Energy Tuning of Polyhedral Kernels on Multicore and Many-Core Architectures

***Abstract*—Reducing application energy consumption is important to improve user experience of embedded systems, e.g. smart phones, GPS navigators etc. However, applications (kernels) are not energy-aware in its design stage. Tuning for energy efficiency is necessary to meet the system's energy constraints. In this work we use energy-aware framework to tune Polybench kernels for energy on two architectures (Intel SandyBridge and Intel MIC). We applied combinations of loop optimizations to the kernels and observed for dense matrix kernels like `2mm` and `gemm`, *the optimization that has the minimum execution time did not consume the least amount of energy*. This means execution time does *not* always correlate to the energy consumption. We also found that the optimizations that work best on one architecture do not carry to another.**

I. Introduction

Reducing application energy consumption is important to improve user experience of embedded systems, e.g. smart phones, GPS navigators etc. However applications (kernels) are not energy-aware in its design stage. Tuning for energy efficiency is necessary to meet the system's energy constraints. Tiwari et al. performed coarse-grain measurement and provided evidence for the existence of opportunities to auto-tune for energy in parallel applications[1]. Existing Polyhedral auto-tuning frameworks have shown promising results in tuning for execution for small computation kernels like the Polybench programs[2]. In this work we use the framework added with fine-grained energy measurement support to enable tuning for energy. We choose 6 representative Polybench kernels (from linear-algebra, machine learning, and stencils code) and tune them on two architectures (Intel SandyBridge and Intel MIC). When tuning the application for better energy usage, some combination of loop optimizations, including loop tiling, loop unrolling, and loop fusion, are performed on kernels along with the auto-parallelization. We observed for dense matrix kernels, *the optimization that has the minimum execution time did not consume the least amount of energy*. This differs from previous observations [3]. We also found that the optimizations that work best on one architecture do not carry to another.

This work has three main contributions: (1) Introducing the energy-aware framework for energy auto-tuning (2) Observing that it is *not* always the case to be able to reduce energy consumption by reducing execution time. (3) Evaluation of architectural differences of energy behaviors corresponding to the same loop transformations.

II. Energy Measurement-and-Tuning Tools

For energy consumption reporting, `RCRtool` is available to any application through a region-based call API. When an application halts elapsed time, energy use, and average computed power is reported for (a) each region and (b) the entire application.

The Intel SandyBridge architecture allows users to track energy usage through the exposed Running Average Power Limit (RAPL) hardware counter. The `RCRtool` records the current energy value of the counter at least 1000 times a second by writing into a shared-memory data structure.

On MIC `RCRtool` tracks power usage through a file updated every 50 millisecond. RCRtool monitors the power at user level and computes the energy consumption over time.

Finding the optimal combination of compiler optimizations requires a compilation framework. The Polyhedral Compiler Collection (PoCC) was used to generate program variants with different optimizations. Intel Compiler v14.0.2 was used as the backend compiler for this work.

III. Benchmarks for Energy Measurement and Tuning

In this work, we evaluate six representative Polybench programs for energy auto-tuning using loop optimizations: `2mm`, `gemm`, `gramschmidt`, `covariance`, `seidel-2d`, and `jacobi-2d`.

Using PoCC, program variants were generated using a different set of the optimizations from the following four groups:

- Loop fusion: smartfuse, maxfuse, nofuse
- Loop unrolling factor: 1, 2, 4, 8
- Loop tiling: 1, 16, 32, 64
- Loop vectorization: on, off

Loop parallelization was always turned on.

IV. Experimental Results

First we report the speedups and energy saving achieved on SandyBridge applying parallelization and loop optimizations. Then we show a cross-architecture

comparison on the best loop optimizations that work the best on one architecture do not necessarily work the best on the other, in terms of execution time and energy consumed.

A. Speedups and Energy Savings

TABLE I
TIME AND ENERGY COMPARISON OF OPTIMAL CONFIGURATION TO BASELINE OF POLYBENCH KERNELS ON SANDYBRIDGE

Benchmark	*Execution Speedup*		*Energy Reduction*	
	SM	LG	SM	LG
2mm	1.48X	1.36X	1.28X	0.77X
covariance	37.86X	122.20X	25.11X	60.31X
gemm	1.46X	1.44X	1.28X	0.78X
gramschmidt	19.41X	21.64X	13.66X	11.72X
jacobi-2d	1.31X	1.48X	1.37X	1.44X
seidel-2d	7.76X	9.60X	7.68X	5.53X

`SM` and `LG` represent the two dataset sizes *small* and *large*. For each benchmark we compare the best optimization sequenced version (`best`) to the original (`O`), measuring execution time and energy consumed. We report the execution speedup as T_{best}/T_0 and energy reduction as E_{best}/E_0. If a given energy value reported in IV-A is less than one, energy consumption increased.

B. Cross Architecture Comparsion

Figure 1(a) shows the best optimization sequences of the six benchmarks chosen from SandyBridge do not perform as good when run on the MIC architecture. Comparing with the optimal optimization sequence on the MIC, the SandyBridge sequences incurred significant increase in execution time and energy consumption for 2mm, covariance, and gemm. The SandyBridge optimization sequences performed as good as the optimal sequence on the MIC for the other three benchmarks. Figure 1(b) compares the time and energy (running on Sandy Bridge) of the best optimization sequences chosen from MIC with those of the optimal optimization sequences chosen from SandyBridge. 2mm, covariance, gemm, and seidel-2d consumed from 200% to 350% more energy and time.

V. CONCLUSION

In this work, we tuned six representative Polybench kernels for energy on an Intel SandyBridge processor and an Intel Xeon Phi coprocessor by applying various loop transformations. For the `2mm` and `gemm` kernels (dense matrix kernels), we observed *non-correlated* speedups and energy savings over the baseline version, i.e. *fastest execution does not guarantee fewest energy consumption*. We also showed that good loop transformations for one architecture do not carry over to other architectures.

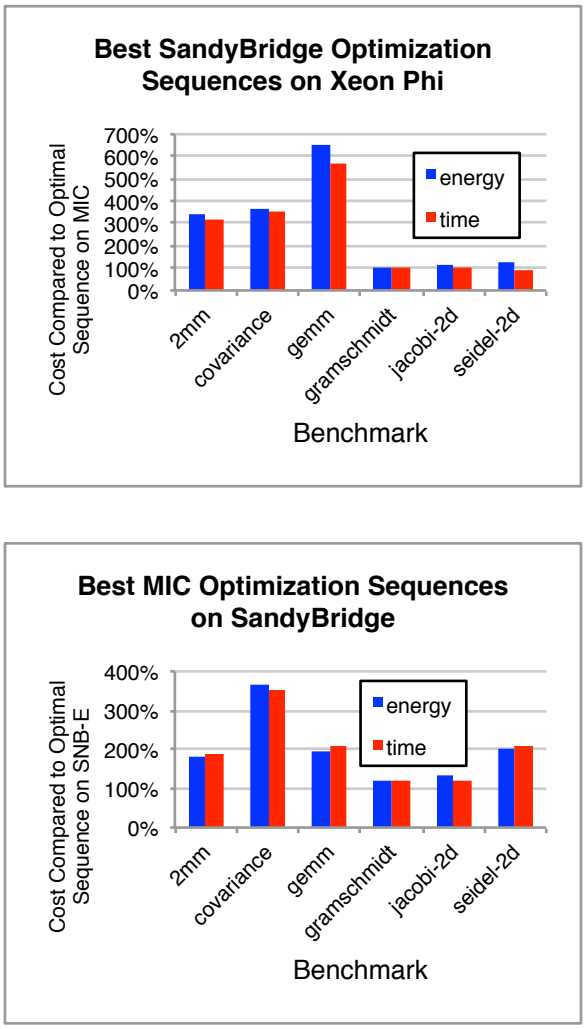

Fig. 1. Cross-Architecture Analysis of Best Optimization Sequences

REFERENCES

[1] A. Tiwari, M. A. Laurenzano, L. Carrington, and A. Snavely, "Auto-tuning for energy usage in scientific applications," in *Proceedings of the 2011 international conference on Parallel Processing - Volume 2*, ser. Euro-Par'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 178–187.

[2] E. Park, J. Cavazos, and M. A. Alvarez, "Using graph-based program characterization for predictive modeling," in *CGO*, 2012, pp. 196–206.

[3] W. Wang, J. Cavazos, and A. Porterfield, "Energy auto-tuning using the polyhedral approach," in *Proceedings of the 4th International Workshop on Polyhedral Compilation Techniques*, S. Rajopadhye and S. Verdoolaege, Eds., Vienna, Austria, Jan. 2014.