)}{CoP(t_k)} - \frac{B_s^*}{CoP(t_k)}| \\ &> |\frac{B_s(k)}{CoP(t_k)} - \frac{B_s^*}{CoP(t^*)}| \\ &> |B_{CRAC}(k) - B_{CRAC}^*| \\ &> \delta_c(k). \end{aligned}$$

With a CoP with a numerical value greater than 1, as expected from Equation (2), we conclude that

$$\delta_p(k) > \delta_c(k). \tag{8}$$

A similar argument can be made for the case of $B_s(k) < B_s^*$. For iteration $k + 1$, our method will update the computing power as:

$$\begin{aligned} B_s(k+1) &= B - B_{CRAC}(k) \\ &= B_s^* + B_{CRAC}^* - B_{CRAC}(k), \end{aligned}$$

which can be re-arranged to

$$\begin{aligned} B_s(k+1) - B_s^* &= B_{CRAC}^* - B_{CRAC}(k) \\ |B_s(k+1) - B_s^*| &= |B_{CRAC}^* - B_{CRAC}(k)| \\ \delta_p(k+1) &= \delta_c(k) \\ \delta_p(k+1) &< \delta_p(k). \end{aligned}$$

Thus, the distance, $\delta_p(k+1)$, between $B_s(k+1)$ and B_s^* is less than the distance, $\delta_p(k)$, between $B_s(k)$ and B_s^*. Therefore, the computing power approaches B_s^* with every iteration and finally converges to B_s^*.