A Study of the Effectiveness of CPU Consolidation in a Virtualized Multi-Core Server System*

Inkwon Hwang and Massoud Pedram
University of Southern California
Los Angeles CA 90089
{inkwonhw, pedram}@usc.edu

Timothy Kam
Intel Corporation
Hillsboro OR 97124
Timothy.kam@intel.com

ABSTRACT

The focus of this paper is on dynamic power management in virtualized multi-core server systems. The paper starts by analyzing the effect of virtualization and CPU consolidation on power dissipation and performance (latency) of such systems, and concludes by presenting two new CPU consolidation algorithms for multi-core servers. The paper also reports an extensive set of experimental results founded on a realistic multi-core server system setup and well-developed benchmarks, i.e., *SPEC2K* and *SPECWeb2009* and obtained through hardware measurements.

Categories and Subject Descriptors

D.4.1 [**Process Management**]: Scheduling

Keywords

Energy efficiency, virtualization, consolidation, and scheduling

1. INTRODUCTION

Today's servers consume large amounts of energy so there is a growing need for energy-aware resource management in multi-core server systems. Virtualization has emerged as a promising solution for eliminating "computing waste" through physical resource sharing in data centers. In this study, we focus on the CPU (i.e., physical core), which is one of top energy consumers in a virtualized system. A common power saving technique for CPUs is *Dynamic Voltage and Frequency Scaling* (DVFS) [1-3]. However, the additional power savings possible through DVFS is becoming smaller and smaller in part due to lower supply voltage levels and a shared power and clock distribution network for the cores. In addition, it is not easy to apply existing DVFS techniques to cores in a virtualized multi-core server system. This is mainly because the existing DVFS techniques require information about running applications to make decisions about the voltage and clock frequency setting, but a virtual machine manager (hypervisor), which resides in a privileged domain, does not have information about applications running on the guest domains because of abstractions [4].

Another energy saving technique is *Dynamic Power Shutdown* (DPS). In particular, some modern CPUs support *Core-level Power Gating* (CPG), which allows individual core to be put into very low power, but non-functional, state. Such CPUs have their own *Power Control Unit* (PCU), which performs DPS; however, we expect that more power savings are possible if there is software level assistance for DPS. This is because the PCU in current servers does not have enough information about the application running on the system. Hence, there is a pressing need to understand when and how the CPU consolidation works with respect to power savings without violating performance constraints.

A common deployment model for high-end multi-core server systems is in data centers, which forms the computational and storage backbone of the digital information provided to end users via the Internet. As shown in Figure 1 , there is a large variance in workload intensity of commercial data centers [5]. To guarantee a required service level agreement (SLA) under the worst-case conditions, servers are typically designed to handle a peak workload condition even if the servers are under-utilized at times. There are very few hours that all servers in the data centers are running at their peak utilization levels. Motivated by this observation, many studies have suggested the use of *virtual machine migration* (VMM) for energy saving [3, 4, 6-8]. In theory, the VMM promises high energy saving, but it is difficult to apply the technique to servers in a data center because of the high overhead of the VMM, e.g., the large system boot time, network traffic caused by the need to transfer the running application and its local context to a new server, and so on. In general, the VMM does not aggressively address the server under-utilization because of its conservative nature (in order to avoid violating SLA.)

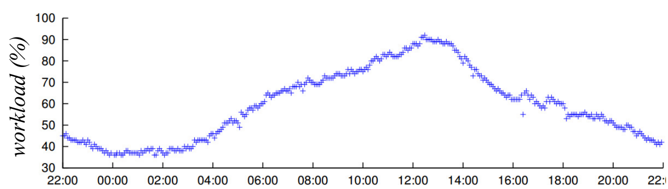

Figure 1 Typical server workload in Facebook data centers [5]

After an investigation of the effect of consolidation on the power dissipation and performance (latency) in virtualized multi-core server systems, this paper presents a consolidation technique for assisting DPS. There are a number of studies that have investigated energy savings due to consolidation. In [9], the authors showed that consolidation across cores in a single four-core-per-processor, two-processor-per-server system offers very small energy savings. However, their server system did not support CPG, so the energy saving potential of core-level consolidation needs to be investigated. In [10], Jacob *et al.* compared CPG and DVFS and showed that CPG can result in 30% energy saving compared to DVFS. However, the results were calculated from a combination of real measurements and estimated leakage power (the adopted leakage power model was quite simple.) In [5], the authors presented a technique called Core Count Management (some variant of the consolidation technique), and reported 35% energy saving. However, they reported both power and performance results based on simulations performed by using simple power and performance models. In our study, we show energy savings and

*This work is sponsored by grants from the National Science Foundation and the Intel Corp.

ISLPED'12, July 30–August 1, 2012, Redondo Beach, California, USA.

latency impacts of consolidation in a virtualized system based on realistic benchmarks and obtained through hardware.

Contributions of our work are as follows: 1) we perform extensive experiments (using well-developed benchmark sets, *SPEC2K* and *SPECWeb2009*) under various conditions in terms of workload intensity, number of virtual CPUs (vCPUs), set of active CPUs (i.e., CPUs at an operational state), and so on. This information is useful for developing consolidation algorithms. 2) We rely on real data to evaluate the energy savings of consolidation, instead of using simulators. Virtualized systems are more complex because of abstraction, e.g. vCPU, so it is difficult for simulators to model the system very accurately. Hence, real measurement is essential for proving effectiveness of consolidation. 3) We introduce two consolidation algorithms and assess their efficacies on the *SPECWeb* benchmark suite. The results demonstrate that the proposed consolidation algorithms can result in improvement on energy efficiency under very realistic environments.

The remainder of this paper is organized as follows. In section II we show analysis which shows how consolidation affects energy and performance. The experimental system setup is explained in section III. Following section IV shows experimental results. Finally, we summarize the results and give guidelines in section V.

2. POWER AND LATENCY MODELS

In this section we show how consolidation affects the average power dissipation and latency through mathematical analysis. As a result we can derive insight about how consolidation affects the power dissipation and latency of tasks. Our analytical predictions about the power/latency tradeoffs in a multi-core server system have been empirically shown to be valid for realistic cases. Note that consolidation is reasonable only when the system is under-utilized, thus the model does not consider thermal issue, e.g. leakage power variation with respect to temperature.

2.1 Average power dissipation

In this section we show the relationship between consolidation and power dissipation. Assume we have a system that has '*N*' CPUs with only '*n*' of them being active (i.e., all workloads have been consolidated to run on '*n*' active CPUs and that the remaining '*N-n*' CPUs are turned off or put in a deep sleep state.) The average power dissipation of the i^{th} CPU (P_i) is intrinsically related to the average CPU utilization (U_i) and can be modeled as follows [6]:

$$\begin{aligned} P_i &= aU_i + b, \qquad if\ U_i > 0 \\ &= 0, \qquad\quad otherwise\ (turn\ off) \end{aligned} \tag{1}$$

where U_i denotes $utilization$ of the i^{th} CPU, $0 \le U_i \le 1$.

U_i is a function of workload (k_i), which can be defined as the number of tasks served by the i^{th} CPU (2). '*K*' tasks are sent to the system every second, and a CPU scheduler evenly distributes the tasks to the '*n*' active CPUs, so each active CPU takes '*K/n*' tasks per second. U_i is thus linearly proportional to the total system workload and inversely proportional to the number of active CPUs. However, this statement may not be valid when the workload is very high. As an example, consider a scenario whereby the utilization is '*m*' for *H=K/n* memory-bound tasks per second sent to the target CPU. If more tasks (say *2*H*) are sent to the target CPU per second, the cache miss rate on that CPU will increase (more precisely, the working sets of the *2*H* tasks will not fit on the cache and hence every task will experience a higher cache miss rate.) This means that execution time of the tasks increases, and therefore, the CPU utilization will increase by more than a factor of two. Coefficient '*c*' captures this non-linear effect:

$$U_i = c{k_i}^2 + dk_i, \qquad \text{where } k_i = K/n \tag{2}$$

Total power dissipation is the sum of average power dissipations of individual core (P_i.) Assuming all cores consume the same amount of power, we have:

$$\begin{aligned} P = \sum_{i=1}^{N} P_i &= \sum_{i=1}^{n} aU_i + b \\ &= acK^2/n + adK + bn \end{aligned} \tag{3}$$

Figure 2 depicts the relationship between the number of active CPUs and total average power dissipation for different coefficients, '*c*' and '*b*', under the same overall workload. Note that lower average power dissipation for the same workload means higher energy efficiency. If '*c*' is equal to zero, the total power dissipation decreases monotonically as more CPUs become inactive, which implies that, in this case, consolidation always reduces power dissipation. However, for higher '*c*', the total power dissipation reaches a minimum, and subsequently, goes up with fewer active CPUs. Hence, '*c*' should be carefully considered when we determine the number of active CPUs to consolidate workload. In addition, higher '*c* decreases power saving (Figure 2 a.) The slope of the graph is related to '*b*'. Larger '*b*' results in steeper slope to the right of the minimum power point and therefore, higher power saving. Note that '*c*' is application-dependent whereas '*b*' is a hardware-dependent parameter (with CPG, we expect bigger '*b*'.)

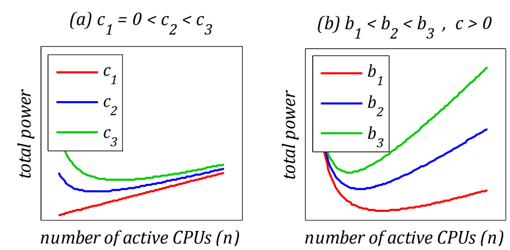

Figure 2 Power dissipation vs. the number of active CPUs under the same overall workload (*K* tasks/s)

2.2 Latency (delay)

As shown in (4), our task latency model has two terms. The first term makes delay larger for higher CPU utilization [6]. This term dramatically increases delay when the system is almost fully utilized. For higher utilization, fewer power state transitions occur, which in turn tends to decrease delay because of lower power state switching overhead. The second term of (4) shows this effect.

$$\begin{aligned} D = D_i = D_{i,1} - D_{i,2} &= \left(\frac{e}{1-U_i} + f\right) - gU_i \\ &= \left(\frac{e}{1-(c{k_i}^2 + dk_i)} + f\right) - g(c{k_i}^2 + dk_i) \end{aligned} \tag{4}$$

where U_i denotes the $average\ utilization$

As shown in the power dissipation analysis, higher '*c*' results in a decrease in potential power saving of consolidation. Figure 3 a. shows the relationship between '*c*' value and delay: higher '*c*' results in larger delay. Note that overall workload of results in Figure 3 is the same (*K* tasks/s.) This implies that consolidation may incur potentially high delay penalty if '*c*' is big.

Figure 3 b. depicts the effect of '*g*'. With a positive value of '*g*', delay decreases as the numbers of active CPUs decreases. This suggests that it is possible to decrease both delay and power dissipation by consolidation if '*g*' is large enough. However, as shown in Figure 3 b, delay increases rapidly when the number of active CPUs becomes very small (since then the CPUs are almost fully utilized), and delay penalty of consolidation becomes significant. Thus, one must carefully choose the number of active CPUs. Coefficient '*g*' is also dependent on the application type.

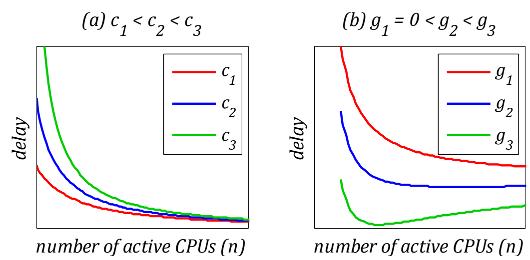

Figure 3 Delay vs. the number of active CPUs under the same overall workload (K tasks/s)

As seen from the above analysis, power saving and delay penalty effects of the consolidation depend on a few coefficients. Some of the coefficients are application-dependent, so there is a motivation to investigate the effectiveness of consolidation under different kinds of applications.

3. EXPERIMENTAL SETUP

3.1 Hardware Testbed

Hardware specification of our system under test is as follows. We have two Intel Xeon E5620 processors; each has four cores, all of which are running at the same clock frequency. Each core has its own dedicated L1/L2 caches but shares an L3 cache with the other cores. Each processor supports seven core frequency levels, from 1.6GHz to 2.4GHz. We cut the 12V CPU power lines and measure the amount of power supplied using a power analyzer.

3.2 XEN – *hypervisor-based virtualization product*

We chose XEN version 4.0.1 for constructing the virtualized system. XEN, which is an open source hypervisor-based virtualization product, provides APIs for managing virtual machines (VMs.) For this study, we run experiments under different configurations in terms of the number of vCPUs, clock frequencies, and the set of active CPUs.

3.3 Service model and Quality of Service

This paper targets a server/client service model where there are many clients, which send tasks to a server and the server responses when it completes the tasks. Turn-around time (from time when a task is sent from a client to the time when a service completion acknowledgement is received by the client) is considered as the delay of the task. For determining the *quality of service* (QoS), we use the 95th percentile delay (Figure 4 .) If the 95th percentile delay is less than the maximum allowed limit, we have met our QoS target. The limit itself is chosen as the 95th percentile delay of a fully-loaded base system (with no consolidation.) The term 'Fully loaded' means that the total CPU utilization is at 80% out of 100%. This is a reasonable value because servers are designed to produce high performance at around 80% utilization levels (performance level drops rapidly as the CPU utilization approaches 100 %.)

3.4 Benchmarks – *mcf, perl (SPEC2K), and SPECWeb*

As shown in Section II, the type of applications affects the effectiveness of consolidation. Hence, we do experiments for three common application types: CPU-bound, memory-bound, and I/O-bound. For the CPU-bound and memory-bound applications we use *perl* and *mcf* benchmarks, respectively. A *perl* shows high *Instruction per Cycle* (IPC) and low *Memory Access per Cycle* (MPC), but *mcf* has opposite characteristics [3]. We develop the WorkloadGen benchmark to control workload level of *perl/mcf* and gather system performance. For I/O-bound application *SPECWeb2009* benchmark is used.

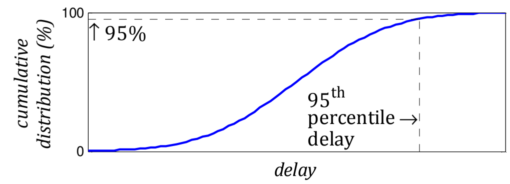

Figure 4 Delay cumulative distribution

3.5 WorkloadGen benchmark

We design and implement a benchmark program, *WorkloadGen*, to generate workload of desired characteristics and to measure performance of the system. It generates tasks and reports throughput and average response time (latency.) Type of tasks and workload intensity (defined as the number of tasks generated per second) are controllable.

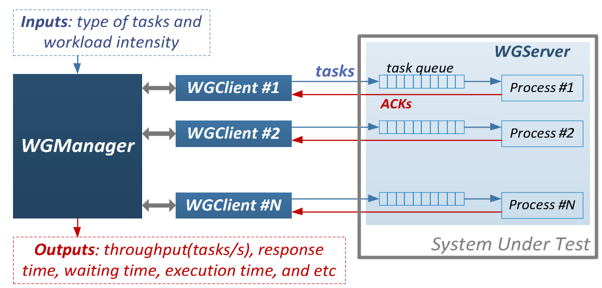

Figure 5 Block diagram of the WorkloadGen

The benchmark consists of three modules (*WGManager*, *WGClient*, and *WGServer*) as depicted in Figure 5 . *WGClients* request tasks to a *WGServer* and report performance statistics to a *WGManager*. *WGServer's* main role is to create workload by executing tasks requested by *WGClients*. When the *WGServer* completes a task, it sends an *ACK* packet to the *WGClient*. Based on the data in the *ACK*, *WGClients* can gather statistics of performance. There are a number of *WGClients*, so it is necessary to control them and gather statistics from them, and this is done by a *WGManager*. The *WorkloadGen* reports statistical performance data: average response time (turn-around time) per task, average waiting time in the queue, and average execution time per task, which are needed for analyzing the overall system performance.

4. RESULTS AND ALGORITHMS

In this section, we report experimental results of *perl*, *mcf*, and *SPECWeb*. Our purpose is to compute the energy saving of the consolidation technique, so all results correspond to an under-utilized server system, i.e., the CPU utilization is around 30% out of 100%. For *perl* and *mcf*, we investigate the delay and energy efficiency of different configurations (i.e., combinations of the number of vCPUs and active CPUs, and clock frequency)

We quantify the *energy efficiency* of a system as '*number of tasks served per unit of energy*.' For *SPECWeb*, we can specify the overall workload level, but instantaneous workload level changes dynamically. Hence, for *SPECWeb*, we verify the energy efficiency of the consolidation through dynamic management. We propose two very practical algorithms for dynamic management, and show up to 15% improvement in system's energy efficiency.

4.1 *perl* & *mcf* benchmark results

The number of vCPUs is an important parameter in a virtualized system: it determines how many CPUs can be utilized by a virtual domain at any time, so the performance of the domain is closely related to this parameter. At the same time, more vCPUs in a system cause higher power and performance overheads, so it will be detrimental if there are too many vCPUs. Figure 6 depicts the overhead caused by unnecessarily managing too many vCPUs. Utilization and delay of *perl* benchmark rapidly increase when the ratio of the number of vCPUs to the number of CPUs becomes greater or equal to three. The *mcf* benchmark shows the same trend. Note that workload level of all cases in Figure 6 is the same to each other. This means that the overhead of managing vCPUs causes this phenomenon. This result implies consolidation needs to be accompanied by dynamic vCPU count management. We maintain the ratio of vCPUs to CPUs to be around two for all test cases presented in this study through dynamic vCPU count control.

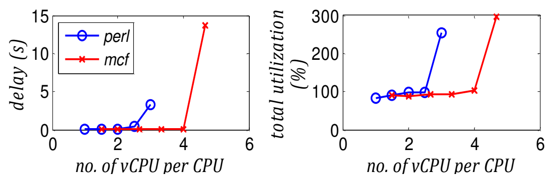

Figure 6 Delay and total utilization vs. the number of vCPUs per CPU (there are 4 active CPUs)

For consolidation, the choice about which set of CPUs is active can have an effect on performance and energy efficiency. Under multi-processor systems, several different CPU selection schemes are possible. In this study, we have only two processors, so we simply investigate two CPU selection schemes: 1) we select half of the required CPUs from one processor and other half from the other processor; and 2) we select all required CPUs from one processor, and only if more CPUs are needed than one processor can provide, we turn on the other processor and select the remaining CPUs from the other processor. We call these schemes as '*symmetric*' and '*asymmetric*' selection scheme.

Figure 7 shows the energy efficiency and delay of schemes. There is only small difference in energy efficiency and delay, but '*asymmetric*' selection scheme is a little bit better for the *perl* in terms of energy efficiency. On the other hand, there is no noticeable difference in energy efficiency and delay for the *mcf*. Hence, we choose '*asymmetric*' scheme for this study.

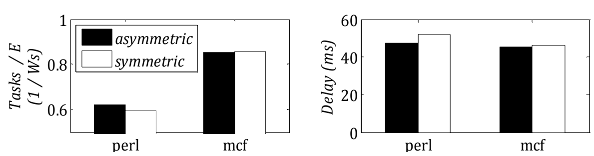

Figure 7 Energy efficiency & delay vs. CPU selection schemes

Figure 8 shows how much energy efficiency can be improved by consolidation without sacrificing QoS. Moreover, it shows that energy efficiency can increase even more if consolidation is accompanied by DVFS. The experimental setup is as follows: there are two guest domains. The '*perl'* case runs *perl* on both guest domains. '*mcf*' case runs *mcf* on both guest domains. '*mixed*' case runs both *perl* and *mcf*, i.e., one domain serves the *perl* while the other domain serves the *mcf*.

Energy efficiency of *perl* is dependent on both the number of active CPUs and clock frequency (Figure 8 a), i.e., fewer CPUs with slower frequency increases the energy efficiency. Delay of *perl* is more dependent on the clock frequency than the number of active CPUs (Figure 8 b), which means that consolidation can be done without performance degradation.

One observation is that the delay of the case with fewer active CPUs is sometimes even smaller than that of the case with larger number of CPUs. For example, 4CPU running at 2.2GHz shows smaller delay than 8CPU at the same frequency. In our model, coefficient 'g' in (4) represents this effect, i.e., a consolidated CPU may end up changing its power states less frequently because of higher utilization level, and this can reduce the delay. The energy efficiency of *mcf* seems to be independent of the number of active CPUs as well as frequency (Figure 8 c.) For different combination of clock frequency and the number of CPUs, there is only less than 3% difference in the energy efficiency. Delay of *mcf* is mainly affected by frequency, but again the difference in the delay is smaller than *perl* (Figure 8 d.) This result suggests DVFS and consolidation are not so effective for *mcf* because there is no noticeable improvement on energy efficiency. The *mixed* case exhibits similar trends to the *perl* case (Figure 8 e and f.)

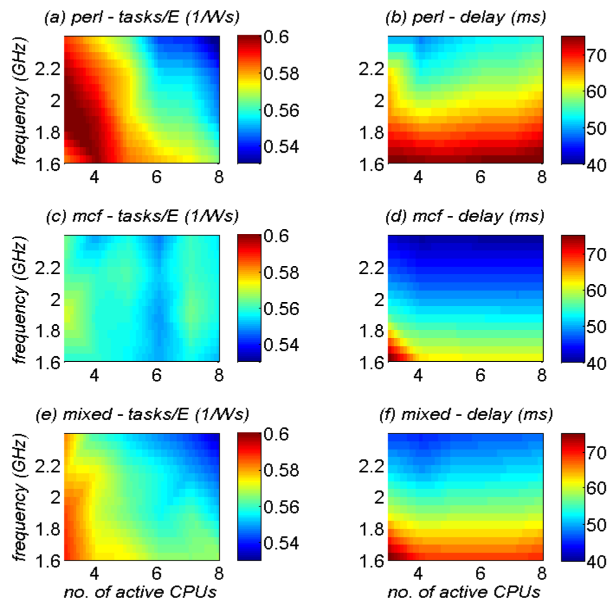

Figure 8 Energy efficiency and delay (Maximum allowed delay of *perl mcf*, and *mixed* is 73ms, 81ms, and 80ms respectively)

TABLE I. shows improvement on energy efficiency of *perl*, *mcf*, and *mixed* cases. Energy efficiency improvement is largest when both consolidation and DVFS are conducted. The *perl* shows biggest improvement and *mcf* shows smallest improvement. This result gives another insight; For VMM, if we consider types of application running on the VMs, we can enhance energy efficiency more. It suggests deploying heterogeneous VMs in a server machine. For example, we may have four domains (two domains serve CPU-bound tasks and others serve memory-bound tasks) and need to deploy them into two server machines. If homogeneous domains are mapped to the same server, energy improvement is 8.1% on average (13.8% from CPU-bound domains and 2.4% from memory-bound domains.) On the other hand, if heterogeneous domains are mapped to a server machine, we achieve 9.7% improvement for both server machines. It is not big difference though in this study, but it implies energy efficiency can be improved through sophisticated VM deployment. It is our future plan to find optimal VM deployments for energy efficiency.

TABLE II. shows coefficients of our model got from experimental results. Our model does not consider DVFS, so we fix frequency (2.4GHz) to find coefficients. R-square value represents how much the model is accurate. Power and utilization equation are quite accurate for *perl*. Delay model of *perl* is acceptable. However, power and delay model of *mcf* is not accurate. The only utilization model is accurate. It is because both power and delay does not change a lot for different number of active CPUs: difference in energy efficiency and delay is less than 3% (Figure 8 c and d.) The difference may be caused by some other factors such as uncertainty of measurement and noise, and the model does not consider those factors. Hence, our model does not fit to *mcf* result. '*c*' of both benchmarks is very small compared to '*d*', which means we can ignore '*c*'. Bigger '*c*' means larger consolidation overhead, but it is very small because we already reduce consolidation overhead by adjusting the number of vCPUs. '*g*' of *perl* is positive, and it implies delay can be reduced by consolidation (Figure 8 b.)

TABLE I. IMPROVEMENT ON ENERGY EFFICIENCY

	Improvement on energy efficiency (%)		
	perl	*mcf*	*mixed*
DVFS	*7.4*	*0*	*6.2*
Consolidation	*10.7*	*1.9*	*8.8*
DVFS +consolidation	*13.8*	*2.4*	*9.7*

TABLE II. COEFFICIENTS OF THE POWER AND DELAY MODELS

	coefficients	***perl***		***mcf***	
		value	R^2	***value***	R^2
power	*ac*	*1.2E-05*	*0.977*	*-0.001*	*0.402*
	ad	*0.066*		*0.099*	
	b	*0.041*		*-0.022*	
utilization	*c*	*- 0.237*	*0.996*	*-0.046*	*0.999*
	d	*6.339*		*4.266*	
delay	*e*	*-143.938*	*0.913*	*-26.758*	*0.766*
	f	*45.796*		*35.419*	
	g	*0.071*		*-0.178*	

4.2 SPECWeb2009 benchmark result

SPECWeb2009 is a well-developed benchmark suite for evaluating web servers which are I/O-bound, so its results can show how consolidation affects delay and energy efficiency of I/O-bound applications. *SPECWeb* requires Simultaneous User Sessions (SUS = 600 in this study) as input. We can specify level of workload by SUS count, but it is only overall workload. Instantaneous workload changes dynamically, so a dynamic management scheme is needed for consolidation. In this section, we start from understanding characteristics of *SPECWeb*. Next, we propose two algorithms for consolidation. Finally, we show experimental results for the proposed algorithms and compare them with the case without consolidation. The '*asymmetric*' scheme is used because it saves more energy than the other. Moreover, consolidation is accompanied by dynamic vCPU count management and DVFS.

4.2.1 Characteristics of the SPECWeb Suite

Web applications are typically not compute intensive [11]; hence, the performance (i.e., the response time per task) is less dependent on the clock frequency of CPUs (Figure 9 a.) One observation is that the number of active CPUs will not be an important factor in setting the web server performance if a sufficient number of CPUs is available. This is because the performance of web servers is highly sensitive to I/O, such as network and disk access. This result also implies that consolidation saves energy without noticeable performance loss for such applications.

Figure 9 b. depicts power dissipation, which is measured for different combinations of the number of active CPUs and clock frequency. One observation is that the power difference between *4CPU* case and *5CPU* case is biggest. This is because only one processor chip is active for the *4CPU* case (*asymmetric* scheme.)

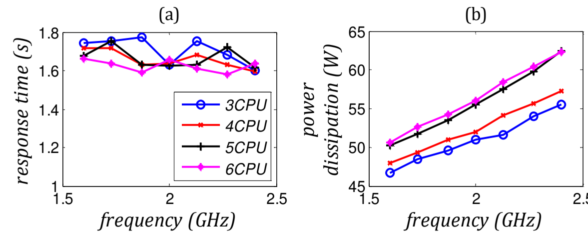

Figure 9 Response time and power dissipation

The relationship between frequency and utilization is needed for designing an effective consolidation controller. Our controller assumes that the workload level observed in the previous decision epoch persists into the current epoch (more sophisticated workload prediction schemes may be employed, but it falls outside the scope of present paper.) CPU utilization under the same workload, however, can be changed if the frequency changes. Hence, the controller must prevent an undesirable situation whereby the active CPUs are overloaded because the chosen frequency is too low. We assume the relationship between utilization and frequency is linear (5), and it is very accurate (Figure 10 .) β is relatively small and can be ignored, so we use a simpler approximated equation (6).

$$(u-\beta)\cdot f = \alpha \text{ , where } \alpha = 150.4, \beta = 29.9 \text{ and } 0 \le u \le 800 \text{ (8 pCPUs)} \quad (5)$$

$$f_i u_i = f_j u_j = \alpha \quad (6)$$

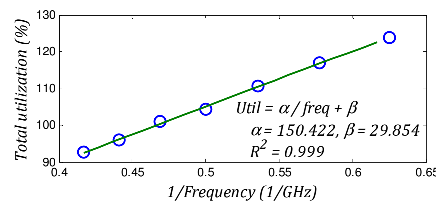

Figure 10 Frequency vs. total utilization

4.2.2 CPU Consolidation Controller

As shown in the previous section, clock frequency and the number of active CPUs are important factors. Hence, we have to carefully select these parameters. In this study, we propose two algorithms, which perform both DVFS and consolidation. They monitor CPU utilization every second, and change frequency and/or the number of active CPUs when desirable. Main idea of the algorithms is to utilize fewer CPUs at slower frequency as possible. It is reasonable for I/O bound applications because performance degradation is not significant unless CPU is almost fully utilized [9]. In addition, our main goal is not to maximize energy efficiency but to show potential of power saving by consolidation. Hence the simple algorithms are enough for this study. To reduce performance degradation, the algorithms change the system configuration conservatively: If a system is overloaded, they promptly increase frequency and/or the number of active CPUs. If, however, the system is under-utilized, they reduce frequency and/or turn off some CPUs when the situation persists for five seconds.

We present two algorithms and main idea of them is quite similar to each other: If the average utilization (u_n) of a CPU is greater than an upper threshold (T_h), they will assign more resource by increasing the clock frequency and/or the number of active CPUs. If the average utilization is less than a lower threshold (T_l), they will decrease frequency and/or the number of active CPUs.

Algorithm I favors using lower frequencies. If more CPU resource is required, it increases the number of active CPUs at the beginning (line 4 through 6.) Only when all CPUs are active and still more CPU resource is needed, it increases the frequency. In case of overbooking, it decreases the frequency first (line 12 through 14.) If the system is still overbooked at the lowest frequency, it will decrease the number of active CPUs (line 16 through 18.)

Algorithm II is similar to Algorithm I, which is not shown here, except that it favors using fewer active CPUs. If the average utilization of CPU is greater than T_h, the highest frequency will be selected. If still more CPUs are needed, it will increase the number of active CPUs. If the CPU is overbooked, it will keep the current frequency and decrease the number of active CPUs.

```
CPU Consolidation Algorithm I
Inputs: N (total number of CPUs), f_n (frequency), c_n (the number of
   active CPUs), T_l/T_h (low/high threshold), and u_n (average utilization)
Outputs: f_{n+1} (frequency), c_{n+1} (the number of active CPUs)
1:  if u_n > T_h then                    // more CPU resource is required
2:     persist = 0
3:     f_{n+1} = f_n                     // keep the same frequency
4:     for c_{n+1} in [c_n+1 , c_n+2 , … , N] do// increase # of active CPUs
5:        if u_{n+1} < T_h then // u_{n+1} = u_n(f_n c_n) / (f_{n+1} c_{n+1}) : new util.(6)
6:           break               // stop here. enough CPU resource
7:     if u_{n+1} > T_h then     // still need more CPU resource (c_{n+1}=N)
8:        f_{n+1} = f_max               // increase frequency to the maximum
9:  else if u_n < T_l then              // CPU resource is overbooked
10:    if ++persist >= 5 then  // under-utilized for 5 decision period
11:       persist = 0, c_{n+1} =c_n // start from the same # of active CPUs
12:       for f_{n+1} in all frequencies (ascending order) do
13:          if u_{n+1} > T_l then
14:             break          // found the smallest frequency. stop
15:       if u_{n+1} < T_l then      // still overbooked
16:          for c_{n+1} in [c_n -1 , c_n -2 , … , 1] do
17:             if u_{n+1} > T_l then
18:                break  // found minimum # of active CPUs. stop
```

Figure 11 consolidation algorithm I

TABLE III. shows the energy efficiency and QoS, which is defined as the percentage of packets that meet the performance specification. As defined before, we consider that there is no performance violation if QoS is greater than 95%. The first row shows results of the base system without consolidation and DVFS. The second test set uses Linux *ondemand* (DVFS) governor [12], which is a default governor of Linux, and it does not support consolidation in general. The last two rows show results from the proposed algorithms. Results are measured through measurements performed on our test bed hardware, so they also include the energy consumption and delay overheads of running the proposed consolidation algorithms. Algorithm I is the best one in terms of energy efficiency—it consumes around 15% less energy compared to the base case, which is better than 'Linux DVFS' case. QoS of Algorithm I is 96.9% which still meets performance requirement. Note that Algorithm I is better than 'Linux DVFS' in terms of performance as well as energy efficiency. Nevertheless additional overhead from turning on/off cores, Algorithm I shows better performance than 'Linux DVFS' because of its conservative nature: the algorithm infrequently changes frequency and active CPU count. Algorithm II improves energy efficiency by around 12% which is also greater than 'Linux DVFS', but it does not meet performance requirement. Algorithm I is outperforms II, and it shows that consolidation with DVFS improves energy efficiency of web servers.

5. CONCLUSION

DVFS has been a promising method for dynamic power management technique, but the energy saving leverage of DVFS decreases as the supply voltage level decreases with CMOS scaling. With HW support, such as *Core-level Power Gating*, consolidation becomes a promising power management technique. However, consolidation needs to be investigated by a more realistic scenario. In this study, we investigate effectiveness of consolidation for different configurations: types of applications, the number of active CPUs/vCPUs, and selection schemes. Through experiments, we presented suggestions for consolidation: 1) Control the number of vCPUs: we recommend that the ratio of the number of vCPUs to the number of CPUs to be less than 3. 2) Avoid aggressive consolidation for memory-bound applications: there is very small energy benefit. 3) Deploy heterogeneous VM domains in a single system: this saves more energy without violating performance. To summarize, this study shows energy saving of consolidation in virtualized systems. The proposed consolidation algorithm achieves about 15% of energy saving for I/O-bound application (*SPECWeb*.) Moreover, it increases the energy efficiency of CPU-bound application by 13.8%. Least improvement (2.4%) was achieved from Memory-bound applications.

TABLE III. SPECWEB BENCHMARK RESULT

	Tasks / E	*QoS (%)*	*Improvement*
Base	*1.098*	*98.0*	*n/a*
Linux DVFS	*1.212*	*93.4*	*10.4*
Algorithm I	*1.264*	*96.9*	*15.2*
Algorithm II	*1.228*	*90.3*	*11.9*

6. REFERENCES

[1] G. von Laszewski*, et al. Power-aware scheduling of virtual machines in DVFS-enabled clusters*. in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*.

[2] P. Pillai and K.G. Shin. *Real-time dynamic voltage scaling for low-power embedded operating systems*. in *Proceedings of the eighteenth ACM symposium on Operating systems principles*.

[3] G. Dhiman*, et al. vGreen: a system for energy efficient computing in virtualized environments*. in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*.

[4] 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*.

[5] O. Bilgir*, et al.*, *Exploring the Potential of CMP Core Count Management on Data Center Energy Savings*, in *3rd Workshop on Energy Efficient Design*.

[6] N. Bobroff*, et al. Dynamic Placement of Virtual Machines for Managing SLA Violations*. in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*.

[7] H.N. Van*, et al. Autonomic virtual resource management for service hosting platforms*. in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*.

[8] C. Clark*, et al. Live migration of virtual machines*. in *Proceedings of the 2nd conference on Symposium on Networked Systems Design \& Implementation - Volume 2*.

[9] M. Pedram and I. Hwang. *Power and Performance Modeling in a Virtualized Server System*. in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*.

[10] J. Leverich*, et al. Power Management of Datacenter Workloads Using Per-Core Power Gating*. in *Computer Architecture Letters*.

[11] D. Meisner*, et al. Power management of online data-intensive services*. in *Proceedings of the 38th annual international symposium on Computer architecture*.

[12] V. Pallipadi and A. Starikovskiy. *The Ondemand Governor*. in *Proceedings of the Linux Symposium*.