Using MuMMI to Model and Optimize the Energy and Performance of HPC Applications on Power-aware Supercomputers

Xingfu Wu and Valerie Taylor
Department of Computer Science & Engineering
Texas A&M University, College Station, TX 77843
Email: {wuxf, taylor}@cse.tamu.edu

Abstract

The MuMMI (Multiple Metrics Modeling Infrastructure) facilitates systematic measurement, modeling, and prediction of performance and power consumption, and performance-power tradeoffs and optimization for Power-aware HPC systems. In this paper, we use the MuMMI to model performance and power. These models focus on four metrics: runtime, system power, CPU power and memory power. We rank the counters from these models to identify the most important counters for application optimization focus, then demonstrate the counter-guided optimizations with an aerospace application PMLB executed on two power-aware supercomputers, Mira at Argonne National Laboratory, and SystemG at Virginia Tech.

1. Introduction

The MuMMI [8, 4] shown in Figure 1 was developed to provide an infrastructure that facilitates systematic measurement, modeling, and prediction of performance and power consumption, and performance-power tradeoffs and optimization for HPC systems. It consists of three main components: Instrumentor, Database and Analyzer. The MuMMI database stores power and energy consumption and hardware performance counters' data with different CPU frequency settings.

The MuMMI Instrumentor (called MAIDE) shown in Figure 2 provides automatic performance and power data collection and storage with low overhead and uses PAPI [5] to collect performance counter data, and uses PowerPack [1] to collect power data on SystemG [6]. The MAIDE provides a way to do source-code level automatic instrumentation for Fortran77, Fortran90, C and C++ programs. As shown in Figure 2, at the end of the program execution, the power and performance data files are automatically transferred to the MuMMI databases using SOAP scripts by the MAIDE. Prior to invoking the SOAP client side to transfer the data files, the data files are converted to SQL scripts based on the database schema so that the SOAP server side receives the data files, and uploads the data to the database. The MuMMI Analyzer (E-AMOM) entails hardware performance counter-based performance and power modeling, and performance-power tradeoff and optimizations.

Recently, we ported the MAIDE to use MonEQ [9] to measure power on Mira [3]. In this paper, we use the MAIDE to instrument the aerospace application PMLB [7] to collect power, performance and hardware performance counter data on Mira and SystemG.

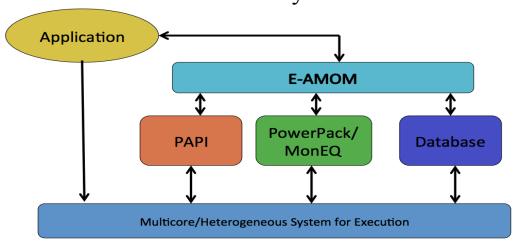

Figure 1. Multiple Metrics Modeling Infrastructure

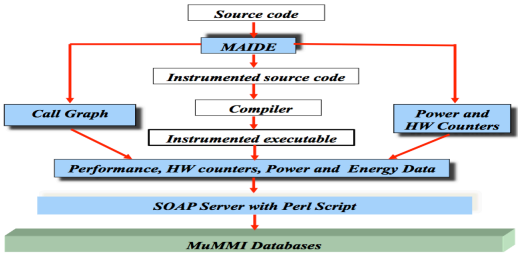

Figure 2. MuMMI Instrumentor MAIDE

2. Modeling and Optimization Framework

Figure 3 shows the modeling and optimization methodology in this work. For a HPC application executed on a power-aware system, we use MAIDE [8] to collect runtime, power (node, CPU and memory), and performance counter data, and automatically upload the data to the database.

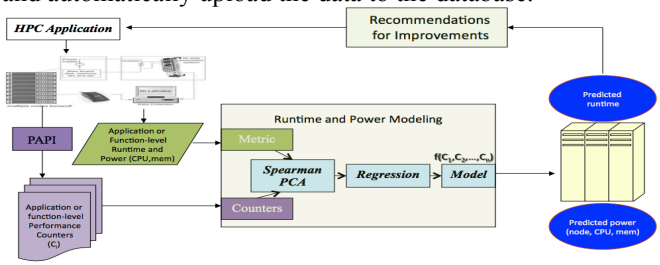

Figure 3. Modeling and Optimization Framework

During the application execution we capture 39 available underived performance counters using the PAPI. All performance counters are normalized using the total cycles

of the execution to create performance event rates for each counter. Then, using Spearman correlation and principal component analysis (PCA), we identify the important performance counters for the four metrics - runtime, system power, CPU power, and memory power, and use non-negative multivariable regression analysis to generate the four models based upon the set of important performance counters and CPU frequencies. Our previous work [8] indicates these runtime and power models are accurate with prediction error rate less than 8% in average for all applications we conducted. Then we can identify the performance counters that are important across the multiple metrics for possible application optimizations.

3. Case Study: PMLB

In the parallel multiblock Lattice Boltzmann (PMLB) aerospace application [7], we use the D3Q19 lattice model (19 velocities in 3D) with the collision and streaming operations. The code is written in C, MPI and OpenMP. The fixed problem size is the 3D mesh with 128x128x128 executed on up to 128 cores on SystemG and with 128x128x128 and 512x512x512 executed on up to 32,768 cores on Mira.

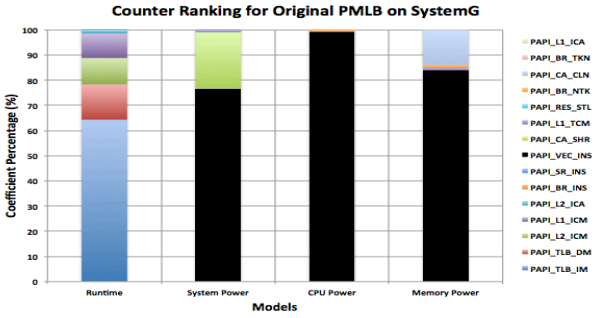

Figure 4. Ranking for the original PMLB on SystemG

Figure 4 shows the performance counter ranking for four models with 15 different counters of the original code PMLB on SystemG, where TLB_IM (instruction translation lookaside buffer (TLB) misses) has the highest rank among these counters. VEC_INS (vector/SIMD instructions) has the second highest rank. Both TLB_IM and VEC_INS are not correlated each other. Therefore, based on the insight, we focus on the counters TLB_IM and VEC_INS for optimization on SystemG.

We utilized the 2MB huge pages for the application execution through the use of libhugetlbfs [2] to reduce the TLB misses, vectorized the code, and used the compiler option *-ftree-loop-distribution* to perform loop distribution to improve cache performance on big loop bodies and allow further loop optimizations like vectorization to take place. The optimized results show the total energy saves by the average 11.28% with the problem size 128x128x128 as shown in Figure 5. Similarly, the total energy saves by the average 15.49% with the problem size 512x512x512 on up to 32,768 cores on Mira as shown in Figure 6.

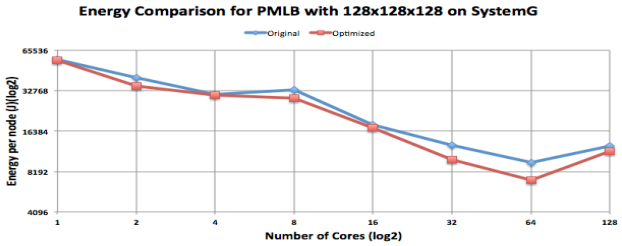

Figure 5. Energy per node on SystemG

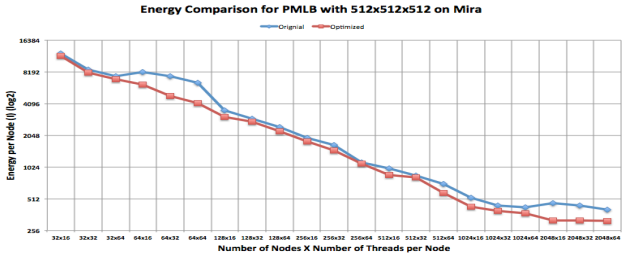

Figure 6. Energy per node on Mira

4. Summary

In this paper, using MuMMI to model the runtime, system power, CPU power and memory power indicates that different performance counters are explored for different models and different applications on different systems because of different application and architecture characteristics. Performance counters' values are correlated with properties of an application that impact performance and power, so identifying important performance counters provides the necessary insights for application optimizations in energy and performance. We believe that our power and performance modeling and optimization methodology can be applied to HPC applications executed on other architectures such as GPU and Xeon Phi by utilizing counters for GPU and Xeon Phi, and can provide optimization guidance for application and system developers for energy efficient development.

References

[1] R. Ge, X. Feng, S. Song, et al., PowerPack: Energy Profiling and Analysis of High Performance Systems and Applications, *IEEE Trans. on Para. and Dis. Sys.* 21(5), 2010.

[2] libhugetlbfs, http://sourceforge.net/projects/ libhugetlbfs/

[3] Mira, https://www.alcf.anl.gov/mira

[4] Multiple Metrics Modeling Infrastructure (MuMMI) project, http://www.mummi.org/info.

[5] PAPI (Performance API), http://icl.cs.utk.edu/papi/

[6] SystemG, http://www.cs.vt.edu/facilities/systemg.

[7] X. Wu, V. Taylor, S. Garrick, D. Yu, and J. Richard, Performance Analysis, Modeling and Prediction of a Parallel Multiblock Lattice Boltzmann Application Using Prophesy System, *IEEE Cluster* 2006.

[8] X. Wu, V. Taylor, et al., MuMMI: Multiple Metrics Modeling Infrastructure (Book Chapter), *Tools for HPC 2013*, Springer 2014.

[9] S. Wallace, V. Vishwanath, et al., Application Power Profiling on Blue Gene/Q, *IEEE Cluster* 2013.