Power Shifting Opportunities on BG/Q Using Memory Throttling

Bo Li
Virginia Tech
bxl4074@vt.edu

Edgar A. León
Lawrence Livermore National Laboratory
leon@llnl.gov

***Abstract*—Power and energy efficiency are major concerns for future exascale systems. Achieving the expected levels of application performance in these power-constrained systems will be challenging. Thus, it is important to understand how an application utilizes power throughout its different phases in order to maximize performance under the operating power budget. The goal is to shift power to the resources on the application's critical path on a code-region basis. In this project, we identify power shifting opportunities for LULESH and pF3D, two applications from Lawrence Livermore National Laboratory. Using a linear regression model to predict the minimum memory speed without affecting performance, we dynamically throttle the memory system on a per-region or per-kernel basis. The results from an IBM Blue Gene/Q system show that we can save a significant amount of dynamic power at a marginal performance cost. Resources on the critical path could use this power to improve application performance.**

I. Introduction

Two major concerns for future exascale systems are power and energy efficiency. Achieving the expected application performance in these power constrained systems requires a better understanding of the different computational phases of an application and their impact on power and energy. A runtime system can leverage this information and selectively allocate power to the resources on the critical path to improve performance. For example, if a code region is compute intensive, allocating more power to the CPU and less power to the memory system may be beneficial.

This study identifies opportunities for shifting power between hardware resources to maximize performance under a fixed power budget. We apply memory throttling to two distinct applications on a per-region and per-kernel basis and demonstrate significant power savings with a marginal effect on performance. A linear regression model based on hardware counters is used to guide the throttling mechanism. This investigation builds on our previous work [1] that analyzed the impact of memory throttling on LULESH, a mini-application for explicit hydrodynamics codes. The analysis of two different codes performed in this study significantly improves the accuracy of our previous regression model by capturing the ratio of computation to data movement. Exploring multiple applications is useful to determine whether this approach is more broadly applicable, while it provides a better understanding of the sensitivity of applications to memory bandwidth at a code phase or kernel granularity.

II. Memory Throttling on IBM Blue Gene/Q

A Blue Gene/Q (BG/Q) system is composed of compute nodes with 16 A2 cores running at 1.6 GHz and 16 GB of 1.33 GHz memory. It provides a built-in hardware feature that controls memory throttling by inserting *idle cycles* between DDR read and write operations. The throttling function *Kernel_SetPowerConsumptionParam* includes a parameter to specify the number of idle cycles ranging between 0 and 126. The memory speed can be altered at a compute node granularity by adjusting the number of idle cycles.

Using the STREAM benchmark, we performed several experiments increasing the amount of memory throttling to determine its impact on memory bandwidth. As expected, a significant decrease in effective bandwidth occurred as we increased the number of DDR idle cycles. We also measured the effective latency of the throttling function, i.e., the number of CPU cycles between the function call and a user application's change in bandwidth. This latency is within 10 μs.

III. Model-based Throttling

The computational characteristics of large applications change with code regions or kernels. For example, some regions are memory intensive while others compute bound. Our approach to power shifting takes this into consideration by applying memory throttling on a per-region or per-kernel basis. The goal is to throttle the memory as much as possible, to save power, without affecting performance. We refer to this memory speed as the *optimal*.

After an extensive space search, we discovered that the optimal memory speed for the applications studied here is a function of the problem size, code region, and concurrency. The optimal speed is dynamically adjusted using a linear regression model based on hardware counters. We use the following counters: number of CPU cycles, integer and floating-point instructions, memory load and store operations, L1 and L2 cache misses, and prefetch operations. These events capture important compute and memory characteristics. Our model is defined as follows:

$$OptIdleCycles = \sum_{i=1}^{n-1} \left(w_i \cdot \frac{c_i}{cyc} \right) + w_n \cdot \frac{c_{comp}}{c_{mem}} \quad (1)$$

A hardware counter is represented by c_i; w_i is a model coefficient generated with offline training; cyc is the number of CPU cycles; and $\frac{c_{comp}}{c_{mem}}$ is the ratio of compute-related to memory-related hardware counter data. Unlike our previous work [1], which used only the first $n-1$ terms, this model explicitly adds a factor to capture a ratio of compute to memory intensity. As shown in Figure 1, our model achieves a greater accuracy, as demonstrated by its R-squared value of 0.83.

IV. Impact on Performance and Power

We apply memory throttling on a per-region and per-kernel basis to LULESH and pF3D, respectively. The amount of throttling is determined dynamically by our regression model, which is trained offline with LULESH and applied to both

Prepared by LLNL under Contract DE-AC52-07NA27344. LLNL-ABS-658958.

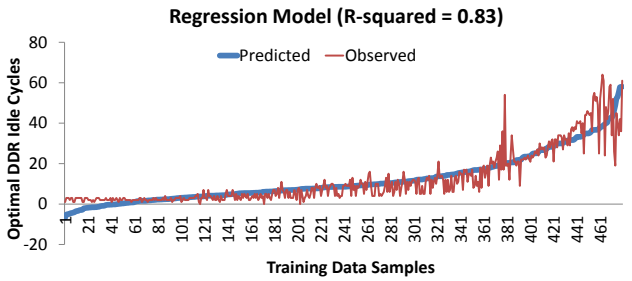

Fig. 1. Accuracy of our linear regression model in predicting the optimal.

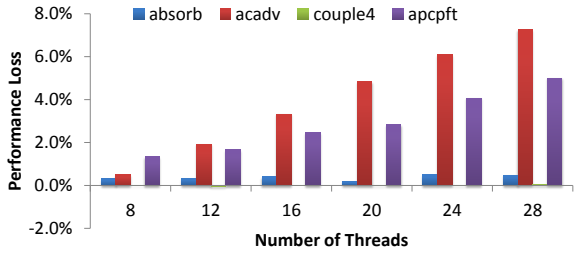

Fig. 3. pF3D performance loss of model-based memory throttling relative to full memory speed. Problem size of 256x384x288.

LULESH and pF3D. The choice of these codes is based on their ability to represent the computational characteristics of real workloads at Lawrence Livermore National Laboratory (LLNL).

The Livermore Unstructured Lagrange Explicit Shock Hydrodynamics (LULESH) mini-application contains data access patterns and computation characteristics of larger hydrodynamics codes. From an application developer's perspective, LULESH includes five regions that consume over 90% of the total execution time. They represent a mixed set of compute and memory intense characteristics.

The second application, pF3D, simulates laser-plasma interactions used at the National Ignition Facility at LLNL. We focus on the single node version of this code, which consists of multiple OpenMP kernels with different computational characteristics varying in computational and memory complexities.

In this study, the memory system is throttled by the number of idle cycles predicted by our model for each of the five regions in LULESH and for four kernels in pF3D. The hardware counters from the initial iterations are passed as input to the model, which then sets the amount of throttling for the reminder of the execution. We measure the performance degradation of our model-based throttling compared to the performance at full memory speed, and show the results for LULESH and pF3D in Figures 2 and 3 respectively.

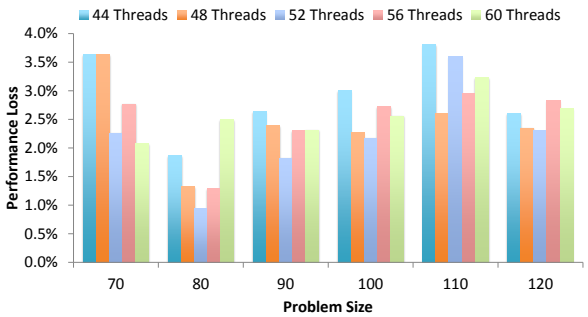

Fig. 2. LULESH performance loss of model-based memory throttling relative to full memory speed.

Figure 2 shows that in most cases our model-based throttling for LULESH has a performance loss of under 3%. In pF3D, as show in Figure 3, the compute-intense kernels *absorb* and *couple4*, have less than 1% performance loss. However, the more memory intensive kernels *acadv* and *apcpft* are sensitive to even slight changes in memory speed lower than their optimal. Additional experiments with other problem sizes and numbers of threads confirmed this finding.

Finally, we consider the impact of our model-based throttling on dynamic power and dynamic energy alongside performance for LULESH. As Figure 4 shows, in most cases, power consumption can be significantly reduced with a marginal effect on performance. The best case scenario saves up to 20% of power with a performance penalty of less than 3%. Currently, we are measuring power and energy for pF3D.

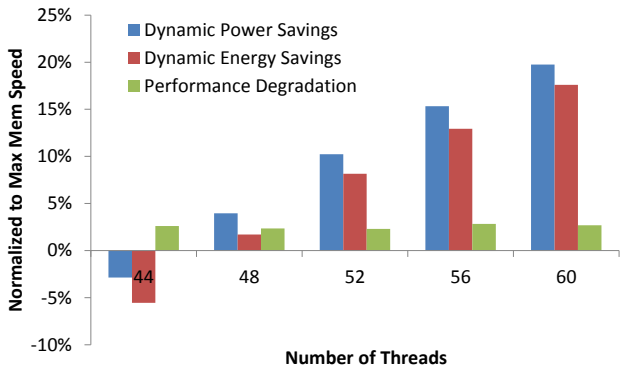

Fig. 4. Impact of model-based memory throttling on performance, energy, and power. LULESH problem size of 120.

V. Conclusions

This work investigates opportunities for power shifting among hardware resources by throttling the memory system on a code region or kernel basis. We employ a linear regression model to approximate the maximum amount of throttling to apply without affecting performance for two important codes at LLNL. Our model-based throttling achieves significant savings in dynamic power, up to 20%, at a marginal performance loss of about 3%. Future work needs to analyze additional applications and apply power shifting on other compute platforms. Further, we plan to expand our research by evaluating non-linear models based on machine learning such as artificial neural networks.

[1] B. Li and E. A. León, "Memory throttling on BG/Q: A case study with explicit hydrodynamics," in *Workshop on Power-Aware Computing and Systems*, ser. HotPower'14. Broomfield, CO: USENIX, Oct. 2014.