Exploring the Potential of CMP Core Count Management on Data Center Energy Savings

Ozlem Bilgir
Princeton University

Margaret Martonosi
Princeton University

Qiang Wu
Facebook, Inc.

Abstract

A data center's power and energy consumption is a crucial design issue, and is often a fundamental determiner of its performance potential. Conventional power/energy management approaches data centers have focused on uniprocessor rather than multiprocessor servers and on techniques such as Dynamic Voltage and Frequency Scaling (DVFS). Given the increasing use of chip multiprocessors (CMPs) and the decreasing leverage of DVFS, it is time to re-examine energy and power management possibilities with an eye towards future data center implementations. In particular, it is interesting to consider whether per-core power management techniques on a multi-core chip can offer sufficient leverage, and whether inter-core resource contention on CMPs may influence what sorts of core scheduling techniques make sense.

In this paper, we focus on cross-server core count management techniques and evaluate a range of design questions for CMP-based data center power managers. Our results are based on a highly-parameterized simulation environment. We evaluate several possible approaches for selecting the number of cores to use, and selecting how the desired core count should be mapped onto the available CMPs. We evaluate these techniques on a real-world workload representing 24 hours of Facebook requests. For the scenarios considered, total core energy savings of 35% are possible from appropriate core count selection and mapping. Overall, this work represents a high-level view of CMPs as data center servers, and can be a guide for more detailed future studies on power management and server design questions.

1. Introduction

Energy consumption and related cooling costs have become a paramount concern for data center design and operation [1, 12]. Electricity costs represent roughly 20-30% of data center operating expenses [10], and cooling costs/equipment also rise when power consumption rises. Therefore, managing data center energy and power dissipation represents a fundamental issue for operating data centers and scaling Internet services. Furthermore, energy management is difficult to do well, because although servers have modest utilizations (10-50% [2]), it is difficult to predict bursts of idle and busy times, and often even idle servers consume as much of 60% of their maximum energy [2, 9].

In response to power and energy concerns, many researchers have proposed dynamic management techniques to mitigate them. Dynamic resource management by consolidating the load to a small subset of servers and entirely turning off the idle servers has been studied widely [5, 6, 21]. Although it theoretically offers good power saving potential, applying it to current data centers is not desirable for a few reasons. First, in many data centers, servers are updated with new code very frequently, and technicians prefer always-ON servers to ease the application of code pushes and possible resulting problems. Second, systems are expected to be robust but it is not guaranteed that server will turn on properly when needed. Third, high boot latencies affect performance because load prediction can only allow partial back-grounding of boot times. For all these reasons and others, turning servers off completely has been avoided by data center operators. Furthermore, nearly all these server control papers have assumed uniprocessor servers, whereas the dominant platform in today's data centers uses chip multiprocessors (CMPs).

Other data center power management techniques have considered Dynamic Voltage and Frequency Scaling (DVFS) applied to processors [11, 17, 22]. Some researchers increase the benefits of DVFS by combining it with resource consolidation [3, 7, 23]. Although DVFS has offered considerable power savings in the past, its leverage is decreasing. For example, the ITRS roadmap [14] predicts that operating voltages will continue to decrease in future technologies, causing Vdd get closer to threshold voltage. This reduces the dynamic range in which DVFS can operate and offer savings. In addition, the prominence of leakage power makes it less favorable to operate CPUs at slow DVFS settings where leakage will be substantial.

The decreasing leverage of DVFS and the increasing prominence of CMPs calls for a reexamination of power control opportunities and trends. In particular, the ability to turn off individual processor cores on a CMP, rather than turning on or off the whole server, offers an appealing granularity at which to manage power and energy. While some initial aspects of per-core power gating have been briefly explored [16, 18, 19], many questions remain. For example, how much power-saving leverage exists from adjusting core counts dynamically, without necessarily turning servers on or off? Also, how do contention issues influence how one should assign jobs to cores? For example, if one has 4 quad-core CMPs and enough current workload for 4 processor cores, consolidating the requests onto a single CMP chip seems appealing from a power perspective, but may lead to extra contention for shared chip resources such as shared caches and on-chip interconnect. Finally, since processor cores can be power-gated off and on more

smoothly and quickly than whole servers, how does this influence power management decisions? This last question is particularly relevant for data center workloads that may alternate unpredictably between frequent bursts of busy and idle periods.

To answer these questions, this paper uses abstract models to explore the large possible design space. We evaluate the different design possibilities using a workload trace derived from real Facebook data center processing. The primary contributions of this work are;

- This paper quantifies potential power savings for a family of core count management techniques in multi-server CMP systems. Although simple, our method consistently satisfies service-level agreements for 75^{th} percentile request latency.
- Based on both a real Facebook workload and other stochastic workloads, we evaluate the potential energy savings offered by our techniques. Our results show that total core energy consumption can be decreased by 35% with our method still satisfying the same 75^{th} percentile latency goal. Effect of core energy saving on total energy consumption varies from 3% to 15% (depending on power breakdown) . Power savings are even larger when data centers see large idle periods.
- We explore how chip-level contention influences the design of core count managers. At periods of very high or very low load, the differences between round-robin (cross-core) versus consolidating schedulers are modest. At intermediate load, however, round-robin is more effective at garnering high performance out of CMPs prone to contention effects. Such issues should be considered by those designing request schedulers for data centers built on large aggregations of CMPs.

This paper is structured as follows. Section 2 explains related work in more detail and describes our family of proposed core count management techniques. Section 3 then describes experimental methodology. Section 4 introduces the results and Section 5 gives discussion about the results. Section 6 offers conclusions.

2. Core Count Management

2.1 Related Work

Due to the reasons discussed in Section 1, server consolidation is not a good candidate for power management real system. Instead of turning servers off, PowerNap [20] proposes using low-power server states. Since the latency of entering a low-power state is much lower than that of server turn off, this method provides a better power saving opportunity. However, this system only works at times where the whole system is idle and this highly depends on workload characteristics. Thus, depending on the application, controlling the low power states of individual components (CPU, memory, disk, fan) allows finer grain power management and can be more desirable.

CPU power management has been an interesting topic to many researchers. Chen et al. [7] and Bertini et al. [3] focused on applying DVFS and server consolidation to heterogeneous unicore-servers for multiple applications. Although these techniques have been successful, our work has an advantage over them because we focus on CMP systems which are becoming more common in high-performance data centers. Moreover, our technique will not be affected by current technology trends unlike DVFS. Although per-core DVFS is also possible, because of current technology trends, we believe that our technique can be an alternative for per-core DVFS as well.

Power-gating in CMPs has been studied by previous researches. Leverich et al. introduce per-core power gating to cut the voltage to the cores to eliminate leakage power [16]. They use utilization information to decide whether to enable or disable a core. Our method differs from this work since we have a cross-server system where the necessary number of ON cores is decided at a front-end level. Therefore, the jobs can be consolidated to a subset of cores. The rest of the cores are kept off for at least one decision period which is high enough to avoid the penalties from frequent turn on and offs. Madan et al. raise the concept of guarded power gating which disables the power manager when there is an unexpected workload behavior or power virus attack to decrease the cost of frequent turn on-offs [18, 19]. This is still in the server level and does not give a global control over multi-server clusters. However, our work will need to use the hardware infrastructure for per-core power gating introduces by these works or recent Nehalem power gating techniques [15].

2.2 Heuristic Core Count Management Approach

We determine the total necessary number of ON cores for the whole system using a look-up table. This look-up table gives the total necessary number of ON cores for different workload ranges in order to achieve a latency goal. It is created beforehand as follows; we turn on cores one by one and for each ON core count, we increase the load rate until the latency goal is exceeded. We record this load rate as the maximum that the given core count can handle.

Turning on cores from different processors or from the same processor also affects the latency and should be taken into account when creating the look-up table. The reason for this effect comes from the resource sharing in CMPs. In CMPs, cores share last level cache, memory controller, interconnection network and I/O hub. This causes a contention problem which affects their performance and energy consumption [4, 8, 13]. For example, if one of the cores frequently evicts cache lines which are necessary for other core(s), than this behavior will cause a degradation in performance and increase in power. For this reason, we first determine the mechanism that decides which processors the ON cores should reside. Using this mechanism, we create the lookup table. At the time we use the look-up table, we use the same mechanism to turn on the necessary number of cores and this ensures that we get the predicted behavior in terms of latency.

An example look up table is given in Table 1. This table shows the maximum allowed load rates for each ON core count to guarantee the 75th percentile latency goal for a system with 4 quad-core servers. All load rates are given as ratios to the maximum load rate. The maximum load rate of 100% is defined as the rate at which 75th percentile latency reaches 250 msec when all cores are ON in all servers. For a given load rate and latency goal, this table is searched to find the necessary number of ON cores.

Core Count	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Latency Goal																
150 msec	0%	4%	9%	14%	19%	25%	30%	36%	42%	48%	55%	61%	67%	73%	80%	86%
200 msec	1%	7%	13%	20%	26%	32%	38%	45%	51%	57%	64%	70%	77%	84%	90%	97%
250 msec	3%	9%	14%	21%	28%	34%	40%	47%	54%	60%	66%	72%	79%	86%	93%	100%

Table 1. An example look-up table for a 4 quad-core-server system. When all resources are available, ie. all cores are ON in all servers, the maximum load rate is defined as 100% when the 75th percentile latency reaches 250 msec. Other load rates are shown as a percentage of the maximum load rate. Each entry tells the maximum load rate percentage that the given number of ON cores can process and the 75th percentile latency is not above the given latency goal. Contention between cores is not taken into account for this table. Additional look up tables are created for the cases where core contention is taken into account.

For example, if we want the 75th percentile latency to be below 200 msec at 15% load, then the table says we need 4 cores ON. For this particular case, core contention is ignored. As a result, performance will not be affected by the distribution of ON cores to the servers. Thus, the same table can be used for different distribution schemes for this case. When contention is taken into account, different tables are created.

Regional approaches, per-application tables, and control theoretic approaches are all related options for future work. Here we focus mainly on CMP aspects of the control opportunities.

2.3 Core Count Management Alternatives

As explained in Section 2.2, different distribution of ON core across servers have different performance and power effects when contention is taken into account. These are;

- Round-Robin Scheme (RR): In this scheme, cores are chosen from the servers in a round-robin scheme. We start by turning on 1 core from each server and continue with 2 cores, etc. This method mitigates the effect of core contention by distributing the jobs evenly to the servers.
- Same-Server Scheme (SS): In this scheme, cores are chosen from the same server until all of its cores are ON. Only after all cores in a server are turned on, cores can be chosen from the next server. Thus, in this scheme, cores are consolidated to the smallest possible number of servers. Therefore, server power contribution will be less. In this method, effect of contention on performance would be more visible.
- Chip Turn On/Off (CT) : In this scheme, the entire chip is turned on/off. This method will have faster reaction to quick changes in load rate since all the cores in the chip are ON even though they are not being used in that period.

Note that, load balancing across servers should also be consistent with the applied core count management technique.By comparing different management techniques, we try to understand how beneficial a technique is under different load rate and contention models.

3. Simulation Methodology

The design and analysis of comprehensive data center simulators are complex processes. Since the goal in this study is to explore the potential of core count management, we implemented an abstract stochastic data center simulator as described below. Section 3.2 discusses its performance and power models. Section 3.3 discusses the workloads used for our evaluation, including a day-long trace based on actual Facebook workload logs.

Simulation Parameter	Explanation	Value
S	Server Count	4
N	Core Count per Server	4
-	Inter-Arrival Time Distribution	Exponential
-	Service Time Distribution	Exponential
$Service\_time_{base}$	Mean Service Time	100 msec
T	Control Period	10min
C_{ratio}	Degradation of Service Time with each Busy Core	0%, 15%
-	$\frac{P_{cores}}{P_{total}}$	10%, 35%, 50%
-	$\frac{P_{core\_idle}}{P_{core\_busy}}$	40%
W	75th percentile Latency Goal	150 msec, 250 msec

Table 2. Simulation parameters used. Parameters with more than one value means that different different values are used for that parameter to see the effect on total energy and performance.

3.1 Simulator Design

When a new request arrives at the modeled multi-server data center, it goes to a front-end device. This device queues requests until they can be sent to an available server. Each server can handle a number of requests equal to its core count, so any queueing occurs at the front end. For simplicity, we study a single-tier data center, but multi-tier requests could also be modeled. It is also assumed that the workload consists of single-threaded applications, with one thread running per core.

The front-end devices use a table such as Table 1 to determine the appropriate number of ON devices at each interval, T, as described in Section 2.2. This means that every T minutes, a new table look up is performed to get a new core count based on the latency target and the observed workload in the past interval. Once table lookup gives the correct core count, load distribution (i.e. *which* cores are on) occurs according to one of the possible heuristics: same-server, chip turn-on, or round-robin as previously described.

The simulator allows us to vary and study the number of servers, number of cores per server, number of ON cores for each server, and power breakdown of servers. Power, energy, request throughput, and request latency are the metrics tracked by the simulator. Requests into the system are modeled based on stochastic arrival and processing

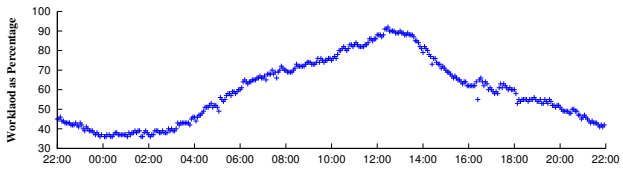

Figure 1. Typical server workload in Facebook data centers. Workload is given as a percentage of the maximum allowed workload.

times. The means and distributions of these times can be varied, and Section 3.3 discusses the workloads modeled here. Performance goal is chosen to be 75^{th} percentile but the method can be adapted to other levels easily. All parameters used in the simulations and their definitions can be found in Table 2.

3.2 Performance and Power Models

The service time for each request is stochastically generated. For this paper, we presume an exponential distribution with a mean of $Service\_time$ although other distributions could be modeled as well. In our simplest performance model, we assume that a core's performance (and therefore a request's service time) is not affected by any inter-core resource contention on the CMP. For other experiments, however, we assume that having multiple busy cores on the same CMP can degrade performance due to contention for shared cache and interconnect. In experiments where we wish to account for this contention, we use a simple linear contention model. In this model, we assume that a core's mean service time increases linearly with the number of other busy cores, weighted by a contention factor:$Service\_time = Service\_time_{base} + (N_{busy} - 1) * C_{ratio}$. N_{busy} is how many of a chip's cores are currently busy, and C_{ratio} is the contention degradation factor. If N_{busy}=1, then the average service time is $Service\_time_{base}$.

The total core power of a server is given by,

$$P_{cores} = N_{busy} * P_{core\_busy} + N_{idle} * P_{core\_idle} \quad (1)$$

and the total power of a server is given by,

$$P_{total} = P_{nonCpu} + P_{cores} \quad (2)$$

Here, P_{core_busy} is the busy power of a core and P_{core_idle} is the idle power of a core. P_{core_busy} and P_{core_idle} are multiplied by the number of busy (N_{busy}) and idle cores (N_{idle}) respectively. Our model distinguishes between idle ON cores (P_{core_idle}) and cores that have been fully turned off (0 power). P_{nonCpu} is the power of all the components except the processor cores themselves (including uncore components on the CMP chip). When multiple busy cores induce resource contention effects, these are modeled as longer processing times as discussed above. Thus the energy impact of resource contention is implicitly calculated through its impact on the idle and busy times of the processors.

3.3 Workload

The main workload used in our simulations is obtained from the real logs of Facebook servers in the web tier which serves mixed page requests to www.facebook.com, mobile.facebook.com, and apps.facebook.com. The most popular page requests include those for news feed, status update, profile change, wall posting, photo loading and uploading, and games. Figure 1 shows per-server workload in a typical 24-hour period starting at 10pm and ending at 10pm the next day. For public release, all workload values can only be given as a percentage of the maximum possible workload. Clearly, daytime request rates are more than double the night time workload, which highlights the potential for power savings with appropriate resource management.

The figure shows the total number of requests arriving in 4-minute windows as logged. To model individual request arrivals, we use the 4-minute workload as a guide, and generate individual inter-arrival times with exponential distribution based on that mean. Facebook's production servers which handle this workload are 8-core, 16-threaded machines. In our simulations, we use 4-core single threaded servers. Thus, we scaled the workload with respect to our server configuration.

Apart from the real Facebook workload, we also generated 3 additional workloads with stationary exponential distributions corresponding to stable load rates at low (5%), medium (40%) and high (85%) levels. These workloads more clearly highlight how different core count management schemes work under different workload levels.

Parameter	Value
C_{ratio}	0%
Load	Facebook Workload
Empty Servers Assigned to Other Jobs	No
75^{th} Percentile Latency Goal	250 msec

Table 3. Default values of key parameters which are used in the simulations.

4. Results

In this section, we discuss how our core count management techniques work under different scenarios. First, we compare the total core energy and 75th percentile latency obtained by our 3 core count management schemes with the "Baseline" case. In the Baseline, all the cores in all servers are ON all the time. Jobs are distributed to these servers in a round-robin scheme. In all core count management schemes, all servers are assumed to start ON. Table 3 shows the parameters used in the simulations unless otherwise stated in particular cases.

4.1 Effect of Different Load Rates

Turning-off unused cores saves idle energy, thus we expect better core energy savings with low loads. In order to verify this, we used the three different levels of workload and the Facebook workload which can be seen as combination of different level workloads.

In Figure 2, we see the normalized total core energy with respect to the Baseline's total core energy at 85% load. As expected, all of the core count management techniques have relatively greater energy savings as the load rate get smaller. Although for 85% load, only 5% energy saving can be obtained by SS, savings reach 80% under 5% load. For the Facebook workload, the savings are 35%. The reason for not having much energy savings under high workload comes from the fact that the most of the cores need to

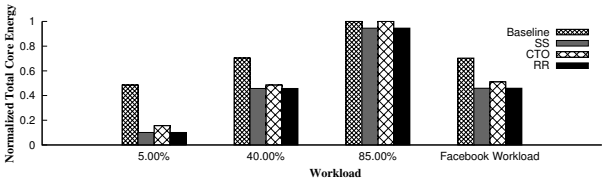

Figure 2. Normalized total core energy at different workload levels. Although energy saving with SS is 35% compare to baseline case under Facebook workload, it can reach up to 80% when load rate is 5%.

1	2	3	4	...	16
590msec	165msec	143msec	142msec	...	142msec

Table 4. 75^{th} percentile latency under 5% load when different number of cores are ON. As can be seen, although the latency is only 165 msec when 2 cores are ON, our techniques do not chose to turn on 1 core because in that case, latency would exceed the goal.

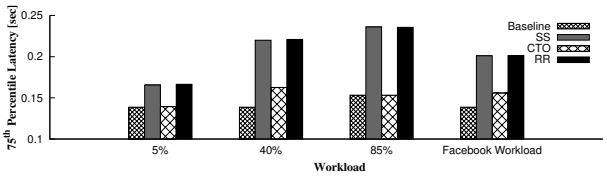

Figure 3. 75^{th} percentile latency at different workload levels with different techniques. All techniques manage to ensure the latency goal of 250 msec.

be ON to satisfy the latency goal. Comparing the total core energy saving with SS, CTO, and RR, we see that there is no difference between SS and RR. This is expected behavior. Since we assumed here that there is no contention between cores in this case, having ON cores in the same server or at different servers does not make a difference energy-wise. The CTO results in slightly higher energy consumption than SS, which again is expected behavior since fewer cores can be turned on.

In Figure 3, we see that all schemes manage to achieve the latency goal of 250 msec. In all cases, CTO has lower latency values as expected. At 5% load, 75^{th} percentile latency is only around 165 msec in SS and RR. This is very small compared to our allowed limit. If more cores were turned off, energy saving would increase. However, our core count management technique does not decrease the number of ON cores, because that would degrade the performance to an undesired level. To illustrate this, Table 4 shows the 75^{th} percentile latency under 5% load for a range of core counts. As can be seen, reducing the core count from 2 to 1 has a big impact on latency and would cause latency to exceed the latency goal. Therefore, our techniques do not chose to turn on 1 core even though the 2-core latency is much better than the expected.

4.2 Effect of Contention

When we take the contention between cores into account, total core energy saving and performance differ. Figure 4 shows that under contention, total core energy consumption is increased in all schemes.This is because the service times, hence busy energy, increase under contention. Schemes where the jobs are distributed in a round robin scheme, ie. Baseline and RR, are affected less than the schemes where the jobs are sent to the same server first, ie. SS and CTO. The reason for this is that when the ON cores

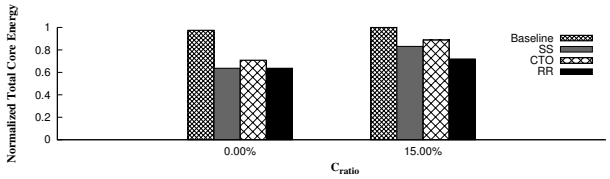

Figure 4. Effect of contention on total core energy with different core count management schemes. Energy gains obtained by different schemes decreased by the contention in different amounts. Increase in total core energy consumption with RR scheme is around 13% while in SS, this increase is almost 30%.

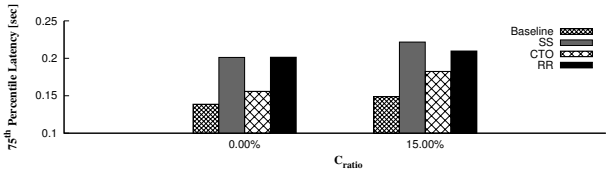

Figure 5. Effect of contention on 75^{th} percentile latency with different core count management schemes. Latency is increased in all schemes under contention. The increase in CTO is higher than the increase in SS because more cores can be busy at the same time, hence are affected by the contention more. The increase in RR is higher than the baseline case because of the similar reason.

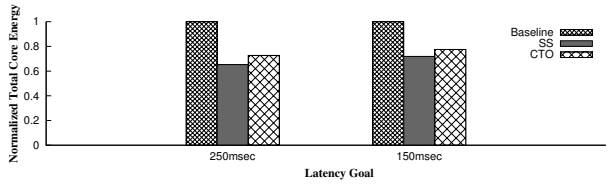

Figure 6. Effect of latency goal on total core energy. With the lower latency goal, total core energy consumptions increase since the idle energy increase with the more number of ON cores. The increase in idle energy increases the energy spent by SS around 10% and CTO around 6%.

are distributed across many servers, the contention effect is less dominant. For example, under contention, increase in total core energy consumption with RR is around 13% while in SS, this increase is almost 30%. When there is no contention effect, SS and RR have the same total core energy consumption, however RR becomes less than SS when the contention takes place.

Figure 5 shows that the latency is increased in all schemes under contention. The increase in CTO is higher than the increase in SS because more cores are available and can be busy at the same time. For example, if there are jobs waiting in the queue in the the SS scheme, CTO schedules these jobs on the extra cores. This will increase the service time because of the contention but at the same time, decrease the queuing latency. The result shows that the increase ratio is around 20% in CTO while the overall latency is still lower than SS where the increase ratio is around 10%. Similarly, the increase in RR is around 4% where in the Baseline case, this ratio is around 7%. Thus, we see that when there more available cores, latency increases because of contention can be higher than the decrease in queuing time coming from the parallel processing of more jobs.

4.3 Effect of Latency Goal

When we have a tighter latency goal, it may be necessary to turn on more cores. Hence saved energy from idle cores de-

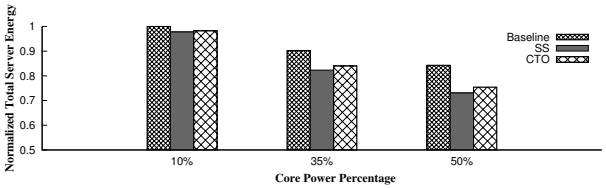

Figure 7. Effect of different core power percentages on total server energy. As the core power percentage increases, the energy gain obtained by core count management techniques increase.

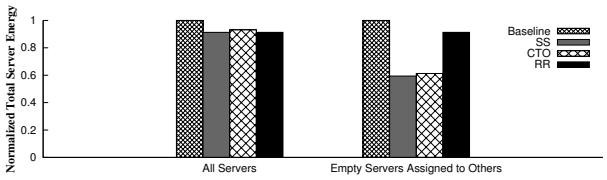

Figure 8. Normalized total server energy when all active servers are dedicated to the main application and when the empty servers are assigned to other jobs. Making use of empty servers results in 41% decrease in energy with SS method.

creases. Figure 6 shows how different schemes are affected by a tighter latency goal. Total core energy consumption is around 10% higher with SS and 6% higher with CTO when the latency goal is 150 msec instead of 250 msec. SS increases more because in CTO, we have a coarser grain granularity in choosing ON core count. Effect on RR is not showed separately, since RR has the same core energy characteristics with SS under when contention is not taken into account.

4.4 Effect of Power Breakdown

Our core count management techniques reduce the energy consumption of the cores. Thus, the effect of our techniques on total power consumption depends on $\frac{P_{cores}}{P_{total}}$. As the core power percentage increases with respect to total power, we will get relatively more energy gain with our techniques. We assumed that for a quad-core machine, 50% of power goes to processor and 30% of this power goes to uncore part. Thus, $\frac{P_{cores}}{P_{total}}$ becomes 35% in this case. In order to see the the effect of different breakdowns, we used a lower (10%) and a higher (50%) core power percentage. Again, it is assumed that there is no contention between the cores. Results are shown in Figure 7. When core power ratio is 35%, the energy savings is 9% with SS. The energy savings decreases to 3% when the core power ratio is 10%. However when we increase core power ratio to 50%, the energy gain increases to 15%. Similar effect can be seen with CTO. Energy savings becomes 2%, 7% and 11% as we increase the power breakdown from 10% to 50%.

4.5 Effect of Server Management

Until now, we talked about the load consolidation in core level. In this part, we want to discuss about consolidating an application to a small subset of servers and use the remaining servers to other applications.

For a single application, when we apply our core count management techniques, some servers stay empty for some periods. When server turn off is not allowed, extra energy consumption by non-productive servers can not be eliminated. As an alternative, empty servers can be assigned to other applications when they are empty. Since we change the ON core count at the beginning of each control period T, we can assign the empty servers to other applications for at least one period. This method will help us to see the distinction of the SS and RR in terms of power because the number of servers they chose are different.

Figure 8 shows the effect of assigning empty servers to other applications on total energy consumption. Energy spent by other applications will not add up to the total spent by the main application. It is assumed that when the server is needed again for the main application, it can be in use instantly. Other jobs can be chosen as low priority jobs so they will not be affected by this change. We see that energy gain s obtained by SS method increases from 9% to 41% with this method. Energy gain from CTO method is also affected by this method and increased to around 39%. However, we do not see much difference in RR because of the fact that in RR, ON cores are chosen from different servers as much as possible. RR does not allow having many completely empty servers, hence energy gain is not affected much for this load. From this figure, it is clear that SS has more advantages in terms of power when the cores do not have contention. We already showed that the latency is same for these two cases.

5. Discussion and Results Summary

By using stochastic simulation, we discuss the effect of cross-server core count management on energy and performance. For Facebook workload, total core energy savings of 35% can be obtained while satisfying the same latency goal. Depending on the cores contribution to total power, total energy savings of 3% to 15% are likely, with even higher savings in some cases.

We can list the key findings in the simulations as follows;

- In general, energy saving is higher at low load rates because cores tend to stay longer in idle at low load rates.
- SS and RR shows similar core energy consumption and latency results when contention effect is ignored. However, when contention is taken into account, we see that RR has higher core energy saving potential than SS since effect of contention is minimized by distributing ON cores to more servers.
- The value of contention ratio has a bigger effect on SS than on RR. This comes from the fact that SS uses cores from the same server as much as possible, hence the effect of contention is higher.
- If use of empty servers by other applications are allowed, SS and CTO has great server energy saving potentials. For example, such offloading can result in 41% energy savings for the main application when the energy consumed by other applications are not accounted.

6. Conclusion

This work uses extensive stochastic simulation and real-world Facebook workloads to explore a range of core count management issues in multi-server CMP systems. In addition to selecting *how many* cores should be actively executing a particular workload, CMP-based systems mean that it

is important to also consider how they should be mapped across the various CMPs.

When one accounts for core-to-core contention for on-chip interconnect and cache resources, round-robin mapping schemes can show energy and performance advantages over other schemes that consolidate ON cores onto a single CMP. On the other hand, same-server schemes more effectively free some of the servers from the workload, so if other uses (e.g. lower-priority applications) for the free servers can be found, they become preferable. Overall, these issues highlight the need for reexamining core-count and core-mapping decisions in the face of widespread CMP servers. We see this as a first step towards more detailed studies that evaluate techniques on real servers and that explore hardware alternatives for further improving power leverage and energy proportionality.

References

[1] L. A. Barroso. The price of performance. *Queue*, 3, 2005.

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

[3] L. Bertini, J. C. B. Leite, and D. Mossé. Power optimization for dynamic configuration in heterogeneous web server clusters. *J. Syst. Softw.*, 83, 2010.

[4] D. Chandra, F. Guo, S. Kim, and Y. Solihin. Predicting inter-thread cache contention on a chip multi-processor architecture. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, 2005.

[5] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001.

[6] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.

[7] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005.

[8] S. Cho and L. Jin. Managing distributed, shared l2 caches through os-level page allocation. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[9] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, 2007.

[10] J. Hamilton. Perspectives. blog on data center design and management, 2011. http://perspectives.mvdirona.com/.

[11] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design*, 2007.

[12] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.

[13] R. Illikkal, V. Chadha, A. Herdrich, R. Iyer, and D. Newell. Pirate: Qos and performance management in cmp architectures. *SIGMETRICS Perform. Eval. Rev.*, 37, 2010.

[14] ITRS. International technology roadmap for semiconductors, 2007. http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[15] R. Kumar and G. Hinton. A family of 45nm ia processors. In *Solid-state circuits conference - Digest of Technical Papers*, 2009.

[16] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis. Power management of datacenter workloads using per-core power gating. *IEEE Comput. Archit. Lett.*, 8, 2009.

[17] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron. Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*, 2002.

[18] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram. Guarded power gating in a multi-core setting. In *Workshop on Energy Efficient Design. (Associated with ISCA).*, 2010.

[19] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram. A case for guarded power gating in multi-core processors. In *The 17th IEEE International Symposium on High Performance Computer Architecture (HPCA-17)*, 2011.

[20] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. In *Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, 2009.

[21] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. *Dynamic cluster reconfiguration for power and performance*. Kluwer Academic Publishers, 2003.

[22] K. Rajamani, F. Rawson, M. Ware, H. Hanson, J. Carter, T. Rosedahl, A. Geissler, G. Silva, and H. Hua. Power-performance management on an ibm power7 server. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, 2010.

[23] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely. Dynamic resource allocation and power management in virtualized data centers. In *IEEE/IFIP Network Operations and Management Symposium*, 2010.