Thermal-aware Adaptive Energy Minimization of OpenMP Parallel Applications

Rishad A. Shafik, Anup K. Das, Sheng Yang, Geoff V. Merrett & Bashir M. Al-Hashimi
School of ECS, University of Southampton, SO17 1BJ, UK, *e-mail:* {*ras1n09,akd1g13,gvm,sy2u12,bmah*}*@ecs.soton.ac.uk*

***Abstract*—This paper proposes an adaptive energy minimization approach that hierarchically applies DVFS, thread-to-core affinity and dynamic concurrency controls (DCT) to minimize the energy consumption and improve lifetime reliability through balanced thermal controls, while meeting a specified power budget requirement. Fundamental to this approach is an iterative learning-based control algorithm that adapts the VFS and core allocations dynamically based on the CPU workloads and thermal distributions, guided by the CPU performance counters at regular intervals. The adaptation is facilitated through modified OpenMP library-based power budget annotations. The proposed approach is extensively validated on an Intel Xeon E5-2630 platform with up to 12 CPUs running NAS parallel benchmark applications.**

***Index Terms*—Lifetime reliability, voltage/frequency scaling.**

I. Introduction

Modern processors typically exhibit increased power density, operating temperatures and their variations. Such high power density and temperatures accelerate the device wearout mechanisms through electromigration, dielectric breakdown, etc. Hence, thermal-aware energy minimization is a key design challenge [1].

Parallel programming model is a key contributor to the continuing performance growth of current and future generations of many-core applications. OpenMP is one such programming model, considered as the *de facto* standard of shared memory multiprocessing [5]. To achieve OpenMP-based dynamic adaptation between energy and performance trade-offs, dynamic voltage/frequency scaling (DVFS) is a major runtime control knob [14]. Dynamic concurrency throttling (DCT) is another effective software knob, which selects the number of concurrent threads during runtime to trade-off performance for energy consumption. Over the years, various approaches using DVFS and DCT separately and also synergestically have been shown; examples include [1], [3], [9] etc.

Existing OpenMP-based energy minimization approaches for parallel applications have the following limitations. Firstly, existing approaches [3], [9], [12] employ DVFS and/or DCT using offline training to learn the system architecture and control parameters, thus demonstrating poor scalability. Secondly, these approaches [9], [12] do not consider control of thermal hotspots and cycling caused by uneven workloads and operating frequencies of the processing cores. Other approaches [11] use POSIX thread-based runtime implementations that require substantial application re-design and are not feasible for OpenMP parallel applications. This paper addresses the above limitations through a novel thermal-aware adaptive energy minimization approach. Section II further details the approach, Section III validates the effectiveness of the approach and finally, Section IV concludes the paper.

II. Proposed Energy Minimization Approach

Fig. 1 shows the proposed adaptive energy minimization approach organized in three steps, highlighting the interactions between application, runtime and hardware. In the first step, power budget annotation is incorporated in the application codes. This annotation, defined within and compiled by the modified OpenMP library (*libgomp*), communicate the overall power budget requirement to the runtime. The runtime, which consists of the OpenMP library and OS routines, uses this power budget to guide the online DCT and power budgeting control step at regular time intervals (we denote this as ΔT_{DCT}). This is then followed by the DVFS control step at smaller regular intervals (we denote this as ΔT_{DVFS})), guided by the monitored performance counters. The proposed energy minimization steps are further detailed in the following.

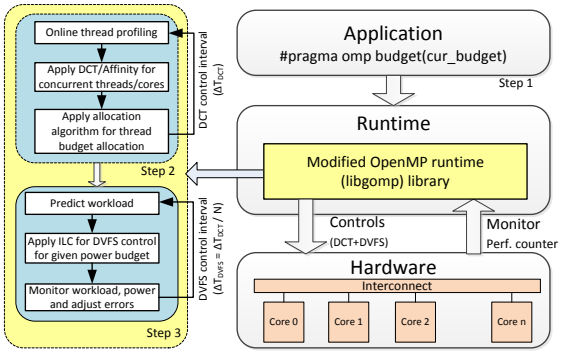

Fig. 1. Proposed energy minimization approach

A. Step 1: Power Budget Annotation

The power budget annotation in the application codes is carried out to enable energy minimization and control application performance (Fig. 1). This is done through annotating the beginning of the master thread code (the main function, not enclosed by any other OpenMP annotations) by `#pragma omp budget(cur_budget)`. The `budget` is introduced to pass the current application power budget (P_{budget}) of `cur_budget` (in mW) to the OpenMP runtime library.

B. Step 2: Online DCT and Thread Budgeting Control

With the specified application power budget, DCT and thread budget controls are applied in the following three phases:

1) Online Thread Profiling: To enable online thread workload profiling, an additional timer-based thread is introduced in the OpenMP runtime library with a period of ΔT_{DCT}. At each ΔT_{DCT}, the affinity of each computing thread (i.e. the core that is executing the thread) is obtained with a reference to the given processor to prevent any preemption and movement to another processor using `get_cpu()`. At this time, three thread and core metrics are evaluated through the performance counters using *LIKWID* [10]: (a) per thread average workload ($\overline{W_d}$) as

$$\overline{W_d} = \frac{1}{N} \sum_{t=0}^{\Delta T_{DCT}} W_{d_t}, \quad d = 0 : D \tag{1}$$

where W_{d_t} is the thread workload at each ΔT_{DVFS} interval. (b) per thread average power consumption ($\overline{P_d}$) as

$$\overline{P_d} = \frac{1}{N} \sum_{t=0}^{\Delta T_{DCT}} P_{d_t}, \tag{2}$$

where P_{d_t} is the observed power consumption at each ΔT_{DVFS} interval due to chosen DVFS (see Section II-B3).
(c) per core temperature standard deviation (std(Temp_c)) as

$$\text{std}(\text{Temp}_c) = \sqrt{\frac{1}{N}\sum_{t=0}^{\Delta T_{DCT}} \left(\text{Temp}_{c_t} - \overline{\text{Temp}_c}\right)^2}. \quad (3)$$

Upon evaluation of these metrics, the timer thread returns the processor reference back through `put_cpu()` and relinquishes the CPU and moves at the back of the thread queue, which allows for other CPUs to execute. The timer thread is statically assigned the same affinity as the main master thread.

2) DCT and Thread Affinity Control: With the evaluated metrics, the DCT control is applied with an aim of improving the system performance, while meeting the given application power budget. Moreover, the thread affinity is also managed with an aim of reducing the thermal cycles caused by uneven workloads and their DVFS controls.

Algorithm 1 shows the DCT and affinity control algorithm used to carry out such optimizations. As can be seen, with the

Algorithm 1 Iterative DCT and Affinity control algorithm

Require: $\overline{W_d}$, $\overline{P_d}$, $\overline{\text{Temp}_c}$ and std(Temp_c)
1: Calculate total power consumption, $P = \sum_{d=1}^{D} \overline{P_d}$
2: **if** P less than P_{budget} **then**
3: Increase number of parallel threads: D=D+1; $D \leq C$
4: **else if** P more than P_{budget} **then**
5: Decrease number of parallel threads: D=D-1; $D \geq 2$
6: **end if**
7: **for** each core: c=1 to C **do**
8: Sort the cores in ascending standard deviation
9: **end for**
10: **for** each core: c'=1 to C **do**
11: Swap thread affinity between cores c' and $C - c'$
12: **end for**

observed average power consumption per thread $\overline{P_d}$, the total power consumption (P) is initially found out (line 1). If P is less than P_{budget}, the number of parallel threads (D) is increased to improve system performance within the power budget (lines 2-3); otherwise, if P is higher than P_{budget}, D is decreased to reduce power consumption (lines 4-5). Decreasing the number of parallel threads reduces number of parallel computing cores and hence cuts down the core leakage power consumptions at the expense of reduced application speedup. The temperature standard deviations are then sorted in ascending order (lines 8-10). Within the sorted list, the thread affinity of the first core is swapped with the last core, the thread affinity of the second core is swapped with the second last and so on (lines 10-12). Such affinity control has the advantage of more even thermal distribution among the cores. Since the DVFS control and memory bandwidth is affected, it, however, can affect the parallel application performance with increased thread synchronization times, which is controlled in the lower level hierarchy through DVFS controls at smaller intervals ($\Delta T_{DVFS} = \frac{1}{N}\Delta T_{DVFS}$).

3) Thread Power Budget Allocation: With the given DCT and affinity controls, the thread power budgeting is reviewed and updated at each ΔT_{DCT} interval. For the new threads joining due to DCT controls in Algorithm 1, the newly allocated power budget for d'-th thread ($P_{d'_{budget}}$) is initially assigned as

$$P_{d'_{budget}} = \frac{1}{D} P_{budget}. \quad (4)$$

However, for the existing threads the power budget is updated based on the online profiled workloads in Section II-B1. The updated power budget of the d-th thread ($P_{d_{budget}}$) is given as

$$P_{d_{budget}} = \frac{\overline{W_d}\left(P_{budget} - \sum_{d'} P_{d'_{budget}}\right)}{\sum_{d=1}^{D'} \overline{W_d}}, \quad (5)$$

where D' is the number of profiled threads ($D' <= D$) and $P_{d'_{budget}}$ is the power budget of the newly joined thread.

C. Step 3: DVFS Control

With the allocated per thread power budgets, the energy minimization is carried out through DVFS based on the predicted workloads at regular intervals (Fig. 1). To effectively predict the time-varying workload at each interval, exponentially weighted moving average (EWMA) is used as the prediction scheme, similar to [13]. Using this scheme, the predicted workload at the t^{th} interval, $\hat{W_{d_t}}$ (in CPU cycles), is given by [13] as

$$\hat{W_{d_t}} = \omega W_{d_{t-1}} + \sum_{i=1}^{E} (1-\omega)^i W_{d_{t-i}} \quad , \quad (6)$$

where W_{d_t} and W_i are the previous observed workloads (in CPU cycles) at the t^{th} and i^{th} decision epochs, $1 \leq i \leq E$, ω is the moving average coefficient and E is the window size (ω and E are evaluated empirically for higher prediction accuracy). Based on the predicted workload $\hat{W_{d_t}}$ in (6) the operating frequency is determined by the iterative learning control (ILC) function as

$$f_{d_t} = f_{d_{t-1}} - \Delta f k_{d_t} K_1; \ if \ \mathcal{E}_{d_{t-1}} \approx 0 \quad (7)$$

$$f_{d_t} = f_{d_{t-1}} - \Delta f K_1 \mathcal{E}_{d_{t-1}}, \ \text{otherwise} \quad (8)$$

where $f_{d_{t-1}}$ is the previous operating frequency, Δf is the frequency differential, K_1 is ILC constant defining frequency scaling steps, $\mathcal{E}_{d_{t-1}}$ is the power budget error incurred due to previous control actions and k_{d_t} is given by

$$k_{d_t} = \exp\left[K_2\left(\frac{\hat{W_{d_t}}}{f_{max} \times \Delta t} - 1\right)\right], \quad (9)$$

where K_2 is a constant, Δt is the time interval and f_{max} is the maximum processor core frequency in the system. The power budget error ($\mathcal{E}_{d_{t-1}}$, in %) in (7) is evaluated as

$$\mathcal{E}_{d_{t-1}} = \frac{1}{P_{d_{budget}}} \sum_{i=1}^{t-1} \left(P_{d_i} - P_{d_{budget}}\right), \quad (10)$$

where P_{d_i} is the observed thread power consumption at the i-th interval. The Δf and f_{max} values can be obtained from the OS (in the case of Linux, from *sysfs* variable).

Energy minimization through ILC is implemented as an additional timer-based thread in the OpenMP runtime library with a period of Δt. This thread is statically assigned the same affinity as the main master thread (i.e. thread 0).

III. Experimental Results

To validate the effectiveness of the proposed approach, a number of experiments are carried out on an Intel Xeon E5-2630 [7] platform, which has a total of 12 cores, organized in two sockets with 6 cores each. Each core operates at a minimum frequency of 1.2 GHz (at $V_{dd} = 0.98V$) and a maximum frequency of 2.6 GHz (at $V_{dd} = 1.35V$); there are also thirteen other intermediate frequencies increasing in 0.1 GHz steps. NAS application benchmarks of class B (medium) and C (large) are executed on Linux kernel version 2.6.32. The average performance and energy consumption are recorded over 100 executions of the applications using the *LIKWID* [10]. The lifetime reliability in mean time to failure (MTTF) is estimated using the model in [8].

Fig. 2.(a) and (b) show the the execution times and energy consumption of the applications for a given power budget of 30W for four different approaches: the proposed approach, Linux's

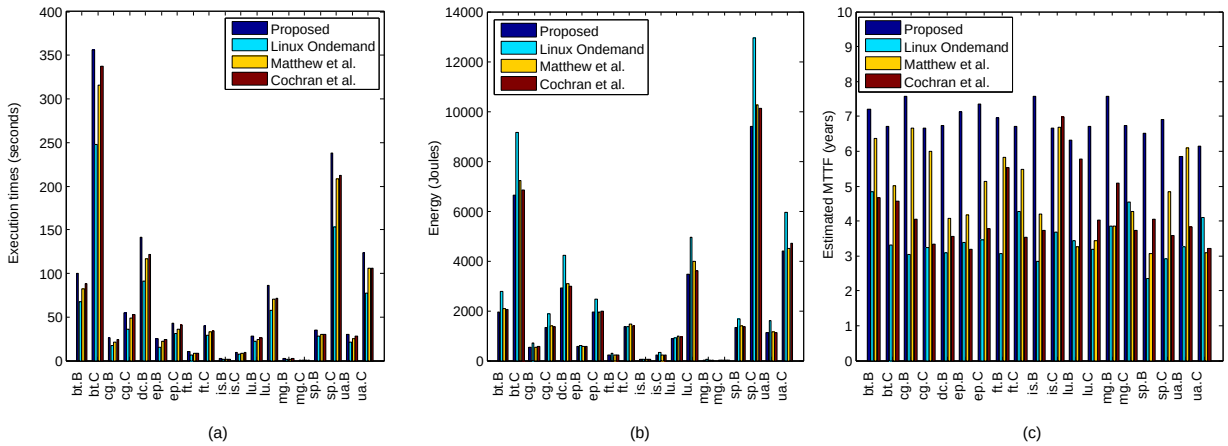

Fig. 2. Comparative (a) performance (seconds), (b) energy consumption (Joules), and (c) lifetime reliability (MTTF in years)

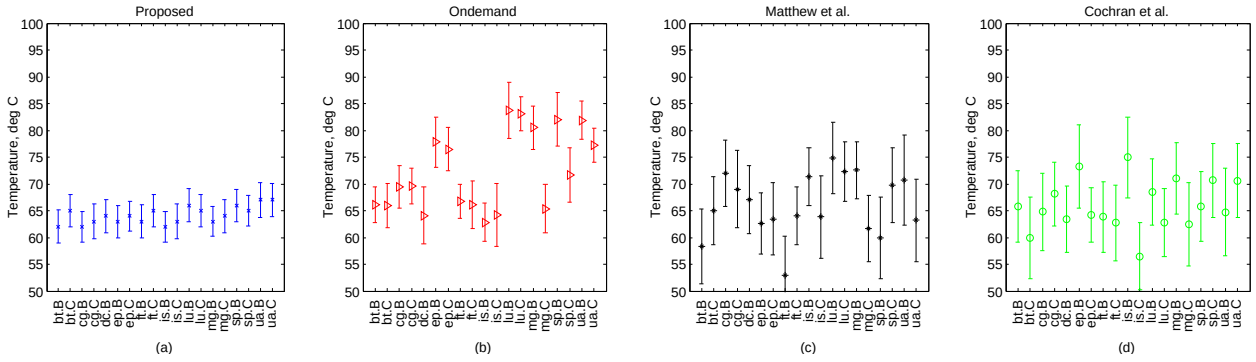

Fig. 3. Comparative average temperatures and their deviations

ondemand governor [14], and the approaches proposed in [9] and [1]. From the figures two observations can be made. Firstly, depending on the nature of the computation of the application, the execution times and energy consumptions vary dynamically. The applications *bt* (class C) and *sp* (class C) exhibit the highest execution times and corresponding energy consumptions, while *mg* and *is* show the lowest execution times and energy consumptions. Secondly, due to power budget constrained energy minimization the proposed approach outperforms the other approaches in terms of the energy consumptions (by up to 15%). The ondemand governor and the approach proposed in [9] consisently outperforms and consumes more energy than the proposed approach due to power budget unaware energy minimizations. Although the approach proposed in [1] is power budget aware but it depends largely on offline training driven look up table for online adaptation, which can scale poorly due to variable thread synchronization times [12]. The proposed approach achieves energy minimization by up to 6% due to online DCT and power budget adaptation (Section II).

Fig. 2.(c) shows the lifetime reliability comparisons of the approaches. As can be seen, the proposed approach offers better lifetime reliabilities based on the average, peak and thermal cycle based model proposed in [8]. The lifetime reliability improvements can be explained by the average temperatures and their deviations shown in Fig. 3. As can be seen, the proposed approach offers better control on the average temperatures and the thermal cycles due to adpaptive DCT and DVFS controls (Section II).

IV. Conclusions

An adaptive energy minimization approach for parallel applications is proposed. The adaptation is facilitated through OpenMP-based power budget annotations in the applications, defined in the modified OpenMP runtime library. Using the power budget annotations, the proposed approach achives thermal-aware energy minimization through hierarchical controls using DCT, thread affinity and DVFS to minimize energy consumption for the given power budget considering the thermal distributions among processor cores. The proposed approach is validated on a many-core platform running various benchmark applications, showing up to 15% reduced energy compared to the existing approaches.

References

[1] R. Cochran *et al.*. Pack & Cap: Adaptive DVFS and thread packing under power caps. *44th MICRO*, pp.175–185. ACM, 2011.

[2] M. Etinski *et al.*. Understanding the future of energy-performance trade-off via DVFS in HPC environments. in *JPDC*, 72(4), pp.579–590, 2012. 7

[3] A.K. Porterfield *et al.*. Power Measurement and Concurrency Throttling for Energy Reduction in OpenMP Programs. in *PDPS*, pp.884–891. 2013.

[4] D. Brodowski and N. Golde. Linux CPUFreq–CPUFreq governors. *Linux Kernel*.

[5] OpenMP. [Online]: http://www.openmp.org/

[6] NAS Parallel Benchmarks. [Online]: www.nas.nasa.gov

[7] Intel Xeon E5-2630. Intel® Xeon® Processor E5-2630 Family (15M Cache, 2.3GHz) [Online]: http://ark.intel.com/products/64593/

[8] J. Srinivasan *et al.*. The Case for Lifetime Reliability-Aware Microprocessors. In *SIGARCH Comput. Archit. News*, 32(2), pp.276–281, March, 2004.

[9] C-M. Matthew *et al.*. Prediction models for multi-dimensional power-performance optimization on many cores. *ICPAC*, 2008.

[10] J. Treibig *et al.*. LIKWID: Lightweight Performance Tools. Ch. in *Competence in High Performance Computing*, pp.165–175. Springer, 2012.

[11] A.K. Das *et al.* Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems. in *DAC*, pp.1–6, June, 2014.

[12] Y. Hwang, and K. Chung. Dynamic power management technique for multicore based embedded mobile devices. in *IEEE TII*, 9(3), pp.1601–1612, 2013.

[13] S. Sinha, J. Suh, B. Bakkaloglu and Y. Cao. Workload-Aware Neuromorphic Design of the Power Controller. in *IEEE JESTCS*, 1(3), pp.381–390, 2011.

[14] V. Pallipadi and A. Starikovskiy. The ondemand governor. *Proceedings of the Linux Symposium*, pp. 215–229, 2006.