PowerPerfCenter: A Power and Performance Prediction Tool For Multi-Tier Applications*

Varsha Apte, Bhavin Doshi†
Department of Computer Science and Engineering, IIT Bombay, Mumbai 76, India
{varsha,bhavin}@cse.iitb.ac.in

ABSTRACT

The performance analysis of a server application and the sizing of the hardware required to host it in a data center continue to be pressing issues today. With most server-grade computers now built with "frequency-scaled CPUs" and other such devices, it has become important to answer performance and sizing questions in the presence of such hardware. *PowerPerfCenter* is an application performance modeling tool that allows specification of devices whose operating speeds can change dynamically. It also estimates *power usage* by the machines in presence of such devices. Furthermore, it allows specification of a *dynamic workload* which is required to understand the impact of power management. We validated the performance metrics predicted by PowerPerfCenter against measured ones of an application deployed on a test-bed consisting of frequency-scaled CPUs, and found the match to be good. We also used PowerPerfCenter to show that power savings may not be significant if a device does not have different *idle* power consumption when configured with different operating speeds.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling Techniques

Keywords

Power, Performance Prediction, Frequency Scaled CPU

1. INTRODUCTION

Performance analysis continues to be a critical step in the life cycle of a server application. Typically, performance analysis starts when the functionality of the application is ready and it is deployed in a testbed. At this stage, comprehensive performance tests are undertaken, in which metrics such as throughput and response time of requests are plotted against load intensity (which is often in terms of "number of simultaneous users" of the application). However, since the production environment and workload of the application is most often different from the test environment, we need performance *models* which can help *estimate* application performance metrics, using mathematical or simulation methods.

The complexity of modern multi-tier networked server applications is such that performance modeling cannot be done manually. Hence several *modeling tools* have been developed, which accept a high-level description of an application and its hardware environment, and produce estimates of the application's performance metrics.

PerfCenter [1] is one such tool, which takes as input the description of the hardware, software and network architecture of a server application and its deployment, and produces estimates based on queuing analysis or discrete event simulation. PerfCenter offers an intuitive specification framework from the point of view of a "data center" operator who needs to host this application in an efficient manner. PerfCenter capabilities, along with examples of how it can be used for modeling the performance of an application, were described in detail in a previous paper [5]. In that paper, the authors show how PerfCenter is used for making configuration decisions such as the number of CPUs in a host, the number of hosts allocated to a server, or to determine the optimal number of threads that a server should be configured with.

In the recent years, *energy* has emerged as a valuable resource that is consumed by a data center. Studies have shown that power costs dominate data center operating costs [4]; hence several power optimization technologies have emerged and are being used in data center hardware. The impact of these technologies can be understood, and design choices analyzed, only if we have modeling tools that can predict energy usage by applications deployed in such a data center.

Energy optimization in server computing has been mainly done by using devices whose power usage can be regulated by manipulating their speeds. "Frequency-scaled CPUs" [6] are an example of a device whose operating frequency (and thus power drawn) can be changed dynamically. Frequency scaled CPUs have become standard in server-grade computers, thus performance modeling tools need to be capable of predicting performance in the presence of such CPUs and other power-managed devices.

In this paper, we present *PowerPerfCenter* - an enhanced and extended version of PerfCenter which allows for specification of *power managed devices* in host machines. It also

*Work-In-Progress Paper

†This research is partially supported by a grant from Tata Consultancy Services, Mumbai, India

ICPE'14, March 22–26, 2014, Dublin, Ireland.

http://dx.doi.org/10.1145/2568088.2576758.

```
host h1[1]
corei5 count 1
corei5 buffer 99999
corei5 schedP fcfs
corei5 power_managed
 governor conservative
disk count 1
disk ...
end

host h2[1]
  ...
end
task send_to_db
  corei5 servt 0.05
end
task to_html_login
  corei5 servt 0.236
end
task get_creds
 core2duo servt 0.003
end
```

```
server web
thread count 150
thread buffer 0
thread schedP fcfs
task send_to_db
task to_html_login
..
end

server db
thread count 150
thread buffer 0
thread schedP fcfs
task get_creds
..
end

loadparams
  thinktime exp(6)
end
```

```
device     lan
corei5     lan1
core2duo   end
disk
end

deploy h1[1] lan1
deploy h2[1] lan1

deploy web h1[1]
deploy db h2[1]
```

```
modelparams
  method
simulation
  type closed
  confint true
  simendtime 6000
  replicationno 5
end
```

```
scenario Login prob 0.18846
  send_to_db get_creds 200 SYNC
  get_creds to_html_login 100
end
```

Figure 1: PerfCenter Input File Snippets

allows specification of a *dynamic workload*, which highlights the impact of power management. It estimates performance metrics in such a scenario, and additionally estimates various *power consumption metrics* that can help analyze the "power-performance trade-off". To our knowledge, *PowerPerfCenter* is the only "software performance modeling" tool that is capable of modeling power-managed devices.

The rest of the paper is as follows: in Section 2 we recap the basic PerfCenter model, and present the details of the power-related enhancements in PowerPerfCenter. In Section 3 we present validation results and an analysis of the power-performance trade-off for the "WebCalendar" application. We conclude the paper in Section 4.

2. POWERPERFCENTER

The PerfCenter system model captures the essential components of a multi-tier application deployed in a data center that are required to analyze its performance. It consists of a set of machines, or `Host`s, which are an aggregation of `Device`s (such as CPU, disk, RAM etc); a set of `Server`s (e.g. Web server, database server) which are described by attributes such as number of threads, and the set of `Task`s that they perform. These servers together constitute a multi-tier application that can be used to fulfill various user requests, which we call `Scenario`s. A scenario is described by a probabilistic call graph of `Task`s that are carried out by various servers to fulfill a request. The calls can be described as "synchronous" or "asynchronous", and are specified with the size of data that is exchanged when the call is made. `Task`s are specified with service time requirements on various `devices`. `Server`s are "deployed" on `Host`s and `Host`s can be deployed on a `LAN`. Finally, two `LAN`s can be connected via a point-to-point WAN `Link`, which is described by its bandwidth and propagation delay.

The workload model is as follows: scenarios are specified with arrival probabilities, and the load is specified with either an open arrival rate, or a number of users and think time, which implies a closed system. Figure 1 shows snippets of a very basic PerfCenter input file.

With this specification, PerfCenter generates the underlying queuing network model and solves it using discrete event simulation. PerfCenter can report various performance metrics such as response time, throughput, utilization etc. at the device, scenario, server and system level. We refer the reader to the earlier paper on PerfCenter[5] for further details on the existing capabilities of PerfCenter.

2.1 Power Managed Devices in PerfCenter

Many devices in a computer today are able to dynamically reduce power usage, when peak power is not required. This is especially applicable to two of the most commonly used devices, namely, CPU and disk. In case of modern CPUs, power saving is achieved by clocking the CPU at a lower rate (through Dynamic Voltage and Frequency Scaling (DVFS) [6]), resulting in effectively *slowing* it down. Hard Disk Drives save power by slowing the spin rate, i.e. spinning down its platters [4]. The power consumed is a function of the operating speeds of these devices.

There are several models that relate the CPU operating frequency, to the power consumed by a machine. A widely accepted model is where power is assumed to have a *static* and a *dynamic* component [6]. The static power component represents the power that is consumed by the powered-on but idle device. The dynamic component is a function of the instantaneous utilization of the device. Also, both static, i.e. idle, and dynamic power depend on the instantaneous operating speed of the device. PowerPerfCenter's power model assumes that when the operating speed of a device is s, and its utilization is ρ, the power consumed is given by $\gamma_i(s) + \rho \times \gamma_d(s)$, where $\gamma_i(s)$ is the idle power, and $\gamma_d(s)$ is the maximum additional power consumed by the device while operating at speed s and at full utilization ($\rho = 1$).

Every device in PowerPerfCenter can optionally be declared as a power-managed device. For a power-managed device, PowerPerfCenter expects the specification of operating speeds (s), idle power consumptions $\gamma_i(s)$, and the maximum dynamic power consumptions $\gamma_d(s)$.

Power-managed devices are controlled by a "governor" which decides when it should change the operating speed and by how much. Currently, PowerPerfCenter's power-managed device abstraction is modeled after frequency-scaled CPUs, and hence it offers four basic governors that are found in a typical Linux implementation [2]: `powersave, performance, ondemand and conservative`. Of these `powersave` represents a static governor which fixes the operating speed to the lowest. `performance` fixes the operating speed to the highest. `ondemand` and `conservative` change the frequencies dynamically. Each of these governors probes the CPU after a fixed duration of time, called `probeinterval`. At each probe, the utilization of CPU in the previous interval is seen. If it is above (below) a threshold, called `up (down) threshold`, then the governor decides to increase (decrease) the frequency. The `ondemand` governor increases the frequency to the highest, and decreases to the next lower step, when thresholds are crossed. The `conservative` governor changes frequency in steps when thresholds are crossed.

Power-managed devices in PowerPerfCenter are specified as follows:

```
powermanagement corei5
speed_levels 1.2 2.26 2.39 2.53 2.66 2.79 2.93
        3.06 3.19 3.19 end
power_consumed 90 122 127  135  140  150  159
```

```
        168  176  176 end
idlepower 56 56 56 56 56 56 56 56 56 56 end
probe_interval 0.080
governor_up_threshold 80
governor_down_threshold 20 end
```

Note that since there are multiple operating speeds of the device, the specified service demands of the `task`s that require such a device are now assumed to be corresponding to the lowest speed of the device.

2.2 Energy Metrics in PerfCenter

A performance analysis of an application running on power-managed devices would be incomplete without estimation of the power consumed by these devices. Thus, PowerPerfCenter offers the following additional metrics to be evaluated when power-managed devices are used: the *average power* consumed by each instance of a power-managed device, the *energy consumed per request*, and the *power efficiency*, which is defined as requests served per unit of energy. Using these metrics, and the existing performance metrics, the power-performance trade off can be effectively analyzed for different scenarios.

2.3 Dynamic Workload in PowerPerfCenter

Devices that can dynamically change their operating speeds and thus power consumed, are useful primarily in a scenario where the *workload* itself is dynamic. Thus, this modeling capability is useful only if we can specify a time-varying workload pattern.

PowerPerfCenter allows the specification of a dynamic workload profile, which is then implicitly *repeated* if the simulation time is longer than the duration of the workload profile. The workload is defined as a set of tuples - each tuple corresponds to a load level and the duration of this load level. The exact specification is as follows:

```
workload cyclic
noofusers 25  35  45  35  30  15  20  end
interval  300 300 300 300 300 300 300 end end
```

When dynamic workload is specified, the performance metrics are produced separately for each load level.

3. RESULTS AND VALIDATION

For illustrating the use of PowerPerfCenter and validating its prediction, we used the "Webcalendar" application deployed on a testbed. Webcalendar is a standard two-tiered calendaring application, provided using a Web server and a database server. It supports various use cases such as Login, ViewDay, ViewWeek, ViewMonth, ViewEvent, etc. The Web server was hosted on a machine with the Intel Core i5 650 processor, 4 GB RAM and 1 TB HDD. The database server was hosted on a machine with the Intel Core2 Duo E4500 2.2 GHz processor, 2 GB RAM and 160 GB HDD. We used the Web server machine to experiment with performance in presence of CPU frequency scaling. The Intel Core i5 in this machine supports frequency settings in the range of 1.2 GHz to 3.19 GHz. The idle power consumption (γ_i) of this CPU as reported in [8] is the same (56W) at all frequencies. The dynamic component (γ_d) of the overall power does depend on the operating frequency and is given in the results in [8].

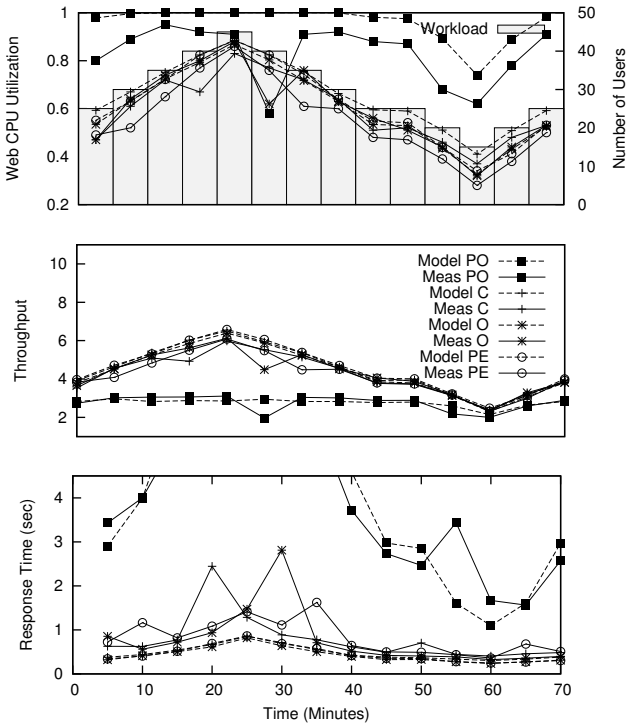

Figure 2: Measured (solid lines) vs model (dotted lines) metrics for the Webcalendar application for the four governors. Abbreviations: PO: Powersave, C: Conservative, O: Ondemand, PE: Performance. The legend shown is common to all graphs.

This testbed was populated with a dataset of 5000 distinct users with their own calendars. Each user's calendar was populated with 100 random events over the year 2013.

We carried out load tests on this deployment and generated graphs of average scenario response times, throughputs and server CPU utilizations as a function of the number of users. We used a load generator (*AutoPerf* [7]) which is capable of generating requests probabilistically, and also reports server statistics at each load level. The load generator machine had the Intel Xeon E5405 2 GHz processor, 8 GB RAM and 1 TB HDD. All the machines were connected via a single 100 Mbps switch.

This application and its deployment was then specified in PowerPerfCenter. Most of the architectural details are readily available. For obtaining the service times of the tasks, we used *AutoPerf*'s server profiling capability. This gives the service demand of each type of request on the Web server and database server CPUs and other resource usage data such as disk and network bytes read and written per request by each host. This completes the system specification.

We tested and modeled this application for a dynamic workload of varying number of users, each with an exponentially distributed think time of 6 seconds, and a pre-specified scenario generation probability. Measured vs model predicted values of the *performance* metrics were compared for various settings of power governors at the Web server. Figure 2 shows the comparison results.

The results show that the performance, ondemand and conservative governors are able to support the load, while the powersave governor runs out of capacity at just 15 users. Furthermore the performance metrics of the ondemand and

conservative governors are almost indistinguishable from those of the performance governor. This might be because as the load increases, the dynamic governors speed up the CPU quickly to reach the level of the performance governor. After the CPU is at its highest frequency, it can be seen from Figure 2 that load does not become low enough for the CPU utilization to cross the low threshold of 0.2, so the dynamic governors do not reduce the CPU frequency, and continue to operate at the level of the performance governor.

PowerPerfCenter is able to predict the throughput and utilization metrics quite accurately. For these metrics, more than 80% of the values had relative error of less than 20%. In case of response time, the match is good for the powersave governor (higher values are not shown in the graph). In case of the other governors, the measured values of the response times were highly variable. Nonetheless, the overall trend is predicted well by PowerPerfCenter.

3.1 Power Performance Trade-Off

The conservative, ondemand and performance governors have fairly comparable performance. Figure 3(a) shows the average response time and the average power consumed with the Core i5 in the Web server host for the four governors. Note that in this graph, the values are as predicted by PowerPerfCenter - we do not have measurement results for power consumption. Workload-1 (w1) is as shown earlier in Figure 2. To study the effect of a more "skewed" workload with a short-lived peak and long-lived low load period, we also analyzed the power-performance trade-off for another workload (Workload-2 (w2): 75 users for 60 seconds, and 3 users for 600 seconds) on the same setup.

In case of both these workloads, we see (from Figure 3(a)) that the powersave governor achieves power savings at the cost of a high response time. But the performance governor consumes only slightly more energy than the ondemand and conservative governors. This happens because for Core i5, the idle power consumption is the same at all operating speeds [8]. The performance governor, because of its high frequency setting, results in low average CPU utilization (especially for Workload-2) and thus the CPU is idle most of the time. Consequently, the difference between average power consumption by the performance governor and the ondemand or conservative governors is not significant.

To understand this further, we carried out a "what-if" analysis by specifying a different CPU in the Web server host model, whose power consumption figures were based on experiments reported in [3] on the Intel Core i7 processor. This processor has markedly different idle power consumption at different operating frequencies. Figure 3(b) shows the response time and power consumption values as predicted by PowerPerfCenter in this case. We can now see the advantage of using the dynamic governors - the ondemand and conservative governors use 17% less power and 40% less power than the performance governor in case of Workload-1 and Workload-2 respectively, with a negligible increase in response time.

4. CONCLUSIONS AND FUTURE WORK

Energy usage by data centers has become a matter of concern in today's world, and has thus resulted in usage of devices that intelligently optimize power by tuning their operating speeds. In this paper, we presented a tool *PowerPerfCenter* which can predict application performance metrics

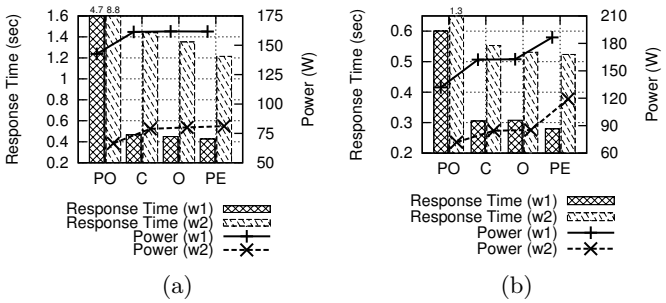

Figure 3: Power-Performance Trade-off for (a) Core i5 and (b) Core i7

in presence of such devices, and which can further estimate power consumption of such devices, for *dynamic* workloads. Comparison of performance metrics predicted by this tool against those obtained from measurements showed a good match. The estimated power usage metrics showed how savings in power usage are significant when the *idle power consumption* of a device is lower at a lower speed. If idle power consumption is the same for all speeds, there may not be much gain in running dynamic speed governors on a device. PowerPerfCenter can provide such critical insights into the power-performance trade-off.

5. REFERENCES

[1] PerfCenter: A Datacenter Application Performance Modeling Tool. `http://www.cse.iitb.ac.in/panda/perfcenter`.

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

[3] Power governors, documentation of linux kernel 2.6.32. `http://www.xbitlabs.com/articles/cpu/display/power-consumption-overclocking_13.html#sect0`.

[4] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *IEEE Computer*, 40, 2007.

[5] A. Deshpande, V. Apte, and S. Marathe. Perfcenter: a performance modeling tool for application hosting centers. In *Proceedings of WOSP 2008*, pages 79–90, New York, NY, USA, 2008.

[6] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar. Critical power slope: understanding the runtime effects of frequency scaling. In *Proceedings of the 16th ACM International Conference on Supercomputing*, pages 35–44, 2002.

[7] S. S. Shirodkar and V. Apte. Autoperf: An automated load generator and performance measurement tool for multi-tier software systems. In *Proc. of the ACM WWW 2007*, pages 1291–1292, New York, NY, USA, 2007.

[8] S. P. Srinivasan and U. Bellur. A novel power model and completion time model for virtualized environments. Technical Report TR-CSE-2014-58, Department of CSE, IIT Bombay, Mumbai, India, January 2014.

Acknowledgements

The authors wish to thank Yogesh Bagul and Rakesh Mallick for their contribution to PowerPerfCenter development.