Power Usage in Servers Centers: Estimation and Management Algorithms

Seminar Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Bhavin Doshi
Roll No: 123059004

under the guidance of

Prof. Varsha Apte

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
April 22, 2013

1 Introduction

Until recently, optimizing the computer systems was aimed only at maximizing their computing performance. While optimizing a computer system, the sole aim was to minimize the response time, limiting it below a certain threshold, called SLA. It was assumed implicitly by designers of these systems, that there is going to be a certain operational cost associated with the system in terms of power consumption. However, there was no effort made to minimize this power usage, partly because the hardware that allows such optimization of power was not available then, and partly because the need for it was never felt.

This trend started changing when power costs started dominating the cost of datacenters. Overall number of datacenters across the world has been increasing continuously. Cost analysis of datacenters reveals that power cost is one of the dominating factors in the datacenter's total cost. Power is used in two main operations of datacenters: computing and cooling. Computing resources (like CPU, disk, memory etc) use power, and dissipate heat. While optimizing the power required for cooling is a problem in the field of heat transfer and powerelectronics, optimizing just computing power is achievable by using advanced techniques like Power-managed CPUs and Cloud Computing.

1.1 Power: A Scarce Resource

In today's world, computing has become a non-separable part of our lives. Banking systems rely on computing as a core method of functioning, without which most of the *plastic* financial transactions would be impossible. Many developing countries like India look forward to shifting many of the governance systems to their new fully digitized avatars. Many businesses which never had a website before, are starting to create their online presence, in form of a website. In future, use of datacenters is going to increase, which makes power a significant factor for consideration while optimizing systems.

As per an estimation[2], total power consumption by datacenters is now 1.1% to 1.5% of world's total power consumption. In 2008, world's total power consumption was 1,43,851 TWh. Approximately 158 to 215 TWh power was hence used by the datacenters. Out of the total consumption, 1,17,076 TWh (or ∼81%)[4] was produced by burning fossil fuels. Due to such heavy dependence on fossil fuels, and limited availability of the same, the cost of power is going to increase faster in future. Hence from the perspective of cost also it is required to consider power as an optimizable resource in computing.

1.2 Power Managed CPUs

Power managed CPUs are the class of CPUs where the clock-frequency of the CPU can be controlled from software layer, in order to achieve different operatalgorithming frequencies as per varying needs, and hence lesser power dissipation. Modern CPUs in desktop, server and mobile range have the said feature of frequency scaling.

1.2.1 Hardware Features

There are various degrees of freedom when it comes to power-managed CPUs. Discrete frequency levels at CPU subsystem level facilitates selecting the operating frequency of the CPU as a whole from operating system level. Fuzzy voltage control (and hence frequency control) to the CPU subsystem enables setting any custom operating frequency. Per-core frequency scaling allows setting an operating frequency for each individual core, independent of other cores' settings. Hot plug facility to dynamically switch on/off the cores as per computing requirement allows the crucial power savings essential in mobile devices.

1.2.2 Software Features

Since Linux Kernel 2.6.32, the dynamic frequency scaling of CPUs is supported. There are multiple configurable settings available to control the module's behavior[3].

The change of frequency is governed in Linux by power governors. There are 5 governors available, which decide the way in which frequency of the CPU would change. They are powersaver, performance, conservative, ondemand and userspace. The powersaver governor always keeps the frequency to its lowest possible value. The performance governor always keeps the frequency to its highest value. The userspace governor takes one frequency value as input, and always keeps the frequency at the user-specified level. The other two governors change the frequency in response to the CPU's utilization.

There are two thresholds defined for the CPU utilization, up-threshold and down-threshold. If the utilization goes beyond up-threshold in a time interval called as probe-interval, then the governors act upon it. Breaching the up-threshold will scale the frequency up. Breaching the down-threshold will scale the frequency down. The conservative governor scales up the frequency step by step, making the frequency rise slower. The ondemand governor scales the frequency up by setting it to its highest value, whenever the up-threshold is breached. Both of these governors scale down the frequency step by step.

1.2.3 Power Dissipation

The power dissipation at any given operating point of CPU can be approximated to a general form. The power dissipation by CPU can be written as $P_{total} = P_{speed} * Utilization + P_{idle}$. P_{idle} is the idle power that is always consumed by CPU. P_{speed} is the additional power consumed by CPU when it is operating a certain frequency. This power consumption model is usually called as CPU utilization based power model.

1.3 Cloud and Virtualization

The power-management features of CPU are useful when used in standalone computers, and are more useful when used with specialized computing environments, like cloud. Cloud computing facilitates not only the short-term power optimization by using frequency scaling, but also provides medium and long-term power optimization by minimizing the use of physical machines used. This subsection introduces the cloud as a special power optimization technique over standalone computing.

As per WikiPedia[1], *"Cloud computing is use of computing resources (hardware and software) that are delivered as a service over a network (usually internet)."* Cloud allows sharing of a physical resource by multiple people, under the illusion of no-sharing. This allows efficient use of idle hardware, if any. The pay-as-you-go types of payment models allow clouds to operate in a cost-efficient way, and still provide customers with low to very low cost services.

In order to allow customers to order services in flexible *quantity*, virtualization is used as the enabling technique. Virtualization allows creation of *virtual* machines (VM) on the top of physical machines (PM). VMs are usually smaller in size w.r.t. the PM. Multiple VMs share the PM's resources. A software called hypervisor[17] manages the multiple VMs, allocates / deallocates the resources, and schedules VM execution. The sharing of resources is supported by hardware in some cases. In the absence of hardware support, the hypervisor enables this sharing of hardware.

1.3.1 Types of Cloud

Amazon EC2 is a cloud which provides hardware machine instances on request, which are fully configurable as per the user requirement. Here the infrastructure is given out to the customer, and hence EC2 is an 'Infrastructure as a Service' cloud, or simply an IaaS cloud. Microsoft Azure is a service where customers can buy instances of machines written in .NET libraries and a Common Language Runtime, a language-independent environment. This type of service is called "Platform as a Service", or simply PaaS cloud. Google AppEngine provides customers with instances of servers having a 3-tier architecture. Here, the software being provided as a service is a server system, and hence it is called "Software as a Service", or simply SaaS cloud[6].

1.3.2 Benefits of Cloud

There are multiple benefits of using the cloud deployment for a datacenter. The biggest motivation to use cloud is that the cloud provides huge benefit of better resource utilization. With the techniques of migration, VMs can be migrated from one PM to the other at runtime in a transparent manner. This allows the cloud provider to tightly pack VMs on least number of PMs possible till the point of their acceptable real-time performance, hence bringing the hardware utilization to the optimum. Other PMs that are not being used can be switched off, hence conserving power, both in terms of computing and cooling. Migration is a medium term solution to the changed computing needs, as migration itself takes some time, and has some associated cost. As a short term solution to the varying computing requirements, frequency scaling can be used in hardware resources to cut corners on the power usage, thereby having reduced overall power consumption.

Apart from all the administrative, logistical and flexibility gains, cloud acts as major enabler of IT presence for business startups. When a business kicks-off, the expected load on the concerned system applications is not predictable. A cloud deployment allows dynamic scale up of the hardware by renting more VM instances. It also allows drastic scale down of the hardware should the load on the application decrease after some time. This potentially allows businesses to set up their e-presence with zero initial investment in hardware. The very model of renting VMs from a cloud provider shifts the risk of hardware failure, as well as various costs like maintenance of hardware, cooling, physical space, and regular maintenance of servers from the business to cloud provider. As the provider is providing services at a larger scale to multiple customers, it can buy the hardware in very large bulk quantities, bringing the hardware resources' cost down.

Rest of the report is organized as follows. Chapter 2 discusses techniques characterization of power in a server system or a datacenter. Chapter 3 discusses resource management algorithms power usage minization. Chapter 4 describes a real life scenario in which potential power savings can be achieved using the techniques described in previous chapters. Chapter 5 summarizes the potential future research topics. Appendix A contains details about the additional papers read as a part of this seminar.

2 Characterization and Application of Power Metrics

When the resource management algorithms are considered for a datacenter, the approach taken is usually of an empirical model. A system of some general characteristics is taken up, and studied under various configurations and operating conditions. Based upon the observations and data points collected, a linear or a quadratic model is built by fitting the data points in to the model and then deriving the model parameters. In recent years, we have had many such models proposed, studied and discussed. A natural question then arises about the comparison of various models, and the qualities of *the* good model.

Koller *et al.*[19] describes the five qualities required in a good model: Predictable Input, Accuracy, Usability, Speed and Heterogeneity Support.

- The model should be predictable even when the operating conditions in which the model was caliberated, changes. This quality of the model is called predictable input. E.g. in a cloud environment, the frequent migrations of VMs result in changed co-located VMs. Calibrated values for a given VM should still hold valid, or the new value should be predictable, even after co-located VMs have changed.

- Accuracy expectation of the model is a classical requirement. It states that the predicted values by the model should be as accurate as possible, having least amount of errors.
- Usability of the model states that the parameters used by the model should be readily available in any datacenter, or the collection procedure for the required parameter should have minimum overhead. The parameters should also be collectible from user space.
- Speed talks about the processing time required to predict a value, once the model is built. Amortized model building time itself should be lesser.
- Heterogeneity support states that the model should be applicable for various applications that are used in a general datacenter. It should be flexible enough to possess the rest four qualities for various classes of applications.

On the top of these five requirements, an accurate and practical power model for heterogeneous applications needs to be application aware. Application awareness is however very difficult to capture in terms of model parameters, but some models like WattApp[19] tries to do that.

In this section, we would look at various empirical approaches to characterize power, as well as various techniques to use the proposed power models to optimize the software system. A technique by Chen[9] establishs a relationship between the variation in CPU frequency and the end-to-end response times of multitier systems. Another technique by Economou[14] relates component power consumption values with the component utilizations, which can then be used to estimate power usage of a live application. The technique by Koller[19] relates the system arrival rate with the VM level power consumption, for a given virtualization ratio. A technique by Ghosh[15] optimizes the counts of hot, warm and cold VMs inside a cloud, given an arrival rate. Finally, Bhattacharya[8] describes interesting observations about performance and power of a system, when software components are debloated for better performance.

2.1 Relation of Frequency Variation With End-to-End Response Time

When components of a complex software system are deployed on various physical machines, it becomes more difficult to relate the end-to-end system performance with the varying frequency of each physical machine. As multiple machine's frequencies are changing, the impact on response time would depend on the service time of each component, and the division of task's total service time among the software components. Other practical complexities like probabilistic division of tasks, multiple customer classes interacting with the system etc. make it difficult to argue about the system behavior under frequency scaling.

In such a scenario, it is better to abstract the system as a blackbox, thereby hiding its complexities. Chen *et al.*[9] have proposed a technique which leverages this abstraction, and then relates the system response time with the changing frequency of the physical machine of each of the component. For simplicity, from now on, we will call it the frequency of the component.

2.1.1 The Model

The technique begins by relating the response time with not only the workload of the system, but also the operating frequencies of each of the component. Responsdisplaystylee time $\bar{rt}(x) = F(w, f_1, f_2, ..., f_m)$. When the system behavior is being sampled, change in the operating condition is recorded as $\Delta x =$

$x_1 - x_0 = F(\Delta w, \Delta f_1, \Delta f_2, ..., \Delta f_m)$. Since we would know the response time at operating point x_0, $\bar{rt}(x_0)$, then we can write the response time at the new operating point x_1 as

$$\bar{rt}(x_1) = \bar{rt}(x_0) + \sum_{i=1}^{m} \frac{\partial F}{\partial f_i}\Big|_{x_0} \Delta f_i + \frac{\partial F}{\partial w}\Big|_{x_0} \Delta w + O(\Delta x^2)$$

With the assumption that Δx is small enough or that F is linear, the higher order terms of Δx can be neglected. It means that we can predict the system response time at any operating point x_1 if we know the $\bar{rt}(x_0)$, as well as the *frequency gradients* of the system (partial derivatives), but neither of the assumptions are true in practice.

The relative change between system response time and operating point is further broken down into a two step relation. Relative change in system response time w.r.t. component response time is called system gradient, $\partial \bar{rt}/\partial rt_i$. Relative change in component response time w.r.t. component frequency is called machine gradient. The non-linear relation between frequency and response time is taken care by a basis function on frequency. $rt = MeanServiceDemand/f(1-U)$ is taken with the assumption of M/G/1/PS behavior of response time. With $U = a \cdot w/f$ (a being transaction specific constant), the basis function $\hat{f}_i = (f_i - a_i w)^{-1}$ is considered for each component. $\partial \bar{rt}_i/\partial \hat{f}_i$ is then calculated.

Since in a system, these two gradients are not measurable separately due to lack of control or direct measurement of component response times, we combine these two gradients by chain rule of derivative. Frequency gradients are hence defined as $\left(\frac{\partial \bar{rt}}{\partial \bar{rt}_i}\frac{\partial \bar{rt}_i}{\partial \hat{f}_i}\right)$ for each host machine. The final composite predictor of response time is then defined in the paper as follows:

$$\bar{rt}(x_1) = \bar{rt}(x_0) + \sum_{i=1}^{m} \frac{\partial \bar{rt}}{\partial \bar{rt}_i}\frac{\partial \bar{rt}_i}{\partial \hat{f}_i}[\hat{f}_i(f_{i_1}) - \hat{f}_i(f_{i_0})]$$

Frequency gradients become unusable when system configuration changes drastically. The gradients must be reconfigured for new configuration.

2.1.2 Measurement of Model Parameters

To estimate the frequency gradients of the systems, the frequency of the system was altered between two point values: current frequency and lowest frequency. This change was brought about w.r.t. a square wave pattern. At the end, Fourier transform was used to get the *frequency spectrum* of response time, which is in-turn used to get an estimate on frequency gradient of the concerned system.

2.1.3 Conclusion

We can analyze the system while it is running on production load, without disturbing it, or taking it down. This unobtrusive analysis is a great advantage of this technique. From this technique, however, it was not clear from the paper as to how would the technique be applied in case of multi-core CPU system. The paper also implicitly assumes that the system would be CPU bound. In case of multiple frequency scaling devices (like frequency scaled IO andisplaystyled CPU), the technique has to be re-thought. In case of software updates to the system, the frequency gradients must be recalculated if the update brought about drastic changes in the system's behavior. System's M/G/1/PS assumption might not be always valid, as the arrivals in complex systems might be batched, or they cannot be periodic. In such cases, a whitebox approach via a simulator like PerfCenter[13] is more accurate.

2.2 Relation of Component Power With Component Utilization

Techniques till now have focused on estimation of power from observed values. These are usually the socket level readings of power. This means that the power is read by connecting an ammeter to the main power supply, and then observing value of the power drawn by the whole system. A component level power estimation is an area which is yet less explored. Component level power observations and estimations require very close instrumentation of the computer circuitry and motherboard. This technique is studied by Economou *et al.*[14].

2.2.1 The Technique

- The technique 'Mantis' considers a blade server for its study, as advanced power management techniques like DVFS and low-power disk. The technique goes down to the server board of the blade server. The measurements of per-component power consumption are best done at per-component level.
- The server board is logically divided into four parts (Quoted from [14]):
  - A 12V plane consisting of processor and memory. This plane consumes more than 90% of the power.
  - A 5V plane consisting of hard-disk
  - A 5V auxiliary plane
  - A 3.3V plane, which along with the 5V auxiliary plane would consist of network, peripherals, regulators, supplies and other miscellaneous components of the system.
- A separate power measurement board is designed to measure power across all the four planes simultaneously. The processor power is further isolated form the first 12V plane, and is measured separately.
- Under these settings, a bunch of caliberation tests are run, just once, to measure the power behavior of the hardware under different conditions. Since the measurement is done at hardware level, the tests are designed to stress each hardware component separately. For generating the load, a utility gamut is used. The OS level utilization values for CPU and IO (possibly from the hardware counters) are also noted using SAR.
- The observed values are then fitted into a linear model, which can be further used to predict the power usage of any application. This linear fitting is done via a linear optimization problem, which tries to minimize the absolute error in per-component predicted power w.r.t. the observed power.
- The granularity of data would also limit their accuracy. While the power values could be observed continuously without any gap, and quantized till a required granularity; the utilization values from the OS are available only at a second's resolution.

2.2.2 Accuracy of the Technique

The technique stands accurate till the general category of hardware is used. When special CPUs like Itanium were tested, there was some deviation in the prediction. Itanium is a server class processor, with instruction architecture of type VLIW (Very Long Instruction Word). This type of instruction has *sub*-instructions inside it, which are targeted towards one CPU component. So in a VLIW instruction, there can be two integer workloads and one floating point computation and so on. Such instruction sets require very complex compilers to produce assembly code that can leverage the hardware capacity of VLIW instructions. CPU's power consumption would then depend on how much a program leverages this facility. This property is measured in the ILP (Instruction Level Parallelism) of a program. As pointed out by [19], the utilization based models are inaccurate because they ignore the application behavior. Here, the same phenomena seems to be occuring. The workload SPECcpu2000-int, which is targeted towards the VLIW facility of the processor, revealed this flaw in the model.

2.2.3 Conclusion

The model's beauty lies in its simplicity of use. Once the caliberation is done, which can be at the board manufacturer level, the available model parameters can be used right away to predict instantaneous and average power for any scenario. This is possible only because the model was application agnostic. This is a model which normal users cannot use unless the support for measurement of values is provided from the hardware vendor. Doing modifications on the board for caliberation might result in voiding the warranty! This technique is hence not directly usable, but subject to industry acceptance on a wider scale.

2.3 Relation of Arrival Rate to System Power Consumption Under Virtualization

Classic prediction techniques of power rely only on the CPU utilization. Such techniques associate the consumed power just with CPU utilization. They implicitly require existence of a single transforming function from utilization to power over the full range of utilization. However, Such a relation between the utilization and power does not always exist. This depends on the nature of concerned system too, on whether it is CPU bound or not. CPU utilization based models also assume that the application is able to make 100% use of CPU with some operating scenario. If a program is not CPU bound, and it bottlenecks on some other power-managed resource instead, then the CPU utilization driven model is not even defined. Current models are application agnostic, and that results in huge estimation errors.

Koller *et al.* have proposed a new technique WattApp[19], which solves the problem of this erroneous prediction. It also talks about resource availability issue in production environments for benchmarking and model-building purposes. WattApp takes M types of servers, and N applications, to perform O(M · N) calibration runs. This data results in 95% accurate predictions.

2.3.1 The Model

WattApp uses throughput as the app specific parameter. For $0 <= \rho <= 1$, the graph for ρ, L2 cache miss count per sec, Network utilization vs λ is observed to be linear for an array of applications via calibration runs on them. This linear graph is fitted into a linear equation with $Power_i = \alpha_i + \beta_i X$. This obviously means that the prediction of the throughput (or load at unsaturated levels) would get us the power estimation. WattApp defines the virtualization ratio d as virtualization overhead for current application when i VMs are deployed on a PM. For some applications, power consumed increases with increased d. Hence virtualization aware power models are proven to be required. The said model would however fetch us only the machine level power, and not the VM level power. This linear graph at a given virtualization ratio d is fitted into a linear equation with $Power_{i,d} = \alpha_{i,d} + \beta_{i,d} X$.

2.3.2 Measurement of Model Parameters

In order to do the calibration runs, idle machines are required. However, in high performance datacenters, availability of idle machines is always scarce. WattApp co-ordinates with the power manager, and *steals* a server when it is just about to be turned off. It then performs calibration runs, and then returns the server to power manager, so that it can be turned off. This is a very clever technique used for utilizing the intermittently idle physical machines in the datacenter for the benchmarking purposes.

2.3.3 Conclusion

The paper conveniently makes a lot of claims which are hard to justify.

> Marginal power drawn by a server has a linear relationship with the marginal increase in application throughput.

This does not fit with the utilization law, when considered under frequency scaling. There is no reasoning given for such statements.

> Power drawn by a set of applications running on the same server can be inferred as a linear combination of the power models of each application running independently.

If virtualization overhead is part of application power model, then taking a linear combination would result in counting virtualization overhead multiple times. There is no explanation given in the paper about handling such cases.

> Application with high I/O activity or low working set need to be aware of the virtualization ratio, while building their power models. On the other hand, for applications characterized by large working set and low I/O activity, we may not need to build power model separately for each virtualization ratio.

This statement tries to correlate two (possibly) completely independent facts: TPC-W- has small working set and TPC-W- has highest impact on power consumption with increased d.

WattApp models assume that application performance is not affected by co-located applications in virtualized environment: This assumption is very very general, and in most cases not true. The assumption holds only for the datacenters running many instances of the same application on a server. However, in utility datacenters, this would obviously be not the case.

2.4 Power and Performance Trade-Offs in IaaS Cloud

In an IaaS cloud, the operating model of the cloud provider's business would expect users to submit *jobs* to the cloud for execution. Upon completion, the result would be given back to the user. Internally, the cloud would create a VM to execute the job. In this scenario, there are three options available about handing of VM. VMs can either be turned off completely once the job is complete, or it can be sent in a deep sleep, or it can be kept on and running even after the job is finished. All the three techniques have their own pros and cons. Shutting down the VMs would conserve energy, but starting a new VM upon job arrival will take more time as more number of VMs would need to be started from scratch. Keeping the VM on would have minimal delay in starting an incoming new job, but in its idle state too, the VM would keep on consuming some energy.

It is then obvious that a good strategy would use a mix of these three types of VMs. In order to decide what mix to use, a CTMC (continuous time markov chain) based analytical model is presented by Ghosh *et al.*[15]. They solve the problem of optimal breakup of VM allotment in three categories (hot, warm and cold), given the number of PMs. Optimality is achieved on the power consumption and performance of the PMs.

2.4.1 The Model

Being an analytical technique, the paper makes many simplifying assumptions about the system behavior. They assume that all VM placement requests are homogeneous VMs with fixed size RAM and fixed CPU cores. They also assume that the power consumption of the PMs is lowest in cold state, more while VM is in warm state, and even more when VM is in hot state. Interacting stochastic sub-models are used to model the behavior of three pools of VM. Each state in the CTMCs have some reward rates, which represent power consumption. This analytical system is solved to estimate the power consumption of the PMs at given system parameters.

The system input parameters are as follows:

- VM[1] job arrival rate is λ
- VM job processing rate is μ
- VM provisioning time on a hot/warm PM is β
- cold/warm VM initiation time is γ
- reward rates (power consumption) for each state of the machine
- each PM's maximum VM holding capacity
- breakup of hot, warm and cold PMs for given n PMs
- mean delay to search a PM from the pool

System state is defined as (VM-queue, VMs-being-provisioned, VMs-running-currently) for a PM. When a new VM arrives, it joins the queue for provisioning, or it starts *getting* provisioned on a hot PM if one is available (has non-full buffer). Otherwise a warm PM is considered for placement if one is available. Otherwise a cold PM is considered if one is available. Otherwise the request is rejected. Upon completion of provisioning, it starts executing its own job. Upon completion, the VM is removed from the PM. Probabilities that a VM request is accepted by a hot/warm/cold PM (P_h, P_w, P_c) are computed from the three CTMCs. Output of hot CTMC (P_h) is given as input to warm CTMC. Output of warm CTMC (P_w) is given as input to cold CTMC. Analysis of the three CTMCs give us the various performance parameters for the system, such as mean queuing delay, drop probability due to buffer full, drop probability due to capacity full etc. These CTMCs can be used by a simulator (authors used SHARPE) to do the analysis for given system parameters, and compute the output parameters.

2.4.2 Conclusion

A simulation or a calibration technique usually has this limitation of non-scalability beyond a certain point. Being an analytical model, the biggest positive point of the technique is that it is scalable for any number of PMs. The main contribution of the paper is about the intuition that for lower power consumption, using more cold PMs is better. But this intuition is proven to be wrong by the paper, as for a given pool breakup, the power consumption is dependent on the load level. This of course means that if good load predictors are available, then the decision of changing the pool breakup (or in other words: shutting down the useless PMs) can be made. This can be a good server consolidation as well as server provisioning algorithm.

2.5 Interplay of Power and Performance in Software Systems

When looked from power perspective, software bloats can be quite expensive. Bad code can cause higher power consumption for the same task that could have been done with lesser power. This increases the cost of execution, which increases the operational cost of running the software. Bhattacharya *et al.*[8] presents a systematic reasoning about the relation between power, performance and software bloat. While analyzing software bloat and performance, bottleneck resources also come into picture. They also present a method to identify potential benefits of removal of bloats, so that the process of bloat removal in a complex system can be prioritized based on benefits.

An interesting fact pointed out by this paper is that the bottlenecks in certain resources decreases the utilization of non-bottlenecked resoures, hence bringing down their power consumption. Removal of bloat does not always result in lower power, as the removed bloat can now allow a non-bottlenecked resource to be utilized better, hence increasing its power consumption. This interplay of power and performance makes it difficult to quantify the improvement post bloat reduction.

[1]a VM here is a job that is submitted for execution

2.5.1 The Observations

Four metrics to quantify the improvement post bloat reduction are as follows:

- Relative throughput: $\frac{Unbloated\ throughput}{Throughput\ with\ bloat}$. This should be as high as possible.
- Relative peak power: $\frac{Unbloated\ power\ at\ unbloated\ peak\ performance}{Power\ with\ bloat\ at\ bloated\ peak\ performance}$. This should be as low as possible.
- Relative pwer efficiency: $\frac{Unbloated\ throughput\ /\ Unbloated\ power}{Throughput\ with\ bloat\ /\ Power\ with\ bloat}$. This should be as high as possible.
- Relative *equiperformance* power: $\frac{Unbloated\ power\ at\ BLOATED\ peak\ performance}{Power\ with\ bloat\ at\ bloated\ peak\ performance}$. This should be as low as possible. This is strongest indicator of reduction in power consumption as all the resources apart from the unbloated resource are still operating at the same utilization point, hence consuming the same power.

A resource can have both power-managed and non-power-managed devices. There is a very elegant solution to capture both types of devices in the same model. With the energy proportionality of devices (α), this fact is captured seamlessly across the system.

2.5.2 System Bottleneck and Bloat

- Reduction of bloat at non-bottlenecked resource would not change the throughput, but will decrease the power. Amount of reduction is subject to the energy proportionality of the device, and the fraction of system power that it consumes. Hence power efficiency improves.
- Reduction of bloat at bottlenecked resource might increase the power consumption. The bottlenecked resource might limit the utilization (and hence power consumption) of other resources. But upon unbloating, the other resource's utilization might increase, which would increase the power consumption. Reduction of bloat would increase the throughput, and increase the power. This might lead to lesser gains in power efficiency.
- Reduction of bloat resulting in shift of bottleneck would yield similar results as reduction of bloat from bottlenecked resource. The throughput gain in this case might be lesser. The power consumption might increase, decrease or remain the same.

2.5.3 Conclusion

The paper explains some very non-intuitive relations between power and performance. These insights can be very useful while analyzing a complex system. But while developing models, or doing a generic analysis, the technique is least useful, as it requires as input the *amount* of bloat that exists in the system. This quantity is not a measurable quantity at runtime, compile time or development time. With amount of bloat being unavailable, all that can be done using interplay model is put a very rough estimate on the bloat, and let the power characteristic of hardware resources dominate in the decision.

Also, the types of bloat that the paper talked about are usually of architectural in nature. E.g. the overtly general designing of software components and too-much of the flexibility which results in more activities at runtime. Such things can't even be identified conclusively as bloat, let alone be quantified. The observations made in the paper are academically interesting as they formalized some of the intuitions, and some of the unknown and non-intuitive relationshipts between performance and power.

3 Queuing Models for Frequency Scaled CPUs

Queuing theory as a concept is extremely rich, with its wide application in various fields, including software servers used in the field of computing. Many datacenter sizing problems can be quickly solved, if we know the expected load on the software system, and its expected performance. This is true, however, only for the datacenters using CPUs with constant speed that is not changeable at runtime. With most CPUs being frequency-scaled, the direct application of standard queuing models do not take into account the variability of frequency. Current sizing techniques consider just the maximum frequency of the concerned CPU, and process of sizing estimation is completed.

As long as we are planning non-virtualized datacenters, this approximation, of considering the maximum frequency of CPU, towards capacity analysis is acceptable, as all the hardware would cater to just the software under consideration, albeit with some possible over-provisioning of hardware. However, with the virtualized datacenters, where we can control the hardware allocation to each VM, it becomes important that the software uses best power-management policy possible. With appropriate power-managers running at runtime inside VMs, the software can *convey* its reduced requirements to hypervisor, and the resources can be accordingly freed up, as suggested in VirtualPower[20], which would obviously allow tighter packing of VMs on a PM. In case where there are no more VMs to be put on a PM, that PM's CPU frequency can be brought down to get short-term power savings. Hence, it is important to have appropriate queuing models for frequency scaled CPUs, through which we can answer the power configuration questions for the software. Queuing models would also allow a bottom-up analysis of a system, unlike the top-down approach taken by empirical models.

3.1 Relation of Power with System Arrival Rate

Power, as a measurable entity, is widely studied w.r.t general electrical equipments, as well as in the field of electronics. For such equipments, the peak power consumption value is defined with physics formulae. Modern computer, being an electronic equipment involving multiple microchips as well as microprocessors, has its peak power consumption value defined. For computing systems, however, the value of peak power consumption is of little practical use, as the peak power is seldom consumed. In the context of computing systems, especially in a datacenter, the quantity of practical interest is the amount of *average power* consumed.

Average power is the time-average of the energy consumption by the computer. Event though instantaneous energy consumption is the sum of energy consumption by each component of the computer, it is dominated by the CPU's energy consumption. This makes the average power consumed by a computer system also variable.

$$AvgPower = \sum_{t=0}^{t_{max}} Energy(t) \tag{1}$$

With availability of power-managed CPUs, the instantaneous operating frequency of CPUs $Energy(t)$ has now become variable. Variation of frequency of CPUs directly mean that the power consumption by the CPU is now variable. Also, as the frequency of the CPU is governed by the utilization of CPU in the previous probe interval (ρ_{t-1}), length of the probe interval, power governor, up threshold and down threshold; we can write:

$$\begin{aligned} Energy(t) &= h_1(f_t) \\ f_t &= h_2(\rho_{t-1}, \text{Probe Interval, PowerGovernor, UpThreshold, DownThreshold, } S_f) \end{aligned} \tag{2}$$

where S_f is set of discrete frequencies settable on the CPU. Its worth noting that in equation 2, everything except ρ_{t-1} on RHS is system configuration parameters.

Also, with Utilization Law for the stable systems, $\rho = \lambda \cdot \mu$, λ being the arrival rate of the system. Hence we can say,

$$\begin{aligned} \rho_t &= \lambda_t \cdot \mu_t && (3) \\ f_t &= h_{23}(\lambda_{t-1}, \mu_{t-1}) \text{ (from equation 2 and 3)} && (4) \end{aligned}$$

Frequency scaled CPUs also makes the service time of the job variable. Service time of a job is dependent upon the operating frequency.

$$\begin{aligned} \mu_t &= h_4(f_t) && (5) \\ \mu_t &= h_{24}(\lambda_{t-1}, \mu_{t-1}) \text{ (from equation 4 and 5)} && (6) \end{aligned}$$

Hence, the service time of jobs can be considered to be load-dependent for the frequency scaled CPUs.

In order to reason through the average power consumption analytically, we must be able to calculate the opearting frequency, which is dependent on the arrival rate for the system. When we have the values of load dependent service time for a system, we can guess the operating frequency with the help of funtion h_{23} (from equation 4), which is unsolved as yet. However, load dependent queuing systems are well-researched.

In the following subsections, we take a look at technique of analyzing load dependent queuing networks.

3.2 Load Dependant General Queuing Networks

An approximate technique of analysis of load-dependent general queuing system under a set of reasonable assumption was done by Akyildiz and Sieber[5]. The technique is analytically iterative in nature, which is applicable for load-dependent nodes of a system. Such load dependent nodes' arrival rate as well as service rate exhibit load dependency. The summary of the solution along with the assumptions is given below.

3.2.1 System Parameters and Solution

A system comprising of multiserver stations, with each server i having service rate μ_i, would have the total service rate as a function of the number of jobs. This service rate can be written as $\mu_i(k) = min(k\mu_i, m_i\mu_i)$ in case of finite number of servers. An infinite server system would have the service rate as $k\mu_i$.

Each server i has a load dependent arrival rate $\lambda_i(k)$. A simple fact is that for any station having k jobs in it, the rest of the system will have $K - k$ jobs, with K being the maximum jobs that the system might have. With this fact, the load dependent arrival rate $\lambda_i(k)$ for a station is obtained by shorting that station, and calculating throughput of the subsystem $\lambda'_c(k)$ via any standard method like MVA or convolution algorithms.

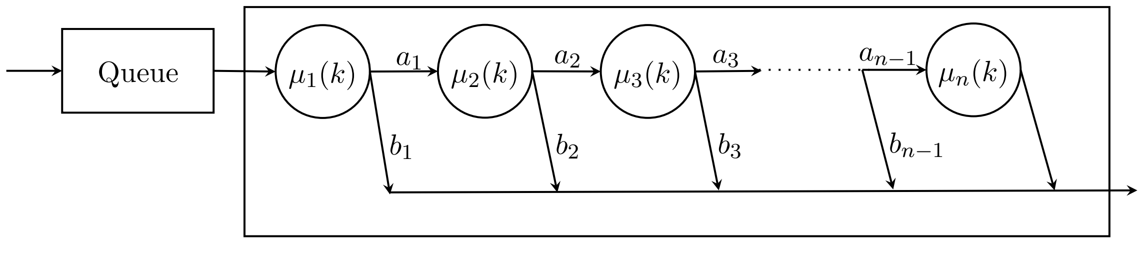

Figure 1: n-Phase Coxian Distribution

Any distribution that has a rational laplace transform can be represented by a sequence of fictitious phases, as shown in Figure 1. Such an approximated distribution would be called a Coxian distribution with n-phases. Furthermore, if the distribution has squared coefficient of variation (c^2) such that $0.5 \leq c^2 \leq \infty$, then the distribution can be approximated as a two-phase Coxian distribution. The server is then assumed to have a service time distribution that has rational laplace transform, and squared coefficient of variation between 0.5 and ∞.

Parameters of the said two-phase coxian distribution would be then determined as follows:

$$\begin{aligned} \mu_{i1}(k) &= 2\mu_i(k) \\ \mu_{i2}(k) &= \frac{\mu_i(k)}{c_i^2} \\ a_i &= \frac{1}{2c_i^2} \\ b_i &= 1 - a_i \end{aligned}$$

The system has the notion of conditional throughput $v_i(k)$, which is solved by Akyildiz and Sieber[5].

$$v_i(k) = f\big(v_i(k-1)\big) \text{ given } a_i,\ b_i,\ \mu_{i1}(k),\ \mu_{i2}(k),\ \lambda_i(k),\ \mu_{i1}(k-1) \text{ and } \mu_{i2}(k-1)$$

If the system state k has k customers in the system, then the steady state probabilities $p_i(k)$ are also given by the authors.

$$p_i(k) = g\big(p_i(0),\ \lambda_i(n),\ v_i(n+1)\ \big) \text{ for } 0 \leq n \leq k-1, \text{ for } 1 \leq k \leq K$$

The basic solution for $v_i(1)$ and $p_i(0)$ is also presented.

$v_i(k)$ and $p_i(k)$ are computed iteratively until the termination test fails. To test the accuracy of the solution, the number of customers in the system is computed from $p_i(k)$ as $\sum_{i=1}^{N}\sum_{k=1}^{K} k \cdot p_i(k)$. If this computed total number deviates more than ϵ from known total customers K in the system, then the accuracy test fails, and one more iteration is done to get fresh estimates. Usually 6 to 10 such iterations are required before we can terminate, but there is no formal estimation of number of iterations that a system would require before getting solved. ϵ usually has a value of 10^{-4} or lesser.

4 Conclusion and Future Work

- **Analytical Power Models**: Power management works with up-threshold, down-threshold, governor and probe-intervals. All the above methods are essentially empirical methods, which are agnostic to these system settings. Currently we cannot derive the optimal values of these configuration parameters, given the expected performance of the system. Creating a queuing model with frequency scaled CPU will allow us to predict power consumption, but also facilitate answering the tuning questions about configuration params of power management in Linux.

- **Whitebox Simulation Approach to Power Estimation**: In certain scenarios where the simplifying assumptions required to work out an analytical model might not be assumable, a tool like PerfCenter[13] can be used to predict the power behavior of the system. Significant discovery of details can be done by using a fine-grained power-performance simulator.

- **Greybox/Blackbox Approach to Power Estimation**: In the simulators like PerfCenter, currently a complete whitebox visibility of the system is required. Even when the visibility of the system behavior is available, it might be too complex to be modeled and analyzed via a whitebox approach. In order to analyze the power behavior of a system, discovery techniques based upon just the system level (as opposed to component level) metrics like system response time, system CPU utilization and system throughput can be searched for. A technique with the said capability of relating the system metrics with system power might be usable in various resource optimization scenarios. Some of the potential scenarios include the VM migration for server consolidation, power-planning of a datacenter, and desktop energy saving schemes similar to LiteGreen[12].

- **Simulators With Extreme Scalability**: As pointed out in [14], full system simulators are extremely slow compared to real hardware. A simulator which has capability of simulating a complete system, to its complexity, can be worked towards. Such a simulator would obviously use more resources than the hardware itself, but completing the simulation fast enough, by using multiple computers simultaneously (parallel and/or distributed computing) can be meditated and researched upon. Discrete event simulations being simpler in design, can incorporate as much of complexity of the target system, with least number of assumptions. This simplicity of designing can be a significant motivation towards working on building a full system simulator.

Appendices

A Additional Papers Read

There is one more paper that has been read as a part of Seminar reading, but it did not fit in the final topic. Summary and details of the said paper are given in this appendix.

A.1 CosMig

In a virtualized datacenter, benefits of virtualization is maximized with VM migration to achieve server consolidation or hotspot mitigation. These migration actions do have some cost associated with them, along with a temporary performance impact on the concerned application's performance. Current models assume the migration to be a function of active memory of VM, and other parameters of the VM under consideration, but they all ignore the co-located VMs. In order to get some performance certainty of VMs, it becomes important to consider impact of migration on the co-located VMs as well in the cost model of migration. Verma *et al.*[21] presents a model to estimate the duration of migration, and estimate the migration's impact on application performance.

A.1.1 The Model

Migration can be described in three parameters: duration of migration, self impact of migration and co-impact of migration. Impact can be in terms of fraction of reduction in actual throughput. The cosmig model consists of two steps. First step is capturing the fixed parameters, which are noted at a given operating point. Operating point consists of multiple factors, like cpu utilization, active memory and type of deployed application. This is called Calibration step. At a fixed operating point, self-impact, co-impact, CPU required for migration and migration duration are noted. With multiple such readings, the second step of estimating rate parameters is carried out, which is called Estimate Step. *Rate parameters* are independent of application. Since impact of co-located VM is similar to that of reduced resources for the current VM, impact of co-located VMs is modeled as a change in operating point. Rate parameters are relative increase in cpu per increase in active memory, peak rate at which memory pages can be copied etc. These steps are performed at 'no resource contention' state. In order to estimate the migration time even with CPU contention, the self-impact is multiplied with 'CPU-Overload', only if there is contention.

```
Begin Estimate
$\tau^*(i) = \tau_{AM}(i) + \frac{AM_i - AM}{MBw^*}$
$CPU_i^{Mig} = CPU_i^{Mig*} + \delta CPU^{Mig}(AM_i - AM)$
If $CPU_i + CPU_j + CPU_i^{Mig} <= CPU_{tot}$
return $\tau^*(i)$, $\pi_s^*(i)$ and $\pi_c(i)$.
Else
$\rho_i = \frac{CPU_i + CPU_j + CPU_i^{Mig} <= CPU_{tot}}{CPU_{tot}}$
return $\tau^*(i)$, $\pi_s^*(i)\rho_i$ and $\pi_c(i)$.
End Estimate
```

Table 1: CosMig Model

AM	active memory
π_s^*	self-impact
π_c	co-impact
τ_{AM}	migration duration
CPU^{Mig^*}	CPU required for migration
δCPU^{Mig}	increase in cpu per increase in active memory
MBw^*	peak memory copy bandwidth

Table 2: CosMig: Notation list

Cosmig models the pre-copy live-migration[10] technique of VM migration, and its not applicable to remus[11] or post-copy[16]. With this technique, the estimation error is reduced from 25% to 5%.

A.1.2 Conclusion and Possible Extensions

The paper assumes for the validity of the technique, that there is a separate network link available on the PM for migration. The paper also assumes that the co-impact would be constant, and only migrating VM's utilization would change. This assumption is invalid for applications with high memory dirty rate. Since such applications take more time to migrate, while they are being migrated, the load characteristics of the co-located VMs might change, which would render the pre-calculated co-impact invalid. The technique would work in the scenario where the VMs are not changing rapidly. In a dynamic environment, where new VMs are getting created and destroyed very frequently, then the requirement of calibration of each pair of VM would make the solution prohibitively slow.

The paper was verified with pHyp hypervisor, which is specially designed for power systems of IBM. The hypervisor might be very tightly integrated with the IBM power system, and hence its performance might be too close to real non-virtualized systems. In case of general purpose hypervisors like Xen[7] and KVM[18], this might not be true, but it might as well be. If there is any extra virtualization overhead in case of KVM or Xen, then the validity of the technique is yet to be verified with those hypervisors.

A major extension to the technique can be consideration of power. The technique can be enhanced to see the impact on power for each VM, and thereby calculating estimated *extra* power used while migration. In the calibration step, this value can be observed, and then can be used in the estimation step to calculate required figure. The final selection of migration destination can be done on the basis of minimum CPU impact, and minimum power impact.

References

[1] Cloud computing, wikipedia.
`http://en.wikipedia.org/wiki/Cloud_computing`.

[2] Growth in data center electricity use 2005 to 2010, analytics press.
`http://www.analyticspress.com/datacenters.html`.

[3] Power governors, documentation of linux kernel 2.6.32.
`http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt`.

[4] World energy consumption, wikipedia.
`http://en.wikipedia.org/wiki/World_energy_consumption`.

[5] I. Akyildiz and A. Sieber. Approximate analysis of load dependent general queueing networks. *Software Engineering, IEEE Transactions on*, 14(11):1537–1545, 1988.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[8] S. Bhattacharya, K. Rajamani, K. Gopinath, and M. Gupta. The interplay of software bloat, hardware energy proportionality and system bottlenecks. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, HotPower '11, pages 1:1–1:5, New York, NY, USA, 2011. ACM.

[9] S. Chen, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders. Blackbox prediction of the impact of dvfs on end-to-end performance of multitier systems. *SIGMETRICS Perform. Eval. Rev.*, 37(4):59–63, Mar. 2010.

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[11] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: high availability via asynchronous virtual machine replication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 161–174, Berkeley, CA, USA, 2008. USENIX Association.

[12] T. Das, P. Padala, V. N. Padmanabhan, R. Ramjee, and K. G. Shin. Litegreen: saving energy in networked desktops using virtualization. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, pages 3–3, Berkeley, CA, USA, 2010. USENIX Association.

[13] A. Deshpande, V. Apte, and S. Marathe. Perfcenter: a performance modeling tool for application hosting centers. In *Proceedings of the 7th international workshop on Software and performance*, WOSP '08, pages 79–90, New York, NY, USA, 2008. ACM.

[14] D. Economou, S. Rivoire, and C. Kozyrakis. Full-system power analysis and modeling for server environments. In *In Workshop on Modeling Benchmarking and Simulation (MOBS)*, 2006.

[15] R. Ghosh, V. Naik, and K. Trivedi. Power-performance trade-offs in iaas cloud: A scalable analytic approach. In *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, pages 152–157, June.

[16] M. R. Hines, U. Deshpande, and K. Gopalan. Post-copy live migration of virtual machines. *SIGOPS Oper. Syst. Rev.*, 43(3):14–26, July 2009.

[17] S. T. King, G. W. Dunlap, and P. M. Chen. Operating system support for virtual machines. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '03, pages 6–6, Berkeley, CA, USA, 2003. USENIX Association.

[18] A. Kivity. Kvm: the linux virtual machine monitor. In *OLS '07: The 2007 Ottawa Linux Symposium*, pages 225–230, July 2007.

[19] R. Koller, A. Verma, and A. Neogi. Wattapp: an application aware power meter for shared data centers. In *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, pages 31–40, New York, NY, USA, 2010. ACM.

[20] R. Nathuji and K. Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 265–278, New York, NY, USA, 2007. ACM.

[21] A. Verma, G. Kumar, R. Koller, and A. Sen. Cosmig: Modeling the impact of reconfiguration in a cloud. In *Proceedings of the 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '11, pages 3–11, Washington, DC, USA, 2011. IEEE Computer Society.