Adaptive Power Efficiency: Runtime System Approach with Hardware Support

Ehsan Totoni

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
E-mail: totoni2@illinois.edu

***Abstract*—Power efficiency is an important challenge for the HPC community and demands major innovations. We use extensive application-centric analysis of different architectures to design automatic adaptive runtime system (RTS) techniques that save significant power. These techniques exploit common application patterns and only need minor hardware support. The application's pattern is recognized using formal language theory to predict its future and adapt the hardware appropriately.**

I will discuss why some system components such as caches and network links consume extensive power disproportionately for common HPC applications, and how a large fraction of power consumed in caches and networks can be saved using our approach automatically. In these cases, the hardware support the RTS needs is the ability to turn off ways of set-associative caches and network links. I will also give an overview of an RTS approach for handling process variation.

I. Introduction

Power and energy related issues are of growing concern in the broad computing landscape. Although semiconductor processes continue to give us more transistors on a chip, thermal dissipation and broader power and energy constraints limit their use. These limitations have direct impact on science and engineering applications in high-performance computing (HPC).

We propose a novel cross-layer approach to improve the power and energy efficiency of HPC systems towards Exascale. The first step is application-centric analysis of current and future systems to provide insights for improvements. We propose automatic runtime system (RTS) adaptations to the hardware based on the analysis. For these approaches to be effective, some minor hardware support that is not provided in the current systems is needed and we investigate those requirements.

Using our cross-layer approach, we develop several RTS-based techniques to save power in different system components such as processor caches [1] and network links [2]. These components are major power consumers across the whole system.

II. Cache Adaptation

A significant amount of power used by a chip is consumed by the cache hierarchy. For example, caches in POWER7 consume around 40% of the processor's power. Yet, the caches may not be utilized equally in various Computational Science and Engineering (CSE) applications and even across different phases of a single application. We develop a scheme to exploit this fact to save power by using the runtime system (RTS) to selectively turning off parts of the cache.

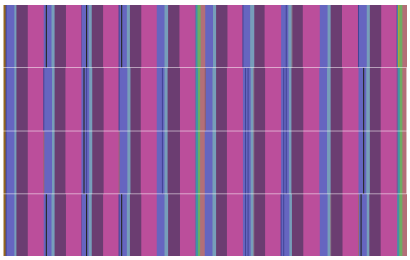


Fig. 2. Timeline view of phases of MILC: time is on x axis and four processors are stacked on y axis. Colors represent different computations. This figure illustrates the regular iterative pattern of MILC.

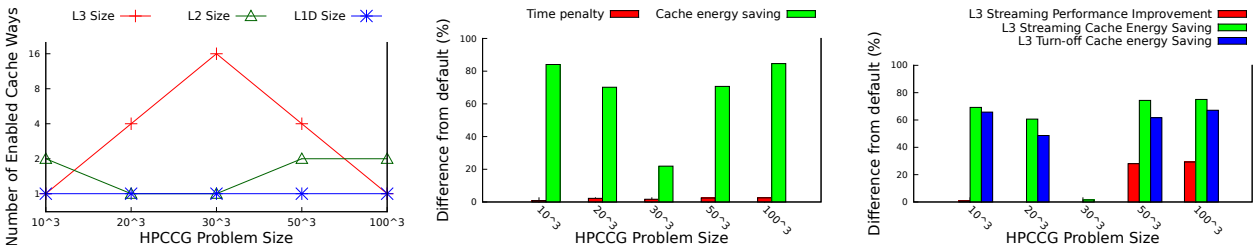
Our scheme takes advantage of the iterative and regular structure of common HPC applications to reconfigure the cache hierarchy. It uses formal language theory to recognize the application's pattern, and the Single Program Multiple Data (SPMD) model to find the best configuration concurrently. It then reconfigures the cache hierarchy by turning the ways of set-associative caches on/off. The overheads are very low since it is done only once. This approach is practical and it only requires minor hardware support since the set-associative caches are partitioned already.

Using Formal Language Theory, the hierarchical iterative structure of an HPC application can be modeled as a *Regular Language*. MILC's pattern illustrated in Figure 2 is equivalent to the regular expression $((a_0a_1a_2a_3)^5b_0b_1b_2b_3)^*$.

Our approach in the RTS can be summarized as follows:

1) Determine iterations and recognize pattern
2) Ensure the computation phases are the same across processors
3) Run different configurations on different processors and find the best in performance and power/energy efficiency
4) Apply the best configuration to all processors
5) Observe the execution and repeat if behavior changes

The RTS is the best agent to direct the reconfiguration since it can recognize the application's pattern easily, without programming effort or hardware implementation overheads. Using cycle-accurate simulations, we demonstrate that 67% of the cache energy is saved on average, while incurring only a 2.4% penalty in sequential computation. Assuming that 70% of the total power of an HPC system is consumed by its



(a) Number of Active Ways

(b) 5% Threshold Energy Saving

(c) L3 Streaming v.s. Simple Turn-off (0.5% Threshold)

Fig. 1. Reconfiguration with different input sizes

processors and 40% of each processors power goes to its caches, 19% of the total power is saved using our approach. Moreover, we established that if hardware allows it, the change of cache strategy to reconfigurable streaming can save up to 75% of cache energy and also improve performance by 30% in some cases.

Figures 1(a) to 1(c) illustrate the behavior and effectiveness of our approach for different problem sizes. Only when the input fits the cache, it is used efficiently and the RTS does not turn ways off. When the input is too small or too large, the RTS turns ways off resulting in significant power savings.

III. Network Adaptation

The interconnection network is a major consumer in the large-scale systems. In future systems, it is expected to consume more than 30% of the system's total power. From this power consumption, up to 65% is allocated to the links and the resources associated with them (and the remaining 35% is mostly consumed by routers). Therefore, saving network power is crucial for keeping HPC systems within a reasonable power budget.

Modern networks are over-provisioned in resources (e.g. links), in order to provide good performance for a range of applications. Although these networks are designed to provide enough bisection bandwidth for the worst case (e.g. all-to-all communication in FFT), not all applications make use of the abundant bandwidth. For example, common HPC applications have nearest neighbor communication pattern. The net result is that many applications do not use a large fraction of links, especially for high radix networks. Figure 3 shows that a small fraction of Dragonfly/PERCS network links are used for some applications.

An effective approach to address this problem and improve energy proportionality is to turn off unused links. Thus, we propose addition of hardware support for on/off control of links (links that can be turned on and off), which can be used by the runtime system to save the wasted power and energy consumption. We show how the runtime can accomplish that by observing the applications' behavior.

Using our basic approach, for commonly used nearest neighbor applications such as MILC, 81.5% of the links can be turned off for a multilevel directly-connected network (around 16% of total machine power, assuming 30% network power

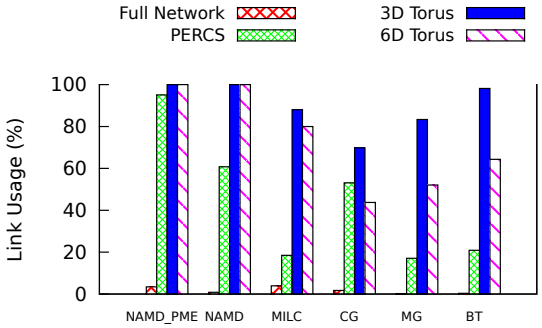


Fig. 3. Fraction of links used during execution of various applications.

budget), and 20% for 6D Torus. Moreover, we demonstrate that approximately 20% of the machine power can potentially be saved for most applications on these networks using a smarter scheduling approach.

IV. Process Variation Heterogeneity

Process variation causes the transistors on the same chip to be different, which results in cores having different frequency and power consumption profiles. This makes scheduling applications under a power budget combinatorially difficult. We model the performance and power consumption of HPC applications on such heterogeneous chips. Based on the models, we propose a scheduling framework using integer linear programming (ILP), which enables effective scheduling with various power consumption and performance constraints. Using this framework, an HPC runtime system can decide how many and which cores of a chip to use depending on the application, the properties of the chip, and the required constraints. Our results show that our framework's chooses configurations that are up to 2.5 times faster than simple heuristics. This will appear in future publications.

References

[1] E. Totoni, J. Torrellas, and L. V. Kale, "Adaptive power efficiency: Runtime system approach with hardware support," in *SC '14*.

[2] E. Totoni, N. Jain, and L. V. Kale, "Toward runtime power management of exascale networks by on/off control of links," in *HPPAC'13*.