Trends in Static Power Consumption

Jörg Lenhardt, Wolfram Schiffmann, Jörg Keller
Faculty of Mathematics and Computer Science
FernUniversität in Hagen, Germany
joerg.lenhardt,wolfram.schiffmann,joerg.keller@fernuni-hagen.de

ABSTRACT

Decreasing feature sizes of modern Complementary Metal Oxid Semiconductor (CMOS) devices lead to an increasing fraction of static and a decreasing portion of dynamic power. Thus, power saving benefits from Dynamic Frequency and Voltage Scaling (DVFS) are diminishing. In addition, the possibility to switch off parts of a semiconductor device become more and more common. Powering off cores is commonly used to reduce the power of a many core system. A consequence of powering off unused parts of a chip is that the relative difference between idle and fully loaded power consumption is increased. That means, future chips and whole server systems gain more power saving potential through power-aware load balancing, whereas in former times this power saving approach had only limited effect, and thus, was not widely adopted. While powering off complete servers was used to save energy, it will be superfluous in many cases when cores can be powered down. An important advantage that comes with that is a largely reduced time to respond to increased computational demand. We include the above developments in a server power model and quantify the advantage. Our conclusion is that strategies from datacenters when to power off server systems might be used in the future on core level, while load balancing mechanisms previously used at core level might be used in the future at server level.

1. INTRODUCTION

Power consumption of computers has been a focus of research for many years, ranging from lower voltage transistors over frequency scaling and power gating to algorithmic approaches. Consequently, different methods for the reduction of power consumption in computers have been applied by different communities at various levels, e.g. reducing power consumption in single cores up to reducing power consumption of a data center by migrating load and switching off unused servers. We present a model for power consumption that reflects recent technological developments and allows the conclusion that at core level, switching off some cores (and migrating workload to remaining cores) gets more important, and hence strategies from data centers might be useful to be applied in single systems. On the other hand, at system level, it seems that switching off complete server systems (and thus long restart times) can often be avoided because of low idle power, and thus load distribution strategies from multicore systems at the time when dynamic power consumption was dominating might be useful to be applied in datacenters.

While in CMOS devices dynamic power only occurs when switching takes place, static power resulting from leakage currents contributes permanently to the power consumption. In the past, the proportion of static power was low and was widely neglected. With decreasing feature sizes static power increases exponentially [13]. Despite the ongoing decrease in feature sizes of modern semiconductor devices the threshold voltage cannot be lowered at former rate. Smaller transistor dimensions lead to proportionally increasing leakage current. As a result, the energy consumption keeps almost unchanged while the chip area gets smaller. Two results regarding power saving potential considering rising static power proportion are important: (1) Powering off parts of a chip is crucial to save energy in future CMOS devices and (2) cores are suitable units for on-demand powering off and on. The idle power consumption of the whole computer system decreases when powering off more parts. Consequently, the gap between idle and full power consumption of servers rises. Formerly, energy efficient load balancing had only marginal potential to reduce power consumption but with increasing full-to-idle power-ratio, it becomes more important.

We distinguish three levels in regard to power and energy: (1) A **single core** can be switched on or off. Above that, it can run at different frequency and voltage levels to reduce power. (2) In a **multicore processor** with n cores usually 1 to n cores are running, 0 if the system is off. (3) The whole **system** can be switched on or off and the system can be loaded between idle (booted but no actual work to do) and 100%. Switching cores on or off is orders of magnitude faster than shutting down or booting a whole system [11]. Due to different effects the full-to-idle power-ratio decreases for single cores. But in multicore environments the ratio increases because at low utilization the load is distributed among fewer cores while the others can be switched off. In this context, we examine to move from (a) *system on/off and DVFS for cores* to (b) *system on and cores on/off with or without DVFS*. The strategies are somewhat reversed. We

examine several models for describing the power consumption using various strategies. This development has some advantageous effects regarding server farms in data centers. Usually, operators are not willing to shutdown servers, e.g. because of the risk that the system does not reboot. Above that, the time to boot a system is magnitudes higher (several minutes) than to power on cores. The response time to random changing conditions is much worse. Moving from powering off and on systems to load balancing to reduce power consumption is a more convenient solution.

The rest of this paper is organized as follows: Sect. 2 presents preliminaries and related work. DVFS and power off in view of rising static power is discussed in Sect. 3. The consequences of this development in regard to semiconductor devices (small) on the one hand and server farm (big) on the other is discussed in Sect. 4. We conclude and give ideas for future work in Sect. 5.

2. PRELIMINARIES

2.1 Workload Models

We assume a divisible workload that is expressed as a percentage of a multi-core CPU's maximum processing capability, and that can be re-distributed over cores. While this seems rather abstract, there are several examples from practice that match this model quite well. First, we consider web-servers, which are multi-threaded applications where each request is served by a separate thread. The number of threads at a certain load level typically is much larger than the number of cores needed to process that load level. Hence, the load can be considered fine-grained enough to be modeled as a divisible load. A similar situation occurs with database applications where different queries are processed in different threads. Although the heterogeneity of the threads' processing requirements can be larger than in a web-server, and although the threads might interact on the data stored in the database, the majority of queries typically contributes small loads and are unrelated, so that the model of a divisible load gives an approximation. Finally, we consider server systems that provide virtualization services by running virtual machines that look like different (physical) computers to customers. Often a server hosts more than one hundred virtual machines, at least many more than it has cores. Thus, each load entity, i.e. virtual machine, only contributes a small share to the total load, so that the workload is again fine-grained enough to be modeled as a divisible load. The difference to the former applications is the duration of the load entities: while web requests and database queries are processed in fractions of a second, and there is a continuous stream of new requests, virtual machines are mostly run for days or even weeks. For short lived requests, load balancing occurs when the requests arrive, by assigning them to a core. For long lived virtual machines, load balancing occurs by migrating the virtual machine to a different core, which contributes an overhead which however is not very frequent and thus can be neglected: the duration of the virtual machines has the consequence that they terminate only seldom and thus load re-balancing is only necessary once in a while.

2.2 Related work

In data centers, the servers' utilization typically lies just between 10 and 50%. Thus, if the servers provide the maximum performance while running at full power a lot of energy is wasted. The general objective is to adjust the power consumption proportional to the requested performance [1]. Dynamic and static power consumption in CMOS devices is explained in [10]. The most popular method to reduce power consumption consists of slowdown the chip's clock by means of DVFS. Unfortunately, the power savings of this technique are limited by the difference between maximum and static power of the chip. Due to the increasing leakage currents on chip level the efficiency of DVFS will diminish as the amount of static power cannot be reduced in the future [12]. In order to achieve energy-proportional computing a shutdown of components like functional units or cores is necessary to achieve significant energy savings [3, 6, 8]. Alternatively, reconfigurable application specific devices (ASIC) on the processor chip [14] and toggling between heterogeneous computing systems with different performance characteristics have been proposed [4]. Several authors examine the consequences of the breakdown of Dennard scaling [5]. The portion of a chip which can be switched at full frequency is dropping exponentially with each process generation due to power constraints. Large fractions have to be dark (idle) or dim (underclocked), cf. e.g. [7, 13]. Thus, both shutdown and slowdown techniques will be needed in the near future. All shutdown techniques suffer from the overhead to reactivate the components when the performance requirements increase later on. This results in delays that will depend on the complexity of the switched off components [2]. Moreover, the overhead can even increase the energy consumption. In [9] a strategy for avoiding those *negative* energy savings is presented. In this paper we investigate the interrelationship between core level and server level power management in the face of recent developments in CMOS devices such as growing importance of static power consumption at core level and diminishing importance of idle power at system level.

3. DVFS AND POWER-DOWN

In this section we present several strategies to reduce power consumption of homogeneous many core systems. In the following, c represents the number of cores, l the load of each core where $0 \leq l \leq 1$, s the static part of power consumption where $0 \leq s \leq 1$, f frequency of chip/core where $0 \leq f \leq 1$. F is a set of frequencies when using discrete frequency levels. We denote scaling, continuous, discrete, and power off by indices *sc*, *cont*, *disc*, and *po*, respectively.

3.1 Continuous DVFS and Power Off

The power consumption of a system using continuous frequencies and DVFS is described in Eq. 1. The power consumption of a single core is the sum of the static part s and the dynamic power, which is the load per core, i.e. frequency, cubed, and weighted with $1 - s$ to get a power range $[0; 1]$. We used a cubic exponent because changing frequency has linear and voltage scaling quadratic influence on the power consumption. The power consumption using the power-off strategy is modeled in Eq. 2. Because the maximum core load is normalized to 1, the number of cores necessary is the current load rounded to the next integer. The cores run at maximum frequency 1, and thus also consume power 1. Fig. 1(a) depicts the resulting power consumption of both strategies for a 16 core system depending on the load l. Static portions $s = 0.1, 0.3, 0.5$ are considered. Powering off is plotted in red, and DVFS in blue. The static power con-

sumption has no influence when applying po_{plain}, so there is only one curve in the shape of a stair. In contrast, using DVFS static power influences the results. Having a low static fraction as found in older devices, DVFS leads to better results than power off except at very low utilization (see fine dashed blue line, $s = 0.1$). A larger static fraction increases power consumption as DVFS has only influence on the dynamic part. With $s = 0.3$ the results of power off are better till a load of 5 (max. total load is 16) has been reached, at $s = 0.5$ powering off is better up to a load of 9. The differences at higher utilization decrease for larger s.

$$sc_{cont}(c, s, l) = c \cdot \left(s + (1 - s) \cdot \left(\frac{l}{c} \right)^3 \right) \quad (1)$$

$$po_{plain}(l) = \lceil l \rceil \quad (2)$$

3.2 Discrete DVFS

In real devices, only a finite set of frequency levels is available. The device runs at one of these levels. Eq. 3 models power for discrete DVFS. The only difference to the continuous version (Eq. 1) is the dynamic part, where a sub-function d is used to choose the lowest discrete frequency level above the load per core.

$$sc_{disc}(c, s, l) = c \cdot \left(s + (1 - s) \cdot d^3(l, c) \right) \quad (3)$$

Fig. 1(b) depicts the power consumption of a 16 core system using continuous (red) and discrete (blue) DVFS plotted for static portions s of 0.1 and 0.5 and discrete frequencies $F = \{0.2, 0.4, 0.6, 0.8, 0.9, 1.0\}$. Discrete DVFS produces a step function where the power consumption is generally higher than in continuous DVFS. Only at points where l/c equals a valid frequency the power consumptions of both variants are equal. As we consider a divisible load, our system would distribute this load as evenly as possible over all cores and thus all cores should be scaled by the system[1] to the same frequency, or onto two adjacent frequency levels as a consequence of the discretization.

3.3 DVFS in Combination With Power Off

To benefit from DVFS *and* powering off, a combination of both can be used. This can be modeled with Eq. 4, which is similar to Eq. 3, except that the number of cores is not a parameter any more. The minimum number of cores is used to execute load l, i.e. l rounded up to the next integer.

$$po_{scale}(s, l) = \lceil l \rceil \cdot \left(s + (1 - s) \cdot d^3(l, \lceil l \rceil) \right) \quad (4)$$

The results in comparison to powering off cores without DVFS are displayed in Fig. 1(c) for a 16 core device. We use static part $s = 0.3$ in this example, and the frequency levels F from the previous subsection. In this case, the strategy leads to better results when l is lower than 9, so in Fig. 1(c) only loads 1 to 10 are displayed. If the load l is low, more frequency levels can be used to reduce power consumption. Increasing the load, fewer frequency levels can be applied till all cores are running at $f = 1.0$. It is easy to understand when assuming a load of 9: Instead of using 10 cores and running at a frequency $f = 0.9$ the algorithm uses 9 cores. In Fig. 1(d) the results of DVFS with and without powering off cores for a 16 core system are displayed. In this example we have a static portion $s = 0.5$. Up to a load l of 9 the combination of powering off most of the cores and apply DVFS leads to a lower power consumption than DVFS alone. Above that, DVFS alone can achieve better results in some cases, e.g. from load 9 to about 9.5. This leads to the insight that in some cases it may be beneficial to use more cores than the minimum possible to achieve lower power consumption.

[1]If the frequency governor does not achieve this, the application or an adapted governor would have to enforce this.

3.4 Decrease Power by Using More Cores

Although counter-intuitive, the previous subsection gave an example where it was advantageous to use more than the minimum number of cores. Eq. 5 models this situation. sc_{disc} from Eq. 3 is used several times starting with load l rounded up to the next integer as input for number of cores. This is tried for all core counts up to c. The result with the minimum power consumption is used.

$$sc_{min}(c, s, l) = min \left\{ sc_{disc}(m, s, l) | m = \lceil l \rceil, \ldots, c \right\} \quad (5)$$

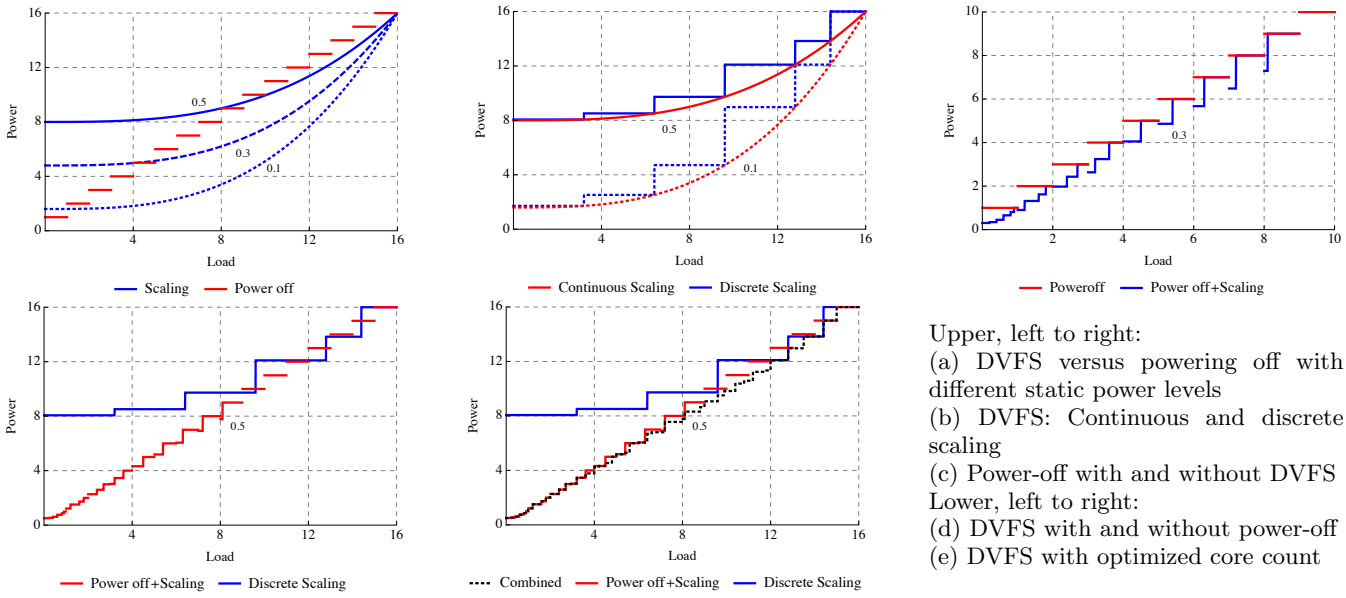
The result for a 16 core system with a static power of $s = 0.5$ is depicted in Fig. 1(e). DVFS only is shown in blue, DVFS with powering off cores in red and sc_{min} in dashed black. For $0 \leq l \leq 4$, the results of sc_{min} are similar to powering off with scaling. After that, the results are better than or the same as DVFS (with or without power off). Especially for $8 \leq l \leq 12$, better results are achieved. Overall, sc_{min} is never worse than the other two strategies.

4. INFLUENCE OF INCREASING STATIC POWER

For a single core, the increasing fraction of static power, together with the restricted possibility for voltage scaling, leads to a diminishing influence of frequency scaling on power consumption. Hence, in case of multicores it might be advantageous to use fewer active cores at higher frequencies. In this respect, the operating system community could take a look at strategies used in datacenters at the level of complete server systems. The good news for operating system researchers is that the capability to forecast workload changes can be restricted at core level, as the time to power up a core is much shorter (milliseconds) than the time to power up a complete system (multiple seconds). Thus, the algorithms to decide on core shutdown and power up can be simpler and thus more aggressive than at system level. For a complete system, the power consumption tends to better scale with the system workload, as unused cores within the processor can be switched off while not needed. As the processor power consumption comprises a notable fraction of the system's total power consumption, and power saving features have been introduced in other system parts as well, such as turning off unused memory banks or hard disks, idle power gets low and it is seldom needed to switch off a complete system for power reasons. The good news for a data center provider is that the strategies when to power up a system can be simplified because a spare system as performance buffer does not hurt the energy budget anymore.

5. CONCLUSIONS

We introduced a simple but reasonable model for the power consumption of a computer system. Our model reduces relations between load (required performance) and power consumption to a small set of parameters. In this way we



Upper, left to right:
(a) DVFS versus powering off with different static power levels
(b) DVFS: Continuous and discrete scaling
(c) Power-off with and without DVFS
Lower, left to right:
(d) DVFS with and without power-off
(e) DVFS with optimized core count

Figure 1: Power curves for different strategies.

are able to analyze the essential features and power management schemes of multicore-based server systems. While current developments in CMOS devices lead to shrinking differences between maximum and minimum power consumption at level of a single core, the possibility to switch off cores leads to growing differences between maximum and minimum power consumption (full and idle power) of complete server systems. As a consequence, the strategies at the level of cores and complete systems can be reversed: Operating systems should give more importance to switch off cores if load is low while data center operators are mostly relieved from the question when or if to switch off complete server systems to minimize energy consumption. This has a tremendous positive side effect. While waking up a core can be done in several milliseconds, waking up a complete server might take a minute. Thus, avoiding to switch off complete server systems allows data center operators to react much faster to sudden increases of load without the previously high energy penalty of running idle systems. Moreover, decreasing the number of system starts per year has the tendency to increase system life time, i.e. is an additional benefit.

6. REFERENCES

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

[2] A. Carroll and G. Heiser. Mobile Multicores : Use Them or Waste Them. *ACM SIGOPS Oper. Syst. Rev.*, 48(1):44–48, 2014.

[3] A. Carroll and G. Heiser. Unifying DVFS and Offlining in Mobile Multicores. In *IEEE Real Time and Application Systems (RTAS) 2014*, Berlin, 2014.

[4] G. Da Costa. Heterogeneity: The Key to Achieve Power-Proportional Computing. *13th Symp. on Cluster, Cloud, and Grid Computing*, pages 656–662, May 2013.

[5] R. H. Dennard, F. H. Gaensslen, Y. Hwa-Nien, V. Rideout, E. Bassous, and A. LeBlanc. Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions. *Proc. of Solid-State Circuits*, 9(5):256–268, 1974.

[6] A. Dhingra and S. Paul. A Survey of Energy Efficient Data Centres in a Cloud Computing Environment. *IJARCCE*, 2(10):4033–4040, 2013.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Power Limitations and Dark Silicon Challenge the Future of Multicore. *ACM Transactions on Computer Systems*, 30(3):1–27, 2012.

[8] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis. Power Management of Datacenter Workloads Using Per-Core Power Gating. *IEEE Computer Architecture Letters*, 8(2):48–51, Feb. 2009.

[9] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram. A case for guarded power gating for multi-core processors. *17th Int. Symp. on HPC Architecture*, pages 291–300, 2011.

[10] S. Reda and A. Nowroz. Power Modeling and Characterization of Computing Devices: A Survey. *Foundations and Trends in Electronic Design Automation*, 6(2):121–216, 2012.

[11] R. Schöne, D. Molka, and M. Werner. Wake-up latencies for processor idle states on current x86 processors. *Computer Science - Research and Development*, pages 1–9, 2014.

[12] E. L. Sueur and G. Heiser. Dynamic voltage and frequency scaling: the laws of diminishing returns. In *Int. Conf. on Power Aware Computing and Systems*, HotPower'10, pages 1–8. USENIX Association, 2010.

[13] M. B. Taylor. A Landscape of the New Dark Silicon Design Regime. *IEEE Micro*, 33(5):8–19, 2013.

[14] L. Wang and K. Skadron. Implications of the Power Wall: Dim Cores and Reconfigurable Logic. *IEEE Micro*, (September/October):40–48, 2013.