Power and Performance Modeling in a Virtualized Server System

Massoud Pedram and Inkwon Hwang
University of Southern California
Department of Electrical Engineering
Los Angeles, CA 90089 U.S.A.
{pedram, *inkwonhw}@usc.edu*

Abstract—**Virtualization has become a very important technology which has been adopted in many enterprise computing systems and data centers. Virtualization makes resource management and maintenance easier, and can decrease energy consumption through resource consolidation. To develop and employ sophisticated resource management, accurate power and performance models of the hardware resources in a virtualized environment are needed. Based on extensive experiments and measurements, this paper presents accurate power and performance models for a high performance multi-core server system with virtualization.**

Keywords- modeling; energy efficiency, virtulization, consolidation, resource sharing, scheduling.

I. INTRODUCTION

The concept of virtualization was introduced a few decades ago, but it was not widely used because it required high performance hardware. However, as hardware performance has improved, virtualization has emerged as a most promising solution for eliminating "computing waste" through physical resource sharing in enterprise computing server systems in general and data centers in particular. Moreover, virtualization makes system maintenance much easier, improves system availability, and reduces the cost of managing/upgrading a large computer system.

Modern data centers not only consume huge amounts of energy, but also their energy needs are increasing very quickly. For example, according the 2007 EPA report[1,2], data centers consumed 61 billion kWh of electrical energy in 2006 and they are on track to consume more than 110 billion kWh in 2011. By the 2020, the data center energy consumption s expected to account for 8% of the electrical energy consumption in the US (and this in spite of continuous improvements in energy efficiency of the newer servers). Fortunately, the average (server) utilization of a typical data center is quite low, so a suitable server management strategy can significantly reduce the data center energy consumption. This is where virtualization enters the picture by allowing each application/client to think it owns the physical resource while in fact that resource is being shared among multiple applications/clients.

An effective policy to reduce the energy cost of data centers is thus by representing each process (which itself is a sequence of tasks) as a virtual CPU (vCPU) and subsequently packing these virtual CPUs into a small number of physical CPUs (pCPU). Of course, the task-level performance constraints (e.g., the turn-around time) imposed by the application/client that generates these tasks must also be met (these are typically formulated in a set of service level agreements between the clients and the data center operators/owners. Regardless every incoming task should be eventually serviced ("fairness" criterion.) To do this mapping from virtual to physical CPUs so as to minimize energy consumptions while meeting various SLAs is a challenging task of immense value to the data center owners. Indeed solving this problem in a cost-efficient and manageable manner will accelerate widespread adoption of cloud computing at least according to the "infrastructure as a service" model.

To develop a provably "good" solution to this hard constrained optimization problem, one will need a better understanding of the power-performance tradeoffs in a virtualized computer system. Questions such as how many virtual domains are needed, what should be the target utilization level of a physical CPU[4,5], how does one related the workload intensity to the server utilization levels, how do we map the vCPU's to pCPU's, what performance level should be set for a given physical CPU, etc. can be answered only if we have quantitative models relating the level of virtualization to power dissipation and performance of the physical CPUs. Instead of answering these questions using simulators, we have opted to rely on actual hardware measurements on two-server computer system, each server comprising of two sockets and each socket having two cores. We run a number of application programs (some synthetic some from the SpecWeb benchmark) to do the experiments, make the measurements, and finally construct our power and performance macro-models. In the process, we provide some insight as to what level of virtualization makes sense for a target enterprise computing system.

We have designed 18 test cases with different configurations, in terms of the number of vCPUs, the set of active pCPUs, and frequency settings for these pCPUs. Power and performance macro-models are obtained from experimental results by regression analysis. Determining the proper number of vCPUs is important: A small number of virtual CPU limits the number of pCPU resources that can be utilized by the system and hence results in power and performance penalty while a large number vCPU's creates extra overhead which again results in power and performance penalty.

To avoid confusion (previous authors have used same terms to refer to different things), we provide some key definitions and terminology that will be used throughout this paper.

- Processor: A package having cores and caches in it
- (physical) CPU[1]: A physical core in a processor
- virtual CPU: A process
- Domain0: Privileged domain
- Guest domain: All other domains in the virtualized system
- Virtual machine: Same as a domain
- CPU utilization: Total of utilization of all CPUs in the system.

II. Experimental System Under Test

A. Hardware Testbed

Our testbed system has the following configuration. We have two processors in the system, which are Intel Xeon E5410 processors. Each processor has four cores and total size of the system memory is 8GBytes. Each processor supports two frequency levels, 2.0GHz and 2.3GHz and its VID voltage range is 0.85V to 1.35V. Fedora version 11 (Linux) is used as an Operating system for domain0. We used XEN hypervisor [8] version 3.4.2 for constructing virtualized system. We cut the power line between the main board and 12V DC power source and measure the power consumed by the processors and on-board voltage regulators. Figure 1 shows photos of our setup in the lab.

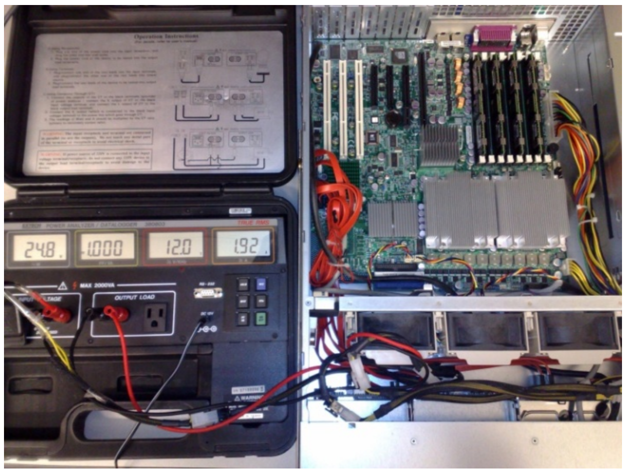

Figure 1. The server system under test and the power analyzer.

[1] CPU means physical CPU unless it is clearly mentioned as virtual CPU

B. WorkloadGen

We designed and implemented a workload generator, *WorkloadGen*, for modeling and measuring performance of the system. The WorkloadGen utility provides parameters for controlling the *workload type*, e.g., CPU, Memory, or I/O intensive, *workload intensity*. By adjusting the input parameters of this program, we can set any workload type and intensity combination. Note that once the task type is fixed, all generated tasks are homogenous and what we do to set the workload intensity is to simply change the *task arrival rate*. There are two module types as depicted in Figure 2. Each *Task Generator* makes tasks that satisfy a target type and intensity profile, and sends them to a *Task Loader*. The *Task Loader* resides in the SUT (System under Test), and loads the tasks sent by *Task Generators*. Task generators and the task loader communicate each other though TCP/IP protocol, so they can locate in physically separated machines.

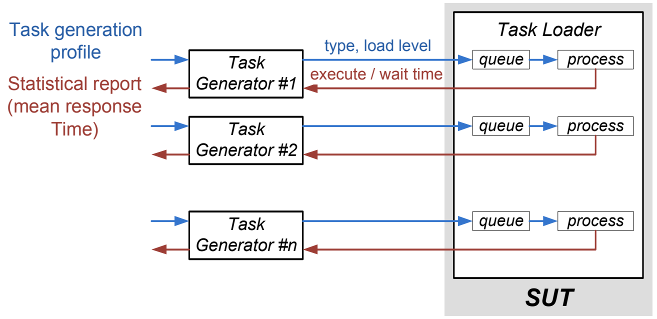

Figure 2. Workload generator block diagram.

Since there is more than one task generator, we also need a *task loader*, which creates a process for each task generator. For example, if three *task generators* connect to a *task loader* and send tasks, the *task loader* creates three processes for serving them; in such a case at most three vCPUs can be concurrently utilized. The *WorkloadGen* reports back statistical performance data: average response time per task, average waiting time in the application queue, and average execution time per task, which are needed for analyzing the overall system performance.

III. Experimental Results

Power dissipation and response time is measured for different configurations. We model power and response time as functions of CPU utilization. In this study, we select response time (as opposed to say throughput) as the performance metric of interest.

A limitation of our experimental study is that we only consider one domain, which is the privileged domain, *domain0*. In general, there is more than one domain in a virtualized server system, and they are dependent on each other, e.g., all I/O between H/W and guest domains must pass through domain0. Hence, analyzing the relationship between domains is important, but it falls outside the scope of the present paper.

A. *Test cases*

There are four groups of test cases as shown at Table 1. The first group, cases 1 through 10, is for modeling power and response time of a single server. All cases in the first group have four vCPUs, but they differ in the frequency level used for the processor as well as the number of *active* CPUs.[2] The second group, cases 11 and 12, is studied for modeling power when different frequency levels are applied to sockets in the same processor. The third group, cases 13 and 14, is studied for exploring effects caused by using different number of vCPUs compared to cases 1 and 5, respectively. The last group, cases 15 through 18, is for power modeling when both processors are used in the virtualized server system.

Bold face L or H frequency annotation in Table 1 means that the corresponding CPUs are active. This does not, however, mean that the all such CPUs are executing tasks at all times; clearly some of the active CPUs may be idle. Non-bold frequency annotation in the table indicates the fact that the corresponding CPUs are idle. All idle CPUs are automatically transitioned to the C1E (extended halt) power saving mode. Intel core-level power management implemented in their power control unit (which is inaccessible to Xen) automatically transitions an idle core into this power saving mode after some timeout period.

One of our findings is that the power consumption of the two processors in the server system is largely dependent on the total utilization of the cores in the two servers, and the frequency setting of the active cores, and not so much on what subset of cores is used by the running domains

TABLE I. LIST OF TEST CASES AND GROUPS (SEPARATED BY DOTTED HORIZONTAL LINES).

Case	***# of vCPUs***	***ID of pCPUs***	***Processor 1***				***Processor 2***			
			0	***2***	***4***	***6***	***1***	***3***	***5***	***7***
1	*4*	*0,2,4,6*	***L***	***L***	***L***	***L***				
2	*4*	*0,2,4*	***L***	***L***	***L***	*L*				
3	*4*	*0,2*	***L***	***L***	*L*	*L*				
4	*4*	*0,4*	***L***	*L*	***L***	*L*				
5	*4*	*0*	***L***	*L*	*L*	*L*				
6	*4*	*0,2,4,6*	***H***	***H***	***H***	***H***				
7	*4*	*0,2,4*	***H***	***H***	***H***	*H*				
8	*4*	*0,2*	***H***	***H***	*H*	*H*				
9	*4*	*0,4*	***H***	*H*	***H***	*H*				
10	*4*	*0*	***H***	*H*	*H*	*H*				
11	*4*	*0,2*	***L***	***L***	*H*	*H*				
12	*4*	*0,4*	***L***	*L*	***H***	*H*				
13	*1*	*0*	***L***	*L*	*L*	*L*				
14	*1*	*0*	***H***	*H*	*H*	*H*				
15	*4*	*0,2*	***L***	*L*	*L*	*L*	***L***	*L*	*L*	*L*
16	*4*	*0,1*	***H***	*H*	*H*	*H*	***L***	*L*	*L*	*L*
17	*4*	*0,2*	***H***	***H***	*H*	*H*	*L*	*L*	*L*	*L*
18	*4*	*0,1*	***H***	*H*	*H*	*H*	***H***	*H*	*H*	*H*

[2] An active CPU is a pCPU which is utilized by some domain(s) or is ON ready to serve the domain(s).

Figure 3 depicts a simplified block diagram of Xeon 5400 series processor. Each processor has two sockets and four cores (CPUs). The two CPUs in the same socket share an L2 cache. Note that ID of cores in processor 0 is even numbered while those in the other processor are odd numbered. This numbering is used in order to be consistent with how XEN assigns ID to CPUs.

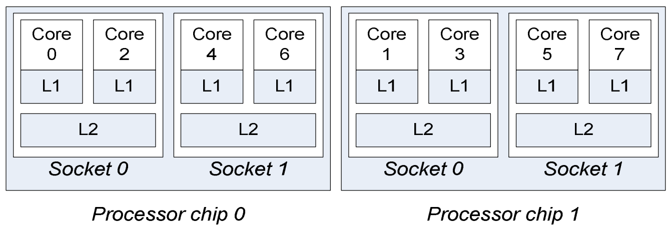

Figure 3. Simplified block diagram of Intel Xeon processor 5400 series (code named Hapertown).

B. *Power modeling*

For now we focus on the single server system (cases 1 through 14). As seen in Figure 4, the relationship between power and CPU utilization is linear. Hence we can model power dissipation as follows:

$$p = a \cdot u + b \qquad (1)$$

where p is power in Watts and u is the total (not average) CPU utilization. Note that the maximum utilization of 2 CPUs is 200%, and so on.

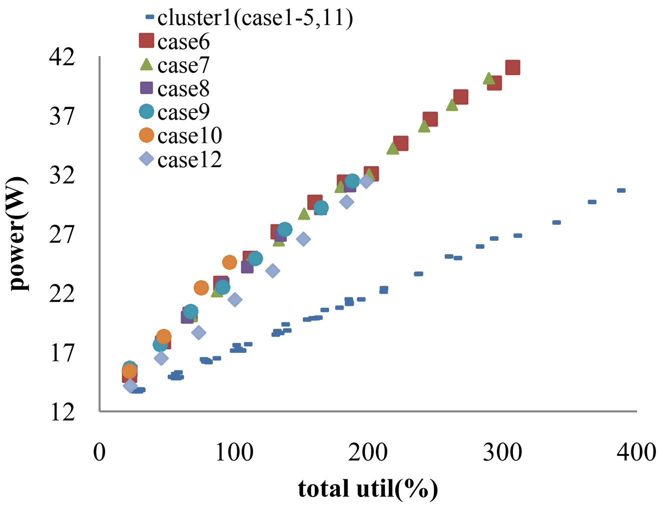

Figure 4. Server power dissipation vs. total CPU utilization

These cases can be divided in two clusters as shown in Figure 4. All cases in cluster1 contain active CPUs that run at low frequency while cluster2 contains cases with active high frequency CPUs.[3]

Based on our experimental results, we state a few interesting facts.

- When all active CPUs are running at the low frequency level, the processor power dissipation is mostly independent of which subset of CPUs is used by the running domains (see cases 1 through 5, and 11.)
- When all active CPUs are running in high frequency, the processor power dissipation is dependent on which subset of CPUs is used by the running domains (see cases 6 through 10.)
- When active CPUs are running at different frequency levels, the processor power dissipation is similar to that of the case which has the same number of active CPUs all running at the high frequency level (see cases 9 and 12.)
- The idle power dissipations are nearly the same for the two frequency levels.
- Two CPUs that are in the same socket run at the same frequency level, regardless of our setting. This, for example, means that if one CPU is set to the low frequency while the other set to the high frequency, both are actually running at the higher frequency.[4]

Table II below reports the regression coefficients (*a* and *b*) for the power macro-model equation (1) for all cases. The table also gives the coefficient of determination, R_{squared}, which provides a measure of how well future outcomes are likely to be predicted by the linear regression model. An R_{squared} value close to 1 indicates a good match between the linear fit and actual data.

TABLE II. COEFFICIENTS OF POWER MACRO-MODEL EQUATION FOR CASES 1-12, 15-18

case	*a*	*b*	$R_{squared}$	***case***	*a*	*b*	$R_{squared}$
1	*0.046*	*12.65*	*0.998*	***9***	*0.096*	*13.62*	*0.997*
2	*0.046*	*12.65*	*0.999*	***10***	*0.127*	*12.48*	*0.996*
3	*0.047*	*12.39*	*0.999*	***11***	*0.047*	*12.38*	*0.997*
4	*0.049*	*12.40*	*0.998*	***12***	*0.097*	*11.74*	*0.998*
5	*0.049*	*12.28*	*0.991*	***15***	*0.046*	*24.60*	*0.997*
6	*0.088*	*14.44*	*0.993*	***16***	*0.085*	*24.58*	*0.998*
7	*0.091*	*13.98*	*0.997*	***17***	*0.091*	*25.65*	*0.995*
8	*0.095*	*13.58*	*0.996*	***18***	*0.128*	*24.63*	*0.998*

Figure 4 and Table II show that, to a first order, the relationship between power dissipation and total utilization of low frequency cases (cluster1) is determined by the frequency level of the active CPUs regardless of the number and specific set of CPUs that are active in each socket. There are of course minor differences. For example, test case 4 has a larger *'a'* value (slope) than case 3. This is because two CPUs in the same socket share L2 cache. Test case 4 has two active CPUs that belong to different sockets (there is no L2 cache sharing), so it consumes more power than case 3 where the two active CPUs share the L2 cache in socket 0. In summary, the regression coefficients of the power vs. utilization equation for all cases of cluster1 are nearly the same. In contrast, there is a big variation in the coefficient values of the power vs. utilization equation for the high frequency cases (cluster2.) Table II shows that the higher the number of active high-frequency CPU's, the lower the slope ('a' value) of the power vs. utilization equation. This observation means, for example, that case 6 (with 4 active high-frequency CPUs) consumes less power than case 10 (only 1 active CPU) at high total CPU utilization level. Note that the apparent difference between power intercepts ('b' values) of these cases is due to the regression fit by a linear equation. The difference between coefficients of cases 8 and 9 is small, but can be explained based on the L2 sharing notion.

The power equation for the cluster2 cases thus includes another term which is the power consumption of shared resources. This term is proportional to the average utilization of the active CPUs. The other terms are proportional to the number of active CPUs. Consequently the new power equation for the high frequency cases may be written as follows:

$$p_H = (\alpha_1 \cdot n + \frac{\alpha_2}{n} + \alpha_3) \cdot u + (\beta_1 \cdot n + \frac{\beta_2}{n} + \beta_3) \tag{2}$$

where p_H is power in Watts, u is the total CPU utilization, and n is the number of active CPUs. The coefficients for our testbed have been computed as follows: $\alpha_1 = 0.0045$, $\alpha_2 = 0.0691$, $\alpha_3 = 0.0532$, $\beta_1 = 0.199$, $\beta_2 = -1.759$, and $\beta_3 = 14.050$

As table II shows we can use a unified equation for all cases in cluster1. For cluster2, slope ('a') and power intercept ('b') are calculated from equation (2).

TABLE III. COEFFICIENTS OF THE UNIFIED POWER MACRO-MODEL EQUATIONS FOR CLUSTERS 1 AND 2

	case	*a*	*b*	$R_{squared}$
cluster1	*1-5,11*	*0.047*	*12.55*	*0.998*
	6	*0.089*	*14.41*	*0.993*
	7	*0.090*	*14.06*	*0.996*
cluster2	*8*	*0.097*	*13.57*	*0.996*
	9	*0.097*	*13.57*	*0.998*
	10	*0.127*	*12.49*	*0.996*
	12	*0.097*	*13.06*[5]	*0.953*

Cases 15 to 18 are for checking whether or not we can extend the power model got from cases 1 to 10 to multi processor systems. Note that the offset coefficients ('b' terms) in these equations is not a simple summation of the offset terms of the corresponding single-server macro-models. Neither is the slope ('a' term) an average of the slopes of corresponding single-server systems.

[3] If any active CPU is running at the high frequency, the corresponding case is put in cluster2.

[4] This observation is from experimental results (although not reported in table I.)

[5] Case 12 is a mixed frequency case, i.e., the power intercept is the average of power intercept of cluster1 and that of case8 in table III.

Table IV compares the (measured) coefficients obtained through regression analysis performed directly on data obtained for the two-server system versus (estimated) coefficients obtained by taking average of the 'a' terms and summation of the 'b' terms of the corresponding single-serve data from Table II. As shown in Table IV, estimated one is not identical to measured one, but the difference is not large; the maximum error is less than 9%. Measured power includes power consumed by voltage regulator, so a plausible reason for this difference is that the voltage regulator conversion efficiency changes based on its load current demand. Another possible reason is error in measurement.[6] In summary, we can use the coefficients in Table II for estimating power dissipation of multiprocessor systems with a relatively small error.

TABLE IV. ESTIMATED VS. MEASURED COEFFICIENTS

case	*estimated*		*measured*	
	a	*b*	*a*	*b*
15	*0.049*	*24.80*	*0.046*	*24.60*
16	*0.088*	*24.76*	*0.085*	*24.58*
17	*0.095*	*25.86*	*0.091*	*25.65*
18	*0.127*	*24.96*	*0.128*	*24.63*

C. *Response time modeling*

In our experiments, inter-arrival times of tasks follow exponential distribution, so, the average amount of time a task stays in the system is as follows [5]:

$$W = \frac{\lambda E[S^2]}{2(1-\lambda E[S])} + E[S] \tag{3}$$

where λ is inter-arrival time, S is a random variable representing the service time, and E[.] denotes the expected value operation. Because relationship between the inter-arrival time and utilization is approximately linear,[7] we can use equation (4) for modeling the system's response time for a task:

$$r = \frac{c \cdot u}{d-u} + e \tag{4}$$

where *r* denotes response time and *u* denotes the total CPU utilization.

Table V shows regression coefficients of the response time equation for test cases. Values of R_{square} establish that equation (4) is quite accurate for modeling the system's response time.

[6] Our current power analyzer reports power dissipation only twice per second.

[7] This statement does not hold true under very high utilization rate, e.g. 95% or higher for a single CPU. Because such a situation should be avoided in any case (due to exponential increase in systems response time), we do not consider it here.

TABLE V. COEFFICIENTS OF RESPONSE TIMEC EQUATION

case	*c*	*d*	*e*	$R_{squared}$
1	*0.013*	*404.5*	*0.045*	*0.997*
2	*0.007*	*290.0*	*0.048*	*0.997*
3,4	*0.014*	*199.4*	*0.044*	*0.991*
5	*0.053*	*109.0*	*0.038*	*0.996*
6	*0.012*	*407.5*	*0.042*	*0.998*
7	*0.009*	*294.0*	*0.042*	*1.000*
8,9	*0.017*	*204.9*	*0.036*	*0.949*
10	*0.058*	*111.1*	*0.029*	*0.989*

Figure 5 depicts response time for test cases 1 to 10. As expected, the maximum utilization is limited by the number of active CPUs times 100%. However, the trends for all cases are similar i.e., the magnitude of change in response time is small under low utilization rates, but it increases exponentially as the utilization reaches the maximum allowed utilization.

Limiting the number of CPUs assigned to a domain (alternatively, guaranteeing a pre-specified total CPU utilization percentage for a virtual domain) is quite useful and common in virtualized systems because it enables performance isolation among the different virtual domains in the system. Hence, how much resource a domain consumes compared to total amount of resources assigned to the domain is an important parameter.

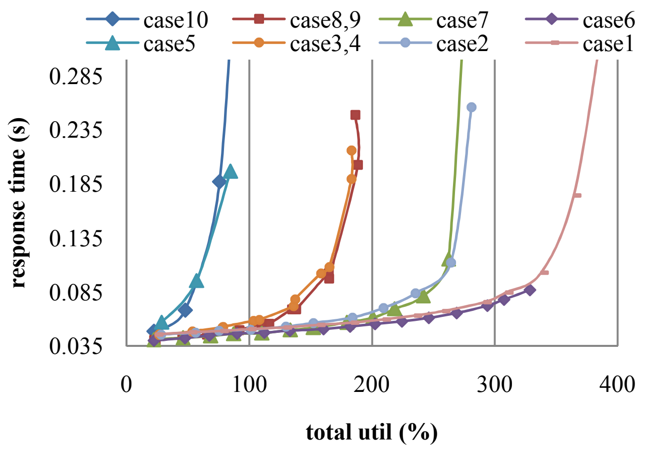

Figure 5. Response time vs. total CPU utilization.

For this purpose, we introduce a new term, *normalized utilization*, which is the total utilization of the CPUs in a given domain divided by the CPU quota (in percentage) allocated to the domain. The range of normalized utilization is between 0% and 100%.

Figure 6 redraws the response time data as a function of the normalized utilization. The response time plots are now very similar to each other except cases 5 and 10, which are using only one active CPU. This is expected. Since in both cases, there is only one active CPU and 4 vCPU's and the overhead of managing these vCPU's becomes large.

A general approach for resource management in enterprise computing systems is to keep the normalized utilization under a pre-defined upper bound in order to guarantee a specified quality of service [9]. Figure 6 proves that *we cannot use a fixed upper bound for all cases*. For example, suppose the maximum acceptable response time is set at 0.1s. Then we can push the CPUs up to a normalized utilization level of 85% in case 6 without violating this performance constraint, but only up to 60% utilization level in case 10. The reason is that case 10 has only one active CPU, and hence, the four vCPUs are executed on that CPU in a time-multiplexed manner. This creates certain overhead that will start to dominate the average response time per request as the number of context switches between the vCPU's increases (which itself happens because of the increase in the number of requests generated by each vCPU). Similarly, the upper bounds for case 5 and case 1 are 58% and 84% normalized utilization levels, respectively.

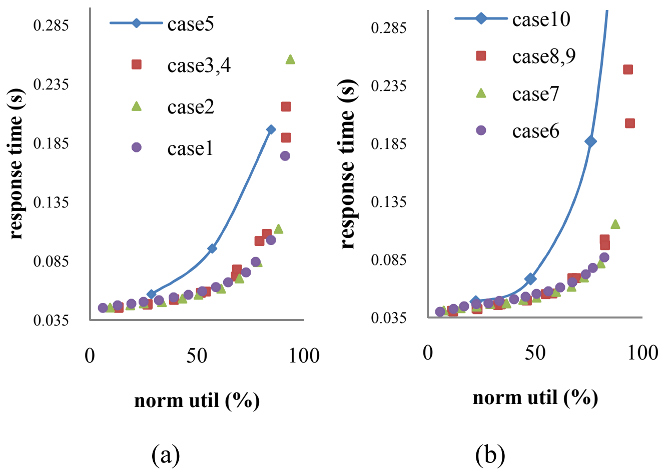

Figure 6. Response time vs. normalized utilization: (a) Low frequency - 2GHz, (b) High frequency - 2.3GHz

To further justify the explanation provided in the last paragraph for the difference among these cases, consider other results. Figure 7 shows the effect of the number of vCPUs on the response time vs. utilization curves. Cases 13 and 14, which also have only CPU but also only one vCPU, exhibit response time curves that always lie below those of cases 5 and 10, which have one CPU but four vCPUs. Clearly the response time curves for cases 13 and 14 lie below those of cases 5 and 10, that is, for a fixed number of active CPUs, the fewer vCPUs we have, the fewer context switches occur, and hence, the shorter is the response time.

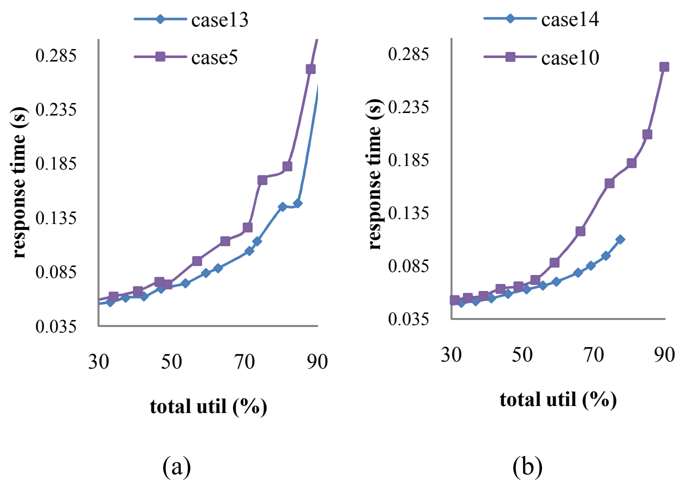

Figure 7. Response time per request for different number of vCPUs: (a) Low frequency-2GHz, (b) High frequency-2.3GHz

D. *Energy-Delay Product for different workload intensities*

As shown in Figure 4, idle power consumption is not zero.[8] Consequently, the *energy cost*, which is defined as average energy consumption per request, decreases as the workload intensity increases (see Figure 8.) Similarly, the average energy consumption per request decreases as the total CPU utilization increases (see Figure 9.) However, under high workload intensity, the response time also increases with the workload intensity (this increase accelerates as the CPU utilization approaches a threshold as seen in Figures 5 and 6). We thus use the energy-delay product (EDP) per request as a cost function to be optimized.

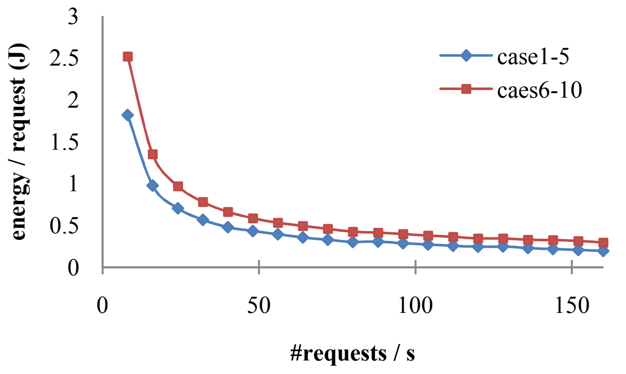

Figure 8. Energy per request vs. the workload intensity (# of requests per second).

[8] In fact a server's idle power consumption is quite high – this effect which is present in all of today's servers (including the Xeon 5400 series used here) is widely acknowledged and referred to as "energy non-proportionality."

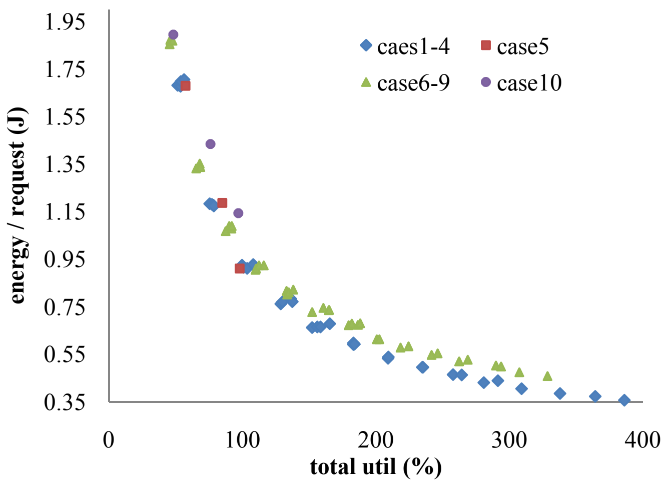

Figure 9. Energy per request vs. utilization.

Figure 10 shows that different cases are *EDP-optimal* for different levels of workload intensity in terms of our cost function. This experimental data motivates the need for dynamically changing the virtualization configuration in a server system. In Figure 10, for workload intensity 105 or more requests per second, case 6 is the best choice in terms of its EDP per request value. The lowest EDP value per request is achieved with case 1 when the number of requests per second is between 50 and 100, and so on. This shows that the dynamic configuration change in a virtualized environment can be quite beneficial in terms of minimizing the EDP value per request.

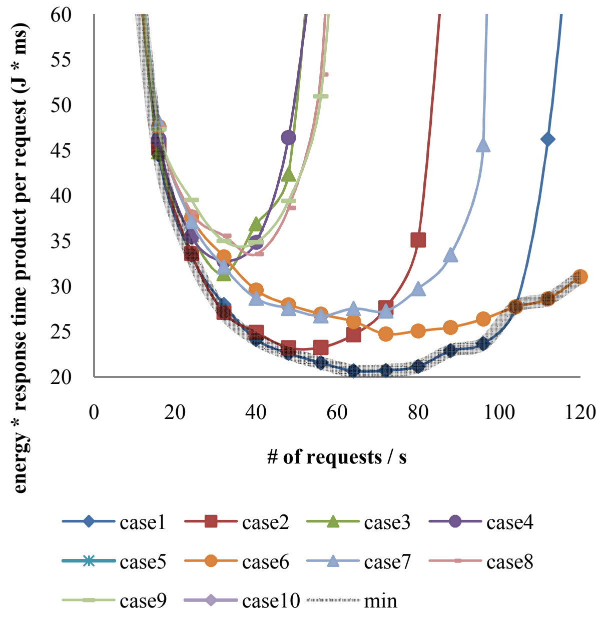

Figure 10. 'Energy delay product' / request vs. # of requests

IV. Conclusion

We present precise power and performance models for a virtualized server system based on Intel Xeon 5400 series. Power dissipation model (as a function of the total CPU utilization) of the virtualized system is similar to one of non-virtualized systems. However, the configuration of virtual machine affects the performance metric i.e., the number of virtual CPUs and physical CPUs greatly influence the system's overall performance.

References

[1] J. G. Koomey, "Estimating total power consumption by servers in the US and theworld. Final report," http://enterprise.amd.com /Downloads/svrpwrusecompletefinal.pdf, 2007

[2] U.S. EPA, "Report to congress on server and data center energy efficiency," U.S. Environmental Protection Agency, Tech. Rep., Aug. 2007.

[3] L. Barroso and U. H¨olzle, "The case for energy-proportional computing," IEEE Computer, Jan 2007.

[4] J. Heo, X. Zhu, P. Padala, and Z. Wang, "Memory Overbooking and Dynamic Control of Xen Virtual Machines in Consolidated Environments," The 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 09) (Mini-Conference), NY, June 2009

[5] G. Khanna, K Beaty, G. Kar, A. Kochut, "Application performance management in virtualized server environments. In: Proc. of the IEEE Network Ops. and Mgmt. Symp., pp. 373-381, Apr 2006.

[6] S. Ross, "Introduction to Probability Models", academic press, 2006

[7] XEN credit scheduler, http://wiki.xensource.com/xenwiki/Credit Scheduler

[8] XEN hypervisor, http://www.xen.org

[9] Y. Wang, X. Wang, M. Chen, and X. Zhu, "Power-efficient response time guarantees for virtualized enterprise servers," in RTSS, pp. 303–312, 2008