Energy Auto-Tuning using the Polyhedral Approach

Wei Wang[1] John Cavazos[1] Allan Porterfield[2]

[1]Dept. of Computer & Information Sciences
University of Delaware
[2]RENaissance Computing Institute (RENCI)
University of North Carolina-Chapel Hill

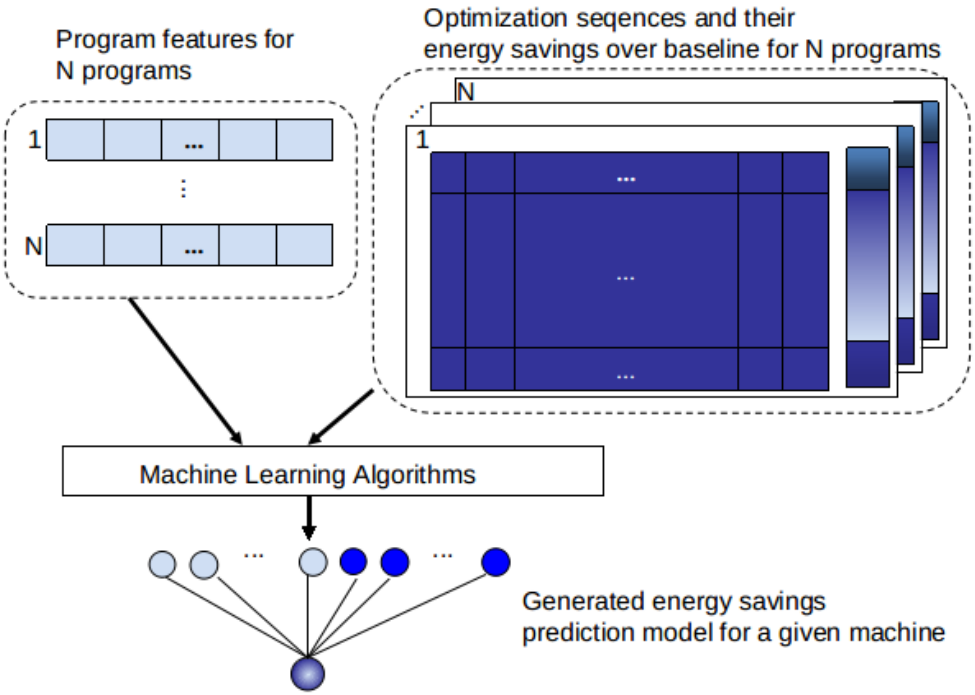
January 20, 2014

Introduction

- Application Energy Consumption
  - Optimizing for lower energy has become critical when we approach Exascale Computing.
- Tuning for faster execution vs. tuning for lower Energy?
  - Knowledge of the relationship between the two will guide auto-tuning process.
- Energy Impact of Polyhedral Optimizations
  - Not well understood.
  - Polyhedral optimizations barely studied on non-trivial/realistic applications.

Auto-tuning Framework

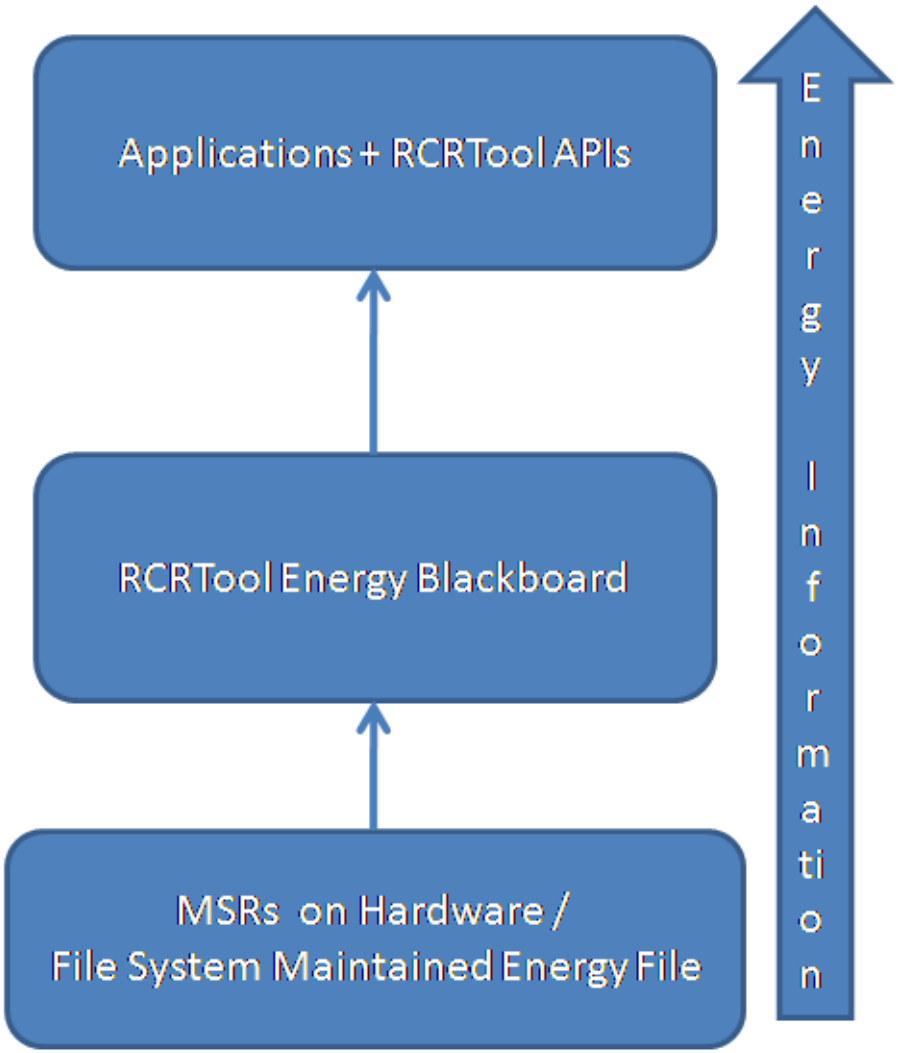
- Program Characterization
  - Control Flow Graph(CFG)
  - Source Code, Performance Counters, ...
- Optimization Sequences
  - Src-to-Src Compiler
- Energy Profiling
  - Energy Related Counters
- Machine Learning Algorithms
  - SVM
  - Linear Regression,...



Auto-tuning for time is very effective, especially using CFG as program feature.
(Refs: Park et al. CGO'11, CGO'12, IJPP'13)

Energy Measurement using RCRTool

- MSRs/Energy File: Instantaneous Energy
- RCRTool Energy Blackboard: Accumulated Energy
- RCRTool API calls: Records energy consumption of executed application codes



Energy Measurement using RCRTool

- Architecture Tested
  - Sandy Bridge, Ivy Bridge
    Shared memory stores MSR counters.
    Update frequency: > 1000/s.
    Supported Language: OpenMP, MPI.
  - MIC
    Shared memory stores energy obtained from PAPI and Intel MICAccessSDK.
    Update frequency: about 20/s.
    Host version and MIC-native version.
    Supported Language: OpenMP (offload and native), OpenCL (host).

RCRTool Exposed APIs

- energyDaemonInit()
- energyDaemonEnter(): Start/Resume measurement when entering a region.
- energyDaemonExit(file, line_no): Stop/Pause measurement upon exiting the region
- energyDaemonTerm()
- energyDaemonTEStart(): Start measuring Time and Energy
- energyDaemonTEStop(): Stop measuring Time and Energy

Exposed APIs-Example

Original OpenMP program

```
1  int main() {
2
3    initialize();
4
5
6    while  {
7
8      #pragma omp parallel for
9      compute_region_1();
10
11
12     serial_code();
13
14
15     #pragma omp parallel
16     #pragma for
17     compute_region_2();
18
19
20   }
21
22
23   finalize();
24 }
```

Added with energy profiling call

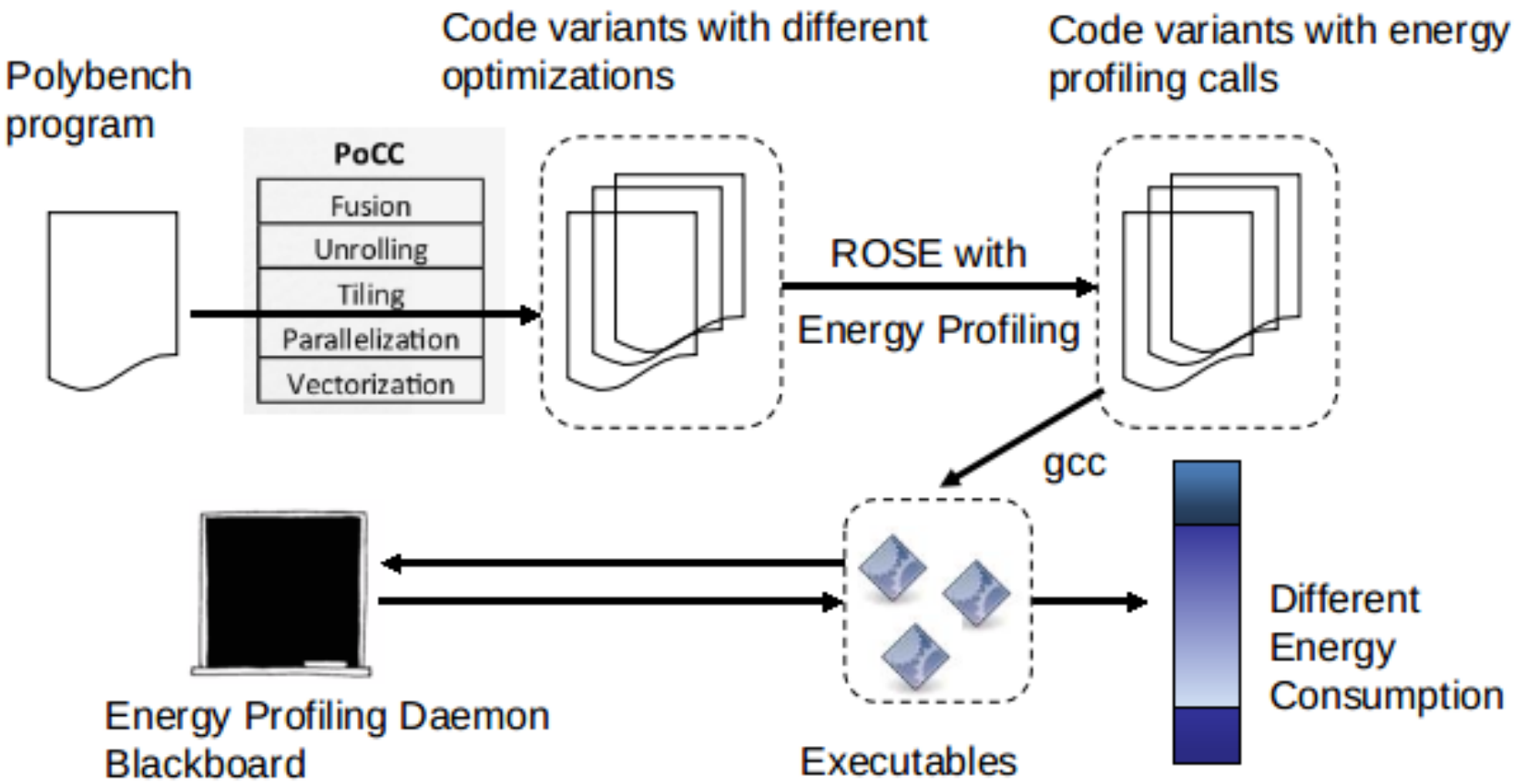
```
1  int main() {
2    energyDaemonInit();
3    initialize();
4
5    energyDaemonTEStart();
6    while  {
7      energyDaemonEnter();
8      #pragma omp parallel for
9      compute_region_1();
10     energyDaemonExit("thefile", 10);
11
12     serial_code();
13
14     energyDaemonEnter();
15     #pragma omp parallel
16     #pragma for
17     compute_region_2();
18     energyDaemonExit("the file", 18);
19
20   }
21   energyDaemonTEStop();
22
23   finalize();
24   energyDameonTerm();
25 }
```

Polyhedral Compilers

Generate code variants of a program containing Static Control Parts (SCoP) using PoCC (Polyhedral Compiler Collection).

- Loop Transformations
- Auto Parallelization (PLUTO)
- Tested Applications
  Existing: Polybench
  New: 2D Cardiac Wave Propagation Simulation, LULESH (C/C++)

Energy Profiling of Different Program Optimizations



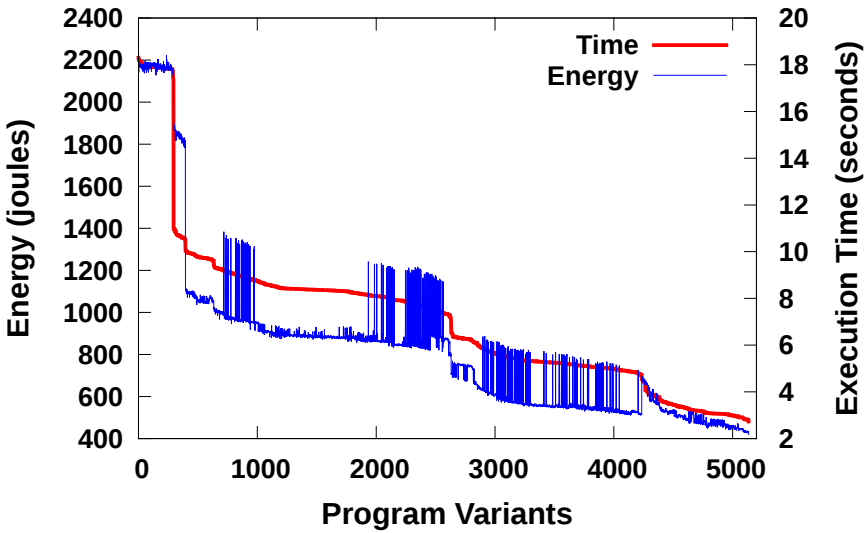
Workflow of energy-aware polyhedral framework

Experiments Setup

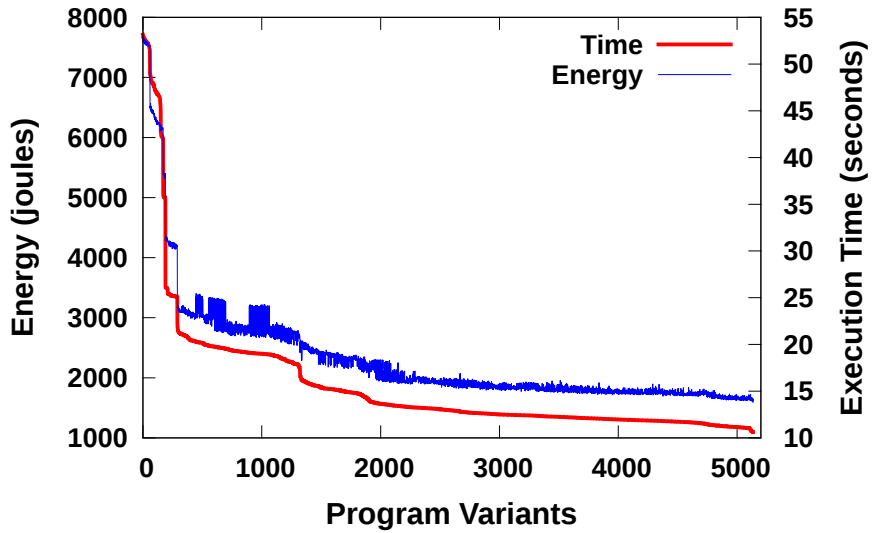
- Hardware
  - Intel Xeon E5-2680 (dual socket 8-core processor with 20MB cache)
  - Xeon Phi coprocessor (61 cores, 1.09GHz, 512KB cache each)
- Software
  - Polyhedral Compilers: PoCC v1.2 and Polyopt v0.2.1
  - Application: Polybench v3.2 and LULESH v1.0 (OpenMP)
  - Back-end Compilers: GCC v4.4.6 and ICC v14.0.0

Energy Consumption and Execution Time Correlation (Polybench)

Covariance Polybench

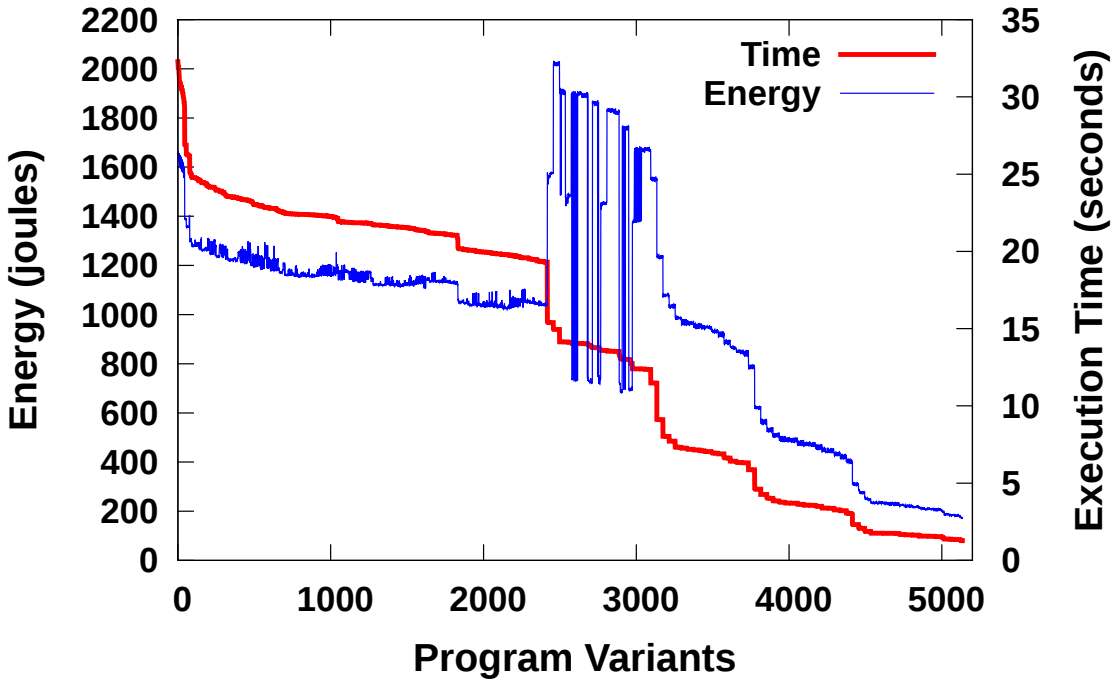


2mm Polybench



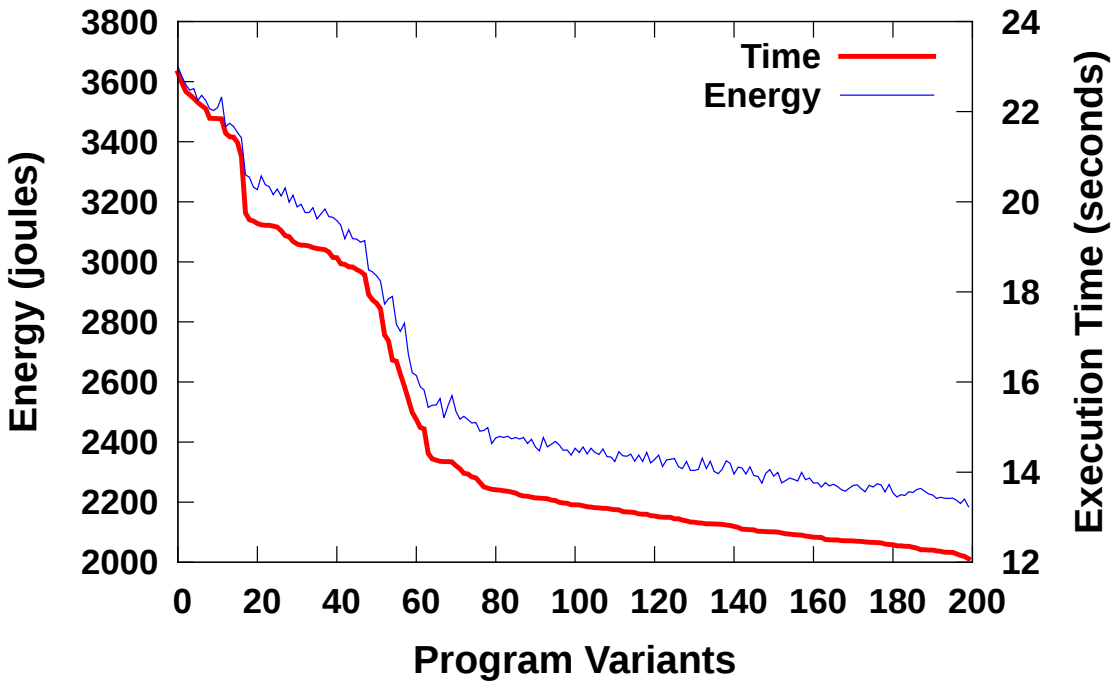
Loop fusion (maxfuse) reduce execution time but increases energy consumption (spikes and the tail in *Covariance* benchmark). Bad tiling configuration increases energy consumption (spikes in *2mm* benchmark). Best optimizations for time are best for energy savings for these two polybench application.

Energy Consumption and Execution Time Correlation (Polybench Stencil Seidel2D Program)



For the stencil program, the correlation between the execution time and the energy consumption is also observed. Jumps in energy usage (and decreased execution time) are results of turning parallelization on.

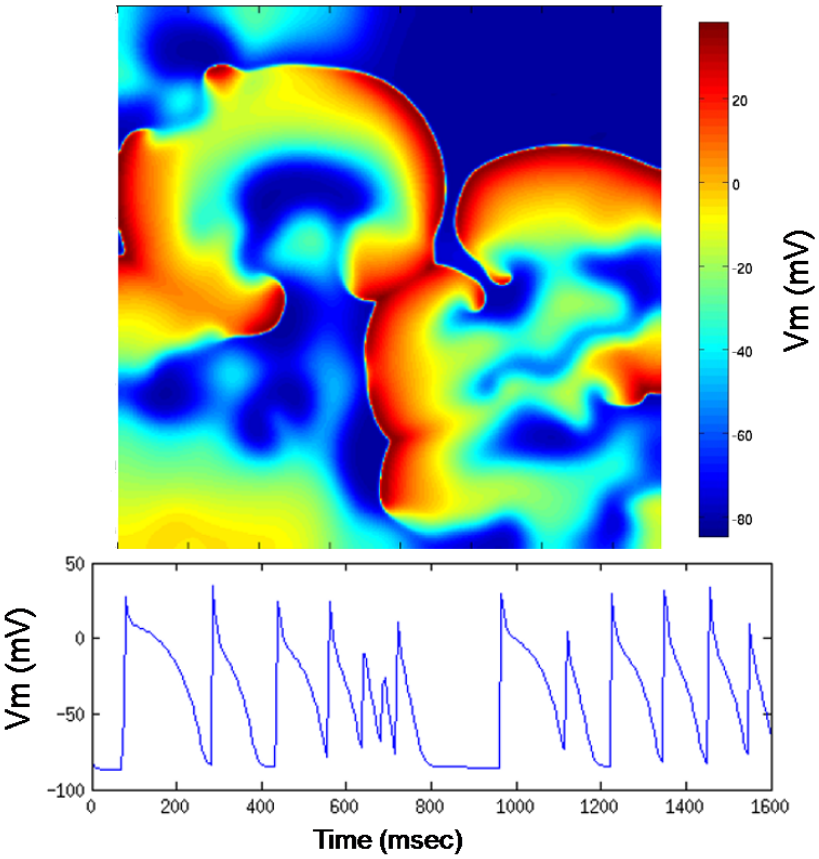
Energy Consumption and Execution Time Correlation (LULESH)



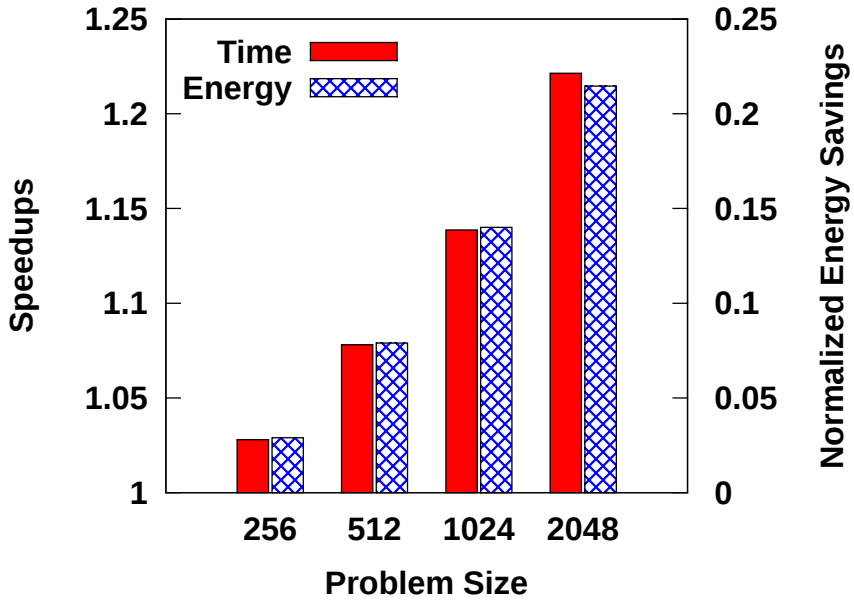
As a larger application, LULESH also displays the similar correlation between energy and time. The best optimized program for time is also for energy. (Note: the graph is from optimizing one loop nest).

Effectiveness of Polyhedral Optimizations on a Realistic Application

2D Cardiac Wave Propagation Simulation

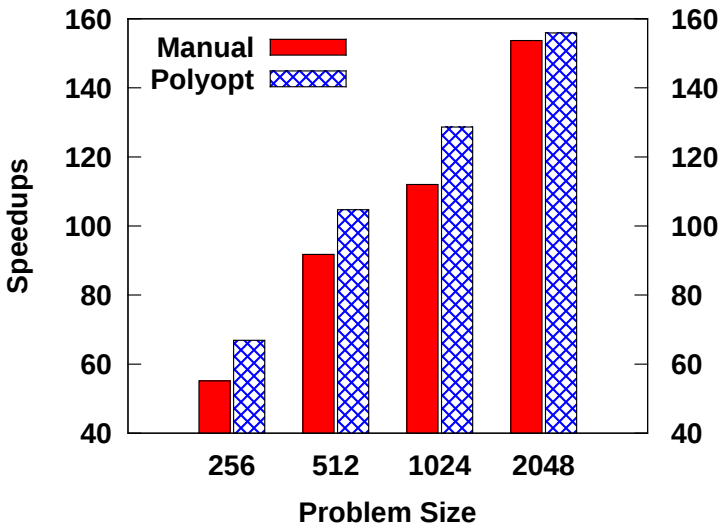


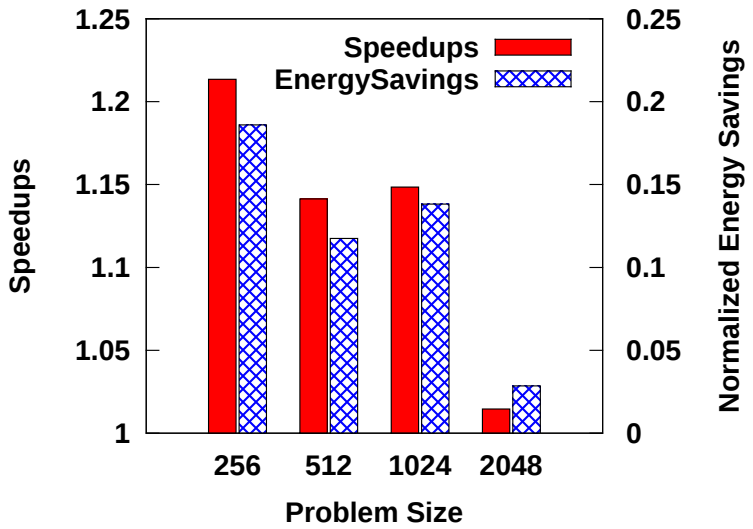
Speedup obtained on a Sandy Bridge system.




```
[PolyOpt] Optimizing Scop #15
[PoCC] Running Pluto
[Pluto] Number of statements: 42
[Pluto] Total number of loops: 85
[Pluto] Number of deps: 164
[Pluto] Maximum domain dimensionality: 3
[Pluto] Number of parameters: 2
```

Results on MIC for Cardiac Simulation



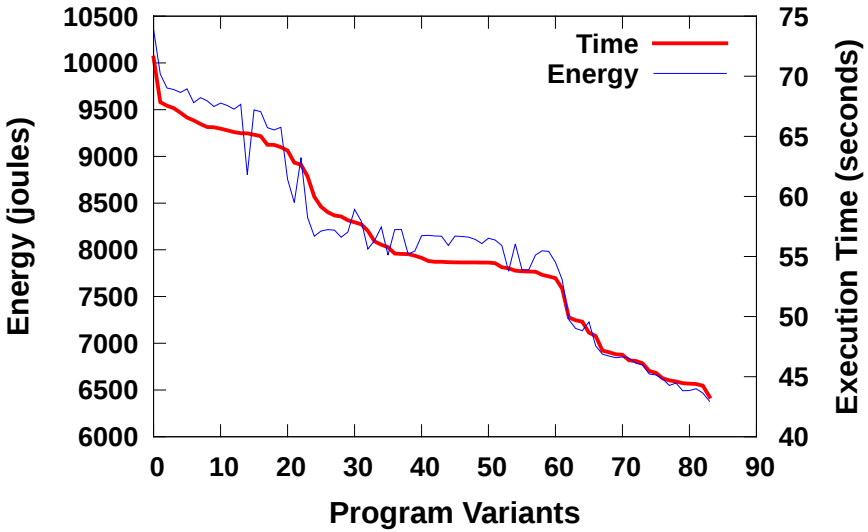


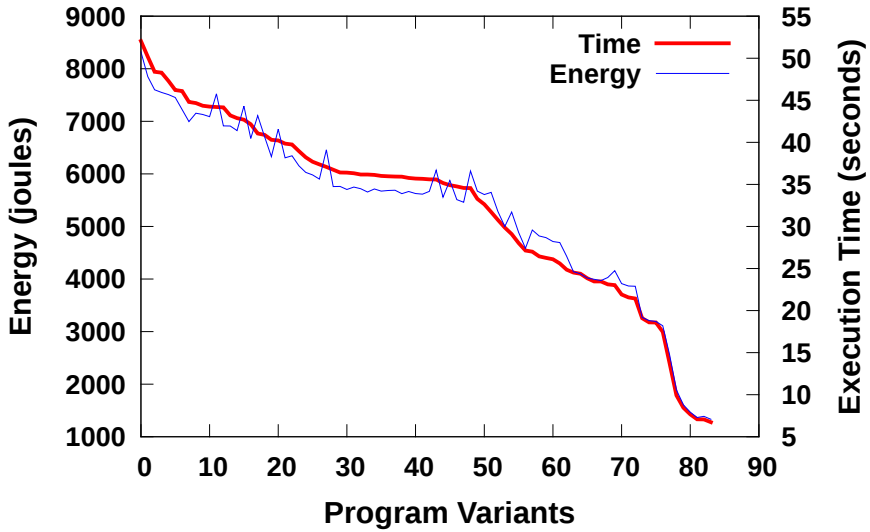
Left: The best optimized PolyOpt program variant vs manual OpenMP (over sequential baseline).

Right: Speedups and energy savings comparing the manual OpenMP with the best PolyOpt program variant.

Conclusion: Polyhedral Approach is effective in optimizing the 2D Cardiac Wave Propagation Simulation.

Energy Consumption and Execution Time Correlation (2D Cardiac Wave Propagation Simulation)





Left: Time and energy correlation on Sandy Bridge

Right: Time and energy correlation on MIC

Conclusion: Energy tracks the time. Saving energy consumption is consistent with improving performance on both processors

Challenges/Limitations using Polyhedral Compilers

- Exposing SCoPs of the application
  LULESH contains six large regions that are potential SCoPs.
- Temporary (array/scalar) variables
  Large number of dependences between statements in a SCoP.
  In LULESH, a human-readable SCoP can easily contain thousands of dependences.
- Temporary variables elimination
  Resulting code is not human-readable and may reduce optimization effectiveness.

Polyhedral Transformable LULESH Code :(

That is part of ONE statement!

```
  dxx(WW) = (1.0)/( (8.) * ( (.125 * ( ((m_x[i+1][j+1][k+1] - (0.5)*dt * m_xd[i+1][j+1][k+1])-(m_x[i][j][k] - (0.5)*dt * m_xd[i][j][k])) - ((m_x
[i+1][j][k+1] - (0.5)*dt * m_xd[i+1][j][k+1])-(m_x[i][j+1][k] - (0.5)*dt * m_xd[i][j+1][k])) + ((m_x[i+1][j+1][k] - (0.5)*dt * m_xd[i+1][j+1][k]
)-(m_x[i][j][k+1] - (0.5)*dt * m_xd[i][j][k+1])) - ((m_x[i+1][j][k] - (0.5)*dt * m_xd[i+1][j][k])-(m_x[i][j+1][k+1] - (0.5)*dt * m_xd[i][j+1][k+
1])) ) ) * (- ((.125 * ( ((m_y[i+1][j+1][k+1] - (0.5)*dt * m_yd[i+1][j+1][k+1])-(m_y[i][j][k] - (0.5)*dt * m_yd[i][j][k])) + ((m_y[i+1][j][k+1]
 - (0.5)*dt * m_yd[i+1][j][k+1])-(m_y[i][j+1][k] - (0.5)*dt * m_yd[i][j+1][k])) - ((m_y[i+1][j+1][k] - (0.5)*dt * m_yd[i+1][j+1][k])-(m_y[i][j][k
+1] - (0.5)*dt * m_yd[i][j][k+1])) - ((m_y[i+1][j][k] - (0.5)*dt * m_yd[i+1][j][k])-(m_y[i][j+1][k+1] - (0.5)*dt * m_yd[i][j+1][k+1])) ) ) * (.1
25 * ( ((m_z[i+1][j+1][k+1] - (0.5)*dt * m_zd[i+1][j+1][k+1])-(m_z[i][j][k] - (0.5)*dt * m_zd[i][j][k])) + ((m_z[i+1][j][k+1] - (0.5)*dt * m_zd[
i+1][j][k+1])-(m_z[i][j+1][k] - (0.5)*dt * m_zd[i][j+1][k])) + ((m_z[i+1][j+1][k] - (0.5)*dt * m_zd[i+1][j+1][k])-(m_z[i][j][k+1] - (0.5)*dt * m
_zd[i][j][k+1])) + ((m_z[i+1][j][k] - (0.5)*dt * m_zd[i+1][j][k])-(m_z[i][j+1][k+1] - (0.5)*dt * m_zd[i][j+1][k+1])) ) )) + ((.125 * ( ((m_z[i+1
][j+1][k+1] - (0.5)*dt * m_zd[i+1][j+1][k+1])-(m_z[i][j][k] - (0.5)*dt * m_zd[i][j][k])) + ((m_z[i+1][j][k+1] - (0.5)*dt * m_zd[i+1][j][k+1])-(m
_z[i][j+1][k] - (0.5)*dt * m_zd[i][j+1][k])) - ((m_z[i+1][j+1][k] - (0.5)*dt * m_zd[i+1][j+1][k])-(m_z[i][j][k+1] - (0.5)*dt * m_zd[i][j][k+1]))
 - ((m_z[i+1][j][k] - (0.5)*dt * m_zd[i+1][j][k])-(m_z[i][j+1][k+1] - (0.5)*dt * m_zd[i][j+1][k+1])) ) ) * (.125 * ( ((m_y[i+1][j+1][k+1] - (0.5
)*dt * m_yd[i+1][j+1][k+1])-(m_y[i][j][k] - (0.5)*dt * m_yd[i][j][k])) + ((m_y[i+1][j][k+1] - (0.5)*dt * m_yd[i+1][j][k+1])-(m_y[i][j+1][k] - (0
.5)*dt * m_yd[i][j+1][k])) + ((m_y[i+1][j+1][k] - (0.5)*dt * m_yd[i+1][j+1][k])-(m_y[i][j][k+1] - (0.5)*dt * m_yd[i][j][k+1])) + ((m_y[i+1][j][k
] - (0.5)*dt * m_yd[i+1][j][k])-(m_y[i][j+1][k+1] - (0.5)*dt * m_yd[i][j+1][k+1])) ) )) ) + (.125 * ( ((m_y[i+1][j+1][k+1] - (0.5)*dt * m_yd[i+1
][j+1][k+1])-(m_y[i][j][k] - (0.5)*dt * m_yd[i][j][k])) - ((m_y[i+1][j][k+1] - (0.5)*dt * m_yd[i+1][j][k+1])-(m_y[i][j+1][k] - (0.5)*dt * m_yd[i
][j+1][k])) + ((m_y[i+1][j+1][k] - (0.5)*dt * m_yd[i+1][j+1][k])-(m_y[i][j][k+1] - (0.5)*dt * m_yd[i][j][k+1])) - ((m_y[i+1][j][k] - (0.5)*dt *
m_yd[i+1][j][k])-(m_y[i][j+1][k+1] - (0.5)*dt * m_yd[i][j+1][k+1])) ) ) * (  ((.125 * ( ((m_x[i+1][j+1][k+1] - (0.5)*dt * m_xd[i+1][j+1][k+1])-(
m_x[i][j][k] - (0.5)*dt * m_xd[i][j][k])) + ((m_x[i+1][j][k+1] - (0.5)*dt * m_xd[i+1][j][k+1])-(m_x[i][j+1][k] - (0.5)*dt * m_xd[i][j+1][k])) -
((m_x[i+1][j+1][k] - (0.5)*dt * m_xd[i+1][j+1][k])-(m_x[i][j][k+1] - (0.5)*dt * m_xd[i][j][k+1])) - ((m_x[i+1][j][k] - (0.5)*dt * m_xd[i+1][j][k
])-(m_x[i][j+1][k+1] - (0.5)*dt * m_xd[i][j+1][k+1])) ) ) * (.125 * ( ((m_z[i+1][j+1][k+1] - (0.5)*dt * m_zd[i+1][j+1][k+1])-(m_z[i][j][k] - (0.
5)*dt * m_zd[i][j][k])) + ((m_z[i+1][j][k+1] - (0.5)*dt * m_zd[i+1][j][k+1])-(m_z[i][j+1][k] - (0.5)*dt * m_zd[i][j+1][k])) + ((m_z[i+1][j+1][k]
 - (0.5)*dt * m_zd[i+1][j+1][k])-(m_z[i][j][k+1] - (0.5)*dt * m_zd[i][j][k+1])) + ((m_z[i+1][j][k] - (0.5)*dt * m_zd[i+1][j][k])-(m_z[i][j+1][k+
1] - (0.5)*dt * m_zd[i][j+1][k+1])) ) )) - ((.125 * ( ((m_z[i+1][j+1][k+1] - (0.5)*dt * m_zd[i+1][j+1][k+1])-(m_z[i][j][k] - (0.5)*dt * m_zd[i][
j][k])) + ((m_z[i+1][j][k+1] - (0.5)*dt * m_zd[i+1][j][k+1])-(m_z[i][j+1][k] - (0.5)*dt * m_zd[i][j+1][k])) - ((m_z[i+1][j+1][k] - (0.5)*dt * m_
zd[i+1][j+1][k])-(m_z[i][j][k+1] - (0.5)*dt * m_zd[i][j][k+1])) - ((m_z[i+1][j][k] - (0.5)*dt * m_zd[i+1][j][k])-(m_z[i][j+1][k+1] - (0.5)*dt *
m_zd[i][j+1][k+1])) ) ) * (.125 * ( ((m_x[i+1][j+1][k+1] - (0.5)*dt * m_xd[i+1][j+1][k+1])-(m_x[i][j][k] - (0.5)*dt * m_xd[i][j][k])) + ((m_x[i+
1][j][k+1] - (0.5)*dt * m_xd[i+1][j][k+1])-(m_x[i][j+1][k] - (0.5)*dt * m_xd[i][j+1][k])) + ((m_x[i+1][j+1][k] - (0.5)*dt * m_xd[i+1][j+1][k])-(
m_x[i][j][k+1] - (0.5)*dt * m_xd[i][j][k+1])) + ((m_x[i+1][j][k] - (0.5)*dt * m_xd[i+1][j][k])-(m_x[i][j+1][k+1] - (0.5)*dt * m_xd[i][j+1][k+1])
) ) )) ) + (.125 * ( ((m_z[i+1][j+1][k+1] - (0.5)*dt * m_zd[i+1][j+1][k+1])-(m_z[i][j][k] - (0.5)*dt * m_zd[i][j][k])) - ((m_z[i+1][j][k+1] - (0
.5)*dt * m_zd[i+1][j][k+1])-(m_z[i][j+1][k] - (0.5)*dt * m_zd[i][j+1][k])) + ((m_z[i+1][j+1][k] - (0.5)*dt * m_zd[i+1][j+1][k])-(m_z[i][j][k+1]
 - (0.5)*dt * m_zd[i][j][k+1])) - ((m_z[i+1][j][k] - (0.5)*dt * m_zd[i+1][j][k])-(m_z[i][j+1][k+1] - (0.5)*dt * m_zd[i][j+1][k+1])) ) ) * (- ((.1
25 * ( ((m_x[i+1][j+1][k+1] - (0.5)*dt * m_xd[i+1][j+1][k+1])-(m_x[i][j][k] - (0.5)*dt * m_xd[i][j][k])) + ((m_x[i+1][j][k+1] - (0.5)*dt * m_xd[
i+1][j][k+1])-(m_x[i][j+1][k] - (0.5)*dt * m_xd[i][j+1][k])) - ((m_x[i+1][j+1][k] - (0.5)*dt * m_xd[i+1][j+1][k])-(m_x[i][j][k+1] - (0.5)*dt * m
_xd[i][j][k+1])) - ((m_x[i+1][j][k] - (0.5)*dt * m_xd[i+1][j][k])-(m_x[i][j+1][k+1] - (0.5)*dt * m_xd[i][j+1][k+1])) ) ) * (.125 * ( ((m_y[i+1][
j+1][k+1] - (0.5)*dt * m_yd[i+1][j+1][k+1])-(m_y[i][j][k] - (0.5)*dt * m_yd[i][j][k])) + ((m_y[i+1][j][k+1] - (0.5)*dt * m_yd[i+1][j][k+1])-(m_y
[i][j+1][k] - (0.5)*dt * m_yd[i][j+1][k])) + ((m_y[i+1][j+1][k] - (0.5)*dt * m_yd[i+1][j+1][k])-(m_y[i][j][k+1] - (0.5)*dt * m_yd[i][j][k+1])) +
 ((m_y[i+1][j][k] - (0.5)*dt * m_yd[i+1][j][k])-(m_y[i][j+1][k+1] - (0.5)*dt * m_yd[i][j+1][k+1])) ) )) + ((.125 * ( ((m_y[i+1][j+1][k+1] - (0.5
```

Conclusion

- Tuning for time can be used as proxy to tuning for energy
  - Energy/time correlation observed for many benchmarks.
  - Optimizations can increase the power and energy, but variant with minimum execution time also has the lowest energy usage.
- Effectiveness
  - On different architectures, improvements as high as 20% in execution time and a similar amount of reduction in energy (for a realistic application) are obtained using polyhedral approach.

Acknowledgment

- EunJung Park, University of Delaware
- Matthew Kay, The George Washington University
- Louis-Noël Pouchet, UCLA
- Albert Cohen, INRIA
- Riyadh Baghdadi, INRIA
- Sven Verdoolaege, ENS