Simulation-Based and Analytical Models for Energy Use Prediction

June 23, 2016

Hristo Djidjev, Stephan Eidenbenz, Balu Nadiga, EJ Park

Los Alamos National Laboratory (LANL)

LA-UR-16-24437

Table of Contents

- Performance Prediction Toolkit (PPT)
- Estimating Per Instruction Energy Use
- Estimating Instruction Counts of Benchmark Applications
- Conclusions/Future work

Performance Prediction Toolkit (PPT): Codesign Performance Modeling Paradigm

- Model *loop structure* of application codes without numerics
- Modeled Application Processes *interact* with and *wait* for other processes on different (or same) entities (-> Discrete Event Simulation)
- Processes *query* hardware resource model through instruction-level task lists for estimated time or energy use
  - Example: Process asks HW "[300 flops, 577 int_ops, 322 mem_access]"
  - HW model response: "274 micro seconds time, 0.284 Joules energy"
- Processes *advance their local time ("sleep")* to mimic computation or other resource usage not modeled in more detail
- Hardware resource model computes time and energy estimates:
  - "First-principles" parameter value (spec sheet)
    - Clock speed, L2-access cycles, RAM access cycles, . . .
  - "Learned value" from data fitting/ML experiments: ***energy use***

Performance Prediction Toolkit (PPT):
Rapid Prototyping Modeling (Python or Lua): Simple, Modular

Code

Hardware Model Library

- Clusters
  Mustang, Trinity, Cielo, Titan
- Nodes, Cores
  AMD Opteron, KNL, MacPro
- Accelerators
  K20X, K40, K6000, M2090, Pascal
- Interconnect
  Gemini 3D Torus
  MPI, OMP
- File system (Lustre)

Application Simulator Library

Benchmark Apps
- PolyBenchSim
- ParboilSim

Production Apps
- SNAPSim
- SPHSim
- SpecTADSim

Simian – Parallel Discrete Simulation Engine

Data

Learned Time and Energy Functions

Application Instrumentation Data

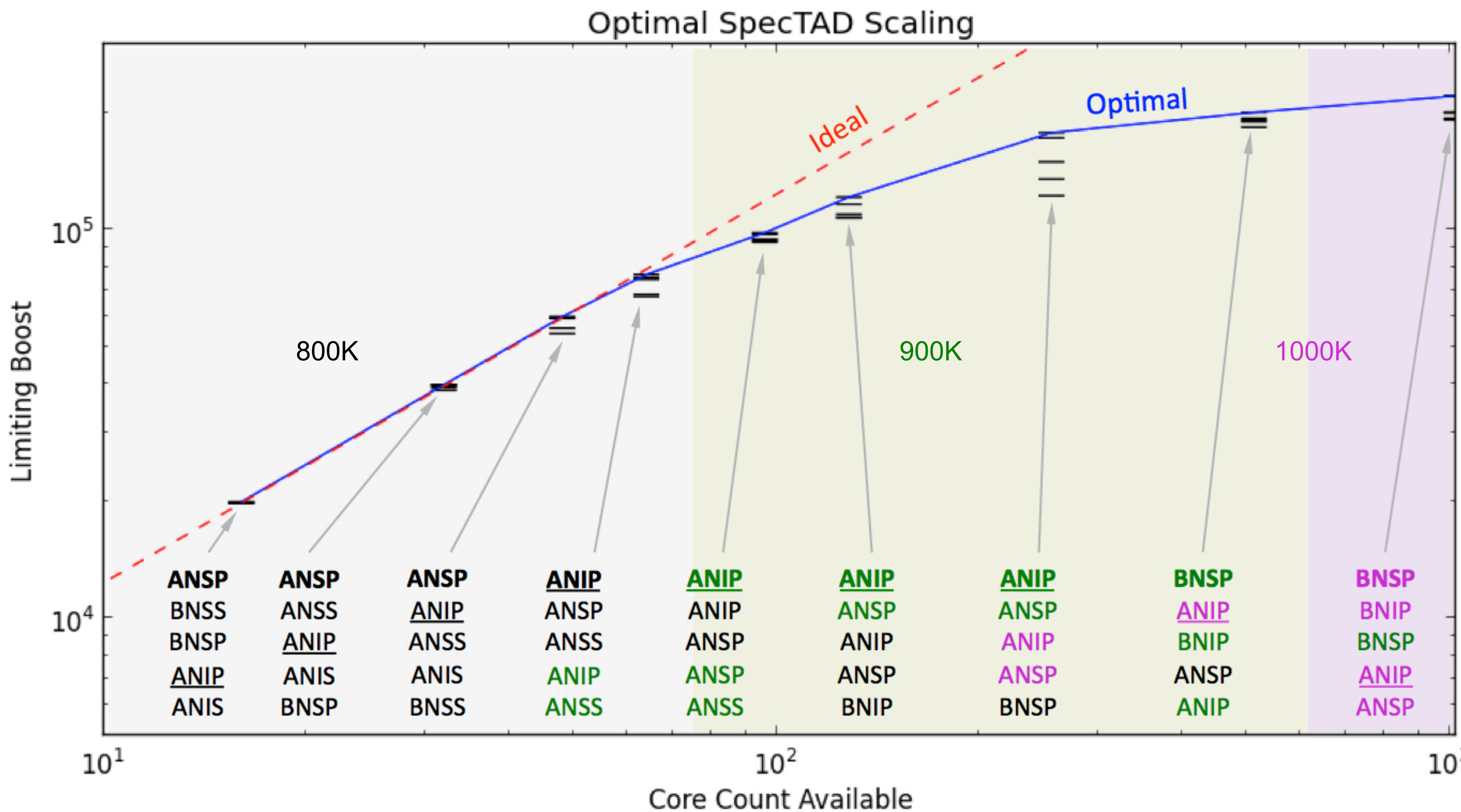
PolyBench, SNAP, SPH, CloverLeaf

Hardware Specs Data

Mustang, Haswell, IvyBridge, SandyBridge, Vortex

SpecTADSim Predicted Performance Results

- Performance Prediction Use Case: Model, Validate, Explore algorithmic design space, Implement most promising variations



Performance Prediction Toolkit (PPT): Previous Results and This Presentation

- Hardware Models
  - nVidia GPUs validation and prediction of next-gen GPU performance [Chapuis et al., ValueTools 2016, Best Paper]
  - MPI/Interconnect models [Ahmed et al., ACM PADS 2016]
  - *Energy use models [Djidjev et al., PMMA 2016]*
- Application Models
  - Accelerated Molecular Dynamics: TADSim [Mniszewski et al., ACM TOMACS 2015], SpecTADSim [Zamora et al., in print, 2016]
  - Deterministic Radiation Transport: SNAPSim [Zerr et al, under review]
  - Smoothed Particle Hydrodynamics: SPHSim [Chapuis et al, WinterSim 2016]
  - *PolyBench Suite Simulator: Analytic energy model embedded in PPT framework [Djidjev et al., PMMA 2016]*

Table of Contents

Energy Use Prediction Models (for an Application)

$$E_{tot} = E_{dyn} + E_{static}$$

Energy modeled as static and dynamic component

$$E_{static} = T_{exec} \times P_{static}$$

Static power measured; execution time predicted by PPT

$$\boxed{E_{dyn} = \sum_i count_{op_i} \times E_{op_i}}$$

Main focus:
How do we estimate energy per operation type?
How do we estimate operation counts in an application?

Instructions or Operations Considered

Name	Operation	Proc	Tool
int_ops	Integer Operations	CPU&GPU	ByFI
flops	Floating Point Operations	CPU&GPU	ByFI
mem_ops	Load and Store Operations	CPU	ByFI
vec_ops	Vector Operations	CPU	ByFI
L2_ops	Level-2 Cache Read/Write	CPU	PAPI
L3_ops	Level-3 Cache Read/Write	CPU	PAPI
dram_ops	DRAM Read/Write	GPU	nvprof
L2_gpu	Level-2 Cache Read/Write	GPU	nvprof
L2_L1	Level-2 to Level-1 Data Transfers	GPU	nvprof
flops_GPU	Floating point operations	GPU	nvprof

Experimental Setup

- Applications: PolyBench Suite
  - 30 linear algebra benchmark codes
  - About 200 different input sizes (across multiple parameters)
- Hardware:
  - Haswell nodes (30 apps, about 6000 cases)
  - Nvidia Tesla K40c (only 11 apps, about 500 cases)
- Measurements/Data collected
  - Timing
  - Energy and power use (using PAPI Rapl and Nvlm)
  - Instruction/Operations counts
    - LLVM-level analysis tool ByFI for hardware independent counters
    - PAPI for hardware counters (L2/3)
    - nvprof for GPU

Finding Energy Use per Instruction from an *Optimization* Problem

Unknown

$$E_{tot}(j) = \sum_i count_{op_i}(j) \times E_{op_i} + T_{exec}(j) P_{stat}$$

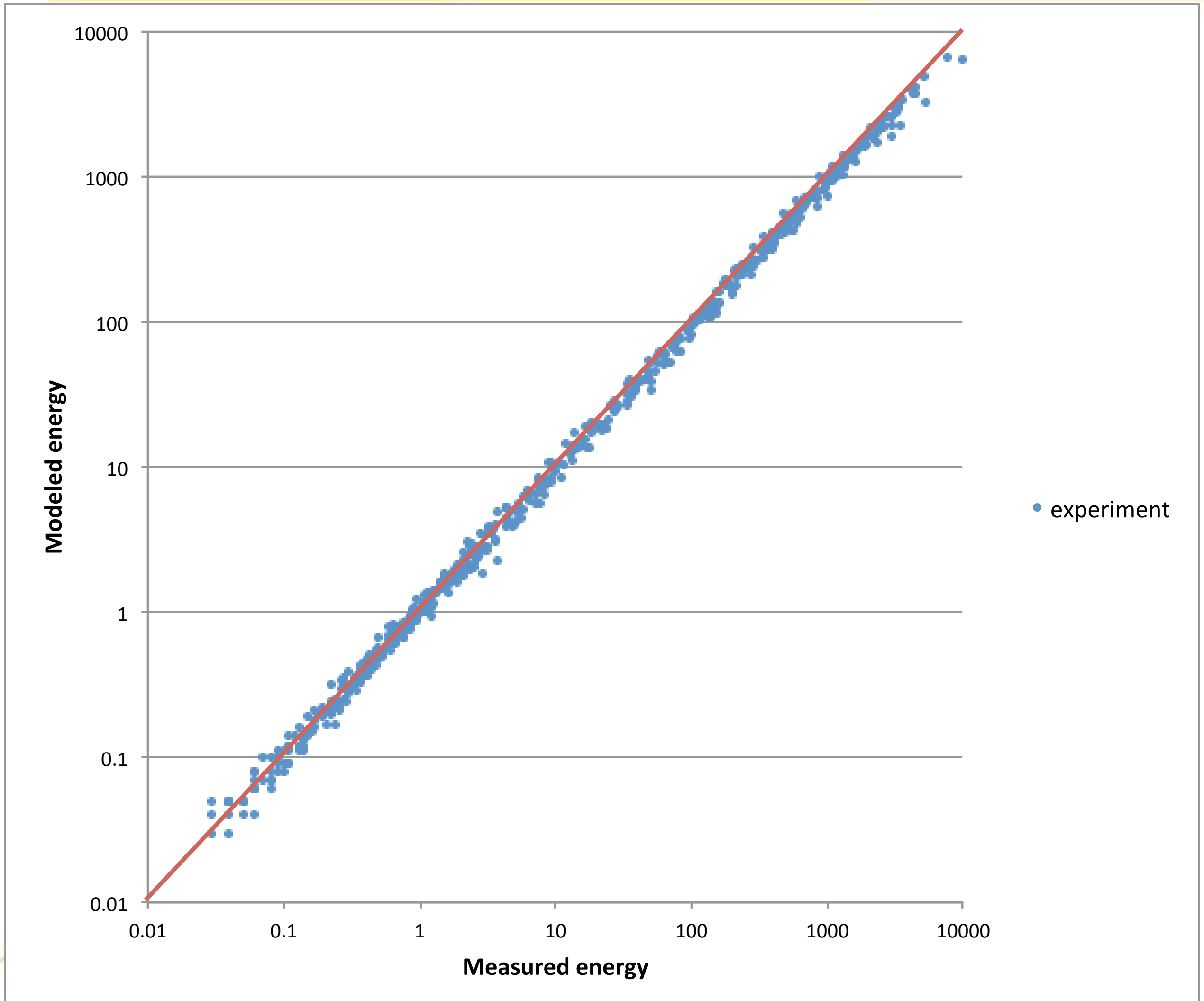
Experiment index

$$Err_{\mathrm{mod}} = \sqrt{\frac{1}{n} \sum_{j<n} \left(\frac{E_{tot}^{j}}{E_{meas}^{j}} - 1 \right)^2}$$

$$\boxed{opt_{E_{op_1},\ldots,E_{op_k},P_{stat}} = Err_{\mathrm{mod}}\left(E_{op_1},\ldots,E_{op_k},P_{Stat}\right)}$$

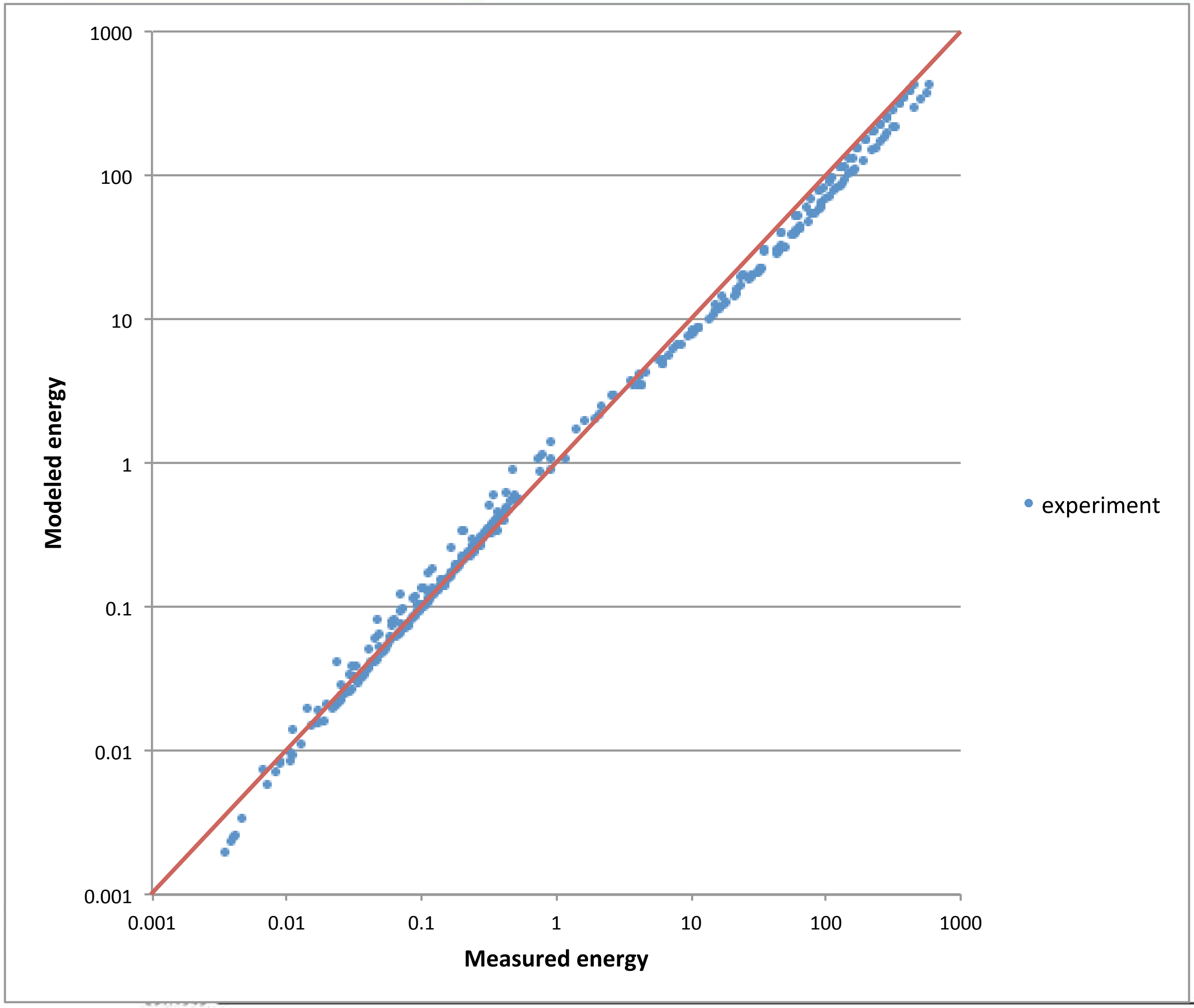
- Solve optimization function subject to equations of E_{tot}
- Minimize, find values using standard solvers (ampl, gurobi)

Haswell Results for Energy Use Per Instruction



- Training data: 10% of 6,000 cases, 90% hold-out set
- Red line is perfect fit
- Visually good fit (caveat: log scales)
- Error: 13%

GPU Results for Energy Use Per Instruction



- Training data: 10% of 500 cases, 90% hold-out set
- Red line is perfect fit
- Visually good fit (caveat: log scales)
- Error: 20%

Results: Energy Use per Instruction

Name	ByFI & PAPI	PAPI	nvprof
int_ops	2.51 e -9 J	2.55 e -9 J	0
flops	1.06 e -8 J	1.08 e -8 J	0
mem_ops	8.57 e -10 J	8.44 e -10 J	-
vec_ops	1.53 e -15 J	5.47 e -15 J	-
L2_ops	-	2.15 e -8 J	-
L3_ops	-	6.65 e -8 J	-
L2_gpu	-	-	3.54 e -9 J
L2_L1	-	-	4.24 e -10 J
flops_GPU	-		3.15 e -11 J
stat_power	17.26 W	16.47 W	57.29 W

Results: Energy Use per Instruction

Name	ByFI & PAPI	PAPI	nvprof
int_ops	2.51 e -9 J	2.55 e -9 J	0
flops	1.06 e -8 J	1.08 e -8 J	0
mem_ops	8.57 e -10 J	8.44 e -10 J	-
vec_ops	1.53 e -15 J	5.47 e -15 J	-
L2_ops	-	2.15 e -8 J	-
L3_ops	-	6.65 e -8 J	-
L2_gpu	-	-	3.54 e -9 J
L2_L1	-	-	4.24 e -10 J
flops_GPU	-		3.15 e -11 J
stat_power	17.26 W	16.47 W	57.29 W

counter-intuitive

- Why? Unwanted correlations, currently investigating in more detail
- A case of the (machine learning) blackbox syndrom: predicts well, hard to believe insights

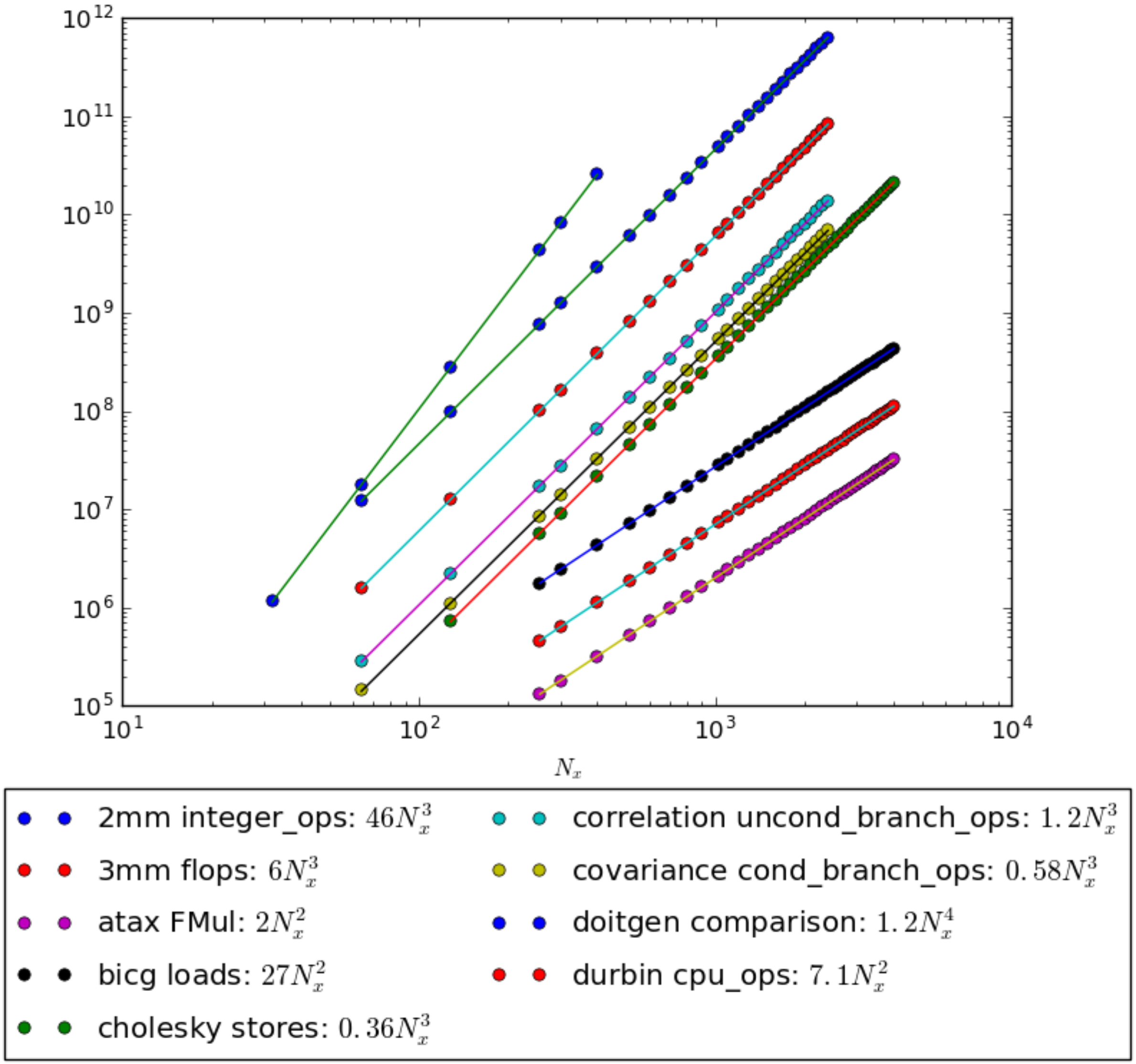
Table of Contents

- Performance Prediction Toolkit (PPT)
- Estimating Per Instruction Energy Use
- Estimating Instruction Counts of Benchmark Applications
- Conclusions/Future work

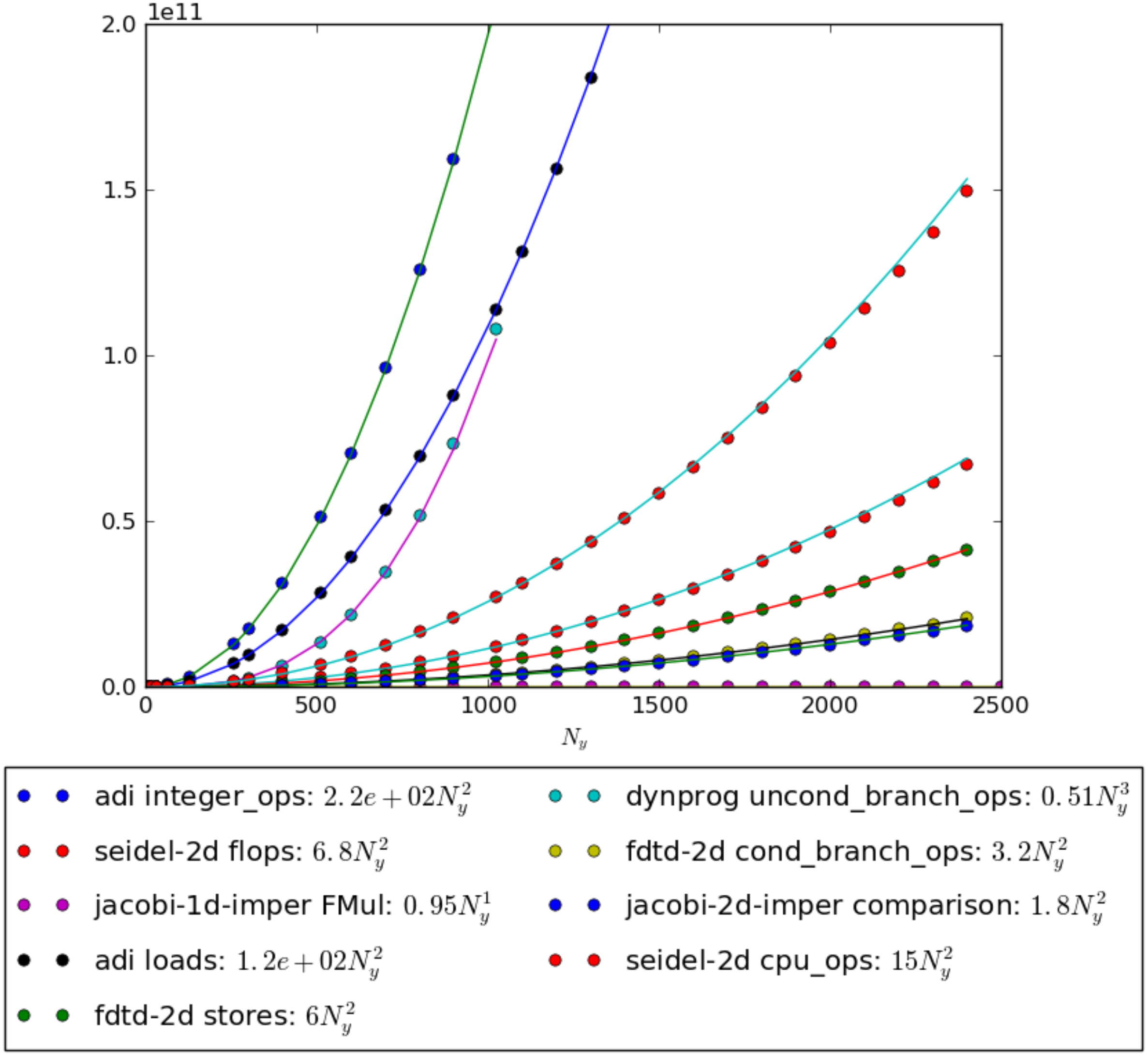
Estimating Instruction Counts

- Same data as before: PolyBench Suite
  - Instruction/Operations counts for 6,000 cases
  - Also: input size parameters: N_x, N_y, N_z (typically matrix dimensions)
- Idea: Predict $count_{opi}$ per PolyBench application *k* by learning/fitting a function f^k(input size)
- For all operations (except L2/L3): practically perfect fit
- L2 and L3 hit rates: 17% and 66% error rate (!)
- Combine with per-instruction energy weights to get analytical formulas for energy use of PolyBench instances

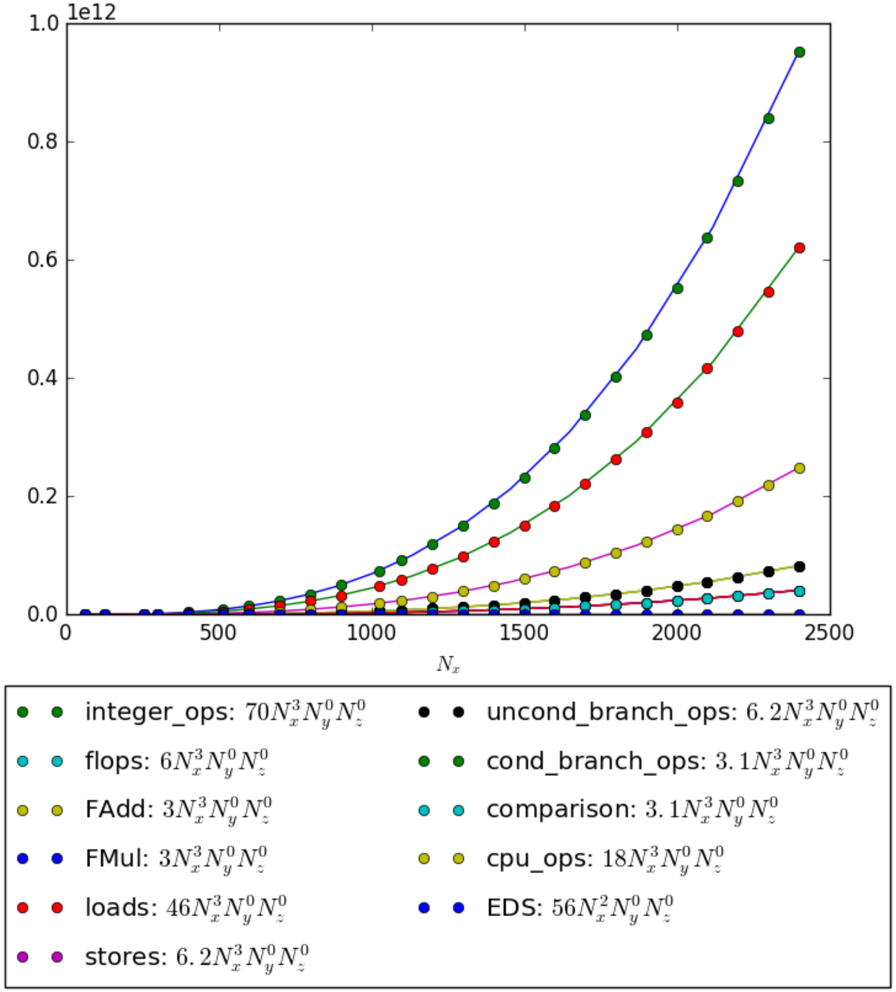
Instruction Count Parameter Prediction Results for Different PolyBench Algorithms (1/2)

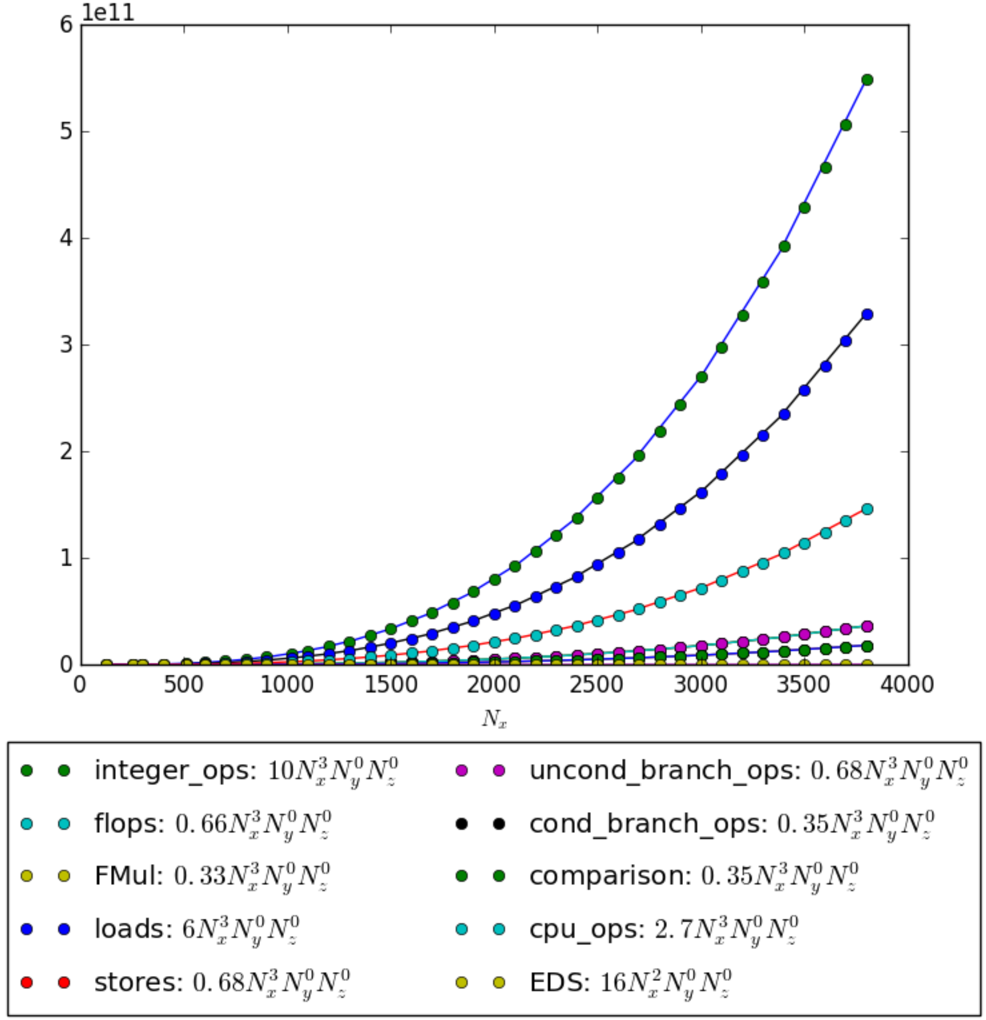


Instruction Count Parameter Prediction Results for Different PolyBench Algorithms (2/2)

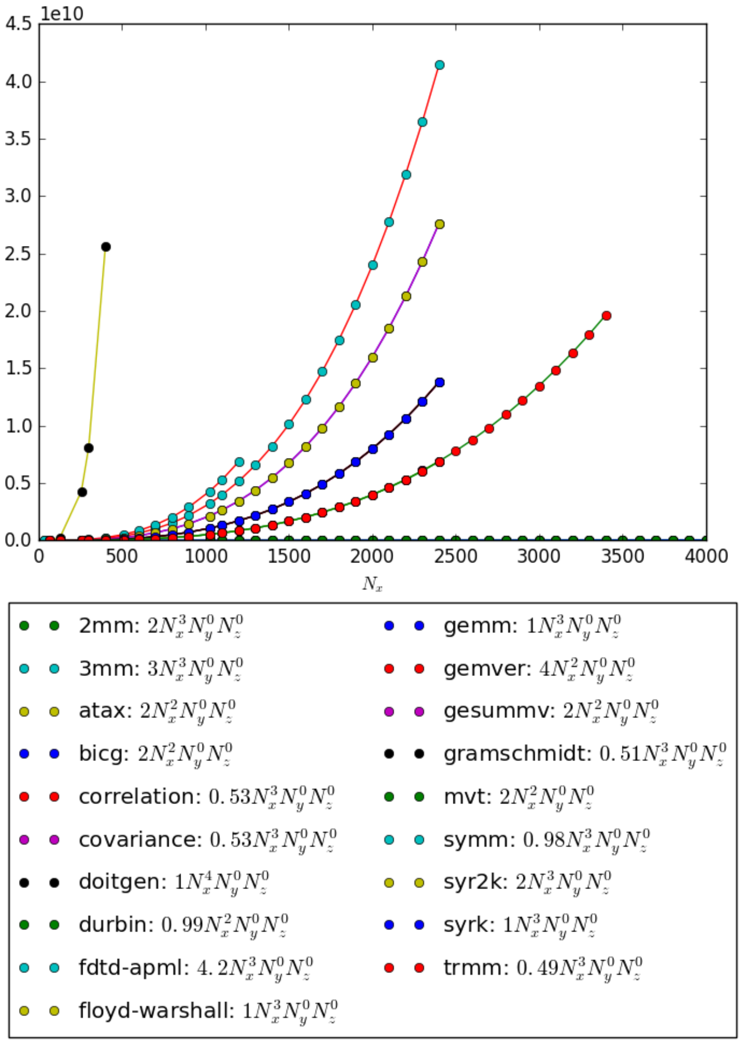


Count Prediction Results: 3MM and LU





Count Prediction Across PolyBench Apps: Fadd and loads



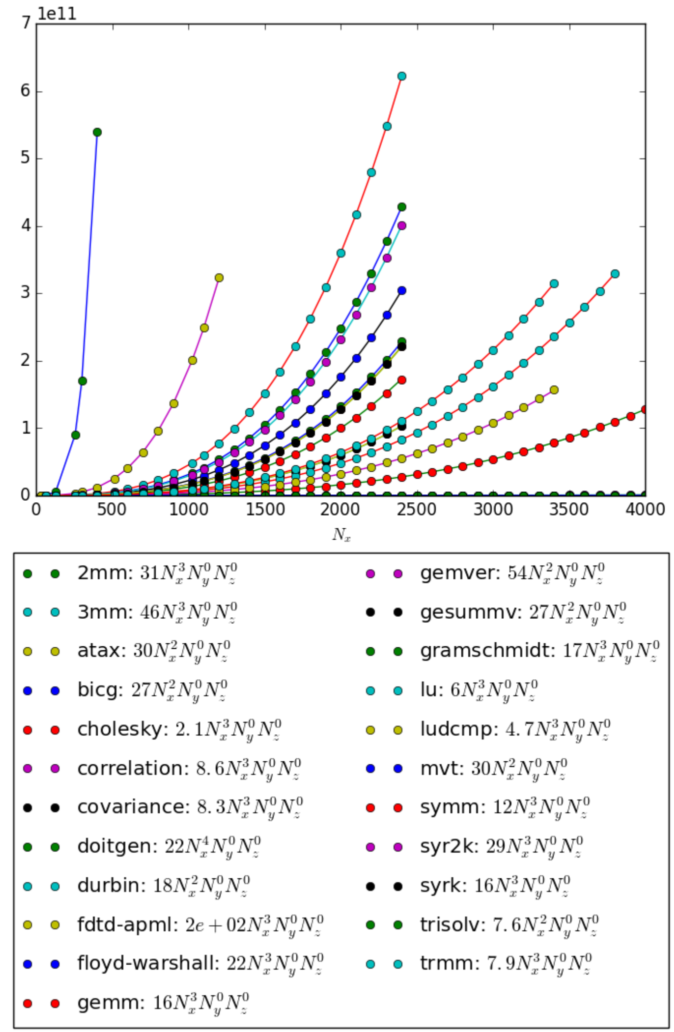


Table of Contents

Conclusions

- Estimating instruction-level energy use weights in context of applications (not micro-benchmarks)
- Energy use predictions in Performance Prediction Toolkit (PPT) in addition to running time predictions
- Initial regression approach results in non-intuitive energy weights, likely due to correlations; good performance in terms of error minimization
- Combination of data analysis and discrete event modeling