A Survey of Dynamic Power Saving Strategies in Real Systems

Sai Kiran Talamudupula
Senior Bios Engineer, Intel, Chandler, Arizona, USA

ABSTRACT

Energy efficiency and energy-proportional computing have become major constraints in the design of modern exascale platforms. Dynamic Voltage and Frequency Scaling (DVFS) is one of the most commonly used and effective techniques to dynamically reduce power consumption based on workload characteristics. The focus of this paper is to survey several energy saving strategies designed for improving power efficiency of CPU and DRAM systems. This paper also presents a characterization of the strategies based on their salient features, to help the research community in gaining insights into the similarities and differences between the them. The aim of the paper is to equip researchers with knowledge of the state of the art energy saving strategies and serve as a quick reference to engineers while they are devising novel energy saving strategies.

Keywords: Dynamic Voltage and Frequency Scaling (DVFS), Power Efficiency, Energy Saving, Survey, Review.

I. INTRODUCTION

The last few decades have witnessed a tremendous rise in the design of scalable applications for various scientific domains. The sheer computational requirements of these applications have forced system engineers to develop ever more power and performance-efficient architectures. For the current topmost petascale computing platforms in the world, it is typical to consume power on the order of several megawatts, which at current prices may cost on the order of several million dollars annually. For operational sustenance of the exascale machines, the power consumption growth rate must slow down and deliver more calculations per unit of power. To address this challenge, power and energy optimizations have been proposed in modern computing platforms at all levels: application, system software, and hardware.

There are several architectural and technological trends that have resulted in increased power consumption of modern computing systems. In modern processors, the number of cores on a single chip has been increasing [4, 40] and consequently to serve the requests of many core architectures, the requirements for memory system have been increasing as well. Apart from the increase in number of cores and size of main memory, the clock rate on these devices has also been increasing rapidly and is only been constrained by the feature size, pipeline subcomputation length and power consumption since dynamic power consumption in a CMOS is dependent upon the operating frequency and voltage.

It is well established that the CPU and the memory subsystem are the major power consumers in a computer system. For example, the CPU consumes about 50% of the total power as was investigated in [10], considering both static and dynamic power consumption. Memory power consumption is a significant component in computer server power profile, which is comparable to or may even surpass processor power consumption for memory-intensive workloads. An early study [39] has reported that, on an IBM p670 server, the average memory power consumption is 1223 watts, compared to the average processor power consumption of 840 watts. A keynote speech from industry in 2011 [40] predicts that memory may consume about 70% in small scale servers in the future, excluding the power consumed by power supply and cooling. The current generation of Intel processors provides different P-states for dynamic voltage and frequency scaling (DVFS) [41,42] and T-states for introducing processor idle cycles (throttling).
In [36], Mei et al review the past research works focusing on GPU DVFS and conduct experiments on GPUs, confirming that GPU DVFS provides lot of potential for energy savings. [37] discusses categories of power saving techniques such as DVFS, thread motion, parallelism etc. and proposes a new technique based on the research work that was reviewed. Zhuravlev et. Al [38] survey scheduling techniques which provide energy efficient computation.

Different from all these works, this paper focuses on dynamic power reduction strategies making use of DVFS/Throttling in modern processors. Although the techniques which improve the performance of an application can consequently reduce the energy consumption as well but only consider the strategies which were devised to reduce the power consumption specifically. Moreover, all the techniques surveyed in this paper have been implemented and tested on real hardware platforms instead of simulation/emulation.

The contributions of this work include 1) an overview of the basic power consumption terminologies commonly used in research literature, 2) Studying the capabilities for reducing dynamic power consumption in modern processors and 3) provide a categorization for dynamic power saving strategies with respect to their design characteristics.

The rest of the paper is organized as follows. Section II provides an overview of types and sources of power dissipation along with the mechanisms to reduce dynamic power in modern processors to apply DVFS/Throttling. Section III describes the salient features of dynamic power saving strategies for their classification, showing the similarity and differences between them. Section IV provides a detailed discussion of various power saving strategies. Finally, Section V provides the concluding remarks

II. POWER DISSIPATION

In this section, we briefly discuss the types and sources of power consumption in modern processors, along with the mechanism to reduce dynamic power consumption, to aid in the discussion of dynamic power saving techniques in the next sections.

A. Static and Dynamic Power

There are three sources of power dissipation in digital CMOS circuits [23] which are summarized as

\[ P = P_{\text{switching}} + P_{\text{short-circuit}} + P_{\text{static}} = \alpha C L V_{dd}^2 + I_{sc} V_{dd} + I_{\text{leakage}} V_{dd} \]

Ps\text{witching} refers to the dynamic component of the power, where CL stands for load capacitance, \( f \) is the clock frequency and \( \alpha \) is the activity factor. Assuming voltage swing is equal to the supply voltage \( V_{dd} \), the short circuit power \( P_{\text{short-circuit}} \) is due to the direct path short circuit current \( I_{sc} \), which flows when both PMOS and NMOS transistors are simultaneously active. The static power consumption \( P_{\text{static}} \) is due to the leakage current \( I_{\text{leakage}} \) which arises due to various factors such as sub-threshold leakage, reverse biased pn-junction etc.

It can be observed from Eq. (1), that only the dynamic power consumption can be modified at application runtime through software since it is dependent upon the frequency of the processor whereas the other two components in general are fixed during the manufacturing process. Therefore, in this paper we review power saving strategies which attempt to reduce only the dynamic power consumption.

B. DVFS and Throttling

The dynamic voltage and frequency scaling (DVFS) mechanism reduces the operating frequency and voltage of the processor on-the-fly during application execution, thereby reducing the dynamic power consumption as per Eq. (1). DVFS can be applied by writing a specific value to the IA32_PERF_CTL model specific register (MSR).

CPU throttling can be viewed as an equivalent to dynamic frequency scaling as it inserts a given number of idle cycles in the CPU execution obtaining a particular operating frequency without changing the operating voltage of the cores. Hence, dynamic frequency scaling is less effective than DVFS in terms of reducing dynamic power consumption, but when used with conjunction with DVFS, it has been shown to reduce the dynamic power consumption significantly [28].
III. CHARACTERISTICS OF THE STATE OF THE ART ENERGY SAVING STRATEGIES

In particular, the existing DVFS based techniques can be characterized through the following features:

a) Application/Library: If a technique makes modification to the application or library code to apply DVFS.

b) Transparent: If a technique applies DVFS transparently without making any modification to the application or library code.

c) Processor/DRAM: If a technique applies DVFS to either one of processor/DRAM or both.

d) Workload Prediction Mechanism: If a technique uses history or trace based workload predictors. The history predictors employ window of past n samples to predict the future workload whereas the trace predictors use the traces of past execution to predict the future behavior [15].

e) Performance Loss Availability: If a technique provides the user with fine grained control of degradation in performance of the application resulting from DVFS usage. In [47], authors discussed that how always selecting a higher performance loss constraint doesn’t always translate into energy savings.

f) DVFS/Throttling: If a technique makes use of DVFS or throttling or both to save power.

IV. OVERVIEW OF THE DYNAMIC POWER SAVING STRATEGIES

Table I categorizes the various dynamic power saving strategies as per their salient characteristics.

We begin by reviewing the linux frequency scaling governors [43, 44] provided by the cpufreq infrastructure. cpufreq is implemented on top of frequency scaling drivers such as acpi-cpufreq, intel pstate etc. The governors are basically an in-built energy saving strategy and only one of them can be active at time which are:

Performance: Statically sets the processor to its maximum frequency.

Powersave: Statically sets the processor to its minimum frequency.

Ondemand: Sets the processor frequency to maximum or minimum if processor is working or idle, respectively.

Conservative: Similar to Ondemand, except it increases/decreases frequency in a stepwise manner.

The Userspace frequency scaling governor is not listed above since it allows userspace programs to set the frequency. Since, all the above-mentioned governors have their own frequency scaling rules, the Userspace governor is chosen to deploy user developed energy saving strategies.

CPU Miser [8] selects a performance loss for the underlying application chosen by the user and divides the execution of an application into intervals of a particular duration. Next, it predicts the execution characteristics, such as memory stalls, of the upcoming interval based on similar recent intervals using the history predictor. CPU Miser primarily considers memory accesses to choose a suitable frequency for a given time slice, and has been shown to attain significant energy gains [8]. It can also either overestimate or underestimate the number of stalls in an application, which can lead to an inaccurate estimation of the frequency. Also, it does not consider the instantaneous power consumption of the unit under test when choosing a suitable frequency.

Adagio [21] discusses critical path analysis to identify the tasks to be slowed down as it does not provide the user with an option to choose a performance loss. The workload prediction mechanism used in Adagio is similar to the history predictor with some feedback added, such that the future behavior of a communication call is predicted based on its last invocation. Lim et al. [19] propose a scheme that applies DVFS during communication phases in MPI applications. Dynamic determination of communication phases and selection a suitable processor frequency to minimize energy consumption are the key features of this scheme. This strategy does not apply DVFS in computation portions of an application and, therefore, may not save a significant amount of energy for an application that has a relatively low communication activity.

Authors in [29] proposed a novel DVFS application strategy, which used both DVFS and Throttling to maximizes energy savings by selecting appropriate values for DVFS and throttling based on the predicted

International Journal of Scientific Research in Science, Engineering and Technology (ijsrset.com)
communication phases considering both the CPU offload (provided by the network protocol, such as Infiniband) during communication and the architectural stalls during computation. [25, 32], discuss the potential pitfalls of the performance-loss based approaches and propose a strategy that depends on the instantaneous power consumption of the computing platform using regression analysis to choose the best processor frequency which would minimize the energy consumption.

[35] implements a runtime communication library named PASCoL making use of Aggregate Remote Memory Copy Interface (ARMCI) which uses both DVFS and polling/blocking mechanisms to achieve energy saving in communication calls. A quad-state runtime system targeting point-to-point calls in MPI was proposed in [27, 33] which applied DVFS in those calls using a trace based workload prediction mechanism. [28, 24] and [16] both focus on collective communication calls such as MPI_AlltoAll which are extremely communication intensive and employ both DVFS and Throttling to achieve impressive energy savings up to 20% for select applications in NAS [3] and CPMD [1].

Timeslice profiling through performance counters and application of DVFS by determining the memory intensiveness of a workload is done in [11]. [12] improves upon [11] by employing advanced modeling of workload and taking into account the out-of-order nature of modern processor pipelines. Authors in [7] devise a strategy which divides the application into various phases, assign it a frequency with and then when this phase is further encountered, it is assigned a frequency which was calculated through experiments and heuristics when that particular phase was first seen.

By determining the memory intensiveness of different algorithmic steps in NwChem [34] in [30], authors in [31] propose a frequency scaling scheme code in application for both processor and memory In [26], a runtime system is proposed which addresses both processor and memory frequency scaling based on a detailed performance model minimizing energy consumption of an application. An important point to note here is that DRAM frequency scaling is currently not supported through software and it was emulated through changes in memory frequency through BIOS in [26].

Table I: Summary of Several Dynamic-Power Saving Strategies as to the Six Salient Characteristics

<table>
<thead>
<tr>
<th>Name</th>
<th>App/Library</th>
<th>Transparent</th>
<th>Proc./DRAM</th>
<th>Prediction</th>
<th>Perf. Loss</th>
<th>DVFS/Throttling</th>
</tr>
</thead>
<tbody>
<tr>
<td>[8]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[21]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[29]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>Trace</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[19]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[25]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[35]</td>
<td>Yes</td>
<td>No</td>
<td>Processor</td>
<td>History</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[14]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>Trace</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[28]</td>
<td>Yes</td>
<td>No</td>
<td>Processor</td>
<td>None</td>
<td>No</td>
<td>Both</td>
</tr>
<tr>
<td>[12]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[11]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[7]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>Trace</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[16]</td>
<td>Yes</td>
<td>No</td>
<td>Processor</td>
<td>None</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[33, 27]</td>
<td>Yes</td>
<td>No</td>
<td>Processor</td>
<td>Trace</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[30, 31]</td>
<td>Yes</td>
<td>No</td>
<td>Both</td>
<td>None</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[18]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[26]</td>
<td>No</td>
<td>Yes</td>
<td>Both</td>
<td>History</td>
<td>Yes</td>
<td>DVFS</td>
</tr>
<tr>
<td>[32]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[4]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>None</td>
<td>No</td>
<td>Both</td>
</tr>
<tr>
<td>[45]</td>
<td>No</td>
<td>yes</td>
<td>Processor</td>
<td>History</td>
<td>No</td>
<td>DVFS</td>
</tr>
<tr>
<td>[46]</td>
<td>No</td>
<td>Yes</td>
<td>Processor</td>
<td>History</td>
<td>No</td>
<td>DVFS</td>
</tr>
</tbody>
</table>
Authors in [18] propose a performance model based on power limiting provided through the running average power limit (RAPL) [5] feature present in Intel processors, which under a performance loss minimizes energy consumption in Intel Xeon Phis [2]. An adaptive core-specific runtime (ACR) is proposed in [4] which makes use of both DVFS in its per core variant and throttling to achieve energy savings in parallel applications. Optimal power allocations using RAPL power limiting features to different components of a compute node along with workload behavior with respect to those applications was studied in [9]. Green GPU strategy [20] for heterogeneous architectures, distributes workload between GPU-CPU so that both the components can finish around the same time and also applies DVFS to GPU cores based on their utilization. [17] compares the energy efficiency of ARM32, ARM 64 and an x86 processor for a GAMESS [22] execution for solution to energy which determined that ARM32 machine was the most energy efficient followed by its 64-bit variant and the x86 processor. Lately, oversubscription [13, 6] has been shown to reduce power consumption and increase performance of an application by executing multiple threads/processes on a single core of a processor.

[45] proposes the energy aware race-to-halt (EARtH) runtime strategy which uses a theoretical model evaluating effect of processor frequency on platform power and determine its effect on application using the workload scalability (SCA) parameter. The EARtH algorithm calibrates itself at system production by measuring power consumption at various processor frequencies and employs it in the theoretical model to determine the optimal frequency which minimizes energy consumption.

FoREST DVFS controller [19], based on the past studies which depicted static power as a function of dynamic power [58, 38], derived through modeling that power ratios at different frequencies for an application are independent of the nature of the application. Next, by computing these power ratios in an online manner, an online algorithm is proposed which depicted significant energy gains. Moreover, it also provides an option for providing a user defined performance loss.

V. CONCLUSION

Recent advances in chip design and the desire to extract maximum performance from the modern computing systems has greatly increased their power consumption. To alleviate this issue, several techniques have been proposed in the past to reduce dynamic power consumption of the processor and memory by modifying their frequency and operating voltage. In this work, we reviewed many dynamic power saving strategies which make use of DVFS and throttling to reduce the dynamic power consumption. We also discussed a categorization of the reviewed strategies as per their salient characteristics to inform the reader regarding their similarities and differences. We hope that this survey work will inform researchers, processor architects and software engineers to regarding the prominent strategies employed to reduce dynamic power consumption and their underlying behavior. Moreover, it will also serve as a quick reference for the designers of future dynamic power saving strategies to compare their work with the past researches.

VI. REFERENCES

[6]. Ellie L. Fought, Vaibhav Sundriyal, Masha Sosonkina, and Theresa L. Windus. Saving time and energy with oversubscription and semi-direct Moller-Plesset second order perturbation


[24]. V. Sundriyal and M. Sosonkina. Per-call Energy Saving Strategies in All-to-all Communications.


