Markus Hähnel, Waltenegus Dargie, Alexander Schill

Analysis of the Scope of Dynamic Power Management in Emerging Server Architectures

Erstveröffentlichung in / First published in:

International Conference on Computer Communications and Networks (ICCCN). Vancouver, 31. Juli – 03. August 2017. IEEE. ISBN 978-1-5090-2991-4.

DOI: https://doi.org/10.1109/ICCCN.2017.8038529

Final edited form was published in "International Conference on Computer Communications and Networks (ICCCN). Vancouver, 2017". ISBN 978-1-5090-2991-4.
https://doi.org/10.1145/2236584.2236587

Analysis of the Scope of Dynamic Power Management in Emerging Server Architectures

Markus Hähnel, Waltenegus Dargie and Alexander Schill
Chair for Computer Networks, Faculty of Computer Science, Technical University of Dresden, 01062 Dresden, Germany
E-Mail: {Markus.Haehnel1, waltenegus.dargie, alexander.schill}@tu-dresden.de

***Abstract*—The architectures of large-scale Internet servers are becoming more complex each year in order to store and process a large amount of Internet data (*Big Data*) as efficiently as possible. One of the consequences of this continually growing complexity is that individual servers consume a significant amount of data even when they are idle. In this paper we experimentally investigate the scope and usefulness of existing and proposed dynamic power management strategies to manage power at core, socket, and server levels. Our experiment involves four dynamic voltage and frequency scaling policies, three different workloads having different resource consumption statistics, and the activation and deactivation of different sockets (packets) of a multicore, multi-socket server. Moreover, we establish a quantitative relationships between the workload (w) and the estimated power consumption ($\hat{p}$) under different power management strategies to make a quantitative comparison of the different strategies and server configurations.**

***Index Terms*—Dynamic power management, DVFS, energy-efficient computing, multicore processor**

I. Introduction

The pace at which the amount and complexity of data increases that are presently stored in and processed and transferred by large-scale Internet servers and data centres has induced the research community and the IT industry to label the phenomenon as the *Big Data* phenomenon. In order to deal with this phenomenon processor, server, and network architectures are undergoing a considerable evolution at a fast rate. As far as multicore processor architectures are concerned, the efficient integration of a large amount of cores in a single chip, complex shared cache hierarchies, fast and efficient simultaneous multi-threading, non-uniform cache architecture, and advanced branch prediction strategies, among others, are being endeavoured both by the industry and the research community. The idea is to significantly increase the number of instructions that can be fetched, decoded, dispatched, issued, and executed simultaneously [1]. Likewise, novel server (mainboard) architectures are being developed to enable the fast and efficient integration of multiple multicore processors and random-access memories in a NUMA architecture.

The energy-proportionality (the ratio of consumed energy to work done) of these architectures has in general been steadily improving because the resource and power utilisation of a server can be adapted, at least theoretically, at core, processor, socket, and server levels. Moreover, presently available processors have more operation frequencies and voltages to scale than their predecessors. Regardless of these possibilities, however, due to the proportionality between the capacity and power consumption of the hardware subsystems, the average power consumption of typical Internet servers is still considerably high.

Both the research community and the industry have been working on dynamic power management strategies for decades. For example, the Advanced Configuration and Power Interface (ACPI) specification defines an open industry standard for configuring, managing, and monitoring the power consumption of hardware subsystems in servers at runtime. Likewise, a substantial body of work exists on dynamic voltage and frequency scaling (DVFS) to adapt the operation frequency and voltages of processors in accordance with the expected idle time duration between two jobs and job completion deadlines specified by applications. Nevertheless, the scope and usefulness of these strategies depend on how quickly and how well they adapt to the ever evolving processor and server architectures. Two of the premises of dynamic power management are that the workload of a server is not constant; rather, it fluctuates over time. Secondly, different workloads have different resource demands (for example, some workloads are predominantly IO-bound, some are CPU-bound, and some are memory-bound). Consequently, an efficient power management strategy attempts to balance the demand for and the supply of resources in a server by switching-off idle resources and by appropriately scaling the operation speed of underutilised resources [2].

In this paper, we experimentally investigate the performance of different dynamic power management strategies using a medium-scale multicore, multiprocessor server architecture. The server consists of four CPU sockets which are interconnected in a mesh topology. Each socket houses a quad core Intel Xeon E5-4603 processor (having 4 physical and 8 logical cores), 16

Final edited form was published in "International Conference on Computer Communications and Networks (ICCCN). Vancouver, 2017". ISBN 978-1-5090-2991-4.
https://doi.org/10.1145/2236584.2236587

GB memory in a NUMA architecture (4 GB for each socket), 10 Gbps Intel NIC, and an XFS-formatted 6 TB hard disk RAID with a theoretical sustained data rate of ∼465 MB/s (the data rate reduces to ∼300 MB/s during heavy contention). We generated different CPU-bound stochastic workloads to dynamically vary the inter job interval (idle state) at the processors. The rest of the paper is organized as follows: In Section II, we summarize related work; in section III, we introduce our experiment set up; in Section V, we present and discuss experiment results and share the insight we obtained from the experiment results. Finally, in Section VI, we give concluding remarks and outline future work.

II. Related work

The idle power consumption of Internet servers becomes significantly high as their complexity and capacity increases. Consequently, different approaches have been proposed and implemented to deal with idle power consumption. The Advanced Configuration Power Interface (ACPI) specification serves as a platform-independent conceptual framework for the seamless integration of different dynamic power management strategies into an operating system for managing the power consumption of the different subsystems of a server. It has three conceptual layers, the software, the ACPI, and the hardware layer. The hardware layer consists of the hardware subsystems the power consumption of which should be managed. Each hardware subsystem defines different operating modes having different power-utility characteristics. The ACPI layer exposes to the dynamic power management strategies the parameters and functions of each hardware subsystems that can be modified and invoked. The parameters are exposed as ACPI registers and function pointers. The software layer consists of two vital power management components, the Operating System-directed configuration and Power Management (OSPM) and the ACPI driver (AML interpreter). The former contains an instance of a dynamic power management strategy for each dynamic reconfigurable hardware subsystem. Therefore, an instance of OSPM, once initiated, takes an exclusive control of all aspects of power management and device configuration, as far as a hardware subsystem is concerned. The AML interpreter (ACPI driver) translates the OSPM language into a language understandable by the functions that interact with the managed hardware subsystem.

Similarly, emerging computer architectures aim to provide rich dynamic power management opportunities. For example, the IBM's PowerX multicore architecture aims to integrate flexible and ACPI compatible dynamic power management features into present and future multicore, multiprocessor server architectures [3]. One of these features is providing separate phase-locked loops (PLL) to individual cores so that their voltage and frequency can be scaled independently. Likewise, some generations of AMD processors provide separate PLL for each core [4]. The increasing number of cores within a single processor package makes dynamic power management essential not only to save energy but also to reduce the thermal effect of high circuit density, which inhibits cores from operating with optimal performance. A high thermal design power (TDP) not only reduces the performance of individual cores but also may damage the whole system. The effect of TDP can be minimised by employing an active cooling mechanism; however, active cooling also introduces its own power consumption. Hanumaiah and Vrudhula [5] aim to minimise the effects of TDP by combining different approaches such as performance-per-watt efficiency as a trade-off between performance and power consumption, DVFS, thread migration, and active cooling.

Maiti *et al.* [6] investigate the consequences of configuration mismatch between core frequencies and core states and propose a framework for core selection, thread-to-core mapping and DVFS. The aim is to select the best distribution of jobs and the appropriate DVFS. Unlike the approach in [5], here optimisation serves one of the two purposes: Either the energy of a server is minimised under performance constraints or its performance is maximised under power constraints. Unfortunately this model considers multicore processors and not multi-socket systems. Brihi and Dargie [7] and Dargie [8] experimentally investigate the scope and usefulness of dynamic voltage and frequency scaling in different AMD and Intel multicore processors dealing with IO- and CPU-bound workloads. One of their contributions is directly measuring the power consumption of the processors by intercepting their power lines and by establishing a relationship between power consumption and CPU utilisation. Similarly, Rossi *et al.* [9] investigate the scope of three DVFS policies (performance, on-demand, power-save) and propose a multi-linear regression model to estimate the power consumption of a processor based on its CPU utilisation, frequency set by a DVFS policy, and time to finish a job.

III. Setup

Most existing dynamic power management strategies aim to minimise the idle states of some of the subsystems of a server, because they consume a significant amount of power even when they are idle. For frequency sensitive subsystems (such as processors), the idle state can be minimised by letting them operate at a slower speed (this is the essential aspect of voltage and frequency scaling). Alternatively, workloads can be aggressively processed, so that the subsystems can be put into a deep-sleep state or switched off entirely. In the latter

Final edited form was published in "International Conference on Computer Communications and Networks (ICCCN). Vancouver, 2017". ISBN 978-1-5090-2991-4.
https://doi.org/10.1145/2236584.2236587

case, the different components of the subsystem can be switched off when they are not needed, even though the subsystem as such is regarded as active. For example, in a processor core, various idle components (such as the memory controller, load store unit, fixed point processor, etc.) can be separately turned off. Foreseeing these two possibilities, the ACPI specification defines the C and P states. In the context of multicore processors, the C state corresponds to managing individual cores by selectively switching off different sub-architecture components and the P state corresponds to operating the various cores at different frequencies. Hence, the scope of a dynamic power management strategy depends on the extent to which the components of a subsystem can be selectively switched off, the number of distinct power modes that can be achieved, and the accuracy with which the expected idle state of the subsystem can be estimated.

A. Dynamic Voltage and Frequency Scaling

For our analysis we identified four types of DVFS policies which can be integrated into a Linux environment (server edition). These are *power-save*, *on-demand*, *conservative*, and *performance* policies (governors) [7], [8], [10]. The *power-save* policy operates cores at the lowest frequency while the *on-demand* and *conservative* policies adapt the clock frequency to the change in the workload of the servers. The last two policies estimate the utilisation of the processor using a moving average technique, predict its future workload (for the next time slot), and scale-down or scale-up the processor's speed accordingly. The essential difference between the two is that the *on-demand* policy scales up the CPU frequency to the maximum whenever an increment in the core's activity is predicted whereas this is done gradually in the *conservative* policy. The *performance* policy operates a core at the maximum frequency. The aim is to complete a task as fast as possible and put the core into a deep sleep. On the system side, the server integrates 16 physical cores into 4 sockets but because of HyperThreading®, the operating system (Ubuntu server edition, version 12.04) recognises 32 logical cores. The available frequencies for scaling are: 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9 and 2.0 GHz. The corresponding processor voltages are set automatically and vary in the range from 0.6 to 1.35 V based on the temperature, leakage, power delivery load-line and dynamic capacitance.

At a socket level, the frequency of operation for individual cores can be different from the one set by a governor. A governor residing in the operating system, based on its knowledge of the present and anticipated workload of each core, may set the optimal operation frequency for each logical core separately. But this decision may not be realised directly by the hardware which has often some restrictions. For example, two HyperThreads running on the same physical core cannot have different frequencies. In order not to exceed any deadline, the core will select the highest frequencies of its active HyperThreads. Secondly, on Intel processors, there is only one phase-locked-loop (PLL) for all cores belonging to a single package (socket) [11]. Consequently, all cores belonging to one and the same package will operate at the same frequency. Once again, the highest frequency of all the active cores will be chosen by the clock frequency controller. We verified this aspect by transcoding a video several times with different configurations (see Table I). Firstly, we set the maximum frequency via a governor for a single core which carried out the transcoding task; the video was transcoded in 44.5 s. Secondly, we set the operation frequency to a minimum level and carried out the same transcoding task, which took 61.9 s to complete. In the first two cases, all the other cores were disabled. Thirdly, we activated an additional core within the same package and set its operation frequency to maximum whilst keeping the frequency of the other to minimum and carried out the transcoding task of the same video with the same core; the task was completed in 44.4 s. Hence, even if different cores belong to the same package may have different loads, they will have the same operation frequency which is determined by the highest workload. To minimise the inefficiency that arises due to the restriction on DVFS, individual cores can be forced to enter into a deep sleep or can be switched-off altogether.

Table I
TRANSCODING DURATION WITH SEVERAL SOFTWARE SET FREQUENCIES. TRANSCODING ALWAYS RUN ON CORE 1.

Core 1	Core 2	Core 3 & 4	Duration [s]
2 GHz	sleep	sleep	44.5
1.2 GHz	sleep		61.9
1.2 GHz	2 GHz		44.4

B. Workload

How well the future workload of a processor can be predicted and how fast its operation frequency (speed) can be set depends on the dynamics of the incoming workload. For this reason and in order to carry out repeated and reproducible experiments, we generated stochastic workloads. Each physical core (leaving HyperThreading unused) was given a video transcoding task, which always required 100 % CPU utilisation. We employed (FFMPEG[1]) to transcode a video from one format to another. By the end of each experiment, the throughput of all policies would be equal (i.e., the amount (in byte) of videos they transcoded would be equal). Transcoding tasks were given to the cores at fixed intervals, but

[1] https://www.ffmpeg.org/

Final edited form was published in "International Conference on Computer Communications and Networks (ICCCN). Vancouver, 2017". ISBN 978-1-5090-2991-4.
https://doi.org/10.1145/2236584.2236587

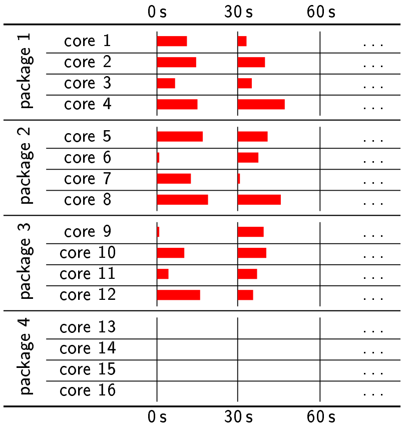

Figure 1. Exemplary workload/video size for each 30 s interval and core when three packages are used.

the sizes of the videos for each interval were random variables with a known distribution function, mean, and variance. As a result, the video transcoding time, and, hence, the idle time between the completion of a task and the arrival of the next task were random variables as well. This is illustrated in Fig. 1.

As can be seen, a transcoding task is given to the cores every 30 s. But the size of a video assigned to a core is randomly selected from a known distribution (normal, exponential, uniform). Consequently, when regarded along the time axis, the idle time of a single core has random duration; likewise, when regarded across multiple cores, the idle time has a random duration. The duration of a single time slot is determined by the time required to complete the transcoding of the longest video.

Thus, we produced the following workload distributions:

- Exponential – $\mathbf{E}\,(\lambda = 15\,\mathrm{MB})$; 0.5 % of the videos have the maximum video size, which is 80 MB for all the experiments.
- Normal – $\mathbf{n}\,(\mu = 15\,\mathrm{MB}, \sigma = 7.5\,\mathrm{MB})$; 2.3 % of the videos have zero size (no video was assigned for the entire time interval) and approx. no video has the maximum video size.
- Uniform – $\mathbf{U}\,(0, 30)$ – The size of the videos assigned to a core varies uniformly between 0 and 30 MB.

C. Measurement

We gradually increased the number of packages involved in our experiments. Thus the workload was started on 1, 2, 3 or 4 packages. We run each experiment for one hour and used YOKOGAWA WT210 digital power analysers to measure the power consumption of the server at a rate of approximately 10 Hz and DSTAT[2] to measure the CPU utilisation at a rate of approximately 1 Hz. Altogether we conducted 48 experiments, 4 different socket configurations, 3 different workload types, and 4 different DVFS policies.

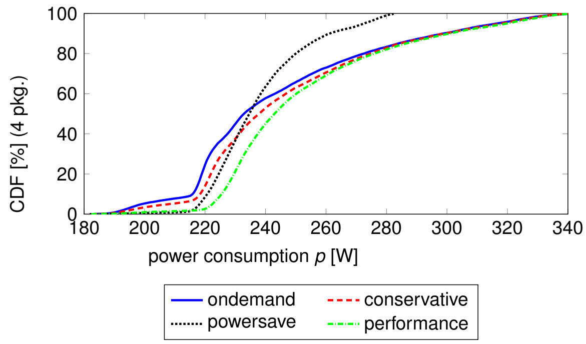

Figure 2. The cumulative distribution function of the power consumption of the server (p) when the exponential transcoding workload was being processed with the four dynamic power management strategies.

IV. Analysis

We employed the cumulative distribution function (CDF) to investigate how the statistics of the CPU utilisation and the power consumption of the server changed over time as its workload fluctuated. Except for the *power-save* policy, which utilised the least amount of resource and consumed the least amount of power, we did not observe an appreciable difference between the CDFs of CPU utilisation of the different scaling policies. Which means, the CPU utilisation is affected by the statistics of the workload rather than by the scaling policies (alternatively, the CPU load average is not a good indicator of the CPU activity at a time granularity less than 1 s). The diversity was the least in the exponential workload with a single socket and the most in the normal workload with all the cores activated. The *power-save* policy often violated transcoding deadlines (a transcoding task should complete within 30 s, which is the interval between the arrival of two consecutive transcoding tasks), which clearly indicates that energy saving can be made only to a certain extent beyond which there is a penalty. Similarly, from the power consumption statistics of the server (refer to Fig. 2 and 3), it can be seen that operating all cores at maximum frequency (with the *performance* policy) did not yield any performance gain as a result of a high power consumption (both *on-demand* and *conservative* have kept deadlines despite power saving).

The *on-demand* and *conservative* policies produced workload-aware, adaptive, comparable and reproducible results. Whilst the difference in the statistics of their power consumption (mean and variance) is insignificant, the *on-demand* strategy of switching the operation frequency to the maximum whenever an increment

[2] http://dag.wiee.rs/home-made/dstat/

Final edited form was published in "International Conference on Computer Communications and Networks (ICCCN). Vancouver, 2017". ISBN 978-1-5090-2991-4.
https://doi.org/10.1145/2236584.2236587

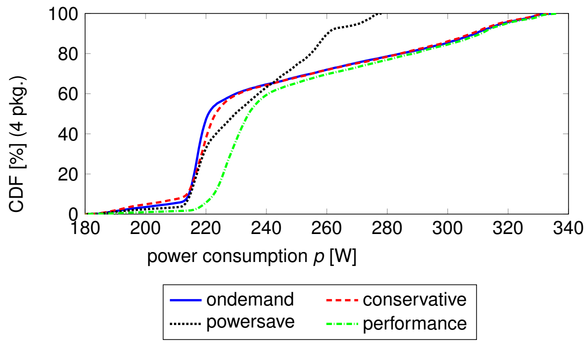

Figure 3. The cumulative distribution function of (p) when the normally distributed transcoding workload was being processed with the four dynamic power management strategies.

in the workload of a processor is anticipated appears to be the best strategy (this observation is consistent with previous observations where the *on-demand* policy produced better results under different workload types and workload statistics [7]). In contrast, the gradual increment in the operation frequency produced an extra frequency-switching cost.

V. Power vs. CPU Utilisation

One of the desirable features of a dynamic power management strategy is its predictability. The existence of a quantitative relationship between the workload of a server, on the one hand, and the resource demand and the power consumption of the workload, on the other enables to predict the outcome of a dynamic power management policy. When a quantitative relationship exists under each power management strategy, it is is possible to identify and choose the best one that suits a given workload type and workload statistics. Interestingly, the three DVFS policies (*on-demand*, *conservative*, and *performance* produced reproducible, linear relationships between the CPU utilisation and the power consumption of the multicore, multi-processor server. The existence of such a relationship has been claimed before (Möbius *et al.* [12] and Dargie [13]), but the claim was made in the absence of any dynamic power management strategy. The parameters of the linear regression varied slightly for different workloads and socket configurations, but this variation was insignificant. Fig. 4 shows the relationship between the CPU utilisation and the power consumption of the server for the normally distributed workload under the *on-demand* policy. The corresponding error is displayed in Figure 5. The power consumption of the server could be estimated from the CPU utilisation with an estimation error of below 2.5% for all the cases.

One of the consequences of the existence of a quantitative relationship between the power consumption and the CPU utilisation of the server is that, if the statistics of one of the quantities is known, it is possible to express the statistics of the other in terms of the known statistics. Suppose, for example, we have the statistics of the CPU utilisation (which we regard as a random variable, $\mathbf{w}$) under a given DVFS policy. Hence, the distribution function of $\mathbf{w}$, $F(w)$, is expressed as:

$$F(w) = P\{\mathbf{w} \leq w\} \tag{1}$$

where w is a positive real number signifying the workload being below $w\%$. Similarly, the density function, $f(w)$, of $\mathbf{w}$ can be determined by differentiating $F(w)$ with respect to w. Once $F(w)$ and $f(w)$ are determined, then useful parameters such as the variance and mean of $\mathbf{w}$ can be determined as well. Moreover, since we have experimentally determined that (refer to Fig. 4):

$$\hat{\mathbf{p}} = a \cdot \mathbf{w} + b$$

The CDF and pdf of $\hat{\mathbf{p}}$ can also be determined, because we have,

$$F(p) = P\{\hat{\mathbf{p}} \leq p\} = P\{a \cdot \mathbf{w} + b \leq p\} \tag{2}$$

Rearranging terms in Equation 2 yields:

$$F(p) = P\left\{\mathbf{w} \leq \frac{p-b}{a}\right\} = F_w\left(\frac{p-b}{a}\right) \tag{3}$$

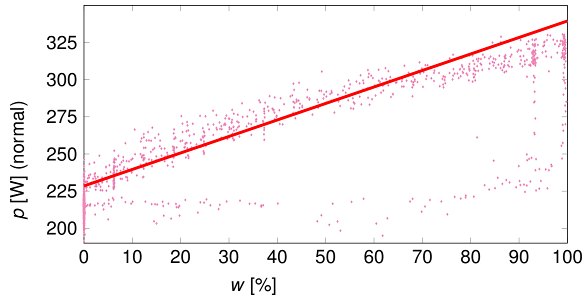

Figure 4. A linear regression of the power consumption p over the CPU utilization w for the normal workload under the *on-demand* DVFS policy.

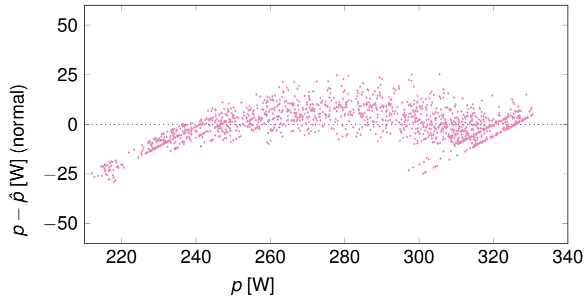

Figure 5. The difference between the predicted power consumption $(\hat{p})$ and the actual or measured power consumption (p) for the normal workloads under the *on-demand* DVFS policy.

Final edited form was published in "International Conference on Computer Communications and Networks (ICCCN). Vancouver, 2017". ISBN 978-1-5090-2991-4.
https://doi.org/10.1145/2236584.2236587

where $F_w(p)$ implies the distribution of $\mathbf{w}$ expressed in terms of p. From Equation 3, it is possible to derive the density function, the mean, and variance of $\hat{\mathbf{p}}$. So, it is clear that once the statistics of $\mathbf{w}$ is available, the power required to handle the workload under a given power management policy can be approximated.

VI. Conclusion

In this paper we investigated the scope and usefulness of different dynamic power management policies in a multicore, multi-processor server architecture. The server with which we experimented was a four socket system, each socket housing a quad-core Intel Xeon E5-4603 processor. We considered different workload statistics, DVFS policies, and server configurations. Our experiment results reveal that the scope of power management at a socket level is limited by the shared phase-locked loop (PLL) which manages the clock frequency for each core. The Intel Xeon E5-4603 processor provides a single PLL as a result of which cores belonging to the same package operates at the maximum frequency set by the DVFS policies to that package. In general, the DVFS policies we examined yield predictable, reproducible, and linear relationships between the total CPU utilisation and the overall power consumption of the server. The relationships enabled us to estimate the statistics (distribution and density functions) of the power consumption of the server when the statistics of the workload of the server is known. This aspect is useful for predicting power consumption, for identifying the best power management strategy, and for scheduling workloads. In this work, tasks were manually assigned to cores in order to examine the relationship between power consumption and CPU utilisation under different DVFS policies. In future our aim is to dynamically assign tasks based on the idle time distribution of cores and to find an optimal configuration that combines heterogeneous power management strategies.

Acknowledgment of Funds

This work is supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 – HAEC.

References

[1] R. Kalla, B. Sinharoy, W. J. Starke, and M. Floyd, "Power7: ibm's next-generation server processor", *IEEE Micro*, vol. 30, no. 2, pp. 7–15, 2010.

[2] R. Jevtic, H. P. Le, M. Blagojevic, S. Bailey, K. Asanovic, E. Alon, and B. Nikolic, "Per-core dvfs with switched-capacitor converters for energy efficiency in manycore processors", in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, 2015, pp. 723–730.

[3] B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, R. Cargnoni, J. a. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blaner, C. F. Marino, E. Retter, and P. Williams, "Ibm power7 multicore server processor", *IBM Journal of Research and Development*, vol. 55, no. 3, 1:1–1:29, 2011.

[4] S. Kamil, C. Chan, L. Oliker, J. Shalf, and S. Williams, "An auto-tuning framework for parallel multicore stencil computations", in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, IEEE, 2010, pp. 1–12.

[5] V. Hanumaiah and S. Vrudhula, "Energy-efficient operation of multicore processors by dvfs, task migration, and active cooling", in *IEEE Transactions on Computers*, vol. 63, 2014, pp. 349–360.

[6] S. Maiti, N. Kapadia, and S. Pasricha, "Process variation aware dynamic power management in multicore systems with extended range voltage/frequency scaling", in *IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2015, pp. 1–4.

[7] A. Brihi and W. Dargie, "Dynamic voltage and frequency scaling in multimedia servers", in *IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 374–380.

[8] W. Dargie, "Analysis of the power consumption of a multimedia server under different dvfs policies", in *IEEE 5th International Conference on Cloud Computing*, 2012, pp. 779–785. DOI: 10.1109/CLOUD.2012.31.

[9] F. D. Rossi, M. Storch, I. De Oliveira, and C. A. F. De Rose, "Modeling power consumption for dvfs policies", in *IEEE International Symposium on Circuits and Systems*, 2015, pp. 1879–1882.

[10] V. Pallipadi and A. Starikovisky, "The ondemand governer", in *Proceedings of the Linux Symposium (volume two)*, 2006.

[11] A Naveh, E Rotem, A Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and thermal management in the intel core duo processor", *Intel Technology Journal*, vol. 10, no. 2, pp. 109–121, 2006. DOI: 10.1535/itj.1002.

[12] C. Möbius, W. Dargie, and A. Schill, "Power consumption estimation models for processors, virtual machines, and servers", *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1600–1614, 2014. DOI: 10.1109/TPDS.2013.183.

[13] W. Dargie, "A stochastic model for estimating the power consumption of a processor", *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1311–1322, 2015.