Energy-Delay Based Provisioning for Large Datacenters: An Energy-Efficient and Cost Optimal Approach

Sriram Sankar
Microsoft Corporation
1 Microsoft Way
Redmond, WA
srsankar@microsoft.com

Kushagra Vaid
Microsoft Corporation
1 Microsoft Way
Redmond, WA
kvaid@microsoft.com

Harry Rogers
Microsoft Corporation
1 Microsoft Way
Redmond, WA
harrog@microsoft.com

ABSTRACT

It is challenging to determine the optimum number of servers required to provision for large online applications because of the conflicting mandates of (a) achieving peak performance needs and (b) minimizing unused datacenter power and capacity. Since Online Services application loads are unpredictable, datacenter operators often conservatively provision for maximum power utilization by characterizing workloads for peak load performance. In contrast, we aim to optimize the service capacity per total cost of ownership (TCO) of an Online Service datacenter deployment by characterizing the energy-delay properties of large scale datacenter workloads. We show that the peak load performance is not the energy efficient point of operation for most applications. We choose two industry-strength workloads (Internet Search and D-Process) and analyze their energy-delay behavior under varying loads. We then calculate the optimal operating point for the specific large-scale application and provision datacenter energy and capacity based on the energy-delay curves. In contrast to workload-based peak power provisioning, we show a 7% benefit in Service Capacity-per-TCO-dollar for the energy-delay characterization methodology in our cost analysis for Online Services Applications.

Categories and Subject Descriptors

C.5.5 [**Computer Systems Organization**]: Computer System Implementation – *Servers.*

General Terms

Measurement, Performance, Design.

Keywords

Datacenter Provisioning, Online Services, Energy-Delay Characterization, Cost Optimal Design.

1. INTRODUCTION

Online services have become an important class of large-scale applications that are being continuously driven by unpredictable user demand. Hence datacenter designers face several unique challenges for energy and capacity provisioning. Given the significant cost incurred in building a large-scale datacenter, it is essential to employ methodologies that deliver performance in an energy-efficient and cost-optimal manner. A conservative approach to server power provisioning is to measure peak power consumption while running the workload, and use that value to extrapolate the total number of servers that can be deployed within a given power envelope for an online service. However, this method *strands* (*i.e.*, leaves unused) datacenter power capacity, since utilization is highly variable during non-peak usage for Internet-based services.

Typical industry estimates for capital expenditure for the construction of large-scale datacenters range from \$10 to \$20 per watt of IT load [2]. Every megawatt not utilized can result in an over spend of \$10M-\$20M. As a simple example, if we allocated per-server power such that we over-provisioned 10% of the datacenter critical capacity, we would strand 1.5MW of available power capacity for a 15MW datacenter - which could have been used for powering an additional 5000 servers (at 300W/server) [3] for the Online Service, thus providing the service with scalability in the infrastructure facility that has already been constructed. Thus, making better use of the critical power capacity in an existing datacenter delays the need to build a new datacenter facility, thereby reducing capital expenditure of a large enterprise.

In this paper, we use two benchmarks – a) a production quality internet search engine (Search) and b) a distributed processing workload (D-Process) to illustrate our approach for two different classes of large-scale applications. We profile performance and power characteristics for these applications. We then determine a server operating point for which the amount of energy consumed to deliver a given level of performance is optimal based on energy-delay curves. We use this operating point for provisioning. Fixing a power budget lower than the maximum usable power capacity includes the risk of overloading datacenter circuit breakers and might result in service unavailability. We mitigate this risk by using server power capping technologies that are available in enterprise servers to limit power consumed to the optimal operating point. We believe that this methodology would provide a better energy and cost efficiency for large datacenters. To that order, we make the following major contributions in our paper:

- We profile two industry strength workloads in production and also characterize their power-performance profiles with representative benchmarks.

- We provide observations about Online Services from our datacenter workload characterization study and show that a conservative provisioning methodology that uses peak power characterization is not energy and cost efficient.
- We provide an optimal provisioning policy based on energy-delay characterization for operational efficiency in large datacenters.
- We present a TCO based cost analysis that shows a 7% benefit in service-capacity per TCO dollar.

The rest of the paper is organized as follows: Section 2 describes Background and Related Work, while Section 3 describes the experiment infrastructure. Section 4 presents our methodology and results. Section 5 discusses an energy-efficient and cost optimal deployment of this methodology. Section 6 presents possible future work. Section 7 concludes the paper.

2. BACKGROUND

2.1 Search Application

Figure 1 outlines the structure of the Internet search engine currently in production. Upon arrival, queries are distributed to many nodes. Examining its subset of the document index, each node returns the top N most relevant pages and their dynamic page ranks. The search engine uses sorted indices to access content served to the user. Queries enter the system through a top-level Aggregator. If the Aggregator cannot satisfy the query from its set of frequently accessed pages (i.e., the Cache), it distributes the query to the Index Serving Nodes (ISNs). The ISNs serve the query in a highly distributed manner. Each ISN is responsible for ranking pages, as well as generating descriptions of relevant pages for the user [4].

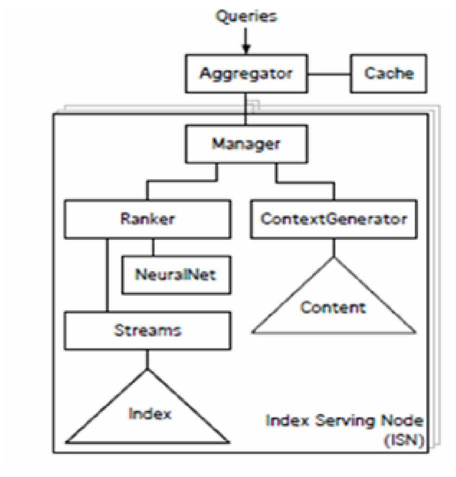

Figure 1: Search Engine Architecture

2.2 Search Performance Requirements

Search application performance is quantified by a combination of Quality-of-Service (QoS), throughput, and latency. The application defines QoS as the minimum percentage of queries handled successfully. For example, a QoS metric of θ percent requires a minimum of θ successful queries for every 100. The other 100-θ queries may time-out due to long latencies for expensive query features or may be dropped due to fully occupied queues. Given a QoS target constraint, we might consider a platform's sustainable throughput, which quantifies the maximum number of queries per second (QPS) that can arrive at a node without causing the node to violate the QoS constraint. If the QPS exceeds the sustainable throughput, QoS degrades.

Query processing must also observe latency constraints. The average response time of queries must fall within a certain number of milliseconds, with additional constraints for the 90th percentile of queries. Latency directly impacts relevance (i.e., documents corresponding to a specific query) by affecting the number of iterative refinements made to a search result. The deployment is also governed by the number of document indices that it needs to service. We consider QPS and the document index to be the factors that determine the Service Capacity of the datacenter.

2.3 D-Process Application

D-Process is a highly parallelized data storage and analysis engine similar to Dryad [18]. It is representative of large-scale data analysis applications including MapReduce [19] and Hadoop, parallel databases [20] and other distributed and streaming programming paradigms used to process massive amounts of data. D-Process essentially implements a distributed data storage and computation platform on a single tier of servers. The D-Process cluster used in production consists of several tens of thousands of servers and is under relatively high load, serving a job stream that almost never runs dry. Typical servers are dual CPU socket machines with 16 to 24 Gbytes of memory and up to four SATA disks. D-Process benchmark performance is measured by the number of standardized jobs that can be completed in an hour.

2.4 Related Work

2.4.1 Power Provisioning

CPU utilization was correlated to system power by Fan *et al*. [2], and it was highlighted that power provisioning in datacenters should not be designed for nameplate power ratings, which is almost always higher than peak power consumed under load. However the authors exclude Web search from power capping and do not provide a method for selecting an optimal operating point lower than observed peak levels. Govindan *et al*. [13] present a statistical model to predict the peak usage of workloads for improved power provisioning, but don't provide optimal methodology for non-peak usage. Server storage was considered to consume a significant portion of power in datacenters and workload characterization was used to estimate storage power in datacenters by Sankar *et al*. [3]. In contrast, our paper is the first to propose the use of energy-delay characterization to determine optimal operating points as effective power provisioning points in a real datacenter scenario for internet driven workloads. Ensemble-level power budgeting policies [14] allow the management layer to optimize for greater power-efficiencies than server level policies. Our methodology can be extended to compute optimal operating points for ensembles of heterogeneous servers and is an interesting area of future exploration. Different workloads have their corresponding optimal operating points. The datacenter operator can plan for each according to the priority of the workload and set an optimal operating point accordingly.

Therefore we believe that our methodology is also orthogonal to peak power balancing techniques like oversubscribing [2], safe overprovisioning [16] and power shifting [15].

2.4.2 *Power Capping*

Server vendors and datacenter solutions providers have started to offer power capping solutions. Intel's Power Node Manager and Data Center Manager [10] were used by Baidu for a proof of concept study for optimizing leased datacenter capacity. However, the power cap value was fixed at performance saturation point based on heuristics. HP's Dynamic Power Capping [11] features dynamically adjustable cap levels, and has provisions for static power cap at different power values. Server power capping control can also be implemented through different approaches including Online self-tuning methodology [7] and a feedback control algorithm that deals with nonlinearity of servers under different performance and power knobs [8]. Power capping can also be used as a technique for a coordinated approach to datacenter power management [9]. In our paper, we use Power Capping as a tool for provisioning in datacenters to prevent service unavailability due to overloading.

2.4.3 *Energy Delay Optimization*

The concept of energy-delay optimization was first used in circuit design [21, 22]. Zhang et al. [23] demonstrated the applicability of sensitivity-based optimization for determining the optimal settings of the design-time parameters of disk drives. A dynamic energy-delay optimization technique was developed for energy reduction in disk drives [24]. Smartphone design [25] also uses energy-delay optimization to extend battery life. In comparison to such previous work, our paper is the first to use energy-delay optimization for server capacity and power provisioning in datacenters.

3. EXPERIMENT INFRASTRUCTURE

3.1 System Setup

The test infrastructure used for the evaluation in this paper was a dual-socket Intel Xeon® L5520 server platform with power capping features, 16GB memory and 4 disk drives. Power measurements were done using a Yokogawa power meter [12]. The operating system used was Windows Server 2008. While the server used here for testing is not the same as what is deployed in production environments, we believe that the methodology and conclusions obtained with this test infrastructure are applicable to actual production scenarios.

3.2 Workload Setup

Internet Search: We use an internally developed Internet search engine benchmark with a mix of input queries traced from production runs. Of the search engine components in Figure 1, we consider a single ISN computing dynamic page ranks. The ISN takes queries arriving from the aggregator and returns sorted page ranks.

The ISN computes page ranks for several thousands of queries of varying complexity after an initial warm-up phase that brings the ISN to a steady state. We sweep the query arrival rate to identify the maximum query service rate that can be sustained by the system. For each query, the ISN computes overall ranks for pages that match the query for a roughly 10 GB index, a subset of the global index that is distributed across several thousands of nodes. The index size is chosen so that it is memory resident to eliminate page faults and minimize disk activity.

D-Process: The D-Process benchmark consists of multiple processes each of which load data from the disk into the memory initially and then run a wide range of processing jobs on the dataset. D-Process stresses the server CPU completely and the total time taken to complete the jobs is taken as a measure of delay metric for the benchmark.

3.3 Metrics

Power (measured in Watts) – In our experiments for Search, we normalize energy consumed to per second interval and hence we use Power reduction as a metric that determines energy efficiency.

QPS (Queries per second) – The performance metric for Search is the number of queries executed by the server in a second. This metric determines the success rates and the SLA of the service.

Service Capacity/TCO – Service capacity of the deployment denotes the measure of document index capacity for a given QPS requirement, represented as a product of the two (capacity*performance). Since large capacity or high performance can be obtained with a larger infrastructure, we normalize the Service Capacity of a deployment by the corresponding TCO. This metric provides a method to distinguish deployments; for instance, a deployment that can serve twice the number of documents against one that can serve only half that number with similar performance (QPS) values. TCO is calculated based on several parameters, including capital expenditure, power, amortization costs etc. The TCO model is derived from the calculations presented by James Hamilton [1].

4. METHODOLOGY AND RESULTS

4.1 Energy-Delay Characterization Methodology

Our methodology for provisioning servers in a datacenter is based on characterizing the energy-delay characteristics of the application running on the server. Multiple references in circuit design [21, 22], disk drive design [23, 24] and smartphone design [25] use energy-delay optimization techniques to establish optimal performance-per-watt. For a given energy constraint and SLA, we optimize 'energy*delay' to find an optimal operating point. We fix server power and measure sustained queries delivered (performance), satisfying the SLA. We calculate the best performance per watt (closest point to the origin minimizing ED), thus providing an optimal methodology for servers that are not energy-proportional.

Selecting a power provisioning value below a server's maximum power consumption has the undesirable side effect of impacting application performance at heavy loads, while selecting a provisioning value equal to the maximum power results in excessive power stranding at non-peak operations. Hence we need to find a point on the energy-delay spectrum at which we get maximum performance-per-watt without impacting performance SLAs. We also require this operating point to be lower than maximum power so we can optimize for non-peak operation as well. From energy-delay optimization theory, we represent the 'opportunity for energy reduction' as "Φ". Φ can be represented as the ratio of the percentage change in energy to a percentage change in delay at any particular instant. For example, Φ = 2 means that a 2% change in energy consumption will produce a 1% change in performance at the given instant. This value varies depending on where we are on the energy-delay curve. Φ can be mathematically represented as

$$\Phi = - (\Delta E/E) / (\Delta D/D)$$

In order to minimize Φ, we need to minimize the numerator energy term, which is the desired result, or increase the denominator. Due to the inverse nature of relationship, higher energy results in shorter delays and vice versa. However the rate at which the relationship changes is not linear. There is a point on the curve at which the proportion of one dimension that is required to contribute to a unit change in the other changes disproportionately. In Figure 2 for instance, we want the delay to be as close to the origin as possible, but we do not want to spend disproportionate amount of energy to achieve this result (on the top left of the curve). Since we have discrete points in our measurements, we approximate the discrete point on the curve that is closest to the origin as the point at which this change occurs. For a curve where an equation can be fitted and a more continuous operation is possible, we have to minimize Φ for the curve equation to determine the optimal operating point. We term this point as optimal since for every point to the left we expend more delta energy for every delta decrease in delay. For every point to the right, there is a higher increase in delay for delta decrease in energy.

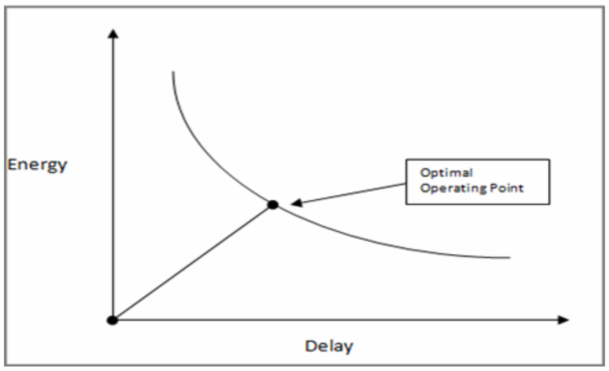

Figure 2: Energy-Delay Optimization Curve

We use power and performance metrics mentioned in Section 3.3 to represent energy and delay respectively. In the context of the Energy Delay curve for Internet Search, we relate the *delay* metric to search throughput. We use sustained query service rate (***QSR***) as the measure of performance throughput (this is the query service rate that is observed for a considerable percentile of the requests at the system). The inverse of the sustained query service rate, denotes the time taken to service a query and is used as a proxy for *delay*. In the case for D-Process, the time taken to complete the D-Process jobs at sustained utilization is taken to be the measure for delay metric.

4.2 Datacenter Utilization Characterization

In this section we provide data from actual measurements of systems in live datacenters. We show that both these benchmarks are CPU intensive. Hence a control algorithm that modulates CPU utilization would essentially throttle the entire application. This is an important result that enables us to use power capping as discussed later in the paper to obtain performance-power characteristics for these applications.

We obtained performance data from multiple production servers actively hosting Search and D-Process services subject to real user-driven loads. We collected performance counters for a period of seven days to account for any weekly patterns in usage. We observed that for both Search and D-Process, the CPU was the primary resource for different type of queries and loads. We present the summary results of this analysis in Table 1.

From Table 1, we observe that both Search and D-Process workloads are really CPU intensive at peak loads. However for many online services, since processor utilization above 80 percent is often undesirable because it impacts quality of service (request latency), the overall service deployment is often provisioned to keep average CPU utilization at moderate levels, typically in the 40 to 60 percent range. Both Search and D-Process applications load their dataset into memory and work off the pre-loaded set and hence they have high memory capacity utilization and corresponding CPU utilization. However as the table clearly shows, none of the two large scale workloads considered here are either disk or network intensive. By controlling processor utilization through processor frequency and voltage scaling, we scale the workload across multiple loads to obtain an energy-delay curve for that specific application.

Observation 1: Both Search and D-Process are CPU-intensive and hence CPU power control policies would throttle the entire application.

4.3 Applicability at Scale

We sample the CPU utilization for a rack of servers hosting the Search application for a period of 24 hours (one sample every 2 minutes) to capture any daily patterns of usage. We present one such chart in Figure 3. As can be observed from Figure 3, the maximum CPU utilization for any given server in the rack and the average CPU utilization of all the servers in the rack follow each other very closely. This essentially signifies that the load balancing for an application like Search would lead to homogeneous spikes in the workload. Hence, a methodology that can be applied at the single server level would scale to entire

Table 1: Workload Utilization Measurements from datacenter servers (Processor Avg Utilization is in 40%-60% bucket)

Applications	Processor (Max)	Avg. Memory Capacity	Avg. Memory Bandwidth	Avg. Disk Capacity	Avg. Disk Bandwidth	Avg. Network Bandwidth
Search	97%	88%	1.8%	30%	1.1%	10%
D-Process	88%	39%	1.1%	52%	0.7%	9%

deployment. However, there are significant opportunities like workload sharing for datacenter level power amortization that is beyond the scope of this analysis.

Since the applications considered here are large scale workloads optimized for specific functionality they have approximately similar usage behavior differentiated only by the user load (Figure 3). In addition, they also take up significant space in the datacenter and hence contribute to a major portion of datacenter capacity and planning. A rack level test or a cluster level test of this methodology would yield similar results because we measure sustained performance and also maintain the same system and workload configuration. All servers of this particular configuration running the Search workload would have the same optimal operating point. Hence, for the energy-delay methodology in this paper, we run the benchmarks at a single server level. Note that cluster level tests become interesting in the presence of heterogeneous systems since each system might have a different behavior, and that is part of our ongoing research effort.

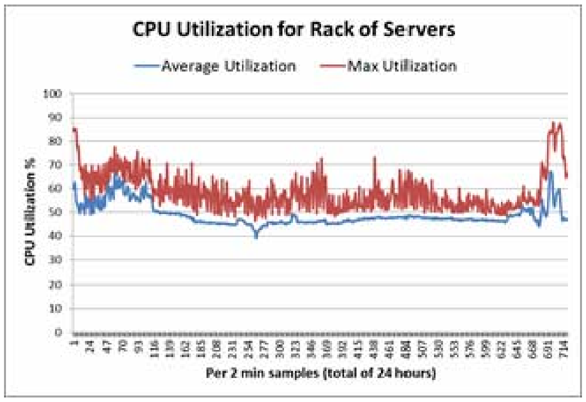

Figure 3: CPU utilization for rack of Search servers

Observation 2: In a load balanced homogeneous online service, Energy-Delay characterization measured at single server level can be applied to the entire workload.

Given Observations 1 and 2, we use the following methodology to measure the energy-delay characteristics and the optimal operating point of the application:

We power cap the server at a set power level through the appropriate CPU throttling functionality for a single server. We then run both the Search and D-Process workloads at their sustained maximum performance levels such that we obtain the performance expended at that amount of power. We plot the energy-delay curves from the above experiment after normalizing the axis for different scales in delay and energy. We calculate the closest point to the origin by using *Normalized Euclidean Distance* between the origin and the points on the curve [26]. This point corresponds to the optimal operating point of the server under test, for each application. We then use this optimal operating point as a reference for datacenter capacity and power provisioning.

4.4 Characterization Results

In this section, we provide the performance-power characterization that we conducted for both Search and D-Process.

4.4.1 Internet Search

For the Search benchmark, we increase the input number of queries at different power caps and measure the queries that are serviced by the system. We also measure the power consumed by the server under test under the different input loads.

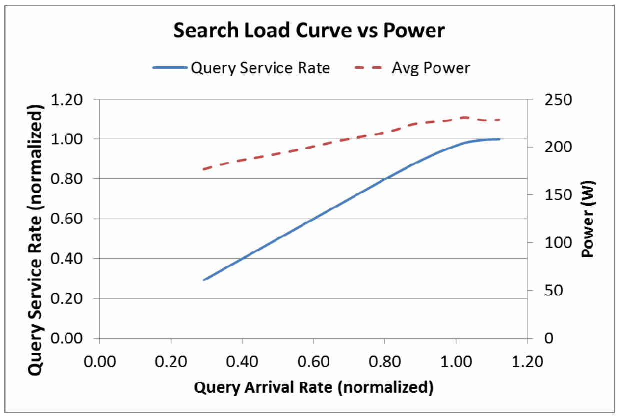

Figure 4: Sustained QPS with input load increase

From Figure 4, we see that as the input load increases we eventually saturate the system, and see no further improvement in the number of queries successfully processed. We conducted an offline analysis to identify the per-component utilization of servers in production and observed that the CPU utilization is proportional to query service rate. As shown in Table 1, this system is not bottlenecked on either memory or disk IO and CPU is the primary bottleneck.

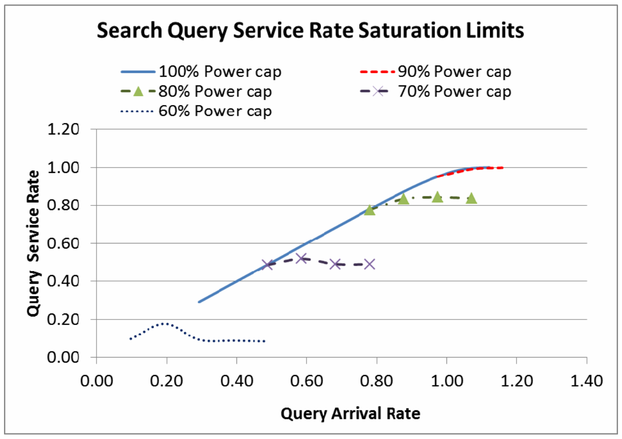

Figure 5: QPS for Search under various power caps

To understand the impact of power on performance, we power capped the server under test and ran the same benchmark at sustained performance at the different power limits. We choose power caps of 60%, 70%, 80%, 90% and 100%, which essentially denote the amount of power that is supplied to the server under test as a percentage of maximum power. Figure 5 shows the results of this experiment. From Figure 5, we see multiple performance curves as measured by the output Query Service Rate on the y-axis. The x-axis contains the input to the server under test, denoted by Query Arrival Rate. For each curve in the graph, we see a particular saturation point, beyond which the system is not able to service additional QPS. We tag this value as the best QPS value that the system can service at the capped power level and determine this Query Service Rate to be the sustained performance at that power level. Through this method, we get (*performance, power*) datapoints that we use to calculate the Normalized Euclidean distance from the origin. The calculations are shown in Table 2.

Table 2: Determining Optimal Operating Point for Search (Normalized Euclidean Distance closest to the origin determines the optimal operating point)

	1/QSR Delay (X)	(X-Mean)/ stddev	Power (Y)	(Y-Mean) /stddev	Normalized Euclidean Distance
60% Power cap	0.02	1.78	173	-1.21	2.154
70% Power cap	0.004	-0.27	193	-0.67	0.720
80% Power cap	0.0025	-0.46	216	-0.04	0.462
90% Power cap	0.002	-0.52	240	0.62	0.811
100% Power cap	0.002	-0.52	265	1.30	1.404
Std Dev	0.0078		36.56		
Average	0.0061		217.4		

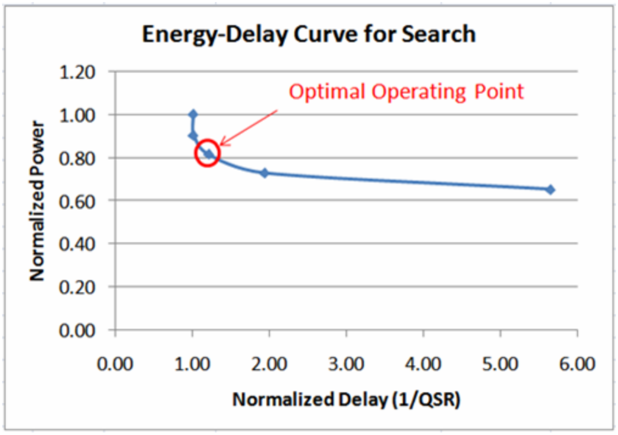

Figure 6: Energy-Delay curve for Search

We plot the Energy-Delay curve from this table in Figure 6. As can be seen from Figure 6, we see that the 80% power cap point is the optimal Energy-Delay operating point for Search application. We use this datapoint for datacenter capacity and energy provisioning.

4.4.2 D-Process

We conducted a similar experiment for D-Process. We fixed the number of D-Process jobs that were scheduled at this server under test, and measured the delay for completion at different power limits. We then normalized the datapoints according to the corresponding scales and obtained the following table. We see from Table 3 that D-Process also has an optimal operating point at 80% of maximum power. The D-Process and Search benchmarks are similar because they are strongly tied to the CPU utilization. Since both these applications use similar systems; we see almost similar energy-delay curves.

Table 3: Determining Optimal Operating Point for D-Process (Normalized Euclidean Distance closest to the origin determines the optimal operating point)

	Delay (X)	(X-Mean)/ stddev	Power (Y)	(Y-Mean) /stddev	Normalized Euclidean Distance
60% Power cap	608	-1.17	173	-1.21	1.687
70% Power cap	648	-0.82	193	-0.67	1.057
80% Power cap	743	0.01	216	-0.04	0.040
90% Power cap	851	0.96	240	0.62	1.140
100% Power cap	858	1.02	265	1.30	1.654
Std Dev	114.16		36.56		
Average	741.6		217.4		

One important difference that could be observed between Search and D-Process is the Normalized Euclidean distance in Table 2 and Table 3. Search starts at a higher distance from optimal operating point at lower power limits. This follows from the fact that Search has lower tolerance on one end of the spectrum since it is bound by SLAs for each of its query types. D-Process is fairly tolerant and has no specific SLAs. The normalized Euclidean distances from the origin for D-Process are hence distributed evenly from the origin.

4.5 Sensitivity Analysis

We measure sensitivity of our methodology to the observed power capping error rates at a) different power capping levels and b) different success rate of the queries (SLA) for Search. Note that the normalized measured power in Figure 6 is not exactly equal to the set capped levels. In Table 4, we observe that, as the power capping level is decreased, the power capping error % increases. This is consistent with power monitoring calibration observations made in [5]. For the power capping mechanism that was available for this work, we observed that power capping control was not able to push power caps down to the fixed levels

Table 4: Power Capping Error Measurements

Setting	Measured Power (W)	Power Cap (W)	Error %
100% cap	265	265	0.0%
90% cap	240	238.5	0.6%
80% cap	216	212	1.9%
70% cap	193	185.5	3.9%
60% cap	173	159	8.1%

as the cap value was lowered.

We also varied our SLA (θ successful queries for every 100) and evaluated the impact on our methodology for three different data points of 99.8%, 98% and 90% success rates. We present the analysis in Figure 7. The delay decreases (as seen from the tail of the curve) with increase in tolerance to query success rate. We also see that since more queries are able to be serviced at reduced SLA, the energy required to complete the same amount of work at higher SLA is not required. However, the shift in this trend is not significant to warrant a change in the static provisioning since all the optimal points are clustered close together as seen in Figure 7 except for the tail of the curve (which is not optimal and hence does not affect the methodology). Hence the optimal operating point methodology is fairly robust for Search. Even with an energy-proportional system [17], in Figure 8, we see that the 80% power cap point is the point of optimal performance per energy use, thus confirming our hypothesis that optimal operating point is indeed energy-efficient.

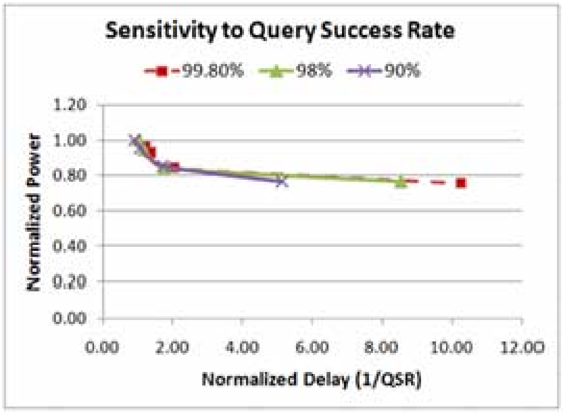

Figure 7: Sensitivity of Energy-Delay curves to Query Success Rate for Search benchmark

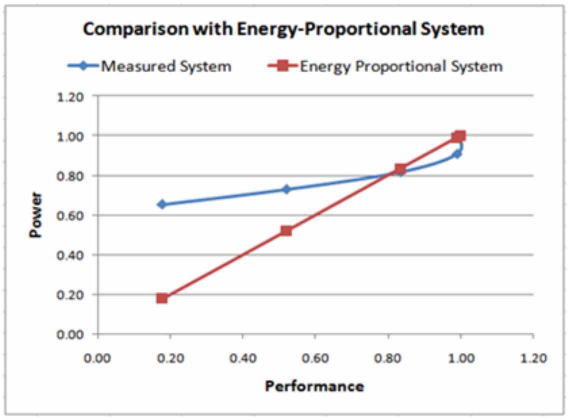

Figure 8: Comparison of Search Power and Performance with an Energy Proportional System

5. ENERGY EFFICIENCY AND COST ANALYSIS

We use a publicly available cost model [1] to estimate datacenter operational costs and illustrate the advantages of power capping. We quantify TCO benefit in terms of aggregate service capacity-per-TCO dollar. The model assumes $200M facility costs for 15MW of critical power with data center power usage efficiency (PUE) of 1.3. (Note that we show that our proposal is beneficial to more efficient datacenters as well). We amortize power distribution and cooling infrastructure costs over fifteen years and the purchase cost of servers over three years. The total cost of operating and managing the datacenter is presented on a monthly basis. The model categorizes the TCO into the following: power (electricity bill for running all servers), servers (purchase cost), cooling and power distribution (data center infrastructure power cost) and others (miscellaneous costs).

Power capping the server at the 80% gives us back 49W per server (max server power of 265W). For a 15MW datacenter, this allows for hosting 11,594 additional servers. For sustained peak input query loads at the data center level, the 80% power cap restricts server performance to 83% of an uncapped server. However the aggregate performance capacity of the deployment is maintained consistently

5.1 Energy Efficiency Analysis

In this section we present the peak load power consumption between two solutions: *Solution A (100% cap)* has the 100% power budgeted servers operating at maximum performance and *Solution B (80% cap)* has 11,594 more servers but operating at 80% power cap. In Table 5, note that the TCO is *higher* because of the number of servers in the datacenter. The calculations assume the maximum power consumed to be 265W at 100% cap and 216W at 80% power cap. The table shows that 56.6K servers at 100% power at peak load consume 240KW more than 68.2K servers at 80% power but both these deployments deliver the same QPS performance. The table shows that *Solution-B* yields better performance per Watt at peak loads. This is a direct result of using Energy-Delay curves for provisioning.

Table 5: Provisioning for Energy-Efficiency (80% cap is energy efficient)

	100% cap	80% cap
Number of Servers	56604	68198
Datacenter TCO	5970969	6665948
Power Consumed	15 MW	14.73 MW
Overall QPS capacity	5660400	5660400
Normalized Perf/Watt	1.00	1.02

5.2 Cost Optimality

In order to best utilize the existing infrastructure, we need to provision the servers to serve the maximum number of documents at the specified performance level. When we have 11,594 extra servers, we can serve ~20% more documents out of the same infrastructure when compared with *Solution A*. This model assumes that there is a continuous growth in number of documents to be served, which is typical of Online Search application.

Table 6: Provisioning for Optimal Cost

	100% cap	80% cap
Number of Servers	56604	68198
Datacenter TCO	5970969	6665948
Overall QPS capacity	5660400	5660400
Document Served	1.0	1.2
Service Capacity	5660400.0	6792480.0
Service Capacity/TCO	0.95	1.02
Power capping advantage	1.07	

Even though the datacenter TCO is higher for the 80% cap solution, the Service Capacity (QPS * Documents Served) is 20% more due the 20% increase in servers that could be fit within the same power and performance requirements. That leads to 7%

increase in Service Capacity/TCO as seen in Table 6. This solution is an important result from the perspective of datacenter provisioning, since we show that a larger number of servers and higher datacenter TCO does not necessarily mean that an online service is less optimal. Indeed among the two solutions, we show that the 80% cap is both energy and cost optimal in this case.

6. FUTURE WORK

In this paper we discussed the provisioning challenge for a large scale application hosted on physical servers. Future work is being done to understand provisioning implications for virtualization and heterogeneous application environments, where multiple applications may be hosted on the same server. In addition, for this paper we assumed a single homogenous server deployment at datacenter scale, which simplified analysis of the effects of power provisioning. When we consider multiple server types that are present for hosting a variety of applications, we need to incorporate those server characteristics to our methodology.

7. CONCLUSION

In this paper, we introduce the Service Capacity-per-TCO-dollar metric and show how we can achieve a 7% benefit for a datacenter deployment by selecting an optimal operating point for a server. This methodology allows additional servers to be deployed within the same datacenter power and performance envelope, thus increasing the overall service capacity. This is a departure from previous approaches where TCO cost and peak performance capacities were the primary determinants for datacenter provisioning. We believe that there is wide topology of design possibilities for provisioning large datacenters and this is one of the first data-driven alternate designs. With enterprises trending toward use of cloud infrastructures, we believe that this method can significantly improve performance, power and cost standpoints for large datacenters.

8. ACKNOWLEDGMENTS

We thank internal and external reviewers for their insightful comments and reviews. We also thank Dileep Bhandarkar and the Datacenter Compute Infrastructure team at Microsoft for their feedback on this work.

9. REFERENCES

[1] Hamilton, J. R. Cost of Power in Large-Scale Data Centers. *http://perspectives.mvdirona.com* (November 2008).

[2] Fan, X., Weber, W., and L., A. Barroso. Power Provisioning for a Warehouse-sized Computer. In *ISCA 2007* (San Diego, CA, USA 2007).

[3] Sankar, S. and Vaid, K. Addressing the Stranded Power Problem in Datacenters Using Storage Workload Characterization. In *WOSP/SIPEW 2010* (2010).

[4] Reddi, V. J., Lee, B., Chilimbi, T., and Vaid, K. Web Search Using Small Cores: Quantifying the Price of Efficiency. *Microsoft Research Technical Report*

[5] *ACPI Specification, Revision 4.0.*

[6] INTEL CORPORATION. *DCMI Specification version 1.00*. 2007, 2008.

[7] Saravana, M. and Govidan, S. Using On-Line Power Modeling for Server Power Capping. In *Workshop on Energy-Efficient Design 2009* (2009), University of Texas and IBM.

[8] Wang, Z., McCarthy, C., Zhu, X., Ranganathan, P., and Talwar, V. Feedback control Algorithm for Power Management of Servers. In *ASPLOS* (2008), Hewlett Packard Laboratories.

[9] Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., and Zhu, X. No Power Struggles: Coordinated Multi-level Power Management for the Data Center. In *ASPLOS* (2008).

[10] INTEL CORPORATION. Dynamic Power Optimization for Higher Server Density Racks – A Baidu Case Study with Intel Dynamic Power Technology. 2008.

[11] HEWLETT PACKARD. Dynamic Power Capping.

[12] YOKOGAWA. Yokogawa Power Meter WT500, http://tmi.yokogawa.com/products/digital-power-analyzers/digital-power-analyzers/wt500-power-analyzer/. Yokogawa.

[13] Govindan, S., Choi, J., Urgaonkar, B., Sivasubramaniam, A., and Baldini, A. 2009. Statistical profiling-based techniques for effective power provisioning in data centers. In *Proceedings of the 4th ACM European Conference on Computer Systems* (Nuremberg, Germany, April 01 - 03, 2009). EuroSys '09. ACM, New York, NY, 317-330.

[14] Ranganathan, P., Leech, P., Irwin, D., and Chase, J. 2006. Ensemble-level Power Management for Dense Blade Servers. *SIGARCH Comput. Archit. News* 34, 2 (May. 2006), 66-77.

[15] Felter, W., Rajamani, K., Keller, T. and Rusu, C. A performance-conserving approach for reducing peak powerconsumption in server systems. In ICS '05: Proceedings of the 19th annual international conference on Supercomputing, pages 293–302, 2005.

[16] Femal, M. E. and Freeh, V. W. Safe overprovisioning: Using power limits to increase aggregate throughput. In 4th International Workshop on Power Aware Computer Systems (PACS 2004), pages 150 – 164, December 2004.

[17] Barroso, L. A. and Hölzle, U. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (Dec. 2007), 33-37. Barroso, L. A. and Hölzle, U. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (Dec. 2007), 33-37.

[18] Isard, M. et al. "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," *Proc. European Conf. Computer Systems* (EuroSys), 2007.

[19] Dean, J. and Ghemawat, S., "MapReduce: Simplified Data Processing on Large Clusters," OSDI, pp. 137-150, 2004.

[20] DeWitt, D. and Gray J. “Parallel Database Systems: The Future of High Performance Database Processing,” *Comm. ACM*, vol. 36, no. 6, 1992, pp. 85-98.

[21] Horowitz, M.A., Stojanovic, V., Nikolic, B., Markovic, D. and Brodersen, R.W. “Methods for True Power Minimization,” Proc. Int’l Conf. Computer-Aided Design (ICCAD ’02), pp. 35-42, 2002.

[22] Stan, M.R. “Low Power CMOS with Sub-Volt Supply Voltages,” IEEE Trans. VLSI Systems, vol. 9, no. 2, pp. 394-400, Apr. 2001

[23] Zhang, Y., Gurumurthi, S.; Stan, M.R.; , "SODA: Sensitivity Based Optimization of Disk Architecture," Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE , vol., no., pp.865-870, 4-8 June 2007

[24] Sankar, S., Gurumurthi, S., Stan, M.R. Sensitivity Based Power Management of Enterprise Storage Systems, International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems , September 2008.

[25] Ra, M.-R., Paek, J., Sharma, A. B., Govindan, R., Krieger, M. H., and Neely, M. J. “Energy-delay tradeoffs in smartphone applications”. In ACM MobiSys, 2010.

[26] Wilson, D. R. and Martinez, T. R. 1997. Improved heterogeneous distance functions. J. Artif Intell. Res. 6, 1-34.