all five workloads. Due to space constraints, we focus the validation discussion on programs *memcached* and *SP*, which are the programs with largest relative error between model and measurement.

Program	Error	
	Time	Energy
EP	1.0%	8.7%
FT	9.3%	9.9%
SP	11.1%	11.3%
memcached	11.4%	10.7%
blackscholes	9.4%	8.2%

Table 5: Relative Error between Model and Measurement

Before the validation experiments, we applied two baseline runs for each program, during which we collect information about the useful work, memory requests and I/O requests, as described in the previous section. To validate the model, we predict the execution time $T(n, f)$ and energy $E(n, f)$ considering the number of cores n and clock frequency f fixed throughout the program execution. We compare this prediction against measurements of $T(n, f)$ and $E(n, f)$. To change the number of cores that the program is using we change the number of worker threads of the programs. For each program, we validate four cores and 13 core frequencies, giving 52 core-frequency configurations. Table 5 summarizes the relative error between measurements and predicted values, averaged over all core-frequency configurations. Overall, the error across all experiments is around 9%, and 70% of the predicted values are under 5% error.

We identify three factors that affect the accuracy of the model. The most significant source of error comes irregularities during execution. For example, *memcached* incurs more instructions on higher core frequencies, which are caused by a polling mechanism used to monitor the network sockets. This significantly increases the energy used, but does not reduce the execution time. This increase causes our model to underestimate by up to 23% the CPU cycles incurred by *memcached* on one core. The second cause for model inaccuracy is the accuracy of the system characterization parameters. In particular, the power values for active cycles, stall cycles and idleness differ by up to 20mW. This variability translates into a slight underestimate of the average power, especially for configurations with low frequencies or low core counts. Third, the measured values of execution time and energy show a variation of 3 to 11%. Figures 6 and 7 show the validation of execution time and energy used for *SP* and *memcached*.

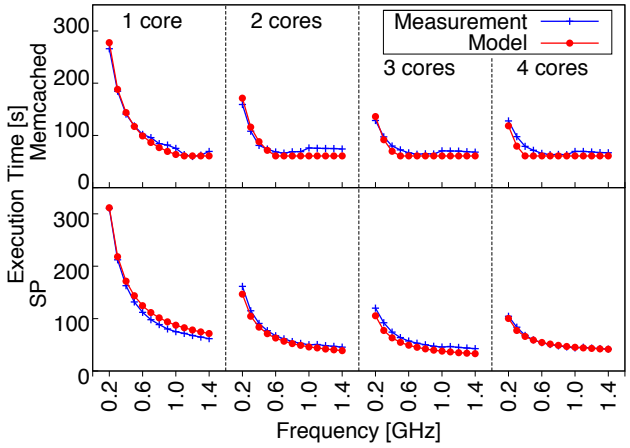

Figure 6: Execution Time Validation

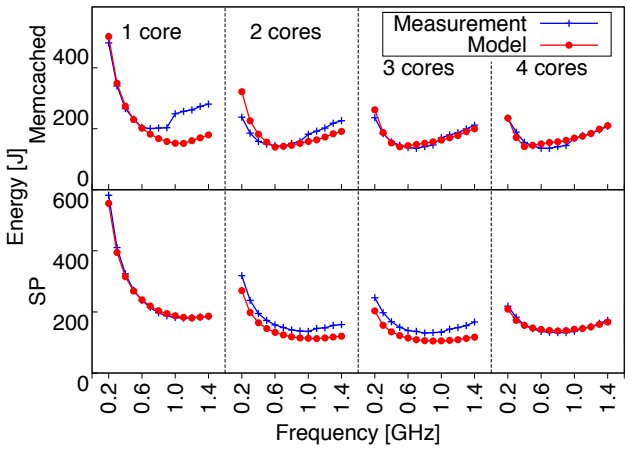

Figure 7: Energy Validation

4. ANALYSIS

We present three studies performed using our model. Firstly, we predict the configuration of number cores and clock frequency, (n, f) that achieves the minimum execution time without wasting energy, and discuss the energy savings that can be achieved. Secondly, we show that some types of workloads incur higher energy cost on low-power multicore than on traditional x64 systems. Thirdly, we show that contrary to intuition, the most energy-efficient way to balance the utilization of core, memory and I/O resources is by adding more memory and I/O resources, rather than by turning off idle cores.

4.1 Optimal Core-Frequency Configuration

We apply the model to predict the configuration of core counts and clock frequency that achieves the minimum execution time. If several configurations achieve execution times within 5% to the minimum, we chose the one with the minimum energy usage, considering that all cores are kept powered on. The results in table 6 shows that only workloads consisting entirely of arithmetic instructions make full use of all four cores. In contrast, all memory- or I/O-bounded programs achieve best performance on less than full core count and around half of maximum core frequency.

Program	Bottleneck	Configuration			
		Min. Time		Min. Energy	
		n	f [GHz]	n	f [GHz]
EP	Cores	4	1.4	4	1.4
FT	Memory	3	1.4	3	0.7
SP	Memory	3	1.4	3	0.9
memcached	I/O	2	0.7	2	0.7
blackscholes	Cores	4	1.4	4	1.4

Table 6: Minimum Time and Energy Configurations on Default Configuration of up to 4 Cores and 1.4 GHz

The predictions for optimal core-frequency configuration allow significant energy savings if un-necessary cores are turned off. For example, for *memcached*, by selecting the minimum time configuration we can power off two cores. Unfortunately, the operating system on our system does not allow selective power off of a subset of cores, and thus we cannot measure directly the energy savings. However, using figures from related work [23, 33], by shutting down two out of the four cores we can estimate a reduction of processor power between a conservative 25% and an optimistic 50%. With these figures, applying the configuration predicted by our model allows for a reduction in total energy savings between

13% and 31%, and without compromising the execution time performance.

4.2 Impact of Memory Bottleneck

We analyze the energy cost of low-power multicores that have limited memory bandwidth. We compare the execution of memory-bounded *SP* on low-power Cortex-A9 with traditional x64 multicore.

The x64 system used is Intel Xeon X5650 at 2.67 GHz, dual processor with 12 cores/24 hardware threads, 24 MB of L3 cache and two memory controllers, each with two memory channels. We ran program *SP* with a large input size of 400 iterations on a grid of 162^3 (input size *C* from NPB). The input results in a working set large enough to exceed the caches of both systems, but fits into 1 GB of main memory.

We apply our prediction for execution time on the low-power multicore for all core-frequency configurations, and observe that the minimum execution time exceeds 17,000 seconds. In contrast, the measured execution time on the x64 system is 330 seconds, using all cores at maximum frequency. The large gap between execution times appears because the Intel system is equipped with much larger caches, and thus, incurs 15 times less cache misses. Furthermore, the main memory bandwidth is around 8 times higher, while the bandwidth of the caches are ten times (for L1) and four times (when comparing Intel L3 to ARM L2) higher. The average power used by the Intel system (with disks turned off) is around 210W.

We extend this analysis for datacenters, factoring the additional power required for power conversion and cooling the systems. The Power Usage Effectivness (PUE) of a datacenter measures the total power required to deliver 1 Watt of IT power. From literature, we identify two PUE bounds: the lower bound is PUE=1.13, in a Google datacenter [3], while an the upper bound is 1.89 [4]. Figure 8 shows the predicted values of energy consumption on low-

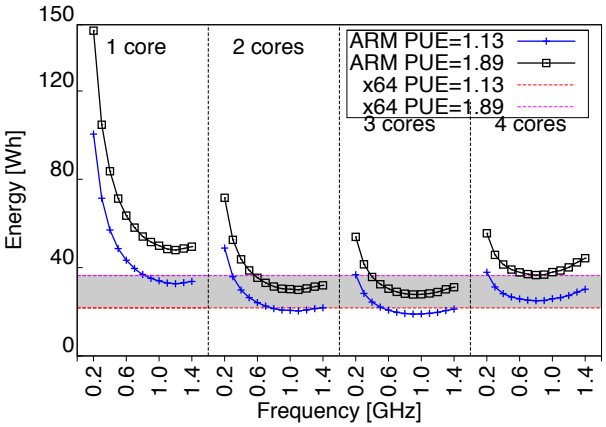

Figure 8: *SP* Energy Usage: ARM Cortex-A9 vs x64

power multicore, for all configurations of cores-frequency, compared with an execution on all x64 cores at maximum frequency, for PUE between 1.13 and 1.89. Few ARM configurations manage to achieve lower energy cost than x64, with less than 7% energy reduction, but at a cost of incurring execution time more than 50 times higher. However, the x64 execution is not energy efficient because all cores are used at maximum frequency even though the memory is the bottleneck. We conclude that memory-bounded programs can be unsuitable for low-power multicores, even if optimizing the core-frequency configuration to achieve best performance at minimum energy cost.

4.3 Energy Proportionality of Server Workloads on Low-power Multicores

Energy proportionality refers to the ability of a system to consume power proportional to their performance [7]. ARM multicores typically have good energy proportionality when used as mobile computers, due to their sleep states and low-power operation [33]. However, our previous analysis shows that resource imbalances lead to large energy wastage in server workloads. Thus, leveraging on the idea that ARM systems are highly configurable, we apply our model to understand how to improve the energy proportional executions of server workloads.

As the key to improving energy proportionality is system balance [5], we apply our model to predict the performance of program *memcached* under different hardware configurations that balance the system resources. Figure 9 shows the response times of different resources for the original hardware configuration (100Mbps Ethernet, one memory controller), when using two active cores. This number of cores is selected as it achieves the best performance at minimum energy cost. For small core frequencies, the CPU work

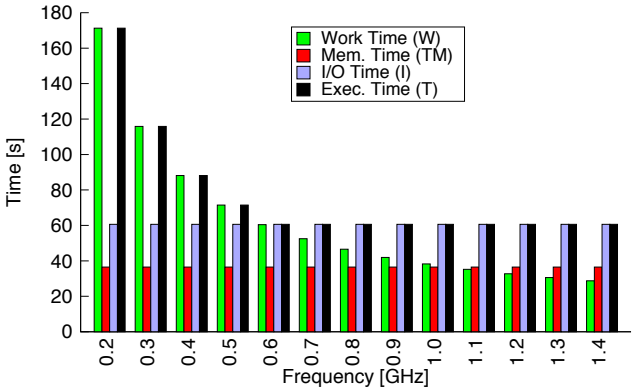

Figure 9: *Memcached* Response Times

time ($W = \frac{w+b}{f}$) is the bottleneck, but at 600MHz the CPU response time matches the I/O response time. Beyond 600MHz, the I/O bandwidth becomes the bottleneck, and the exection time does not reduce anymore.

We analyze the performance impact of replacing two system components. First, the 100Mbps Ethernet is replaced with 1GBps Ethernet, without modifying any other component. In this analysis, we consider a gigabit Ethernet adapter with a power consumption of 600mW, which is typical for a power-efficient network card. The I/O time for *memcached* is composed of transfer time $I_T = 56$s and blocking time $T_B = 5.1$s. With a Gbit Ethernet, the total I/O time becomes $I = 10.7$s. However, because the I/O device is memory mapped, we consider that the gigabit Ethernet will utilize 125MB/s out of the 800MB/s memory bandwidth. Thus, s_M increases from 38 ns to 45ns. Applying Equation 16, T_M increases from 36.5s to 47s. Thus, the effect of moving to 1GBit Ethernet is a reduction from of execution time from $T = 61$ s to $T = 47$ s, and the system bottleneck becomes the memory. Due to the increase in I/O power by 400mW and due to the increase in stall cycles, the average power increases by approximately 500mW. Figure 10 shows that total energy decreases by switching to a Gigabit Ethernet because the decrease in execution time offsets the increase in average power. Since the new bottleneck is the memory, we consider next the impact of doubling the effective memory bandwidth (ARM Cortex-A9 systems can be configured to up to quad-memory channels, while the next generations ARM Cortex-A15 and ARM Cortex-A50 support more outstanding memory requests and can be configured to use LPDDR3). With the double memory bandwidth, the memory

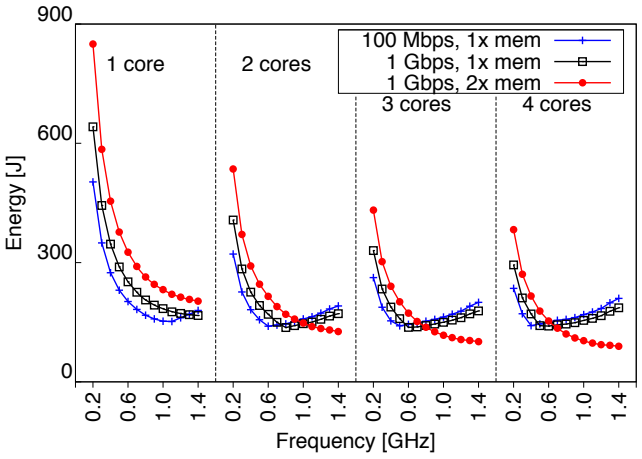

Figure 10: *Memcached* with 1 Gbit Ethernet and Double Memory Bandwidth

response time drops to 18.1s, and the system bottleneck becomes the core. We consider a pessimistic scenario, where power consumption is the quadruple of the original memory system (100mW idle memory power and 1W active memory power). However, the energy consumption still decreases by more than 50%, and *memcached* becomes CPU-bounded. It achieves best performance and minimum energy when using all cores at full frequency.

This analysis showed that reducing the imbalance among core, memory and I/O leads to lower energy usage. However, counter to intuition, we showed that balancing the resources by adding *more hardware resources* is the key to improving energy proportionality, even if this results in higher average power consumption. Considering that future ARM systems such as ARM Cortex-A15 and the 64-bit ARM Cortex-A50 family target much improved memory and I/O throughput, it is expected that future multicore systems based on low-power ARM multicores will deliver better energy-proportionality than current ARM Cortex-A9, even at a cost of higher power usage.

5. RELATED WORK

Previous work on understanding the performance and energy efficiency of server workloads can be broadly classified based on three criteria: (i) level of analysis (i.e. microarchitecture level, system level or cluster/datacenter level performance) (ii) workload characteristics and (iii) approach for understanding the energy usage. We discuss each category, contrasting our approach with previous research.

Level of Analysis. A large body of related work considers using voltage and frequency scaling to balance core and memory performance in single-core workloads [26, 28, 39, 40]. However, with the reduction of transistor size with each technological generation, an increasingly large component of the energy cost is caused by fixed leakage power required to keep the components powered on [23]. As such, DVFS techniques are facing diminishing returns, as they are able to reduce an increasingly small fraction of the energy cost [22]. At the other end of the spectrum, in clustered systems or datacenters, a dominant performance issue is the energy wastage due to low processor utilization. Research in this area has focused on understanding how to increase utilization by workload scheduling [15, 38] or the impact of adjusting system power [13, 19]. At multicore level, a large body of work addresses the question of how the throughput of multicore system is affected by the dichotomy of few *brawny* cores versus many *wimpy* cores, given a fixed performance-density budget [12, 17, 29]. However, these studies do not consider the impact of off-chip resources such as main memory or I/O periferals.

Our findings are orthogonal with previous work. Because we focus on understanding the impact of memory and I/O constraints on execution perfromance, our model can predict the optimal number of cores for a workload, our model allows a better understanding of the impact of work-aggregation or different scheduling techniques in clusters of multicore servers. Furthermore, our work enables a greater reduction of energy over DVFS techniques developed for single-core systems, because it can reveal the number of unnecessary cores that can be turned off.

Workload Characteristics. The suitability of low-power systems to achieve energy-efficient executions of different type of workloads has been studied in the past for many types of systems [25, 35], including ARM [33] and Intel Atom systems [5]. Previous work on understanding the power and performance of ARM systems focused mostly on embedded [33] or mobile workloads [8], which have mostly CPU and GPU requirements. Closer to our targeted workloads is FAWN, a cluster of low-power “wimpy" AMD Geode nodes [5], designed for energy-efficient I/O intensive applications. However, FAWN nodes are single-core 500 MHz processors, thus they are severely bottlenecked by the CPU processing ability and by I/O storage bandwidth. In contrast, the multicore Cortex-A9 systems addressed by us are bottlenecked by memory or network speed. Furthermore, our analysis includes non-I/O bounded HPC programs that are more impacted by the “memory wall”. Work that addresses specifically data-intesive workloads are characterized by [6] in terms of traffic characteristics and by [32] for opportunities of power savings in datacenters. Our work, although using much narrower traffic patterns, validates the conclusion that system balance is key for energy proportionality.

Modeling Approaches. Previous analytical models for predicting execution time include trace-driven analysis for predicting the speedup and speedup loss due to data-dependency [37], the impact of network communication [20] or memory contention of HPC and real-world programs in traditional multicore systems [27]. In contrast, our focus is not on performance loss, but rather on understanding energy wastage due to resource imbalance. A significant body of research has addressed analytical modeling of computer power and energy using models derived from first-order principles or linear regression [10, 21, 24, 26, 28]. Closest to our model is the approach presented by Curtis-Maury et al. [10]. They derive an empirical analytical model for predicting the impact of program concurrency and core frequency on energy usage of HPC programs in traditional multicores. Their approach is based on linear regression over measured data acquired over many training runs, but without considering the impact of resource contention. In contrast, we provide closed-form equations that explicitly model the impact of number of cores, allowing us to study directly the impact of multicore on resource imbalances. More importantly, our analysis suggests that increasing power usage can lead to energy reductions, if the added power contributes to balancing the resource utilization.We share the methodology of modeling the power usage by correlating static power characteristics with hardware events counters with [24, 26, 28]. We do not use linear correlation methods, but derive close-form equations for the power and energy consumption. Our work furthers these studies by considering the case where multiple cores execute the same workload, modeling the impact of multicore on resource contention. This increases the predictive value of our models. Furthermore we use validation against direct measurement of server workloads covering HPC, web-hosting and financial computing.

6. CONCLUSIONS

This paper proposes a trace-driven analytical model for understanding the energy usage of server workloads on low-power multicore systems. We model the effects of multicore on achieving energy-efficient executions of representative server workloads covering high performance computing, web hosting and financial computing. The key idea is the modeling of the overlap between resource demand in a program. Since the power consumed is the product of power utilization and execution time, the model first estimates the execution time of a program by considering the overlap between response times incurred by cores, memory and I/O resources. CPU time, which accounts for both cores and memory response time, is modeled as an M/G/1 queuing system. Our workload characterization shows that bursty memory traffic fits a Pareto distribution and non-bursty memory traffic can be modeled using an exponential distribution. Validation shows a relative error of 9% between model and measured execution time and energy. Applications of our model to analyze the optimal core-frequency configuration, impact of memory bottleneck and energy proportionality in multicore systems reveal a number of insights. We observe that low-power multicores may not always deliver energy-efficient executions for server workloads because large imbalances between cores, memory and I/O resources can lead to under-utilized resources and thus contribute to energy wastage. Next, resource imbalances in HPC programs may result in significantly longer execution time and higher energy cost on ARM Cortex-A9 than on a traditional x64 server. In this instance and without compromising execution time, our model predicts core frequency configurations that balance the resources with energy reduction of up to one third. Finally, we show that higher memory and I/O bandwidths can improve both execution time and energy utilization, even if it means higher power usage. Thus, it is expected that ARM Cortex-A15 and the ARM Cortex-A50 family, which target larger memory and I/O bandwidths, will deliver more energy-efficient servers than currently available ARM Cortex-A9.

Acknowledgements

We thank the anonymous reviewers and our shepherd for their constructive comments, and Lavanya Ramapantulu for helping us to uncover some aspects of the ARM Cortex-A9 processors.

7. REFERENCES

[1] *Chip maker Calxeda receives $55 million to push ARM chips into the data center*, Oct 2012. http://www.webcitation.org/6BSIjQzCM.

[2] *Dell Reaches for the Cloud With New Prototype ARM Server, PCWorld Magazine*, May 2012. http://www.webcitation.org/6BVBj0Oyz.

[3] *Google Data Center Efficiency: How We Do It*, Oct 2012. http://www.webcitation.org/6C8PjIMYd.

[4] *Uptime Institute 2012 Survey*, Oct 2012. http://uptimeinstitute.com/2012-survey-results/.

[5] D. G. Andersen et al. Fawn: a fast array of wimpy nodes. *Proc of SOSP*, pages 1–14, 2009.

[6] B. Atikoglu et al. Workload analysis of a large-scale key-value store. *Proc of SIGMETRICS/PERFORMANCE*, pages 53–64, 2012.

[7] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, Dec. 2007.

[8] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. *Proc of USENIX ATC*, pages 21–21, 2010.

[9] J. Corbet et al. *Linux Device Drivers, 3rd Edition*. O'Reilly Media, Inc., 2005.

[10] M. Curtis-Maury et al. Prediction models for multi-dimensional power-performance optimization on many cores. *Proc of PACT*, pages 250–259, 2008.

[11] F. M. David et al. Context Switch Overheads for Linux on ARM Platforms. *Proc. of ExpCS*, 2007.

[12] J. D. Davis et al. Maximizing cmp throughput with mediocre cores. *Proc of PACT*, pages 51–62, 2005

[13] A. Gandhi et al. Optimal power allocation in server farms. *Proc of SIGMETRICS*, pages 157–168,2009.

[14] A. Gandhi et al. Are sleep states effective in data centers? *Proc of IGCC*, pages 1–10, 2012.

[15] D. Gmach et al. Workload analysis and demand prediction of enterprise data center applications. *Proc of IISWC*, pages 171–180, 2007.

[16] J. L. Hennessy and D. A. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach*. 2006.

[17] M. Hill and M. Marty. Amdahl's Law in the Multicore Era. *Computer*, 41(7):33–38, 2008.

[18] U. Hölzle. Brawny cores still beat wimpy cores, most of the time. *IEEE Micro*, 30(4), 2010.

[19] T. Horvath and K. Skadron. Multi-mode energy management for multi-tier server clusters. *Proc of PACT*, pages 270–279, 2008.

[20] Y. Hu et al. I/o scheduling model of virtual machine based on multi-core dynamic partitioning. *Proc of HPDC*, pages 142–154, 2010.

[21] V. Kumar and A. Fedorova. Towards better performance per watt in virtual environments on asymmetric single-isa multi-core systems. *SIGOPS Oper. Syst. Rev.*, 43(3):105–109, July 2009.

[22] E. Le Sueur and G. Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. *Proc of HotPower*, 2010.

[23] E. Le Sueur and G. Heiser. Slow down or sleep, that is the question. *Proc of USENIX ATC*, 2011.

[24] A. W. Lewis et al. Runtime energy consumption estimation for server workloads based on chaotic time-series approximation. *ACM Trans. Archit. Code Optim.*, 9(3):15:1–15:26, Oct. 2012.

[25] K. Lim et al. Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments. *Proc of ISCA*, pages 315–326, 2008.

[26] M. Y. Lim et al. Softpower: fine-grain power estimations using performance counters. *Proc of HPDC*, pages 308–311, 2010.

[27] F. Liu et al. Understanding How Off-chip Memory Bandwidth Partitioning in Chip Multiprocessors Affects System Performance. *Proc of HPCA*, 2010.

[28] C. W. Lively et al. Power-aware predictive models of hybrid (mpi/openmp) scientific applications on multicore systems. *Computer Science - R&D*, 27(4):245–253, 2012.

[29] P. Lotfi-Kamran et al. Scale-out Processors. *Proc of ISCA*, pages 500–511, 2012.

[30] K. Malladi et al. Towards energy-proportional datacenter memory with mobile dram. *Proc of ISCA*, pages 37–48, 2012.

[31] D. Meisner et al. PowerNap: Eliminating Server Idle Power. *Proc of ASPLOS*, pages 205–216, 2009.

[32] D. Meisner et al. Power management of online data-intensive services. *Proc of ISCA*, pages 319–330, 2011.

[33] R. Mijat. System level benchmarking analysis of the cortextm-a9 mpcore. *ARM Connected Community Technical Symposium*, 2009.

[34] D. Molka et al. Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System. *Proc of PACT*, pages 261–270, 2009.

[35] A. S. Szalay et al. Low-power Amdahl-balanced blades for data intensive computing. *SIGOPS Oper. Syst. Rev.*, 44(1):71–75, Mar. 2010.

[36] Y. C. Tay. *Analytical Performance Modeling for Computer Systems*. Synthesis Lectures on Computer Science. Morgan & Claypool Publishers, 2010.

[37] B. M. Tudor and Y. M. Teo. A practical approach for performance analysis of shared-memory programs. *Proc of IPDPS*, pages 652–663, 2011

[38] A. Verma et al. Server workload analysis for power minimization using consolidation. *Proc of USENIX ATC*, 2009.

[39] M. Weiser et al. Scheduling for reduced cpu energy. *Proc of OSDI*, 1994.

[40] A. Weissel and F. Bellosa. Process cruise control: event-driven clock scaling for dynamic power management. *Proc of CASES*, pages 238–246, 2002.