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

William Killian, Wei Wang, Eunjung Park, John Cavazos

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE, UNIVERSITY OF DELAWARE

Introduction

- Reducing application energy consumption is important in embedded systems.
- Applications are not energy-aware in its design stage.
- Tuning for energy efficiency is necessary to meet system energy constraints.

METHODOLOGY:

- We tuned for energy consumption on Intel® SandyBridge and Intel® Knight's Corner.
- We applied polyhedral optimizations to kernels
- We found the optimization sequence yielding minimal energy consumption.

EVALUATION:

- In-depth comparison on SandyBridge of execution time and energy consumption
- Cross-architecture comparison of optimization sequences

Energy Measurement

- Used `RCRtool` [3] to report region-based energy consumption
- Recorded elapsed time, energy use, and average power for region + application

SANDYBRIDGE:

- `RAPL` hardware counter used
- 1000Hz+ update frequency
- Measures energy, computes power

KNIGHT'S CORNER:

- Built-in power measurement used (`/sys/class/micras/power`)
- 20Hz update frequency
- Measures power, computes energy

Polyhedral Optimizations

- Use the polyhedral model of an AST to transform loops
- Examples include loop unrolling, tiling, fusion, autoparallelization, and autovectorization
- Used the Polyhedral Compiler Collection (PoCC) [2] for automatic version generation

LOOP UNROLLING:

```
// no unrolling
for (int i = 0; i < n; ++i) {
    C[i] = A[i] + B[i];
}

// Unroll factor=4
for (int i = 0; i < n / 4; ++i) {
  C[4*i+0] = A[4*i+0] + B[4*i+0];
  C[4*i+1] = A[4*i+1] + B[4*i+1];
  C[4*i+2] = A[4*i+2] + B[4*i+2];
  C[4*i+3] = A[4*i+3] + B[4*i+3];
}
```

SAMPLE USAGE OF POCC:

```
$ pocc --pluto-parallel --pragmatizer --mark-par-loops
    [--pluto-prevector] [--pluto-tile N[xN]*]
    [--pluto-fuse (maxfuse,nofuse,smartfuse)]
    [--pluto-ufactor N] 2mm.c
```

Optimization Flag Affecting:

LOOP FUSION -- LOOP UNROLLING -- LOOP TILING
AUTOPARALLELIZATION -- AUTOVECTORIZATION

TILE SIZES: 1,16,32,64 UNROLL FACTORS: 2,4,6,8

Note: we always selected an optimization sequence with autoparallelization. All other optimizations were optional.

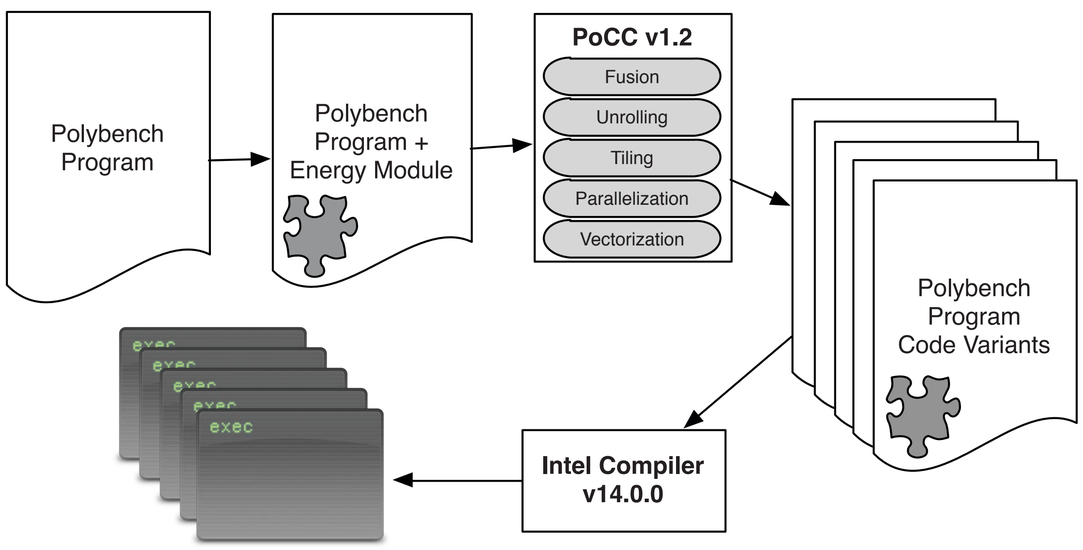
Benchmarks

- Synthetic benchmark is a subset of the Polybench kernels [1] ranging from various domains

POLYBENCH KERNELS:

- `2mm` -- two matrix multiplication (D = A.B, E = C.D)
- `covariance` -- covariance computation (data mining)
- `gemm` -- matrix multiplication (C = alpha.A.B + beta.C)
- `gramschmidt` -- decomposition (linear algebra)
- `jacobi-2d` -- 2D Jacobi stencil (fluid dynamics, image processing)
- `seidel-2d` -- 2D Seidel stencil computation

Compilation Workflow



1. Original Polybench program with no modifications
2. Change generic *Polybench* header file to include energy monitoring API (RCRtool)
3. Use *PoCC* to generate search-space versions of original polybench program
4. Use *Intel® Compiler* to compile the version codes targeting Knight's Corner or SandyBridge
5. Execute all binaries and manually analyze energy/performance results

Results

- Generated 2535 versions of each benchmark with two different dataset sizes (Small and Large)
- CPU configuration: dual-socket quad-core Intel® Xeon® CPU E5-2603 @ 1.80GHz, 32GB DDR3
- MIC configuration: 60-core Intel® Xeon® Phi Coprocessor 5110P @ 1.053 GHz, 8GB GDDR5
- Compilation configuration: **CPU:** `-O3 -xHOST -openmp` **MIC:** `-O3 -mmic -openmp`
- Ran each benchmark 5 times and took the average of the middle three executions

Metrics	$Speedup = \frac{T_0}{T_{best}}$	$E_{reduction} = \frac{E_0}{E_{best}}$

ENERGY ANALYSIS

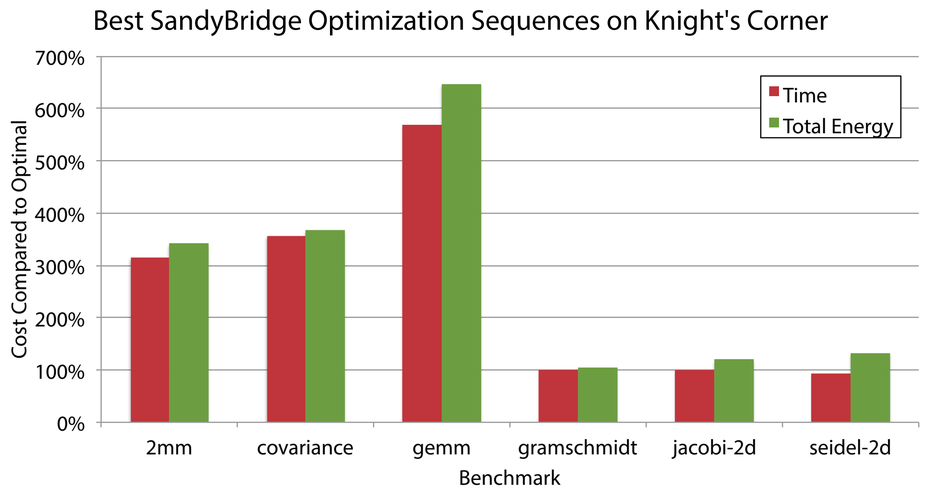
Performance and Energy Improvement on Intel® Xeon® Processor

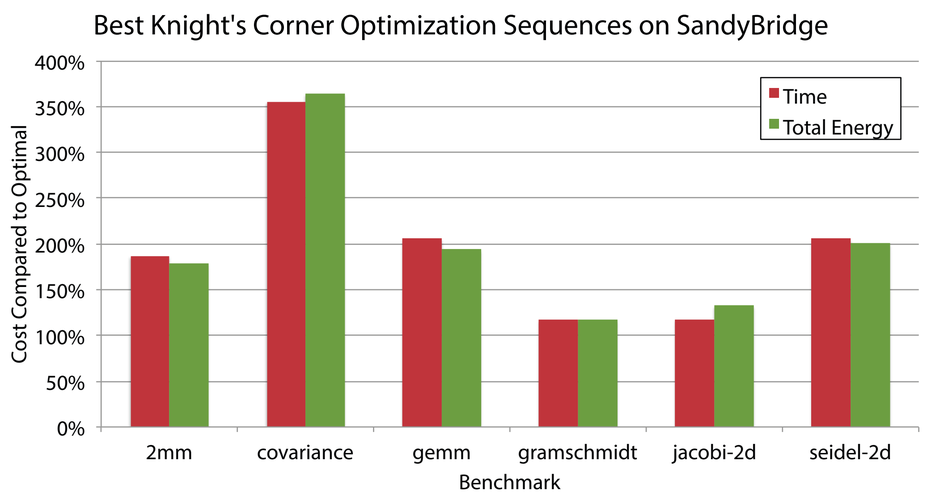
Benchmark	*Speedup*		*Energy Reduction Factor*	
	SM	LG	SM	LG
2mm	1.48x	1.36x	1.28x	0.77x
covar	37.86x	122.20x	25.11x	60.31x
gemm	1.46x	1.44x	1.28x	0.78x
gramschmidt	19.41x	21.64x	13.66x	11.72x
jacobi-2d	1.31x	1.48x	1.37x	1.44x
seidel-2d	7.76x	9.60x	7.68x	5.53x

Performance and Energy Improvement on Intel® Xeon® Phi Coprocessor

Computation for non-parallelized versions took too long to execute on the platform (> 1 hour for the large dataset). We only used the Many-Integrated core architecture for cross-architecture comparison

CROSS-ARCHITECTURE COMPARISON





These graphs indicate that good loop transformations for one architecture do not carry over to other architectures. Performance improvements at times approached or even beat the best optimization sequence. When analyzing total energy consumption, this trend no longer holds.

Conclusion

- Polyhedral optimizations yield up to 60x energy reduction and 120x speedup (28% minimum)
- Non-correlated speedup observed on dense matrix kernels
- Loop transformations do not carry over well across architectures
- Adapting energy-aware API to new benchmarks is easy with RCRtool

FUTURE WORK:

- Explore new benchmark suites and architectures (ARM, Haswell, Silvermont)
- Expand search-space to include more polyhedral optimizations; reduce search breadth
- Improve energy measurement granularity on Knight's Corner

SOURCES / RELEVANT WORK:

[1] L. Pouchet. "Polybench/C" available: http://www.cse.ohio-state.edu/~pouchet/software/polybench/
[2] L. Pouchet. "PoCC - The Polyhedral Compiler Collection" available: http://www.cse.ohio-state.edu/~pouchet/software/pocc/
[3] Allan Porterfield, Rob Fowler, Min Yeol Lim. RCRTool: Design Document. Technical Report TR-10-01, RENCI, North Carolina, Feb. 2010
[4] 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. EuroPar'11. Berlin, Heidelberg: Springer- Verlag, 2012, pp. 178–187
[5] E. Park, J. Cavazos, and M. A. Alvarez, "Using graph-based program characterization for predictive modeling," in CGO, 2012, pp. 196–206
[6] 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
[7] W. Baek and T. Chilimbi. Green: A framework for supporting energy-conscious programming using controlled approximation. PLDI, June 2010
[8] J. Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. ACM Trans. Comput. Syst., 22(2):137–179, May 2004