Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management

Rotem Efraim[1], Ran Ginosar[2], C. Weiser[2], and Avi Mendelson[3]

Abstract—The EARtH algorithm finds the optimal voltage and frequency operational point of the processor in order to achieve minimum energy of the computing platform. The algorithm is based on a theoretical model employing a small number of parameters, which are extracted from real systems using off-line and run-time methods. The model and algorithm have been validated on real systems using 45nm, 32nm and 22nm Intel® Core processors. The algorithm can save up to 44% energy compared with the commonly used fixed frequency policies.

Index Terms—Power Management, Heterogeneous cores

1. Introduction

Power efficiency is a key design parameter for modern processors. DVFS is the current common method for achieving the best performance for a given power budget by controlling the voltage and frequency of the CPU, e.g., by demand based algorithms [1] or for meeting Service Level Agreement (SLA) in data centers and servers [2]. Recently, new multicore based products introduce heterogeneous core architectures that combine fast, high power cores with slower, power efficient cores aiming at even better energy efficiency [3], [4] and [5]. However, controlling CPU power has limited impact on the overall energy efficiency of the computing platform due to energy consumption of other platform components: While lowering the core's voltage and frequency decreases core power and energy, computation time is lengthened resulting in an increase of energy consumed by other platform components.

Fig 1 exemplifies energy consumption of different platforms as a function CPU voltage and frequency. The target performance is achievable by operating at f_1 or faster. When the CPU power dominates total power, the energy follows the dotted curve, and minimum energy is achieved when the CPU operates at f_1. Indeed, this model is assumed in many existing designs [1] and studies [6]. More recently, when power dissipation in the rest of the platform has been considered, it has been realized that when the rest of the platform consumes significantly higher power than the CPU, platform energy follows the dashed curve, and the most energy efficient policy is Race To Halt (RtH) [7]. In many practical systems, however, power is balanced between CPU and the rest of the platform and energy is represented by the solid curve in Fig 1. In such systems the minimum energy point may happen at some intermediate frequency. This paper demonstrates this observation for the first time and presents a novel Energy Aware Race to Halt (EARtH) algorithm that identifies that minimum energy point at run time. The paper also validates the EARtH algorithm on real platforms.

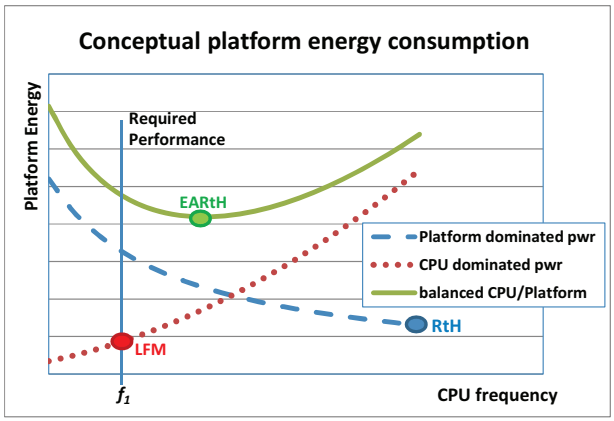

Fig. 1: Conceptual total energy in three different platforms. The minimum energy point depends on which portion of the platform dominates power consumption: LFM for CPU dominance, max frequency when rest of platform dominates, and EARtH point when power is balanced.

When considering a minimum energy point, relevant factors that affect power and performance should be accounted for. The relation of frequency to performance or overall execution time depends on parameters such as CPU and platform architecture, workload-dependent memory access patterns and memory organization. Different cores in a heterogeneous CPU may also have an effect, but in this research we focus on a single micro-architecture and leave the study of heterogeneous systems for future work. Other relevant parameters include platform and CPU power as function of workload and of CPU frequency. The EARtH algorithm presented in this paper accounts for all these parameters, and the paper demonstrates collecting them on real platforms. We instrumented platforms with two types of the Intel® Core i7 processors manufactured on 45, 32 and 22nm processes: A standard voltage CPU and an Ultra-Low Voltage (ULV) CPU. The algorithm was tested using 37 different benchmarks and at different temperatures. The paper shows that EARtH algorithm achieves the optimal minimum platform energy operation points, saving up to 44%

1. *Intel corporation*
2. *Technion – Israeli institute of technology*
3. *MS R&D and Technion – Israeli institute of technology*

Manuscript submitted: 18-Jul-2012. Manuscript accepted: 14-Aug-2012.

Posted to the IEEE & CSDL on 10/5/2012
DOI 10.1109/L-CA.2012.32 1556-6056/12/$31.00 © 2012 Published by the IEEE Computer Society

compared to either of the fixed frequency policies, namely Race-to-Halt (RtH) and Lowest-Frequency-Mode (LFM) operation points.

This paper makes the following contributions:

- **Minimum platform energy** may be achieved at an intermediate processor frequency.
- **An analytical model** calculates the minimum energy frequency using a small number of parameters.
- **EARtH algorithm** finds the optimal energy point in real platforms at run time. The algorithm is based on the required CPU and platform parameters, produced offline and collected at run time.
- **EARtH algorithm and model validated on different real platforms.** Results are accurate to within 2.2% of measured reference. Algorithm was validated over various process technologies and two different CPU types.

2. THE THEORETICAL MODEL

A workload run can be characterized as two well distinct phases: active and idle. The active phase is further split into interleaved off-chip memory-bound intervals (t_{MEM}) and CPU-bound intervals (t_{CPU}) [9],[10], [11]. While changing the CPU frequency changes the CPU run-time inversely proportional to the frequency, off-chip memory-bound intervals are not affected by the CPU frequency. Rather than measuring the time intervals directly, we used the method described below. At first glance the power and energy consumption of modern platforms as a function of CPU and system frequency seem hard to predict. However, once the different power and energy components are properly categorized, order emerges. We categorize power dissipation into the following components:

- CPU power, consumed at run time, comprises both dynamic and leakage parts, and depends on frequency and voltage.
- Platform active power, dissipated by platform as a result of workload activity. Can be further divided into two sub categories:
  - Fixed energy: During workload execution, a fixed amount of data is transferred to memory, disk drives etc. If spread over longer time, the power is lower and vice versa. , This activity is a function of the application foot print and does not depend on CPU frequency and therefore translates to fixed energy.
  - Constant power: Memory and platform peripheral devices may consume power as long as there is activity in the system. That power can be turned off during platform idle times. The energy impact of this power is proportional to the run time of the workload and therefore inversely related to CPU frequency.
- Platform Idle power dissipated by the platform regardless of workload activity (display, DDR self-refresh) Unlike constant power, it is not turned off.

We look for the minimum of the sum of all resulting energy components. CPU frequency affects the energy resulting from only CPU power and platform active constant power, and hence the optimization process focuses on them. While other, more complex dependencies exist on the platform, our study shows that those are second order effects and can be ignored by the model with minimal impact on overall accuracy.

Our model is described by the following parameters:

f_0 - *Reference lowest frequency* of *the CPU*
f_c - *Frequency, relative to* f_0. $f_c = f_{actual} / f_0$.
t_c - *CPU bound run-time at* f_c. t_{c0} *is* t_c at f_0
t_m - *Memory bound run-time, fixed for all* f_c
P_0 - *Lowest CPU power consumed at* f_0
P_c - *CPU power at* f_c. *Power scales as a function of frequency* $P_c = P_0 * F(f_c)$,
P_l - *Platform active constant power at* f_c.

Given the above notations, the frequency-dependent part E_f of platform energy is:

$$(1)\quad E_f = (t_c + t_m)\cdot(P_c + P_l) = \left(\frac{t_{c0}}{f_c} + t_m\right)\cdot(P_{co}\cdot F(f_c) + P_l)$$

For the purpose of optimization it is also more convenient to consider energy relative to the platform energy E_{f0} at the reference point f_0:

$$(2)\quad \frac{E_f}{E_{f0}} = \frac{\left(\frac{t_{c0}}{f_c}+t_m\right)\cdot(P_{co}\times F(f_c)+P_l)}{(t_{c0}+t_m)*(P_{co}+P_l)} = \left(\frac{t_{c0}}{(t_{c0}+t_m)}\cdot\frac{1}{f_c} + \frac{t_m}{(t_{c0}+t_m)}\right)\cdot \left(\frac{P_{c0}}{(P_{c0}+P_l)}\cdot F(f_c) + \frac{P_l}{(P_{c0}+P_l)}\right)$$

We define two platform and workload terms. One is CPU to Platform Power Ratio (CPR), namely the ratio between CPU power at f_0 and total platform power:

$$CPR = \frac{P_{c0}}{(P_{c0}+P_l)};\quad \text{Clearly } 1 - CPR = \frac{P_l}{(P_{c0}+P_l)}$$

The CPU power is a function of workload characteristics. It can be calculated as described in [12], [13] and [15]. Furthermore, modern CPUs report this value via a built-in power metering, and it is used in this study [14]. Note that leakage power dependency on temperature is accounted for in this power measurement. Platform active constant power component P_l does not depend on the workload; it is characterized once for this optimization. CPR≈1 implies that the platform power is dominated by CPU power while CPR≈0 implies that the rest of the platform dominates the total platform power; in real platforms, CPR lies in between these extremes.

The second parameter we define is scalability, the ratio of CPU-bound time to the total execution time, computed at f_0. We define workload scalability (SCA) as:

$$SCA = \frac{t_{c0}}{(t_{c0}+t_m)}, \text{ and clearly } 1 - SCA = \frac{t_m}{(t_{c0}+t_m)}$$

SCA is a workload characteristic that represents the performance dependency on CPU frequency. High scalability (SCA≈1) indicates that the performance is CPU bound and tightly related to frequency while low scalability (SCA≈0) indicates that the performance is memory bound and not impacted by frequency. On modern CPU

architectures it is not possible to measure workload time intervals t_c and t_m directly because they are tightly interleaved. SCA however can be extracted at run time by collecting execution parameters, as explained in [9],[10] and [11]. Furthermore, new CPUs (e.g., Intel® Core™ Sandy Bridge) uses memory stalls counters to generate scalability metric [14]; that metric has been used in this study.

The platform energy can be now expressed as:

$$\frac{E_f}{E_{f0}} = \left(SCA \cdot \frac{1}{f_c} + 1 - SCA\right) \cdot (CPR \cdot F(f_c) + 1 - CPR) \quad (3)$$

To minimize energy, we need to find the frequency that minimizes equation (3). This equation implies that the relative platform energy is a function of overall run time (which is inversely related to frequency), of the CPU power (reflected in $F(f_c)$, and depends non-linearly on frequency), and of SCA and CPR, characterizing the platform and the workload. A typical CPU power is a polynomial function of frequency $P_c \propto f_c^{\alpha}$ but the algorithm applies to other functions as well.

An interesting observation can be extracted from equation (3): CPR≈1 implies that the platform power is dominated by the CPU and the least frequency mode (LFM) policy is preferable. For CPR≈0, the power is dominated by the platform and RtH policy will achieve lowest energy operation. While previous work was limited to these two frequency extremes, we extend the analysis to the entire range in between and the extreme points become special cases of the model.

3. EARtH Algorithm

We propose the EARtH run time algorithm (Fig 2) and validate its predictions on real systems.

The EARtH algorithm requires a one-time characterization procedure and a run time module. At system production the CPU and platform power is measured at several frequencies. Based on these measurements $P(f_c)$ is obtained and stored for future use. In the case of polynomial dependency $P_c \propto f_c^{\alpha}$, only α is stored. Platform measurements yield P_l which is needed for calculating CPR. These measurements are performed once after production. At run time, the EARtH algorithm calculates CPR and SCA and finds the frequency that minimizes energy. An analytical solution of equation 3 does not yield a simple solution. Since the function is discrete with small number of frequencies, we choose a simple linear search to find the minimum energy point. This optimal frequency is passed on to frequency selection procedure (e.g. in the OS), which balances this information with the required performance level.

4. Results

We validate the predictions of the EARtH algorithm on platforms employing state-of-the-art 45, 32 and 22nm quad core Intel® 32nm Core i7 processors. Fig 3 describes measured energy that shows the existence of a minimum total energy point in an intermediate frequency.

We validated the model on two types of processors: a high power CPU (α~2.4) and an Ultra-Low Voltage CPU (α~1.5) intended for Ultrabook computers.

```
EARtH Algorithm

// Parameter initialization. Offline characterization at system
// design. Parameters stored in BIOS, or power up procedure

Get Pl                  // Get Platform Run Time Power
Get α                   // Characterize F(fc);
                        // Function can be polynomial,
                        // table or any other function.

// Run time optimization control
Every time interval {
    Pc = CPU power      // Sample CPU power meter;
                        // internal meter or calculated
    CPR = Pc/(Pc+Pl)
    Get SCA             // Read CPU monitor or use
                        // collected statistics.
    Fopt =min((SCA * 1/fc + 1 - SCA) * (CPR * fc^α + 1 - CPR))
    Freq = Get Operating System frequency request
    F(resolved) = max(Fopt, Freq)
}
```

Figure 2: The EARtH algorithm

The platforms were instrumented to measure CPU power and total platform power. First we ran a set of 37 components of Spec-2000, Spec-2006 and SYSmark [16] at two different ambient temperatures on the two CPUs at 8 different frequencies. The measured minimum energy achieved (per workload, temperature and over all frequencies) served as our reference. The EARtH algorithm predicted the optimal frequency, and the predictions were found to be within 2.2% or the reference. Note: We extracted α at fixed load and measured P_l at idle state. This procedure is not benchmark dependent and therefore applicable to any workload, as shown in this study.

Fig 4.a, 4.b and Table 1 show the potential energy savings of EARtH compared to two static frequency policies when no minimum frequency is required. The horizontal axis lists all benchmark runs, sorted in each chart according to energy savings level. Evidently, RtH is the better static policy for the low voltage CPU because the power cost of higher frequency is low compared to the rest of the platform. On the other hand, for the standard voltage

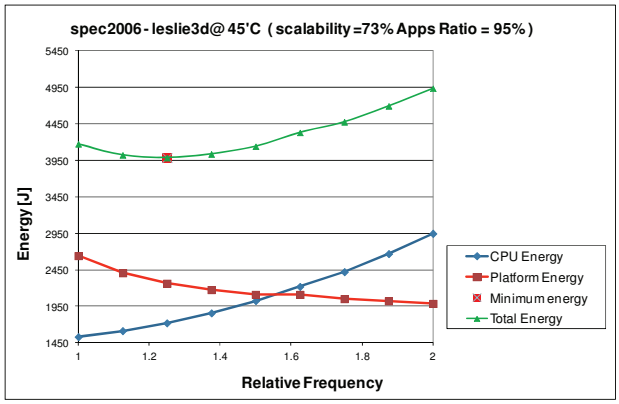

Figure 3: Energy consumption of the CPU, platform and total energy measured on a 32nm Intel® Core™ 2 duo, running SPEC2006. The measurements demonstrate the existence of minimum energy consumption at an intermediate frequency point.

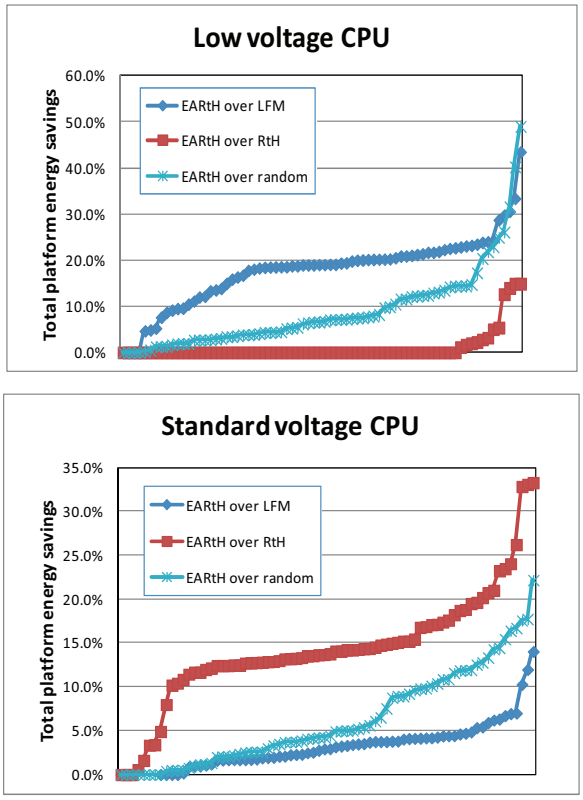

Figure 4: energy savings of EARtH algorithm for (a.) low voltage CPU and (b.) standard voltage CPU compared to LFM, RtH and random frequency policies

CPU, LFM rather than RtH is the better static policy, because the CPU consumes higher power. For comparison, a policy that randomly selects the frequency is also shown. While the random frequency policy may save energy relative to one static frequency policy or another, EARtH algorithm outperforms all three policies (all potential energy savings by EARtH are positive).

TABLE 1 SUMMARIZES THE AVERAGE AND MAXIMUM ENERGY SAVINGS OF EARTH COMPARED TO A STATIC POLICY.

Energy Savings	Standard CPU	Low Voltage CPU
Average over RtH	15.9%	1.6%
Max over RtH	33.1%	15.2%
Average over LFM	4.8%	18.4%
Max over LFM	17.0%	43.6%

5. CONCLUSIONS

The EARtH algorithm achieves the lowest platform energy required to complete a computational task by controlling voltage and frequency of the processor. We have described an analytical model for finding the minimum energy point, based on a small number of physical parameters collected at production time and at run time. The paper describes how to practically extract these required parameters on real platforms. We have demonstrated the existence of such minimum energy point, and validated the EARtH algorithm, by measurements conducted on real platforms with high and low power Intel® Core i7 CPUs manufactured on 45, 32 and 22nm processes. Energy consumption of 37 benchmarks of the SPEC2000, SPEC2006 and SYSmark at different ambient temperatures was measured. The predictions computed by the EARtH algorithm were verified against reference measurements with accuracy of 2.2%. The EARtH algorithm achieved up to 44% energy savings on these benchmarks compared to fixed voltage and frequency policies.

REFERENCES

[1] Advanced Configuration and Power Interface (ACPI) Specification, [online], Available: www.acpi.info/
[2] Fan, X., Weber, W., and Barroso, L. A. 2007. Power provisioning for a warehouse-sized computer. In Proceedings of the 34th Annual international Symposium on Computer Architecture
[3] Nvidia Kal-El, http://www.nvidia.com/content/PDF/tegra_white_papers/Variable-SMP-A-Multi-Core-CPU-Architecture-for-Low-Power-and-High-Performance_v1.1.pdf
[4] Marvell ARMADA 682, http://www.marvell.com//company/press_kit/assets/Marvell_ARMADA_628_Release_FINAL3.pdf
[5] ARM Big little http://www.arm.com/products/processors/technologies/bigLITTLEprocessing.php
[6] C. Isci, A. Buyuktosunoglu, C. Cher, P. Bose and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," In Proc. 39th Annual IEEE/ACM Int. Symp. on Microarchitecture, 2006.
[7] G. Dhiman, K. K. Pusukuri and T. Rosing, "Analysis of Dynamic Voltage Scaling for System Level Energy Management," Proc. HotPower 08 Workshop Power-Aware Computing and Systems, Dec. 2008.
[8] Patterson, M.K.; , "The effect of data center temperature on energy efficiency," Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008.
[9] C. Kihwan, R. Soma and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 24(1), 18-28. January 2005.
[10] C. Hsu and W. Feng, "Effective dynamic voltage scaling through CPU-boundedness detection," Proc. 4th Workshop Power-Aware Computer Systems, December 2004.
[11] Canturk Isci, Gilberto Contreras, and Margaret Martonosi. 2006. Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. In Proc. 39th IEEE/ACM International Symposium on Microarchitecture (MICRO 39).
[12] F. Bellosa, "The benefits of event driven energy accounting in power-sensitive platforms," Proc. 9th ACM SIGOPS European workshop: beyond the PC: new challenges for the operating platform, Sep. 2000, Kolding, Denmark.
[13] G. Contreras andM. Martonosi, "Power prediction for Intel XScale® processors using performance monitoring unit events," Proc. Int. Symp. Low Power Electronics and Design, Aug. 2005
[14] Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, Doron Rajwan, "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," IEEE Micro, vol. 32, no. 2, pp. 20-27, March-April 2012
[15] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," In ISLPED '01, pages 135–140, 2001.
[16] Standard Performance Evaluation Corporation, [online], Available: www.spec.org/