

An Ecosystem for the New HPC: Heterogeneous Parallel Computing

Wu FENG

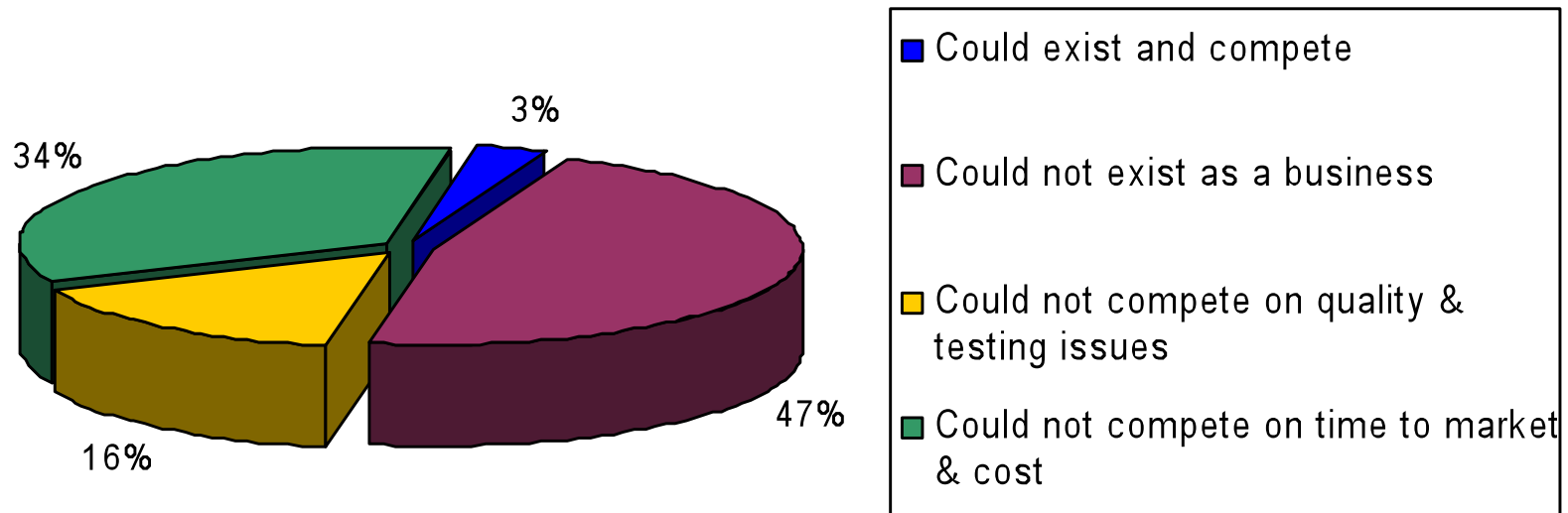
Dept. of Computer Science and Dept. of Electrical & Computer Engineering
Virginia Bioinformatics Institute

Japanese 'Computnik' Earth Simulator Shatters U.S. Supercomputer Hegemony

Tokyo 20 April 2002 *The Japanese Earth Simulator is on-line and producing results that alarm the USA, that considered itself as being leading in supercomputing technology. With over 35 Tflop/s, it five times outperforms the Ascii White supercomputer that is leading the current TOP500 list. No doubt that position is for the Earth Simulator, not only for the next list, but probably even for*

Importance of High-Performance Computing (HPC)

Competitive Risk From Not Having Access to HPC




Data from Council of Competitiveness.
Sponsored Survey Conducted by IDC


China Wrests Supercomputer Title From U.S.

By ASHLEE VANCE

Published: October 28, 2010

A Chinese scientific research center has built the fastest supercomputer ever made, replacing the United States as maker of the swiftest machine,

 RECOMMEND

 TWITTER

 LINKEDIN

Computnik 2.0?

Tianhe-1A: Computnik Revisited?



- The Second Coming of Computnik? Computnik 2.0?
 - No ... “only” 43% faster than the previous #1 supercomputer, *but*
 - \$20M cheaper than the previous #1 supercomputer
 - 42% less power consumption
- The Second Coming of the “Beowulf Cluster” for HPC
 - The further commoditization of HPC

Vision

- **Commoditizing *personal supercomputing* for the masses**



Tianhe-1

2.5×10^{15} floating-pt ops per sec
= 2.5 petaflops (Linpack benchmark)

iPad2: 1.5×10^9 flops = 1.5 gigaflops
 $1.5 \text{ gigaflops/iPad2} * 9.3\text{M iPad2 (Q2 2011)}$
= 14 petaflops (Linpack benchmark extrapolated)

- **A software ecosystem**
 - ... for supporting heterogeneous parallel computing
 - ... by exploiting intra-node parallelism
 - ... to commoditize personal supercomputing for the masses

CPU Core Counts ...

- Doubling every 18-24 months
 - 2006: 2 cores
 - Examples: AMD Athlon 64 X2, Intel Core Duo
 - 2010: 8-12 cores
 - Examples: AMD Magny Cours, Intel Nehalem EX
- Penetrating all markets ...
 - Desktops
 - Laptops: Most in this room are multicore
 - Tablets: Apple iPad 2, HP TX1000, Sony S2
 - Cell Phones: LG Optimus 2X, Motorola Droid X2

A world of ubiquitous parallelism ...

... how to extract performance ... and then scale out

Paying For Performance

- “The free lunch is over..” †
 - Programmers can no longer expect substantial increases in single-threaded performance.
 - The burden falls on developers to exploit parallel hardware for performance gains.
- How do we lower the cost of concurrency?

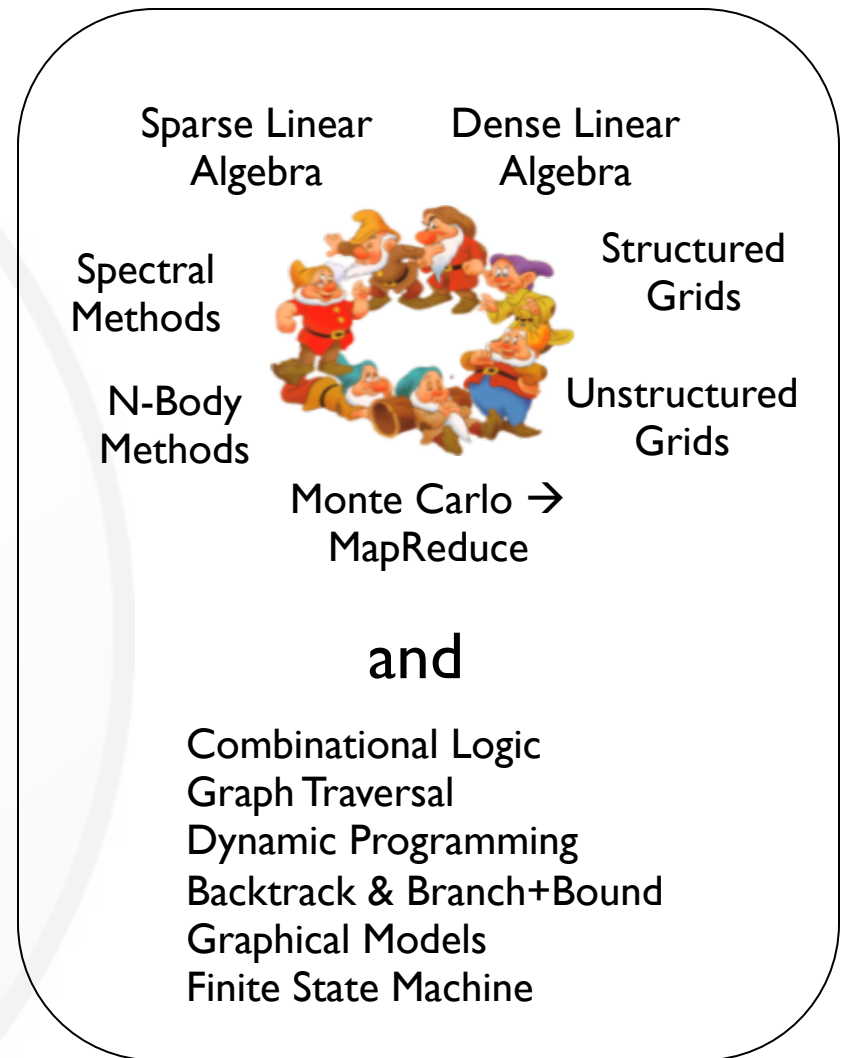
† H. Sutter, “The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software,” *Dr. Dobbs’s Journal*, 30(3), March 2005. (Updated August 2009.)

The Berkeley View †

- Traditional Approach
 - Applications that target existing hardware and programming models
- Berkeley Approach
 - Hardware design that keeps future applications in mind
 - Basis for future applications?
13 computational dwarfs

A computational dwarf is a pattern of communication & computation that is common across a set of applications.

† Asanovic, K., et al. *The Landscape of Parallel Computing Research: A View from Berkeley*. Tech. Rep. UCB/EECS-2006-183, University of California, Berkeley, Dec. 2006.



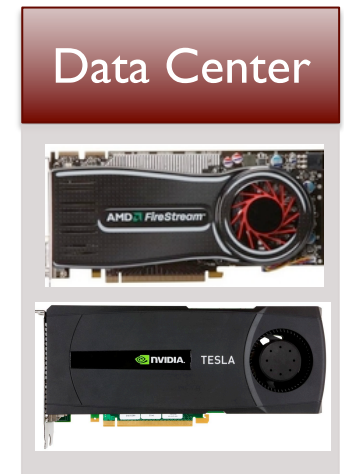
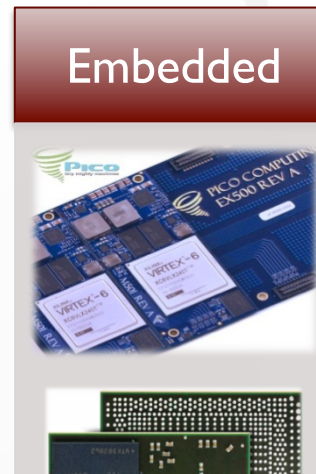
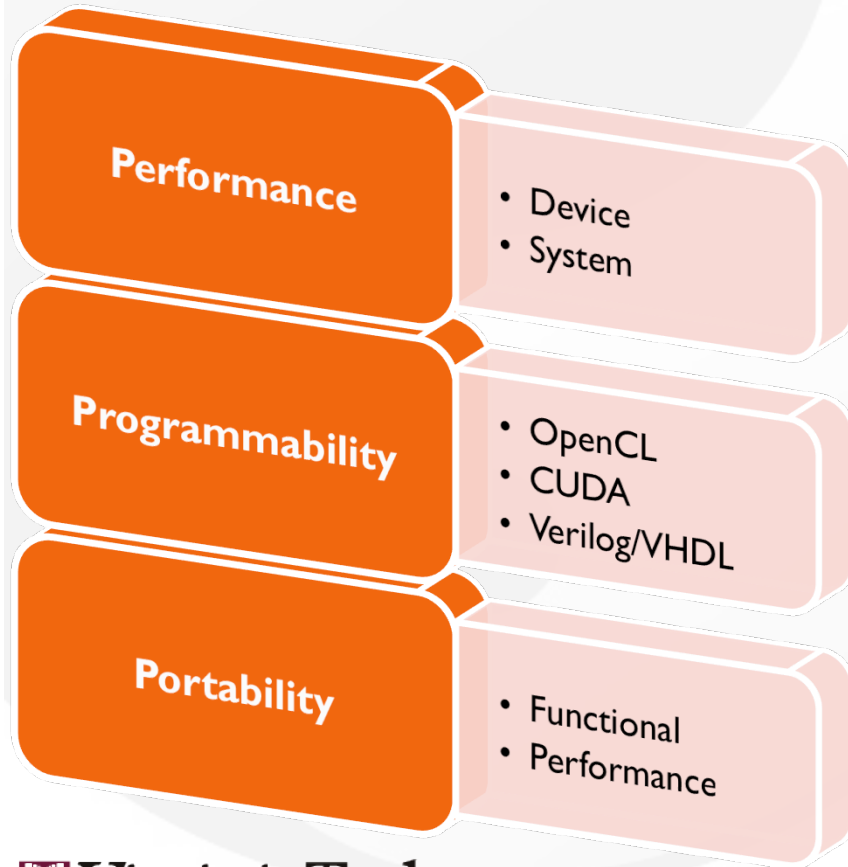
Project Goal

An Ecosystem for Heterogeneous Parallel Computing

- Deliver personalized supercomputing to the masses
 - Heterogeneity of hardware devices for a “cluster on a chip” plus ...
 - Enabling software that tunes the parameters of the hardware devices with respect to performance, power, and programmabilityvia a benchmark suite of computational dwarfs and apps

Multi-Dimensional Understanding of 3 P's

- A multi-dimensional understanding of how to optimize performance, programmability, portability or some combination thereof



Design of Composite Structures

Science & Engineering Principles

Math Algorithms:
Solvers, Optimization,
Model Reduction,
Dynamic Multi-Precision Support

Exascale Simulation Framework

Verification, Validation, and Uncertainty Quantification

Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

Architecture-Aware Optimizations

Performance & Power Models

Run Time

Affinity Cost Models

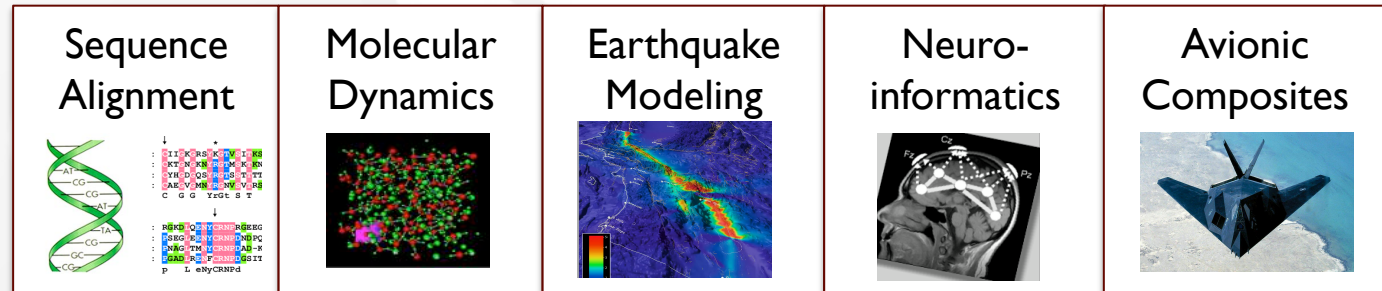
Task Scheduling System

Software Ecosystem

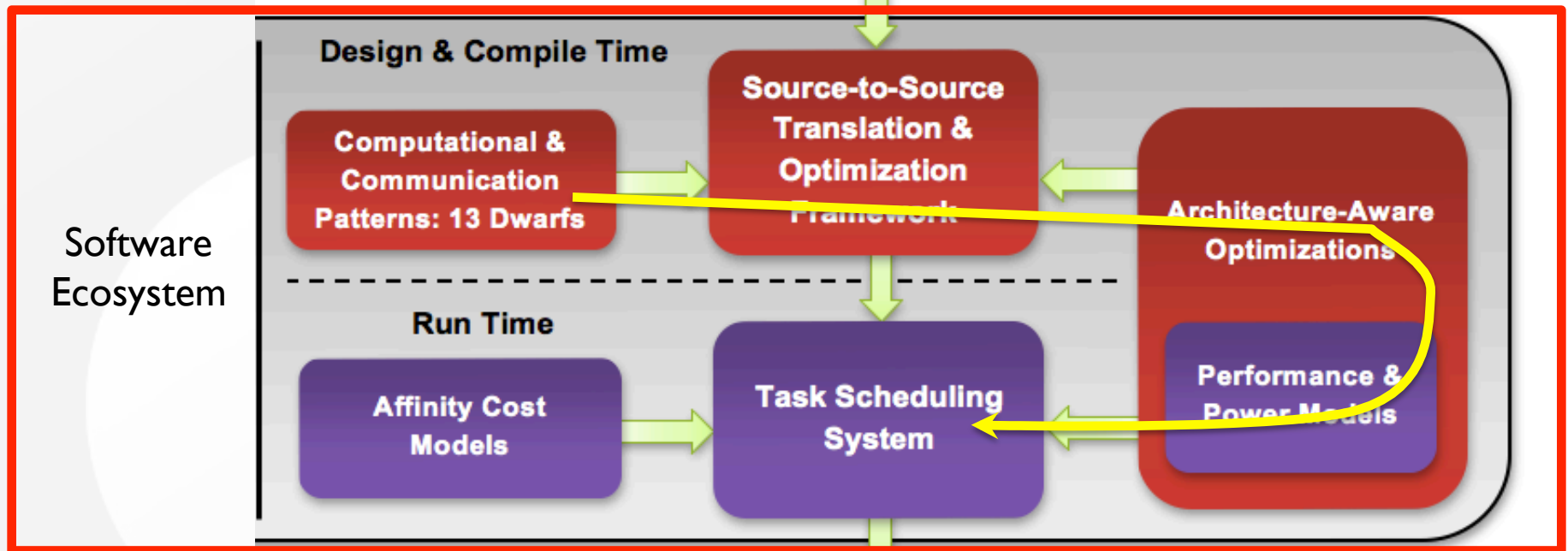
Heterogeneous Parallel Computing (HPC) Platform

An Ecosystem for Heterogeneous Parallel Computing

Applications

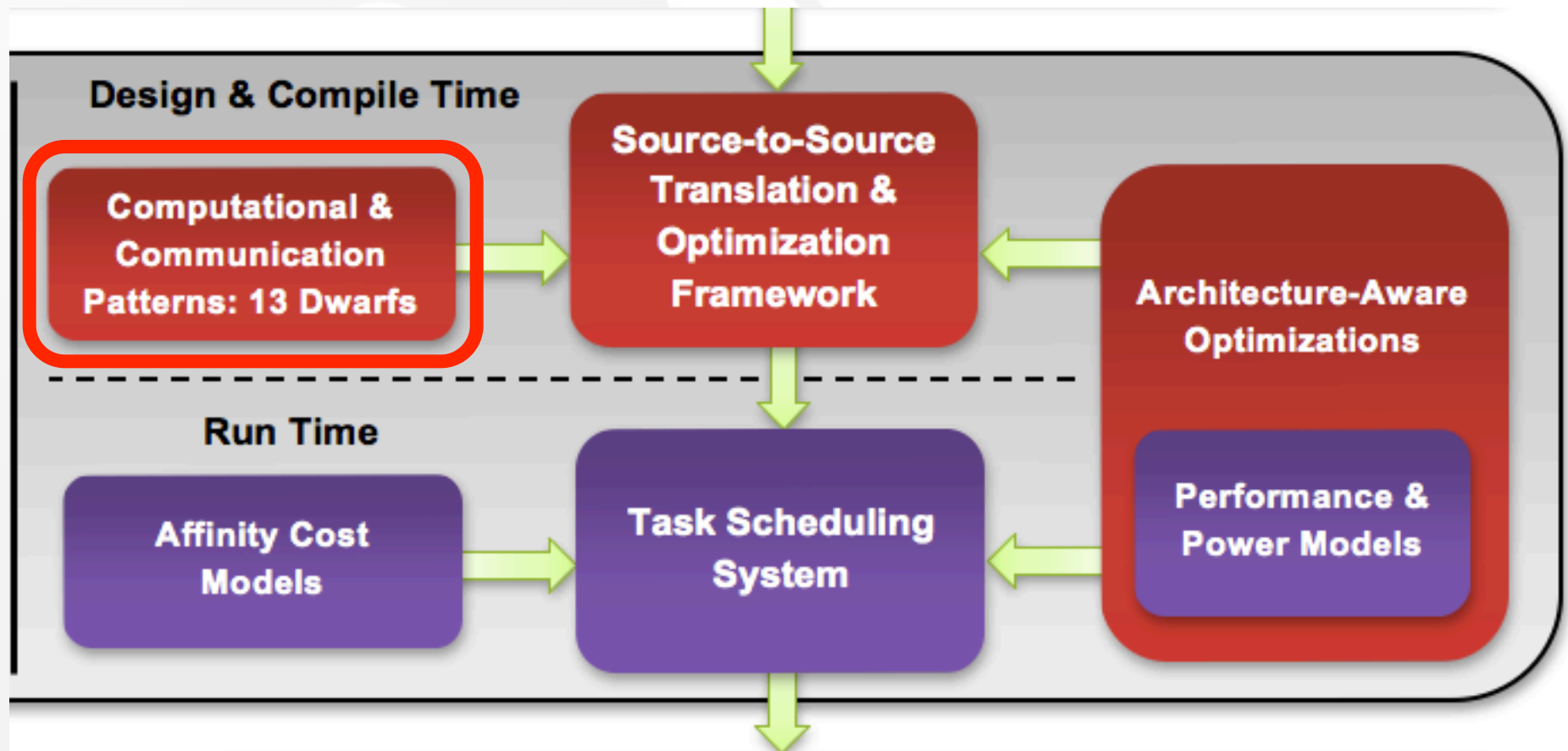


Software Ecosystem



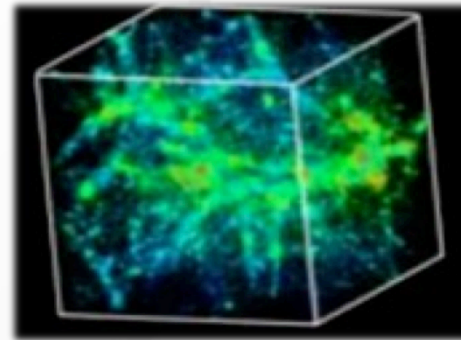
Heterogeneous Parallel Computing (HPC) Platform

Roadmap

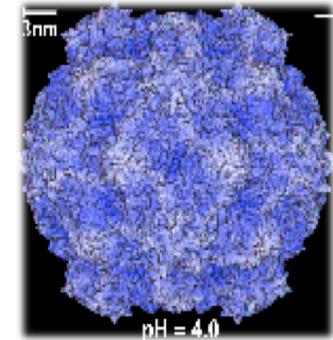


Example of a Computational Dwarf: N-Body

- Computational Dwarf: *Pattern of computation & communication ... that is common across a set of applications*
- N-Body problems are studied in
 - Cosmology, particle physics, biology, and engineering
- All have similar structures
- An N-Body benchmark can provide meaningful insight to people in all these fields
- Optimizations may be generally applicable as well



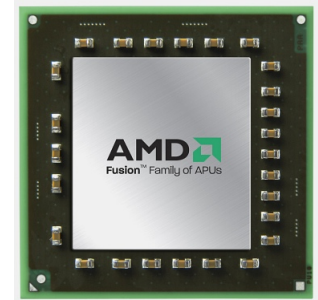
RoadRunner Universe:
Astrophysics



GEM:
Molecular Modeling

First Instantiation: OpenCL & the 13 Dwarfs

- Goal
 - Provide common algorithmic methods, i.e., dwarfs, in a language that is “write once, run anywhere” (CPU, GPU, or even FPGA), i.e., OpenCL



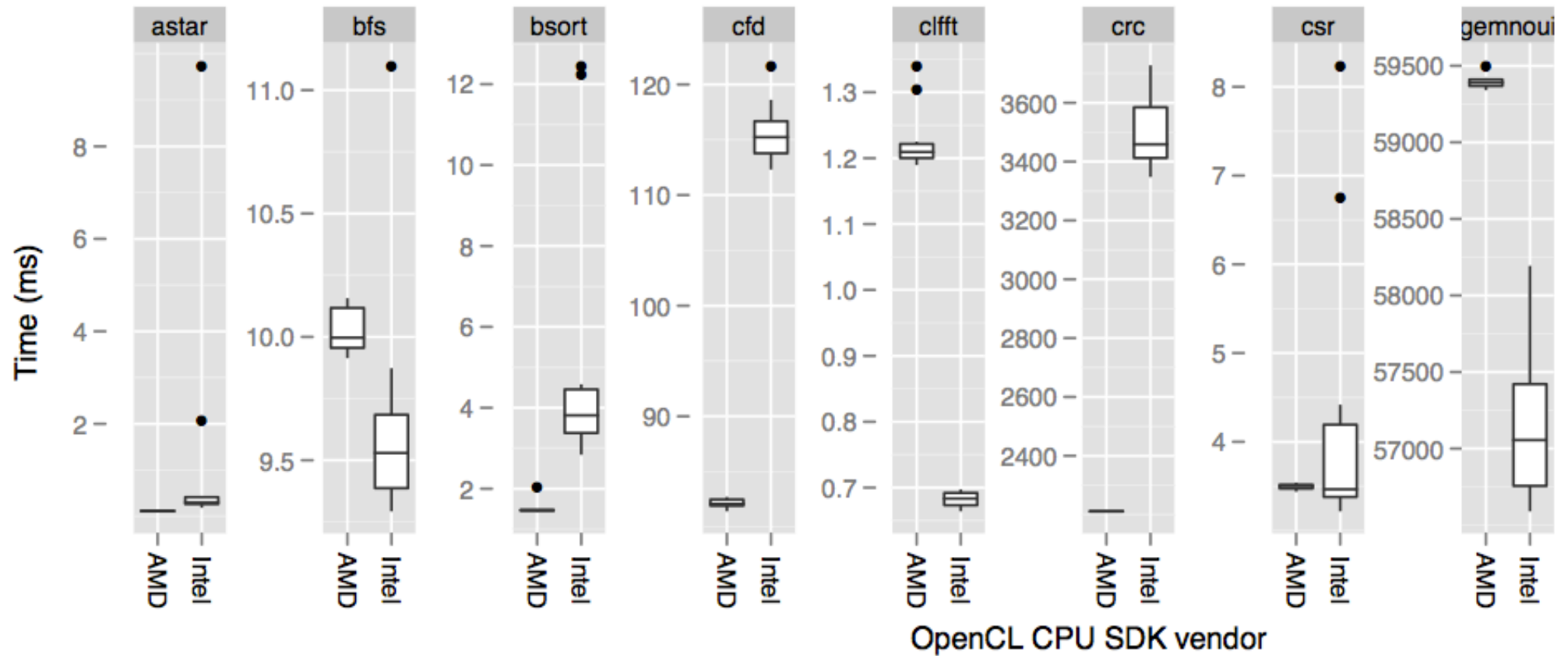
- Part of a larger umbrella project (2008-2012) funded by the NSF Center for High-Performance Reconfigurable Computing (CHREC)



Status of OpenCL and the 13 Dwarfs

Dwarf	Done	In progress
Dense linear algebra	LU Decomposition	
Sparse linear algebra	Matrix Multiplication	
Spectral methods	FFT	
N-Body methods	GEM	
Structured grids	SRAD	
Unstructured grids	CFD Solver	
MapReduce		StreamMR
Combinational logic	CRC	
Graph traversal	Breadth-First Search	
Dynamic programming	Needleman-Wunsch	
Backtrack and Branch-and-Bound		N-Queens, Traveling Salesman
Graphical models	Hidden Markov Model	
Finite state machines	Temporal Data Mining	

Execution Time (ms) of Dwarfs with SDKs



The Future is Fusion

Experimental Set-Up: Machines and Workload

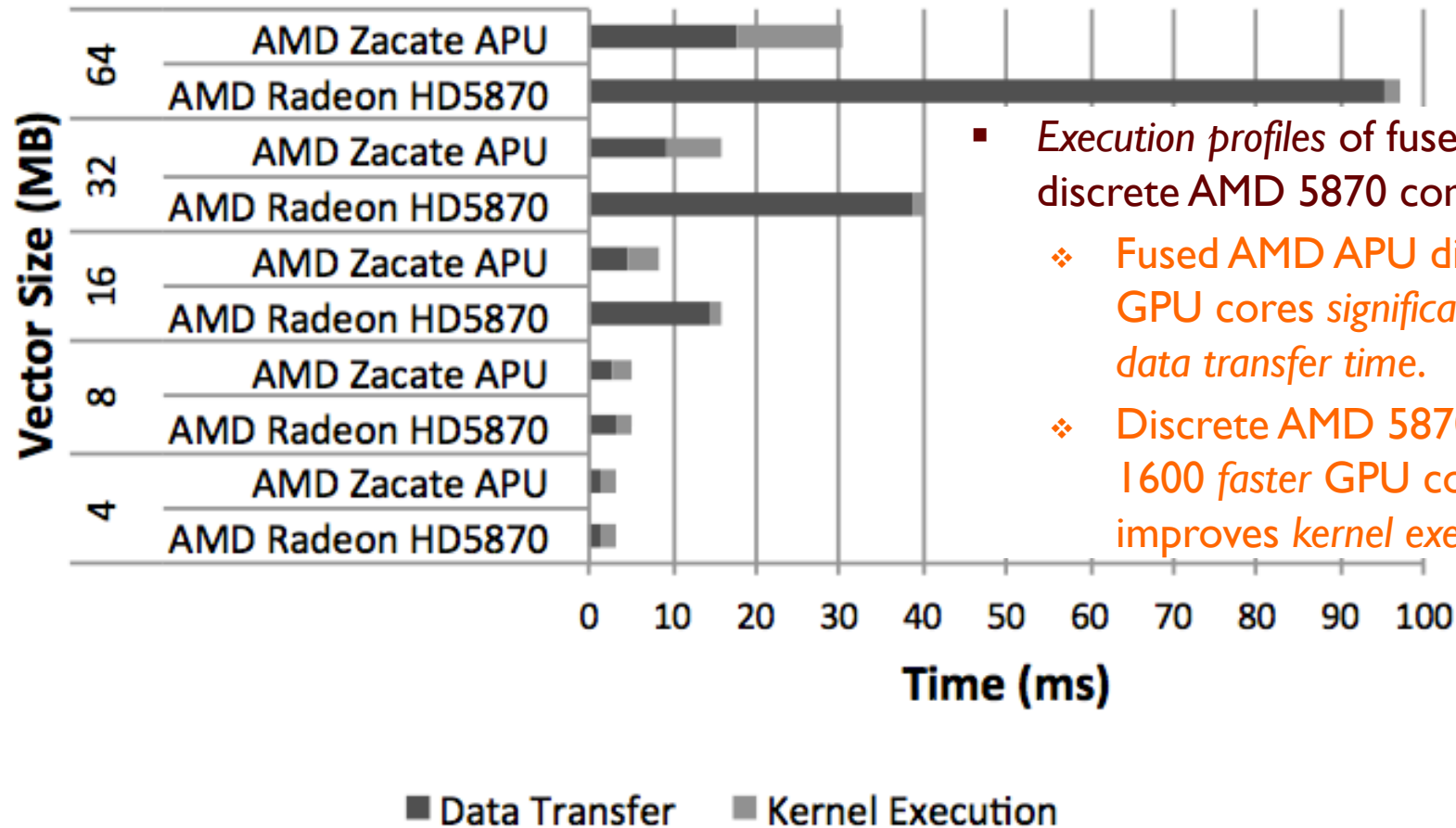
Platform	AMD Zacate APU	AMD Radeon HD 5870	AMD Radeon HD 5450
Stream Processors	80	1600	80
Compute Units	2	20	2
Memory Bus Type	NA	GDDR5	DDR3
Device Memory	192 MB	1024 MB	512 MB
Local Memory	32 KB	32 KB	32 KB
Max. Workgroup Size	256 Threads	256 Threads	128 Threads
Core Clock Frequency	492 MHz	850 MHz	675 MHz
Peak FLOPS	80 GFlops/s	2720 GFlops/s	104 GFlops/s
Host:			
Processor	AMD Engg. Sample @1.6 GHz	Intel Xeon E5405 @2.0 GHz	Intel Celeron 430 @1.8 GHz
System Memory	2 GB (NA)	2 GB DDR2	2 GB DDR2
Cache	L1: 32K, L2: 512K	L1: 32K, L2: 6M	L1: 32K, L2: 512K
Kernel	Ubuntu 2.6.35.22	Ubuntu 2.6.28.19	Ubuntu 2.6.32.24

- OpenCL and the 13 Dwarfs

- Sparse Linear Algebra: SpMV
- N-body: Molecular Modeling
- Spectral: FFT
- Dense Linear Algebra: Scan and Reduce (SHOC @ ORNL)

M. Daga, A. Aji, W. Feng, “On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing,” *Symposium on Application Accelerators in High Performance Computing*, Jul. 2011.

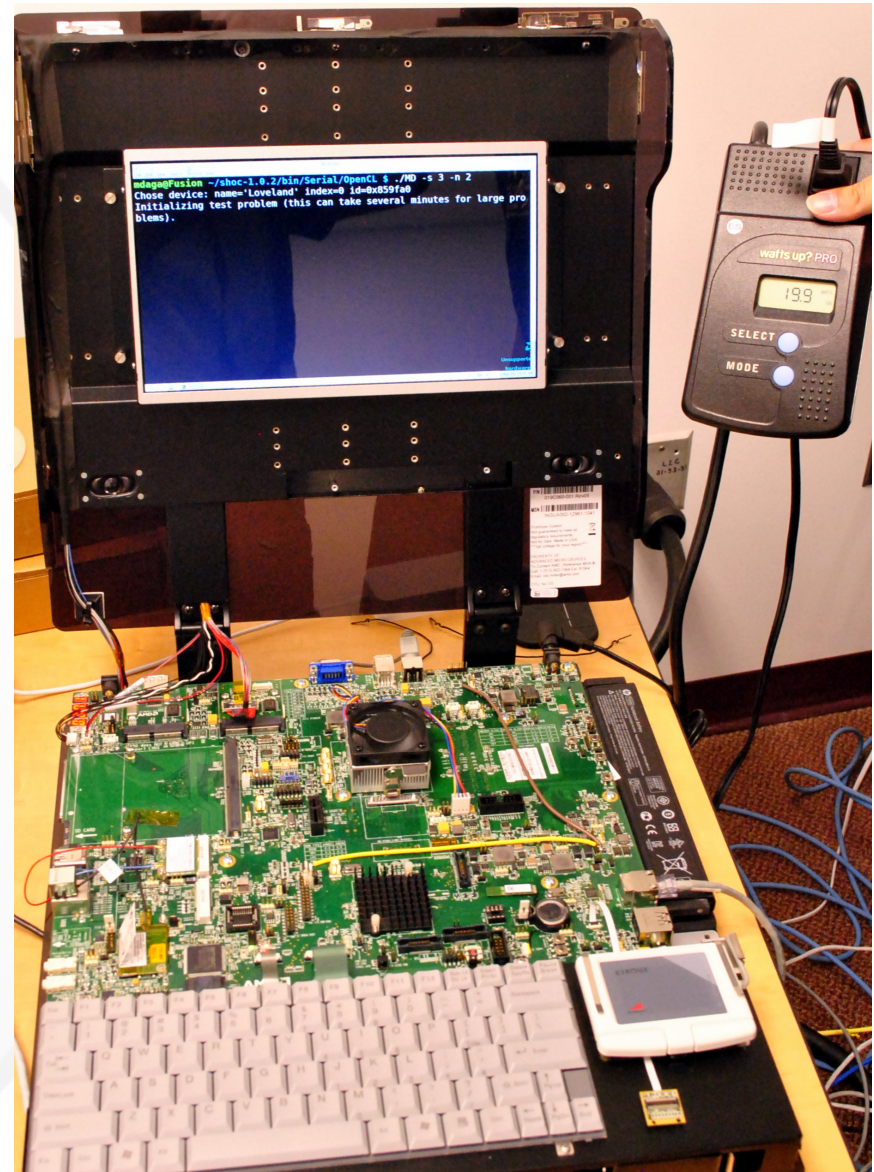
Performance: Reduction (Dense Linear Algebra)



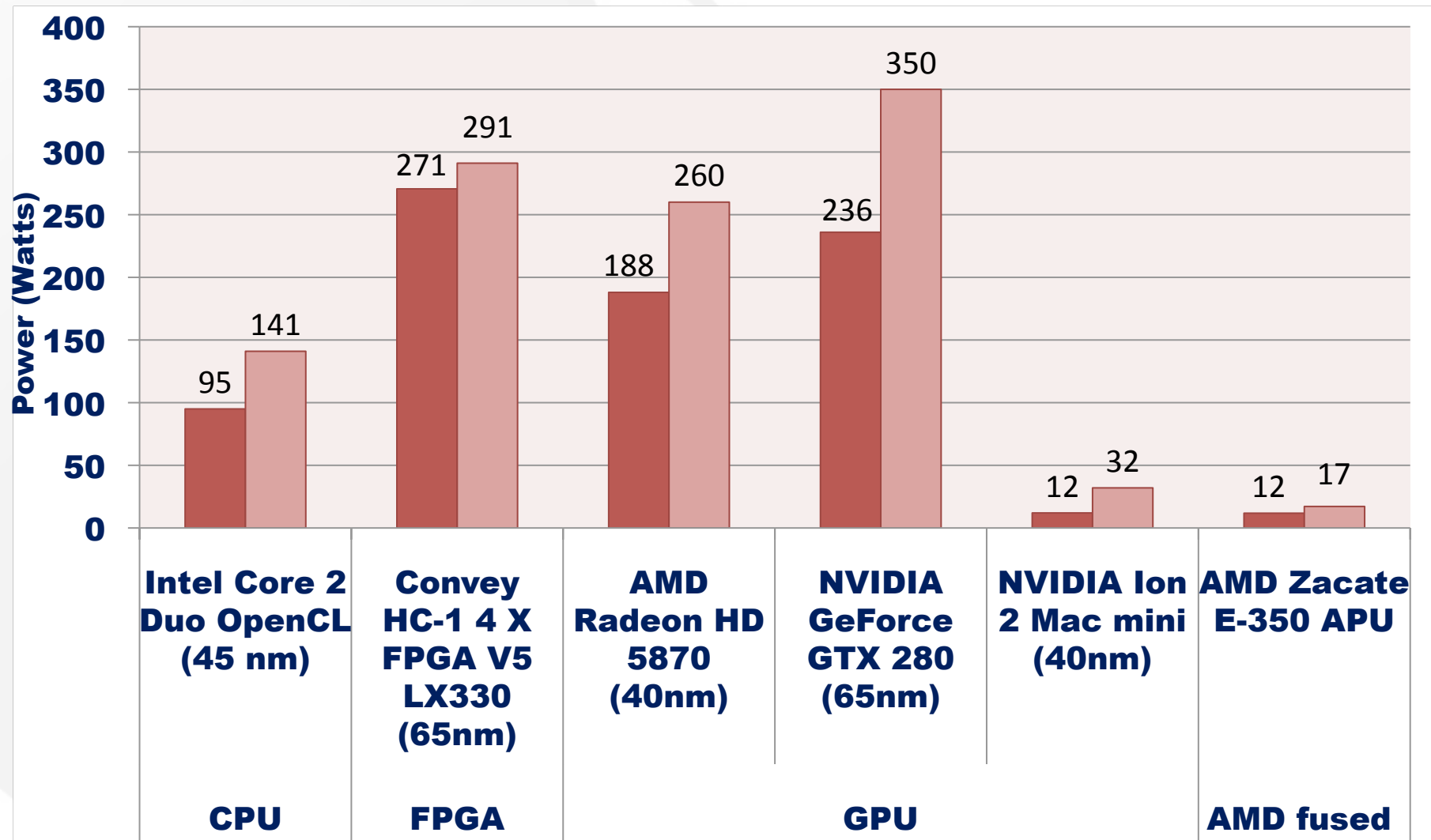
- Execution profiles of fused AMD APU & discrete AMD 5870 complementary.
 - ❖ Fused AMD APU die with only 80 GPU cores significantly improves data transfer time.
 - ❖ Discrete AMD 5870 die with 1600 faster GPU cores significantly improves kernel execution time.

System Power

- AMD Fusion APU
 - At idle: 12 watts
 - At load: 17 watts
(Spectral Method: FFT)
 - At load: 20 watts
(N-body: Molecular Modeling)
- AMD Radeon HD 5870 Machine w/ 2-GHz Intel Xeon E5405
 - At idle: 188 watts
 - At load: 260 watts



Total System Power: Idle vs. At Load (w/ FFT)



Status of OpenCL & the 13 Dwarfs

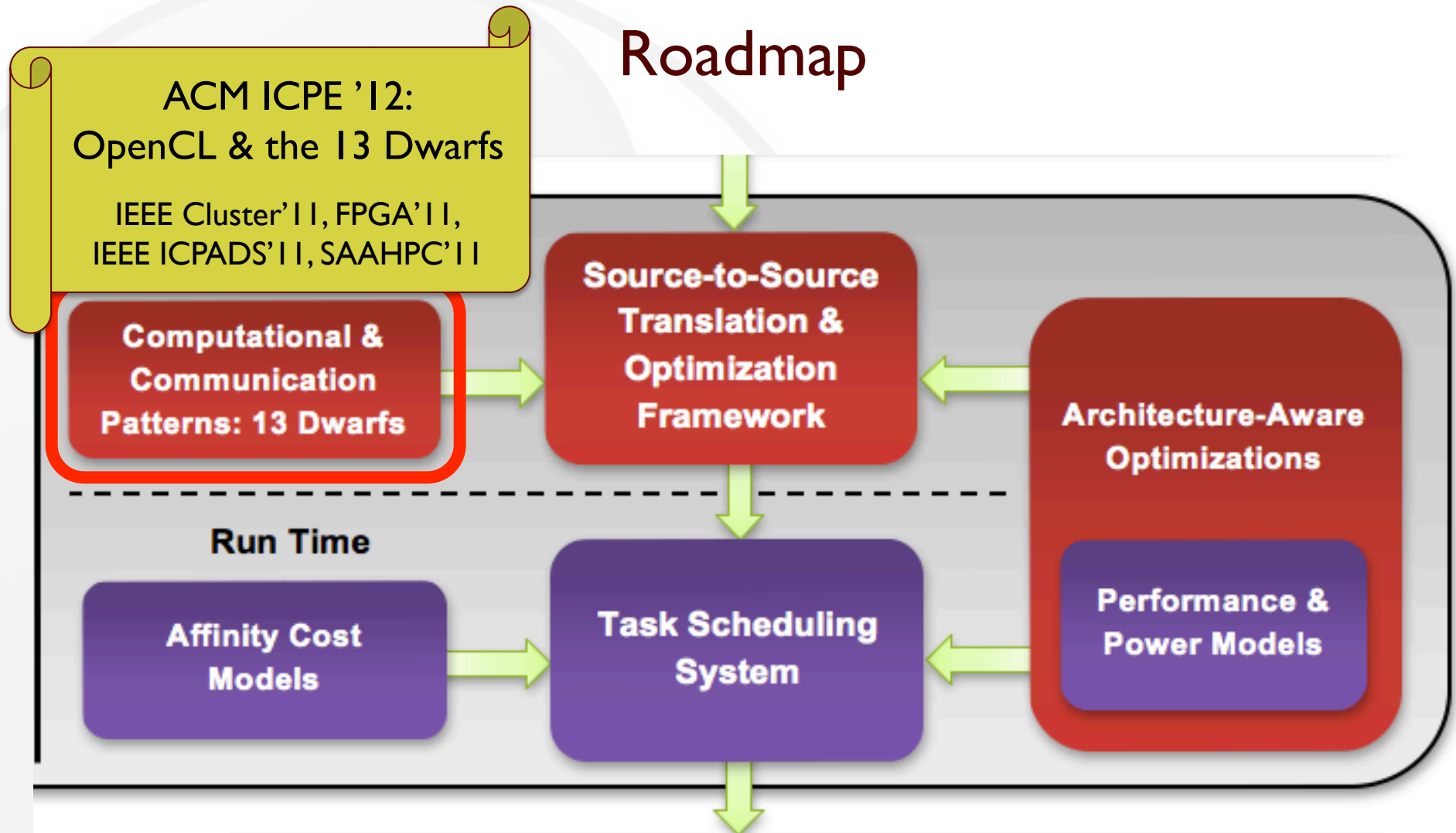
2009 – 2011

Dwarf	Done
Dense linear algebra	LU Decomposition
Sparse linear algebra	Matrix Multiplication
Spectral methods	FFT
N-Body methods	GEM
Structured grids	SRAD
Unstructured grids	CFD solver
MapReduce	
Combinational logic	CRC
Graph traversal	Breadth-First Search (BFS)
Dynamic programming	Needleman-Wunsch
Backtrack and Branch-and-Bound	
Graphical models	Hidden Markov Model
Finite state machines	Temporal Data Mining

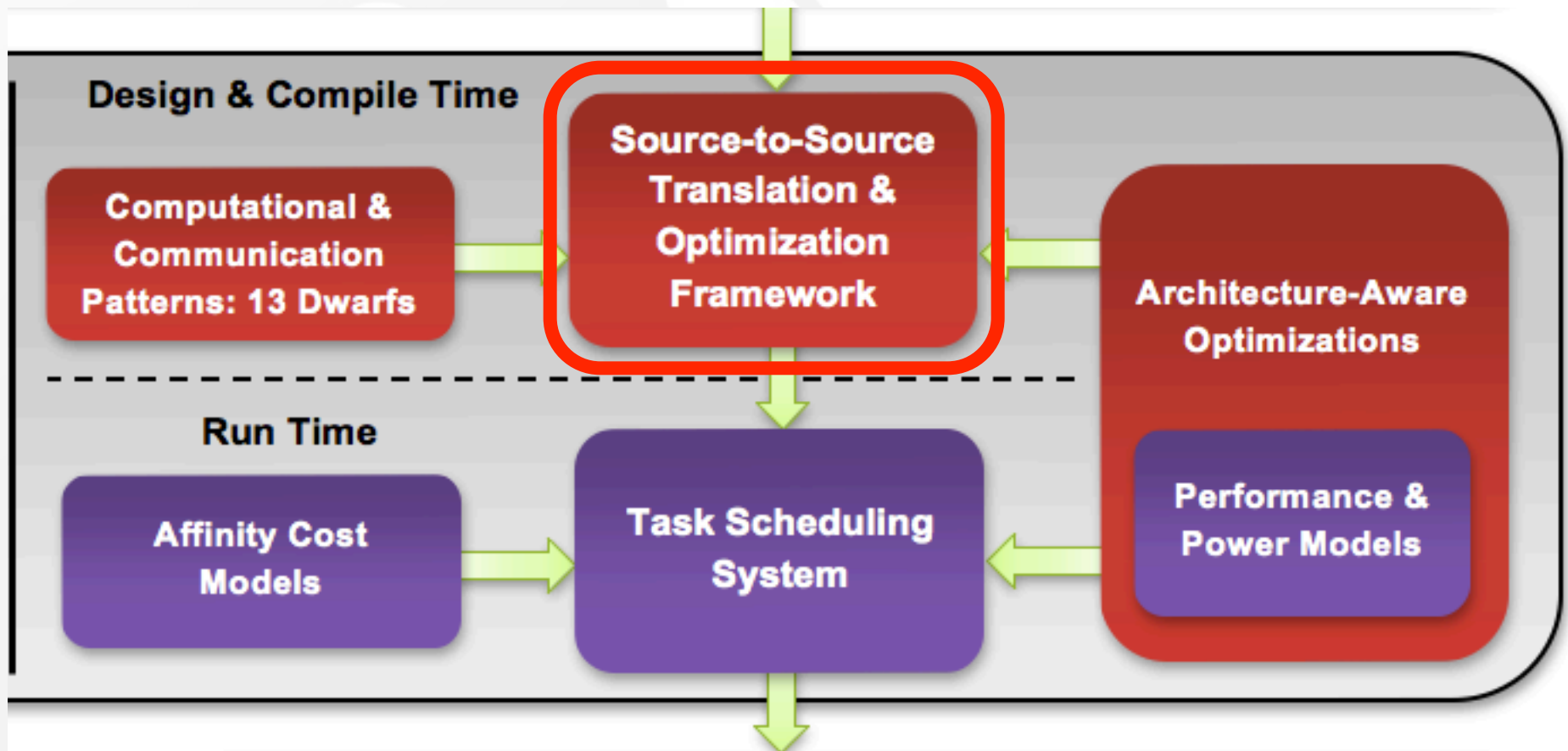
88x → 371x

Performance, Programmability, Portability

Roadmap



Roadmap



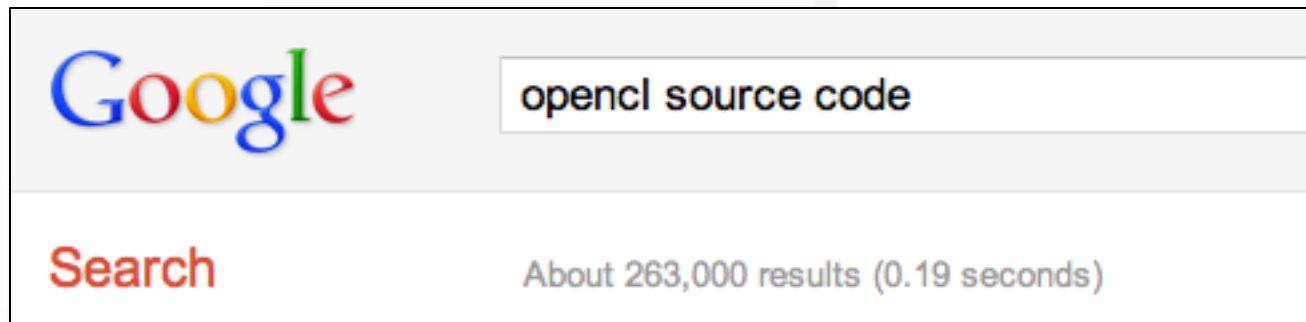
Prevalence



Google

Search About 1,170,000 results (0.50 seconds)

First public release February 2007



Google

Search About 263,000 results (0.19 seconds)

First public release December 2008

Search Date: 2011-09-14

CUDA-Accelerated Applications

GOVERNMENT & DEFENSE

Ikena: Imagery Analysis and Video Forensics
Signal Processing Library: GPU VSIPL
IDL and MATLAB® Acceleration: GPULib
GIS: Manifold

MOLECULAR DYNAMICS, COMPUTATIONAL CHEMISTRY

OpenMM library for molecular dynamics on GPUs
GROMACS using OpenMM
NAMD molecular dynamics
VMD visualization of molecular dynamics
HOOMD molecular dynamics
Acellera: ACEMD bio-molecular dynamics package
BigDFT: DFT (Density functional theory) electronic structure
MDGPU
GPUGrid.net

LIFE SCIENCES, BIO-INFORMATICS

GPU HMMER
DNA Sequence alignment: MUMmerGPU
LISSOM: model of human neocortex using CUDA
Silicon Informatics: AutoDock

ELECTRODYNAMICS AND ELECTROMAGNETIC

Acceleware: FDTD Solver
Acceleware: EM Solutions
Remcom XStream FDTD
SPEAG Semcad X

CST Microwave Studio
Quantum electrodynamics library
GPMAD : Particle beam dynamics simulator

MEDICAL IMAGING, CT, MRI

RealityServer
GPULib: IDL acceleration
Acceleware: Imaging Solutions
Digisens: SnapCT tomographic reconstruction software
Techniscan: Whole Breast Ultrasound Imaging System

OIL & GAS

Acceleware: Kirchoff and Reverse Time Migration
SeismicCity: 3D seismic imaging for prestack depth migration
OpenGeoSolutions: Spectral decomposition and inversion
Mercury Computer systems: 3D data visualization
ffA: 3D Seismic processing software
Headwave: Prestack data processing

FINANCIAL COMPUTING AND OPTIONS PRICING

SciComp: derivatives pricing
Hanweck: options pricing
Exegy: Risk Analysis
Aqumin: 3D Visualization of market data
Level 3 Finance
OnEye (Australia): Accelerated Trading Solutions
Arbitragis Trading

MATLAB, LABVIEW, MATHEMATICA, R

CUDA Acceleration for MATLAB
Accelereyes: Jacket™ engine for MATLAB
GPULib: mathematical functions for IDL and MATLAB
Integrating Simulink with CUDA using S-functions
Enabling GPU Computing in the R Statistical Environment
Mathematica plug-in for CUDA
National Instruments LabView for NVIDIA GPUs

ELECTRONIC DESIGN AUTOMATION

Agilent EESof: ADS SPICE simulator
Synopsis: Sentaurus TCAD
Gauda: Optical proximity correction (OPC)

WEATHER AND OCEAN MODELING

CUDA-accelerated WRF code

VIDEO, IMAGING, AND VISION APPLICATIONS

Axxon Intellect Enterprise Video Surveillance Software
Pflow CUDA Plugin for Autodesk 3ds Max
RUIINS Shatter CUDA Plug-in for Maya
Bullet 3D Multi-Physics Library with CUDA Support
CUDA Voxel Rendering Engine
Furryball: Direct3D GPU Rendering Plugin for Maya

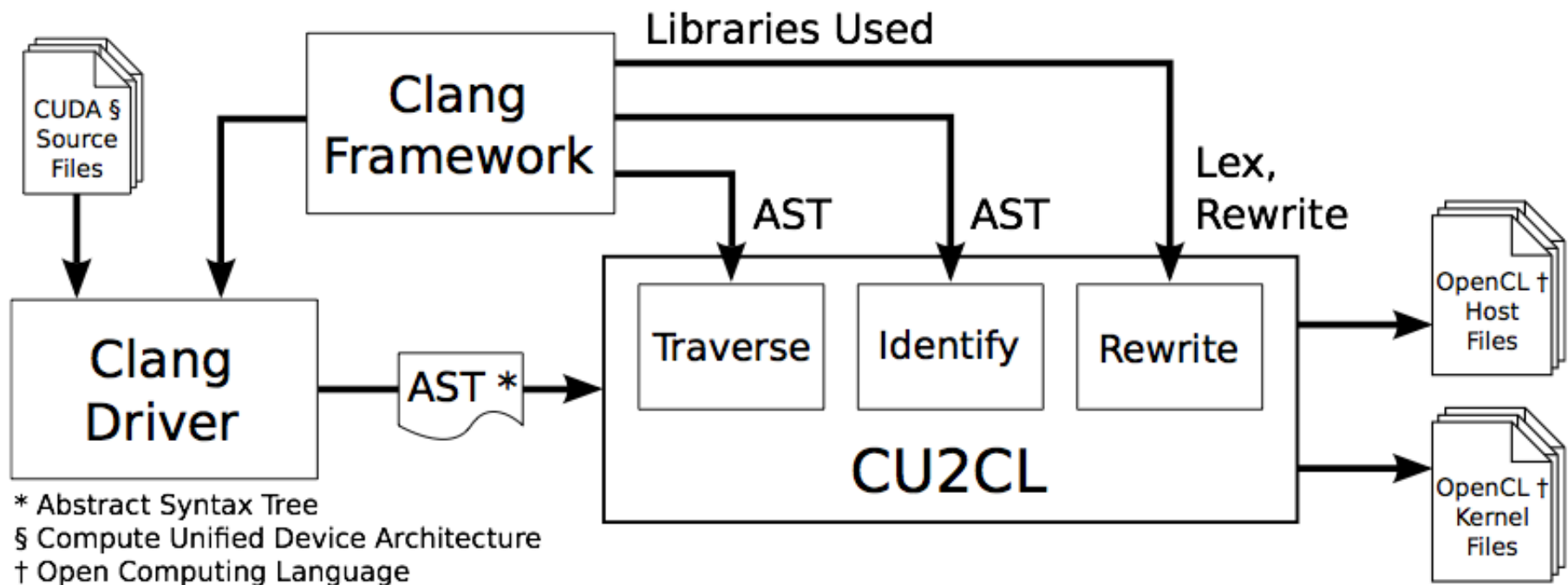
See: http://www.nvidia.com/object/cuda_app_tesla.html

CU2CL: CUDA-to-OpenCL Source-to-Source Translator†

- Implemented as a Clang plug-in to leverage its production-quality compiler framework and target LLVM bytecode.
- Covers primary CUDA constructs found in CUDA C and CUDA run-time API.
- Performs as well as codes manually ported from CUDA to OpenCL.
- *Others: OpenCL-to-CUDA and OpenMP-to-OpenCL*

† “CU2CL: A CUDA-to-OpenCL Translator for Multi- and Many-core Architectures,” 17th IEEE Int’l Conf. on Parallel & Distributed Systems (ICPADS), Dec. 2011.

CU2CL Translation and Performance

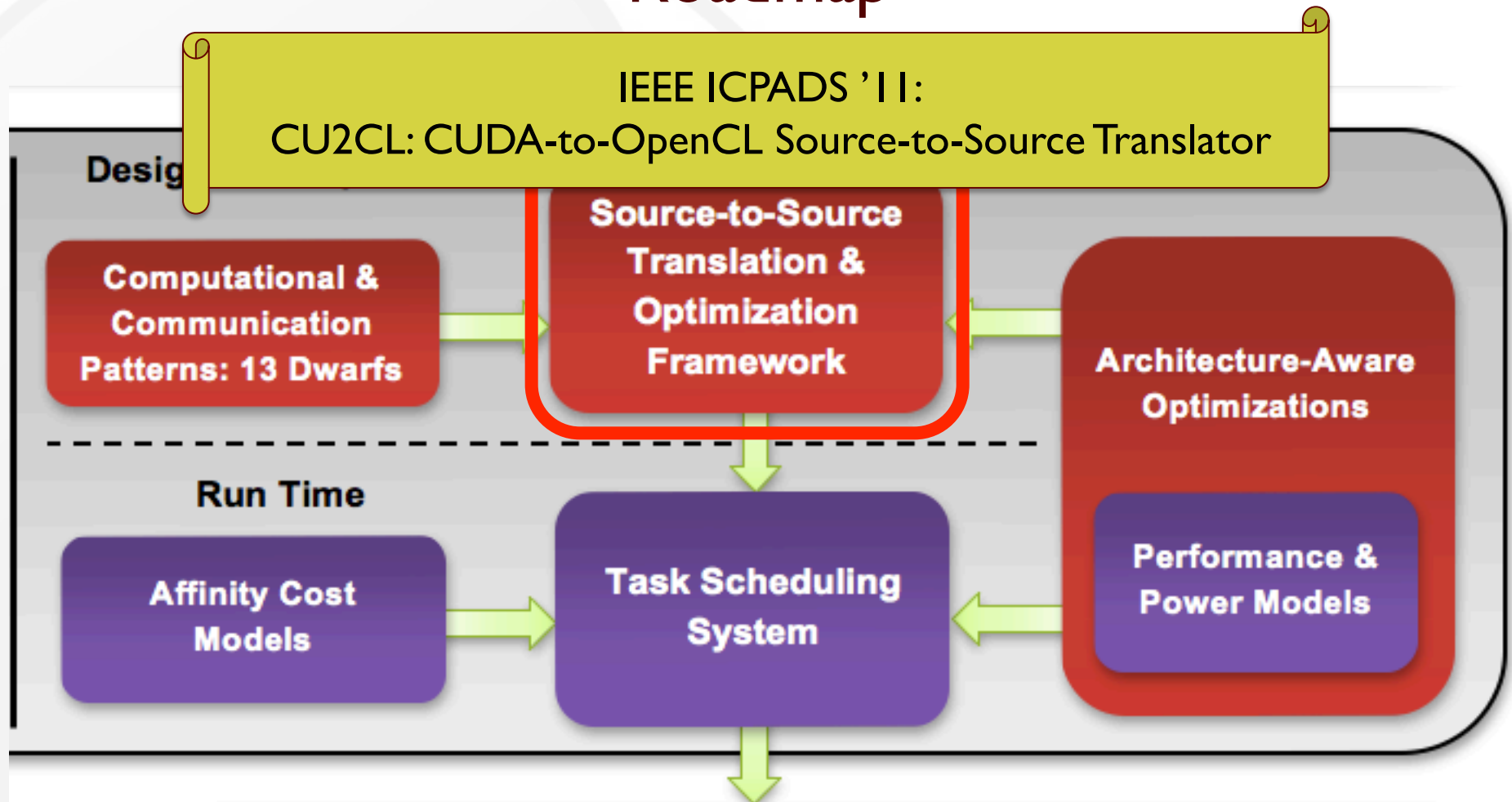


- Automatically translated OpenCL codes (via CU2CL) yield similar execution times to manually translated OpenCL codes

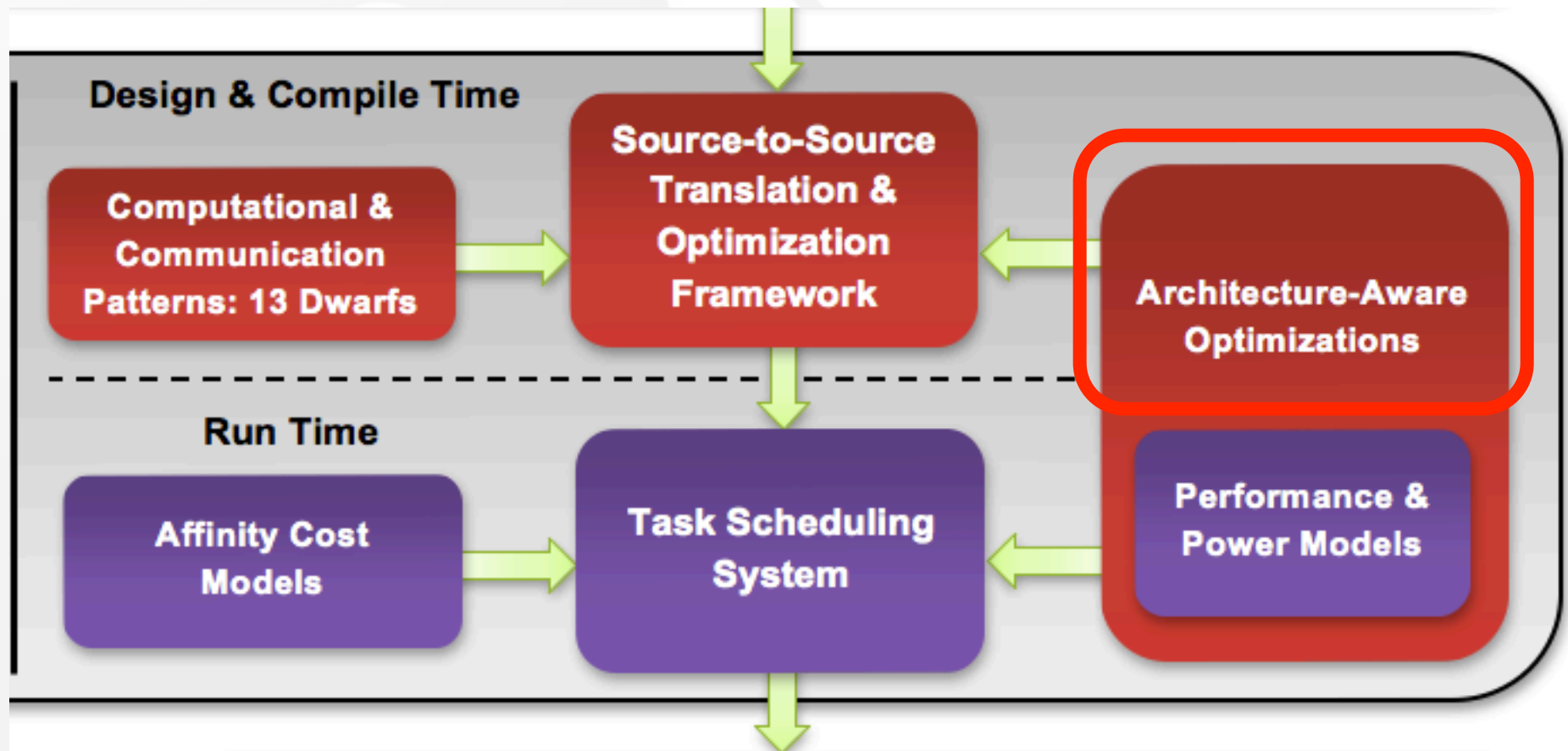
CU2CL Coverage

Source	Application	Lines	Changed	Percentage
CUDA SDK	BlackScholes	347	4	98.85
	matrixMul	351	2	99.43
	scalarProd	171	4	97.66
	vectorAdd	147	0	100.00
Rodinia	Back Propagation	313	5	98.40
	Hotspot	328	7	97.87
	Needleman-Wunsch	418	0	100.00
	SRAD	541	0	100.00
Virginia Tech	GEM: n-body molecular	2,511	5	99.80

Roadmap

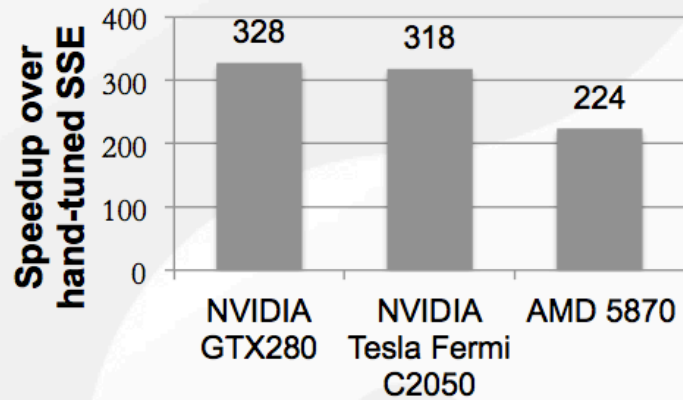


Roadmap

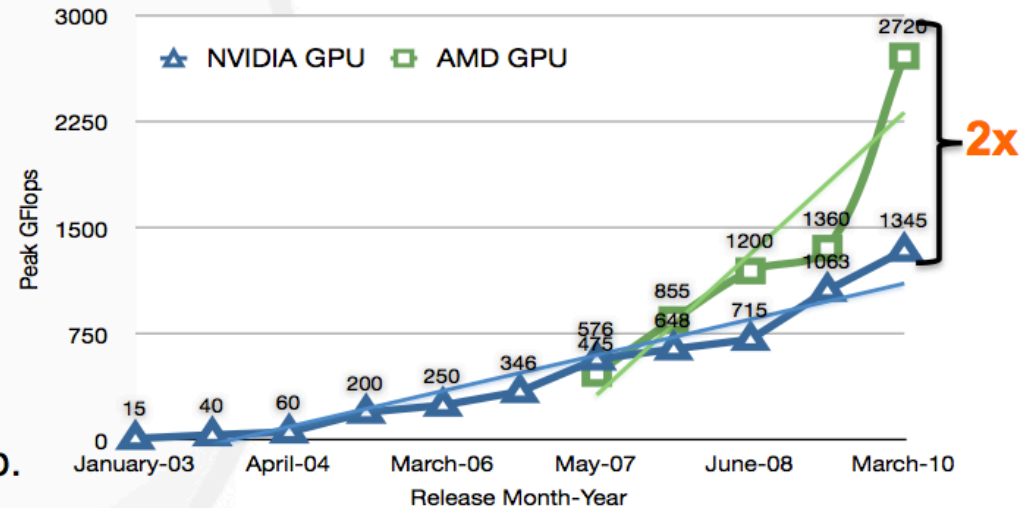


Computational Units *Not* Created Equal

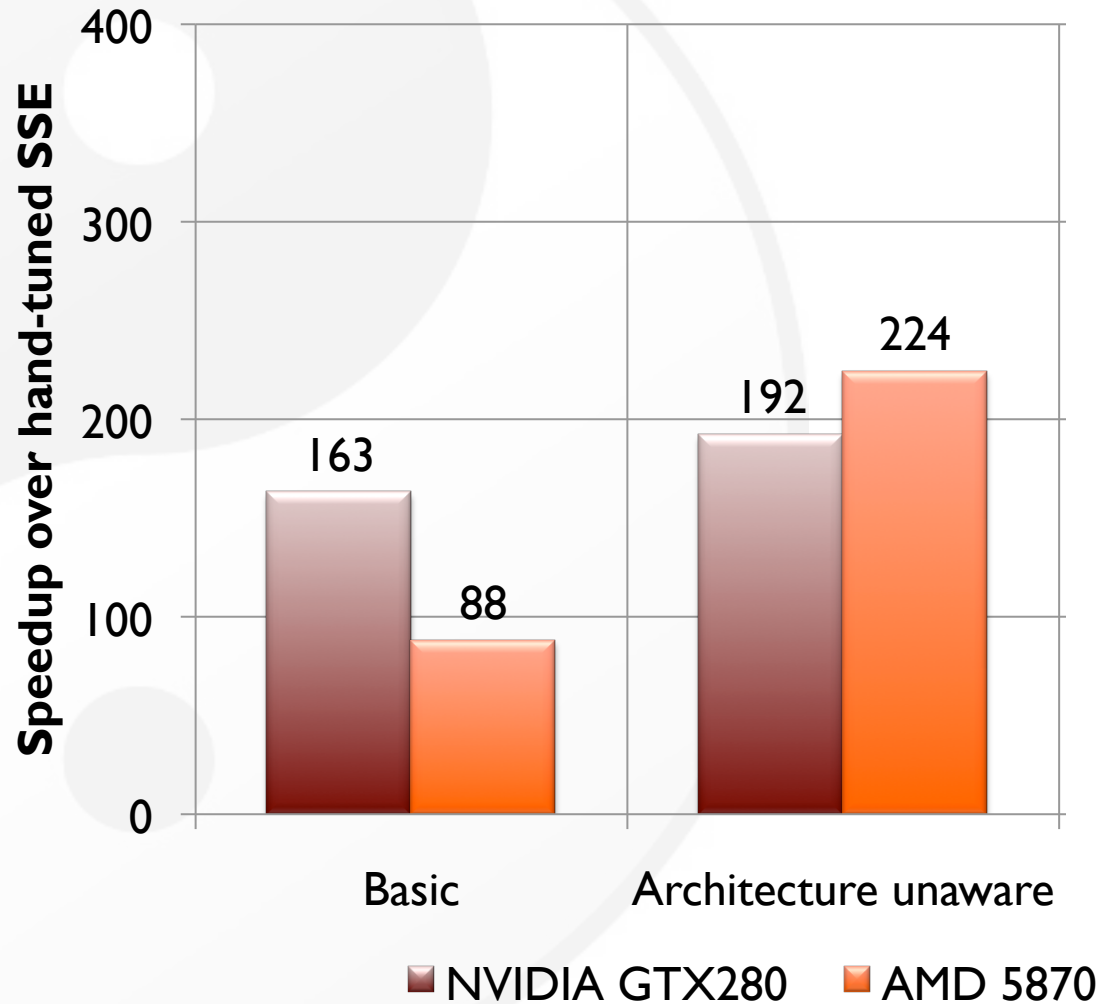
- “AMD CPU \neq Intel CPU” and “AMD GPU \neq NVIDIA GPU”
- Initial performance of a *CUDA-optimized* N-body dwarf



Performance of a Molecular Modeling App.

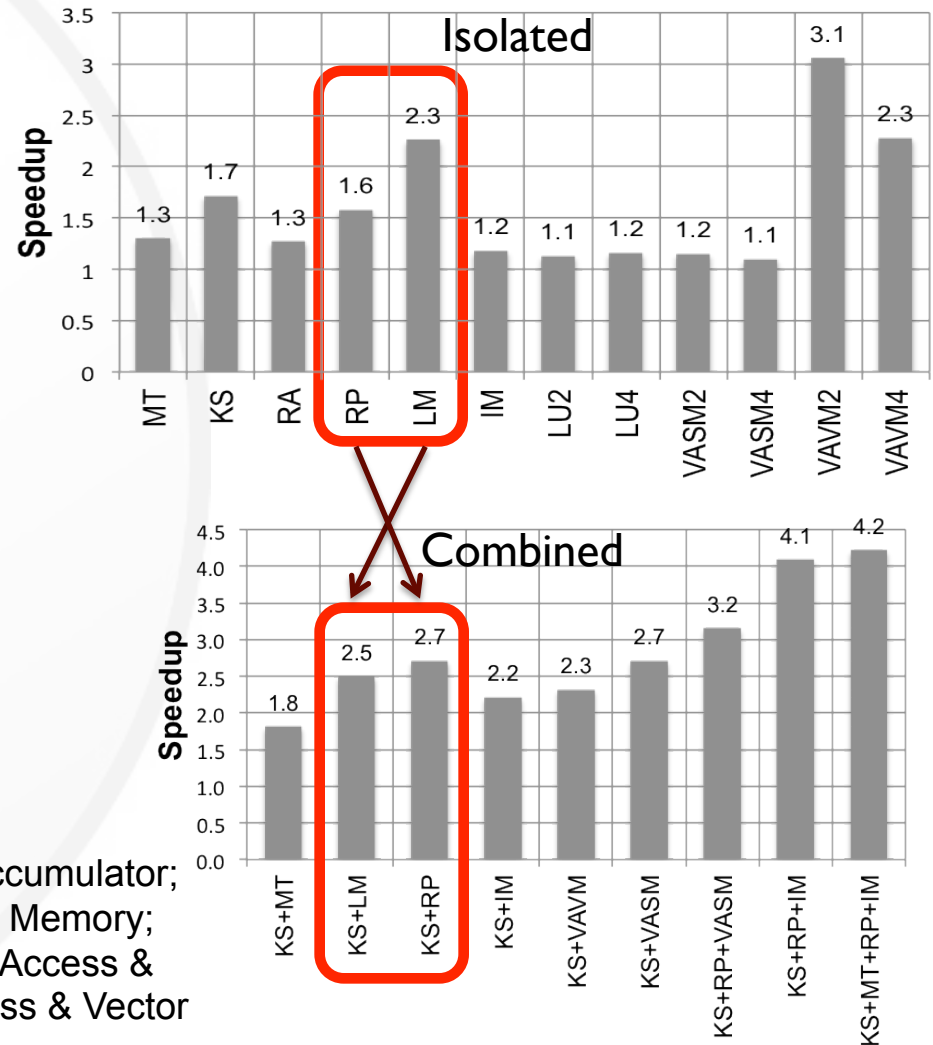


Basic & Architecture-Unaware Execution



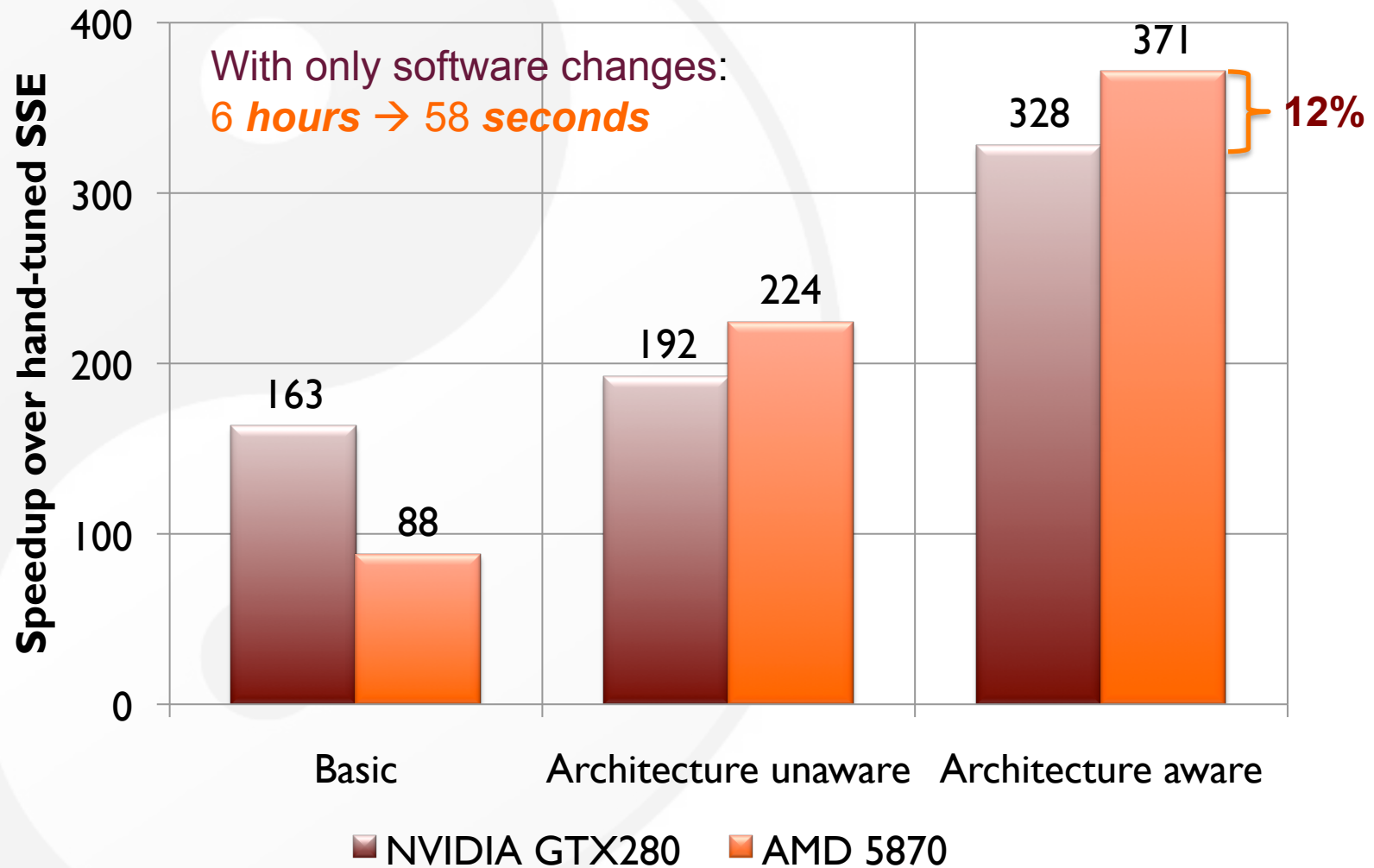
Architecture-Aware Optimization (N-body Code for Molecular Modeling)

- Optimization techniques on AMD GPUs
 - Removing conditions → kernel splitting
 - Local staging
 - Using vector types
 - Using image memory
- Speedup over basic OpenCL GPU implementation
 - Isolated optimizations
 - Combined optimizations

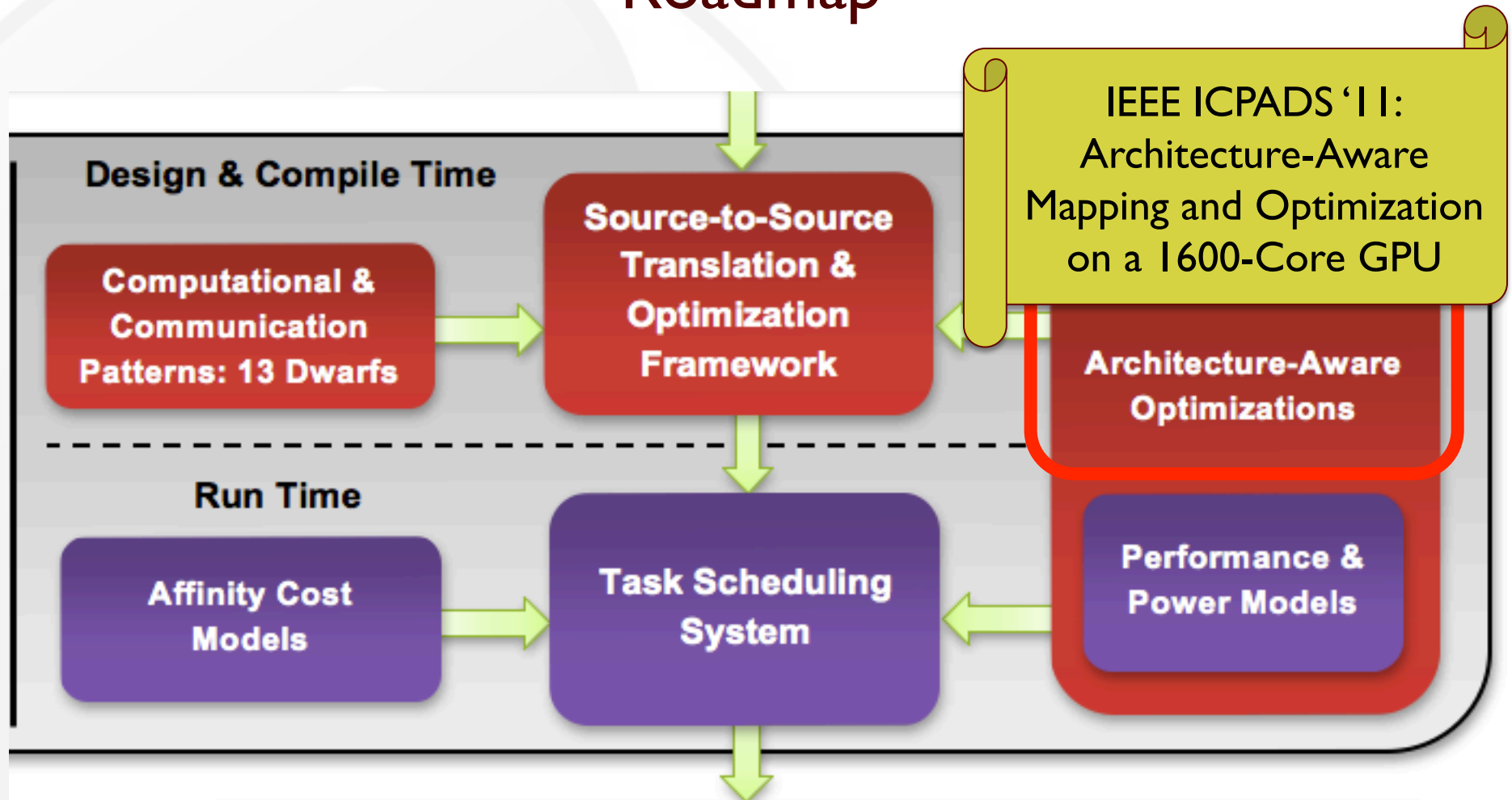


MT: Max Threads; **KS:** Kernel Splitting; **RA:** Register Accumulator;
RP: Register Preloading; **LM:** Local Memory; **IM:** Image Memory;
LU{2,4}: Loop Unrolling{2x,4x}; **VASM{2,4}:** Vectorized Access &
 Scalar Math{float2, float4}; **VAVM{2,4}:** Vectorized Access & Vector
 Math{float2, float4}

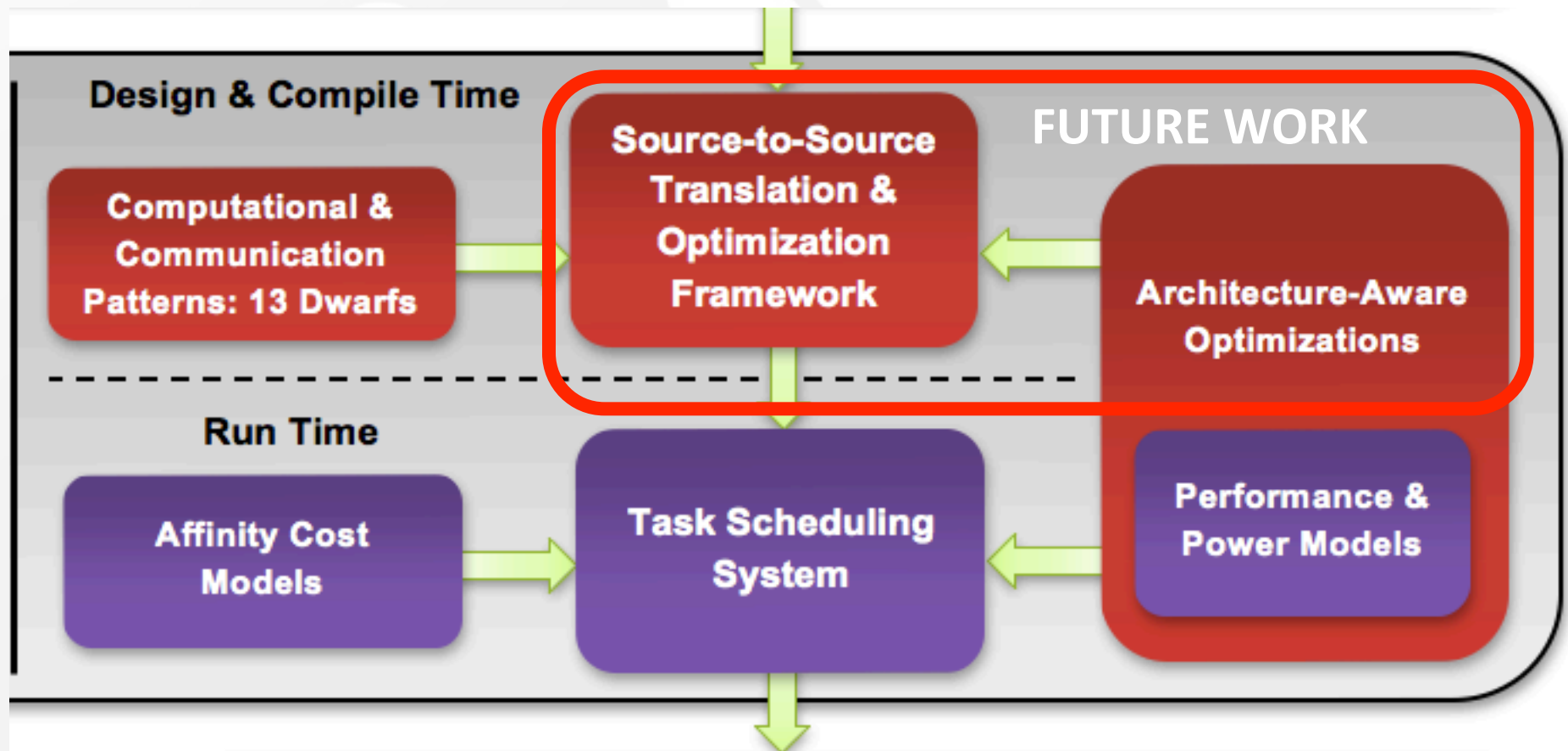
Summary: Architecture-Aware Optimization



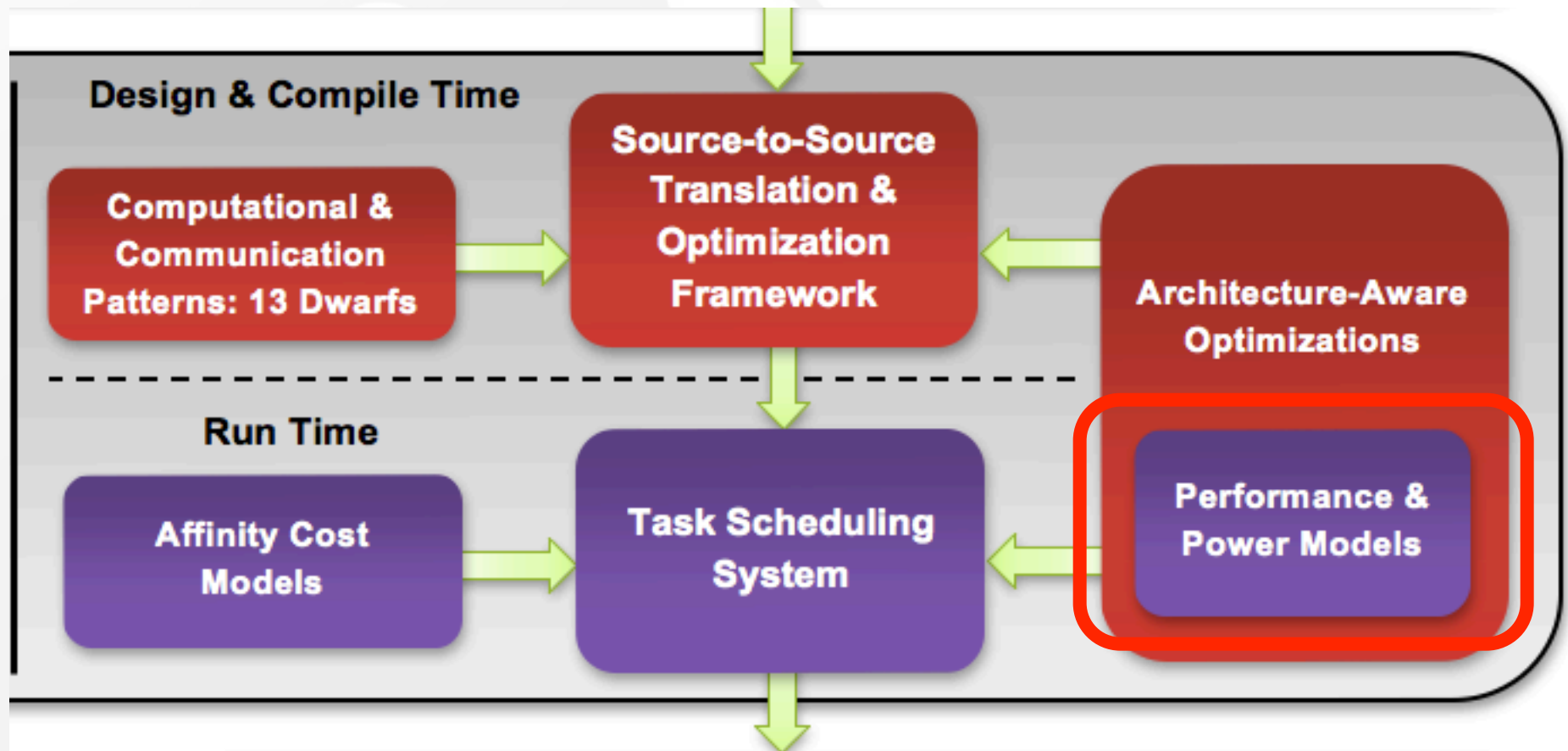
Roadmap



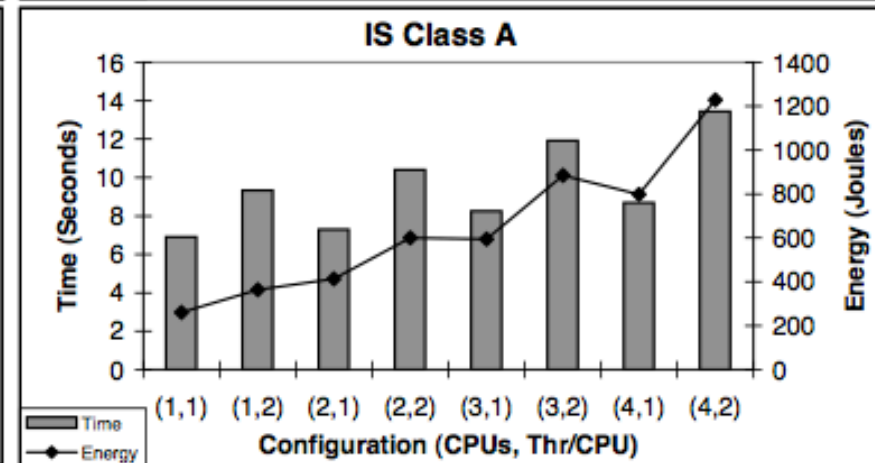
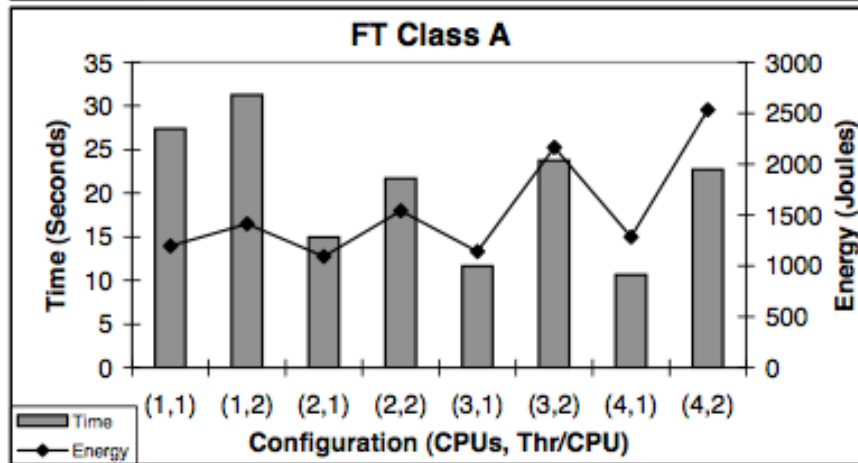
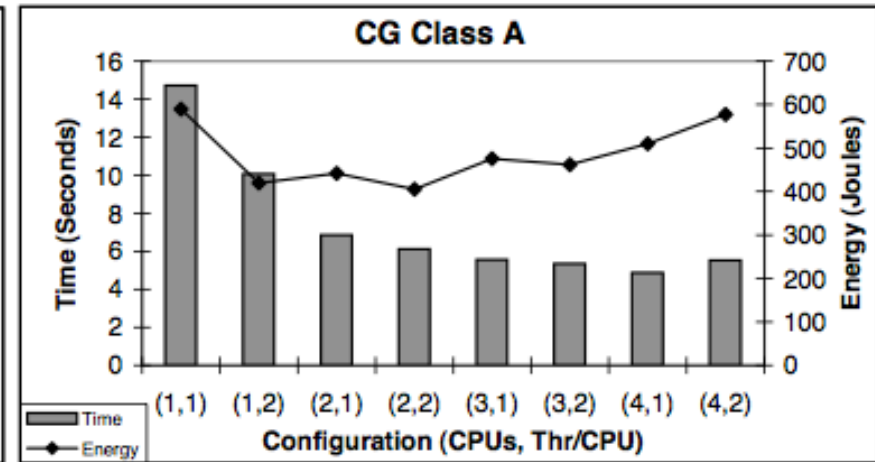
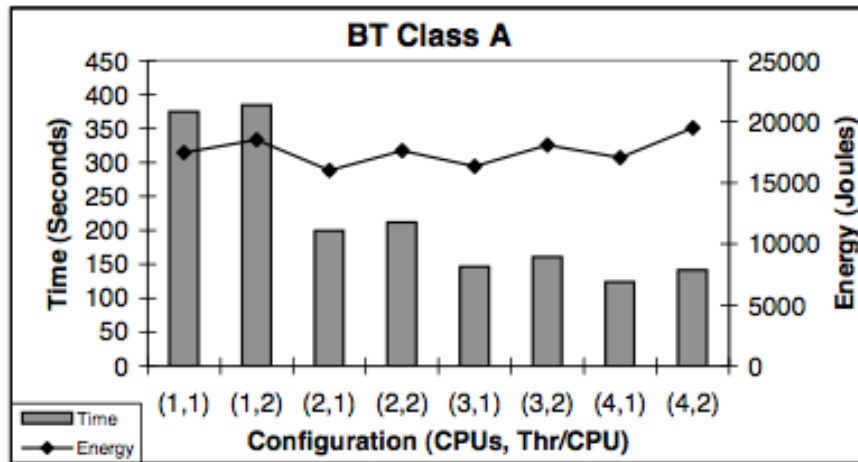
Roadmap



Roadmap



Need for Performance & Power Modeling



Source: Virginia Tech

Performance & Power Modeling

- **Goals**
 - Robust framework
 - Very high accuracy (Target: < 5% prediction error)
 - Identification of portable predictors for performance and power
 - Multi-dimensional characterization
 - Performance → sequential, intra-node parallel, inter-node parallel
 - Power → component level, node level, cluster level

Problem Formulation:

LP-Based Energy-Optimal DVFS Schedule

- **Definitions**
 - A DVFS system exports n $\{ (f_i, P_i) \}$ settings.
 - T_i : total execution time of a program running at setting i
- **Given a program with deadline D , find a DVS schedule (t_1^*, \dots, t_n^*) such that**
 - If the program is executed for t_i seconds at setting i , the total energy usage E is minimized, the deadline D is met, and the required work is completed.

$$\min E = \sum_i P_i \cdot t_i$$

subject to

$$\begin{aligned}\sum_i t_i &\leq D \\ \sum_i t_i / T_i &= 1 \\ t_i &\geq 0\end{aligned}$$

Single-Coefficient β Performance Model

- Our Formulation

- Define the relative performance slowdown δ as

$$T(f) / T(f_{MAX}) - 1$$

- Re-formulate two-coefficient model as a single-coefficient model:

$$\frac{T(f)}{T(f_{max})} = \beta \cdot \frac{f_{max}}{f} + (1 - \beta)$$

with

$$\beta = \frac{W_{cpu}}{W_{cpu} + T_{mem} \cdot f_{max}}$$

- The coefficient β is computed at run-time using a regression method on the past MIPS rates reported from the built-in PMU.

$$\beta = \frac{\sum_i \left(\frac{f_{max}}{f_i} - 1 \right) \left(\frac{\text{mips}(f_{max})}{\text{mips}(f_i)} - 1 \right)}{\sum_i \left(\frac{f_{max}}{f_i} - 1 \right)^2}$$

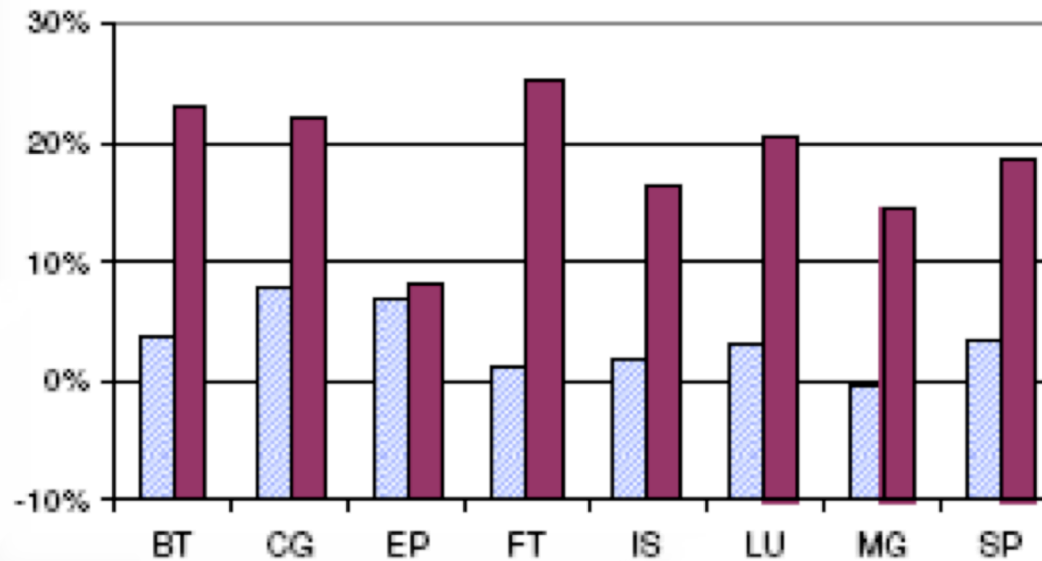
C. Hsu and W. Feng.
“A Power-Aware Run-Time System for High-Performance Computing,” *SC/05*, Nov. 2005.

NAS Parallel on an AMD Opteron Cluster



NAS/NPB3.2-MPL, C.16

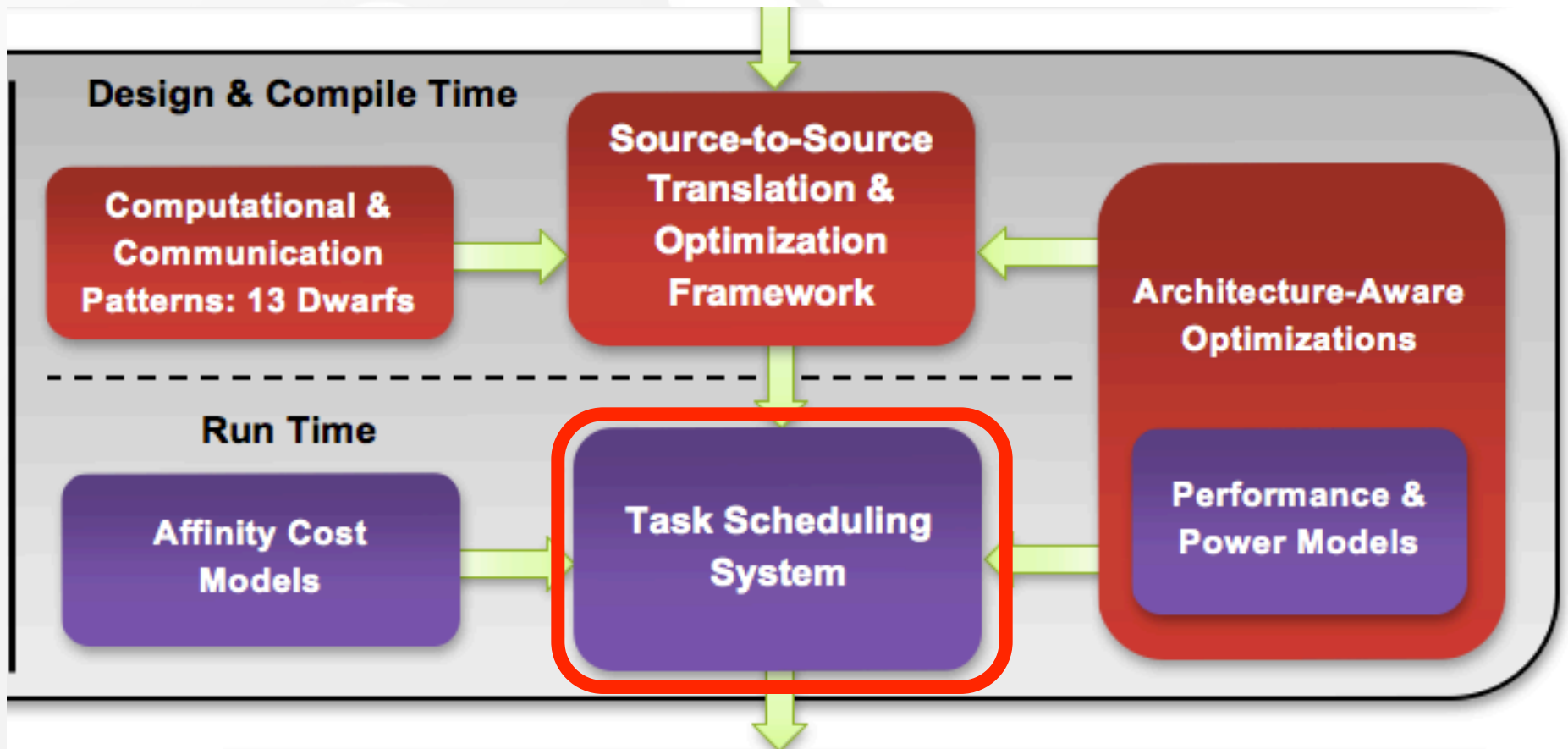
Slowdown Savings



Energy reduction (15%)
Performance improvement ↗

“A Power-Aware Run-Time System for High-Performance Computing,” SC/05, Nov. 2005

Roadmap



What is Heterogeneous Task Scheduling?

- Automatically spreading tasks across heterogeneous compute resources
 - CPUs
 - GPUs
 - APUs
- Specify tasks at a higher level (currently OpenMP extensions)
- Run them across available resources automatically

Goal

- A run-time system that intelligently uses what is available resource-wise and optimize for performance portability
 - Each user should *not* have to implement this for themselves!

How to Heterogeneous Task Schedule (HTS)

- Accelerated OpenMP offer heterogeneous task scheduling with
 - Programmability
 - Functional portability, given underlying compilers
 - Performance portability
- How?
 - A simple extension to Accelerated OpenMP syntax for **programmability**
 - Automatically dividing parallel tasks across arbitrary heterogeneous compute resources for **functional portability**
 - CPUs
 - GPUs
 - APUs
 - Intelligent runtime task scheduling for **performance portability**

Programmability: Why Accelerated OpenMP?

```
#pragma omp parallel for          \  
    shared(in1,in2,out,pow)  
for (i=0; i<end; i++){  
    out[i] = in1[i]*in2[i];  
    pow[i] = pow[i]*pow[i];  
}
```

Traditional OpenMP

```
#pragma acc_region_loop          \  
    acc_copyin(in1[0:end],in2[0:end])\  
    acc_copyout(out[0:end])      \  
    acc_copy(pow[0:end])  
for (i=0; i<end; i++){  
    out[i] = in1[i] * in2[i];  
    pow[i] = pow[i]*pow[i];  
}
```

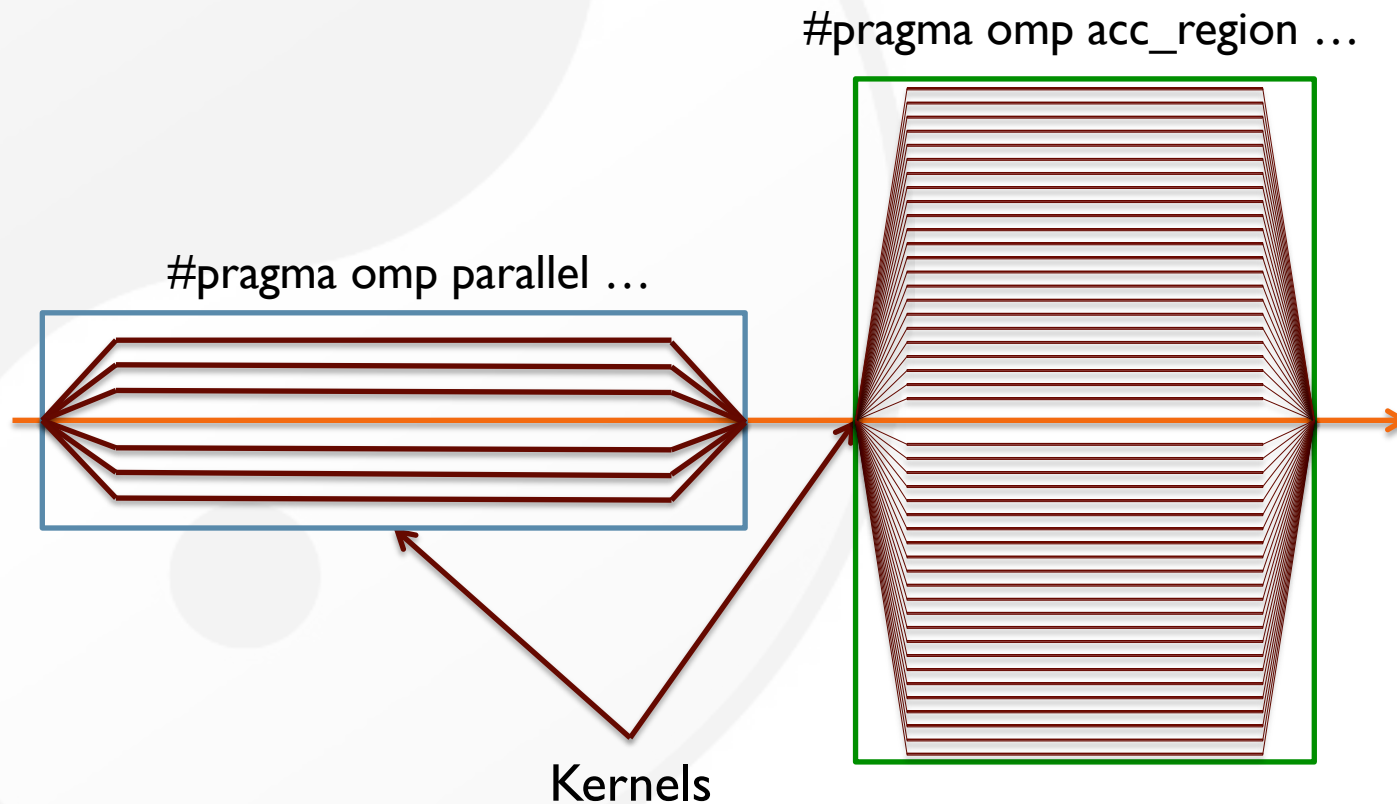
OpenMP Accelerator Directives

```
#pragma acc_region_loop          \  
    acc_copyin(in1[0:end],in2[0:end]) \  
    acc_copyout(out[0:end])        \  
    acc_copy(pow[0:end])           \  
    hetero(<cond>[,<scheduler>[,<ratio>\  
    [,<div>]]])  
for (i=0; i<end; i++){  
    out[i] = in1[i] * in2[i];  
    pow[i] = pow[i]*pow[i];  
}
```

Source: T. Scogland et al., IPDPS 2012

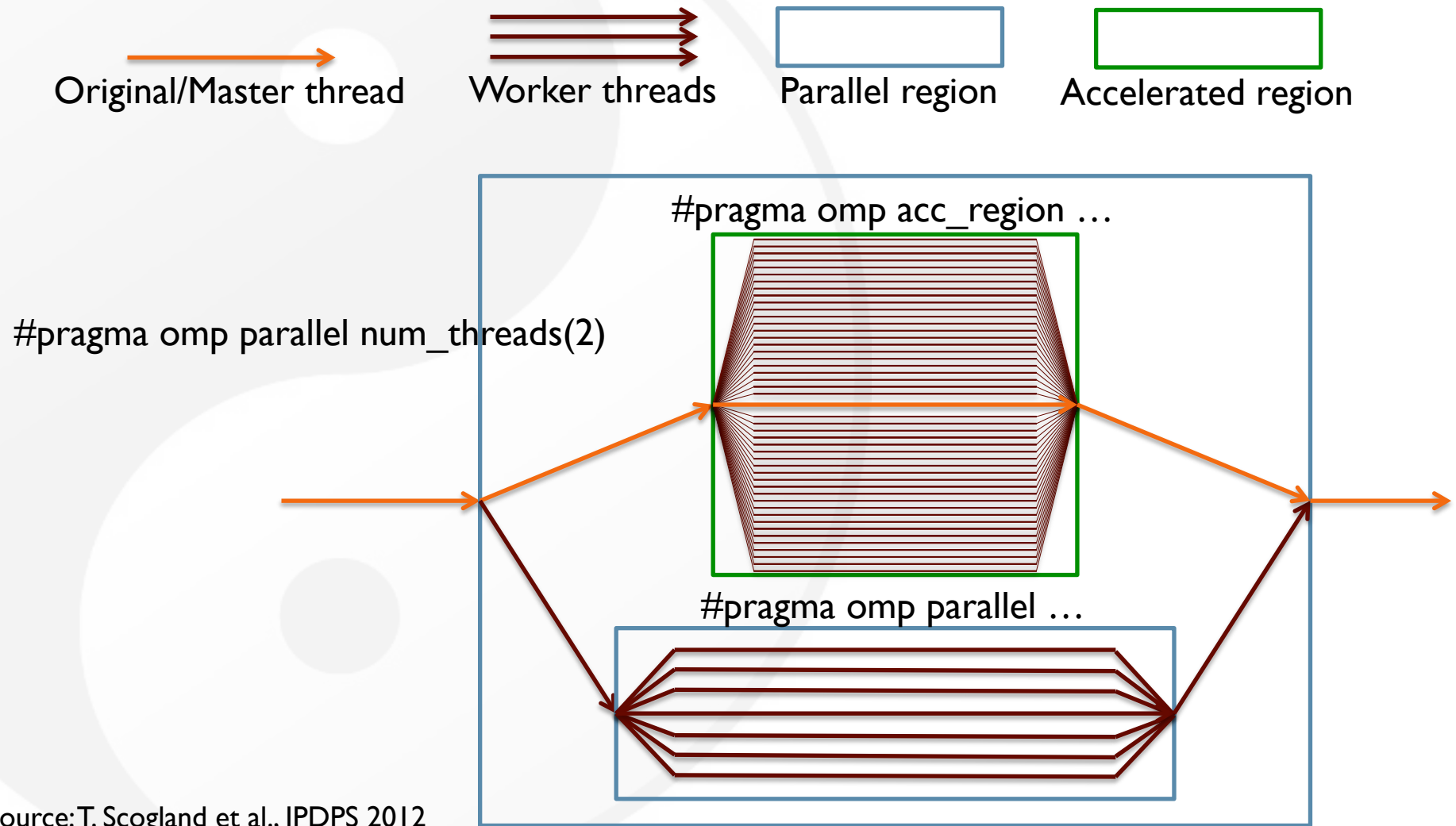
Our Proposed Extension

OpenMP Accelerator Behavior



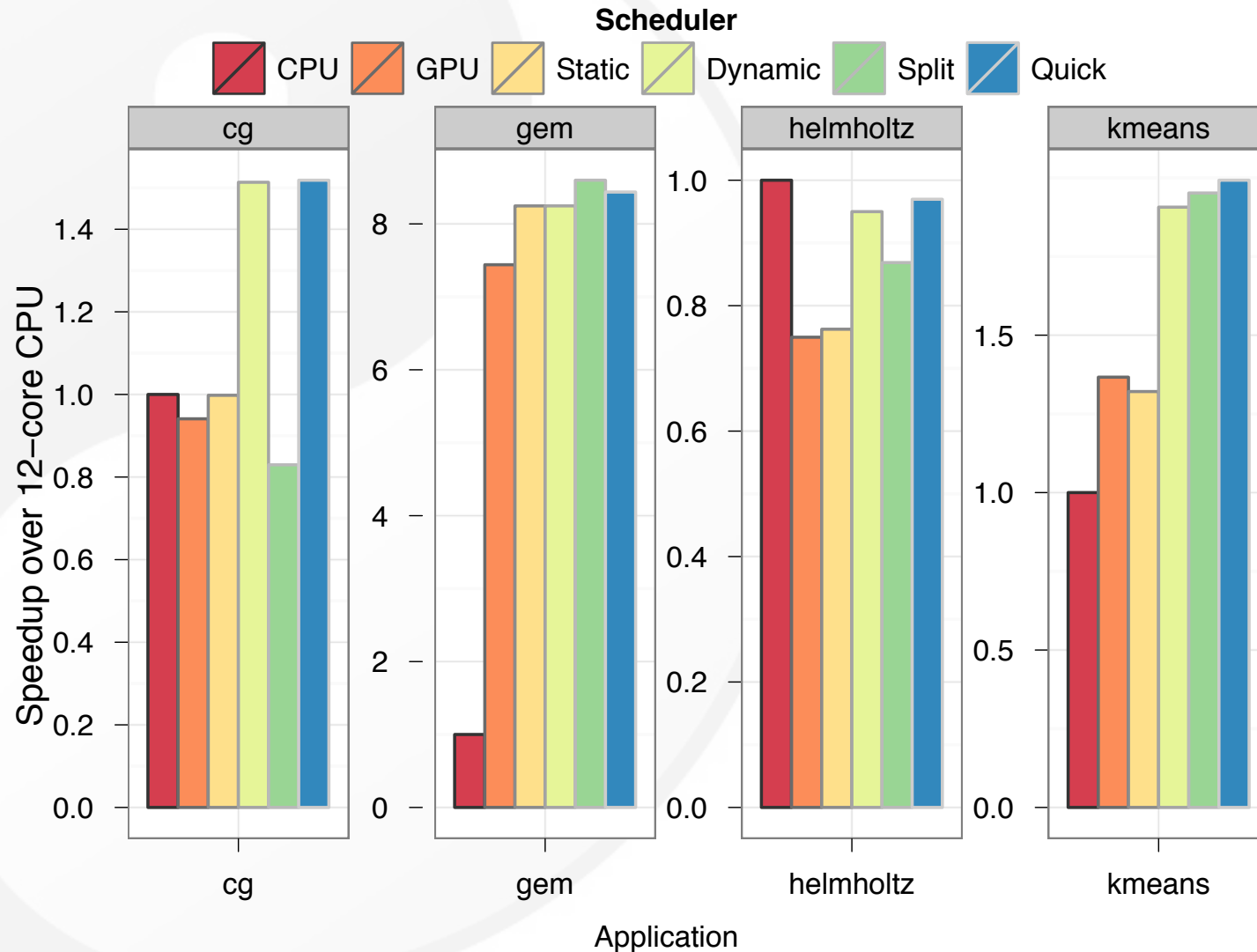
Source: T. Scogland et al., IPDPS 2012

DESIRED OpenMP Accelerator Behavior



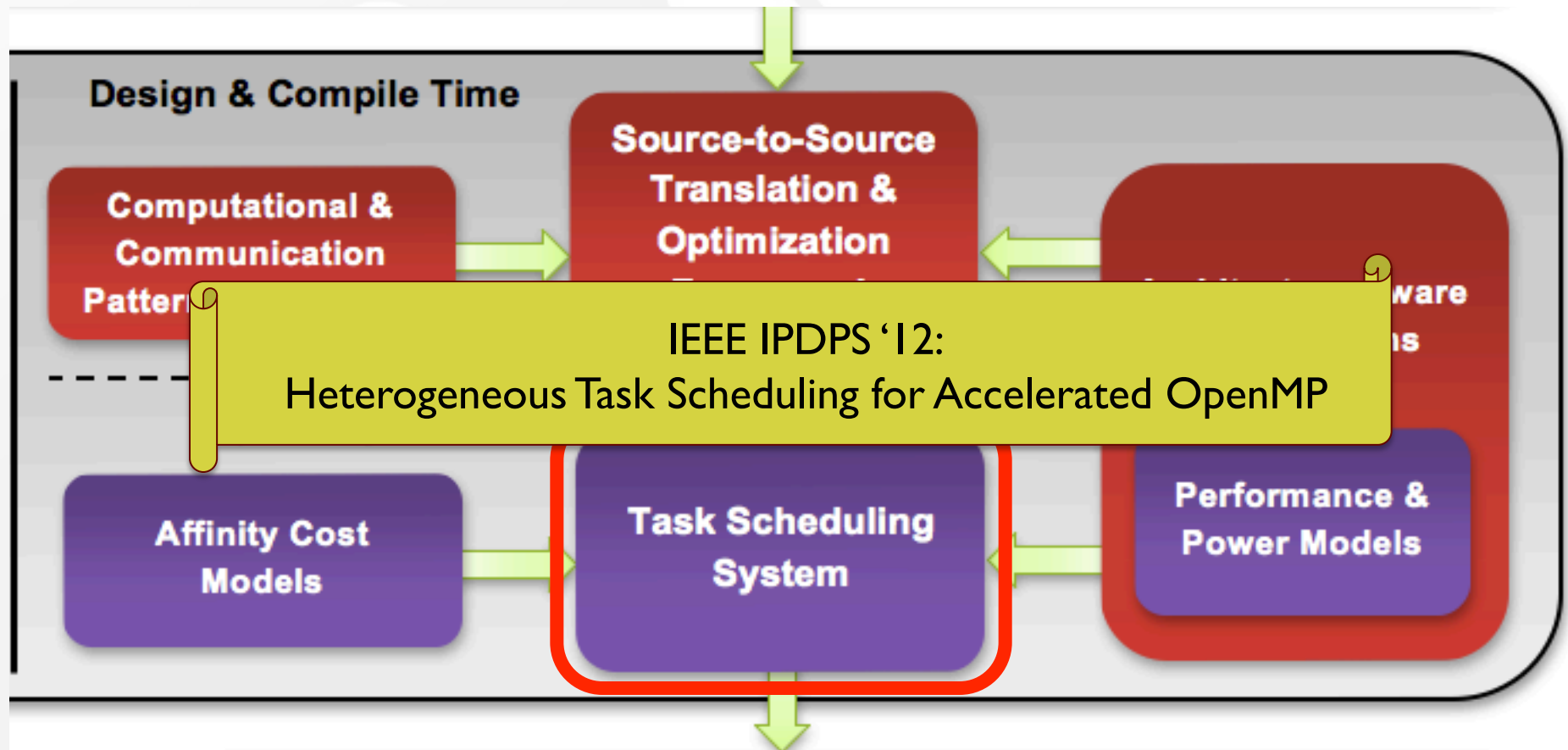
Source: T. Scogland et al., IPDPS 2012

Results Across Schedulers for all Benchmarks

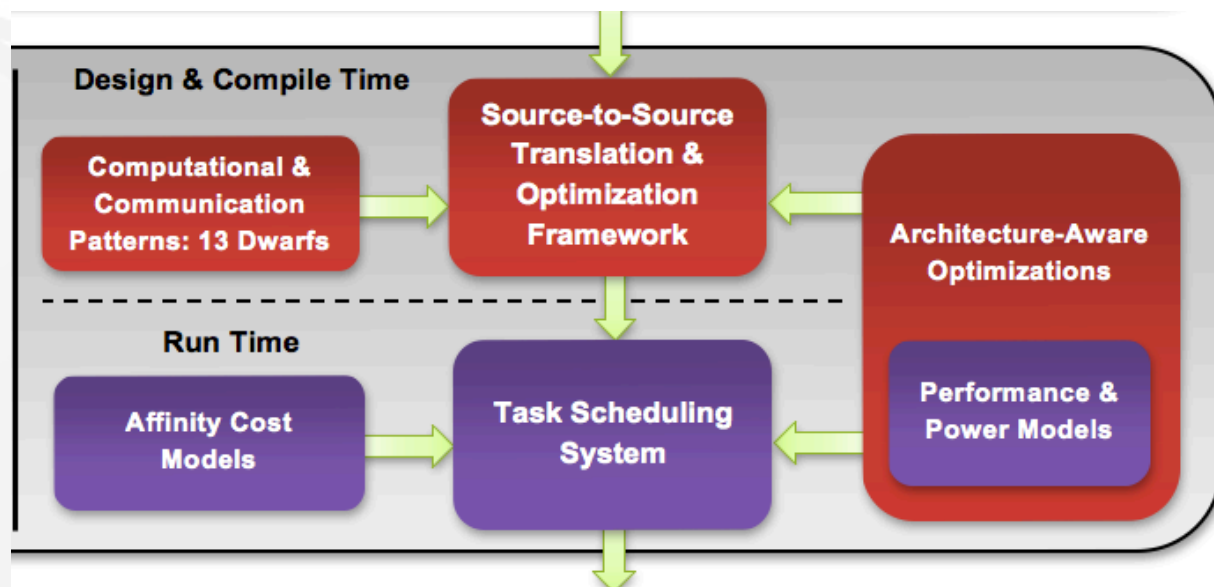


Performance, Programmability, Portability

Roadmap



An Ecosystem for the New HPC: Heterogeneous Parallel Computing



Much work still to be done.

- OpenCL and the 13 Dwarfs
- Source-to-Source Translation
- Architecture-Aware Optimization
- Performance & Power Modeling
- Affinity-Based Cost Modeling
- Heterogeneous Task Scheduling

Beta release pending
CU2CL only & no optimization
Only manual optimizations
Preliminary & pre-multicore
Empirical results; modeling in progress
Preliminary with OpenMP

Recent Publications

- M. Daga, T. Scogland, W. Feng, “Architecture-Aware Mapping and Optimization on a 1600-Core GPU,” *17th IEEE Int’l Conf. on Parallel & Distributed Systems*, Dec. 2011.
- M. Elteir, H. Lin, W. Feng, “StreamMR: An Optimized MapReduce Framework for AMD GPUs,” *17th IEEE Int’l Conf. on Parallel & Distributed Systems*, Dec. 2011.
- W. Feng, Y. Cao, D. Patnaik, N. Ramakrishnan, “Temporal Data Mining for Neuroscience,” *GPU Computing Gems*, Editor: W. Hwu, Elsevier/Morgan-Kaufmann, Feb. 2011.
- K. Bisset, A. Aji, M. Marathe, W. Feng, “High-Performance Biocomputing for Simulating the Spread of Contagion over Large Contact Networks,” *BMC Genomics*, 2011.
- M. Elteir, H. Lin, W. Feng, “Performance Characterization and Optimization of Atomic Operations on AMD GPUs,” *IEEE Cluster*, Sept. 2011.
- M. Daga, A. Aji, W. Feng, “On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing,” *Symp. on Application Accelerators in High Performance Computing*, Jul. 2011.
- A. Aji, M. Daga, and W. Feng, “Bounding the Effect of Partition Camping in Memory-Bound Kernels,” *ACM Int’l Conf. on Computing Frontiers*, May 2011.
- S. Xiao, H. Lin, and W. Feng, “Accelerating Protein Sequence Search in a Heterogeneous Computing System,” *25th Int’l Parallel & Distributed Processing Symp.*, May 2011.
- W. Feng with cast of many, “Accelerating Electrostatic Surface Potential Calculation with Multi-Scale Approximation on Graphics Processing Units,” *J. Molecular Graphics and Modeling*, Jun. 2010.
- W. Feng and S. Xiao, “To GPU Synchronize or Not GPU Synchronize?” *IEEE Int’l Symp. on Circuits and Systems*, May-June 2010.

Funding Acknowledgements



VirginiaTech

AMD



IBM



AMD



HARRIS



XILINX



Wu Feng, wfeng@vt.edu, 540-231-1192



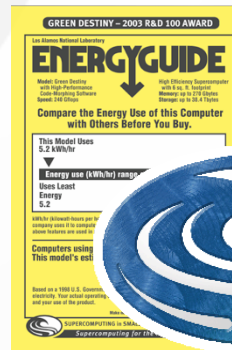
<http://synergy.cs.vt.edu/>



<http://www.chrec.org/>



<http://www.mpiblast.org/>



SUPERCOMPUTING
in SMALL SPACES

<http://sss.cs.vt.edu/>



<http://www.green500.org/>



<http://myvice.cs.vt.edu/>

"Accelerators 'R Us"

<http://accel.cs.vt.edu/>

An Ecosystem for the New HPC: Heterogeneous Parallel Computing

- Deliver personalized supercomputing to the masses
 - Heterogeneity of hardware devices *plus* enabling software that tunes for *performance* (speed & power), *programmability*, and *portability* via a benchmark suite of computational dwarfs and apps

