An Ecosystem for Heterogeneous Parallel Computing

Wu FENG

Dept. of Computer Science and Dept. of Electrical & Computer Engineering

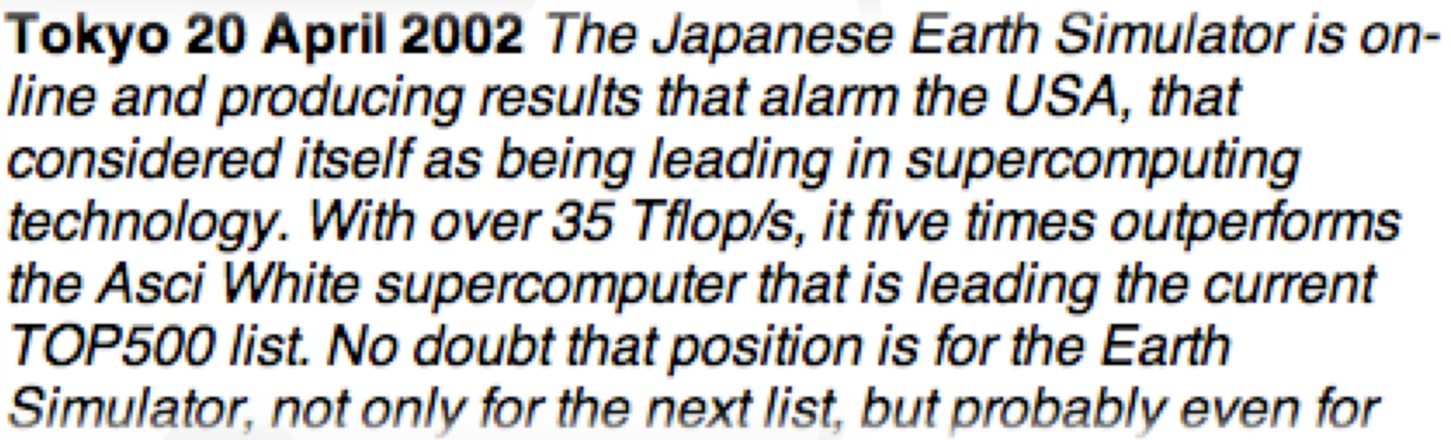
[Dept. of Cancer Biology and Translational Science Institute at Wake Forest University]

VirginiaTech
Invent the Future

POC: wfeng@vt.edu, 540.231.1192

SyNeRGy
synergy.cs.vt.edu

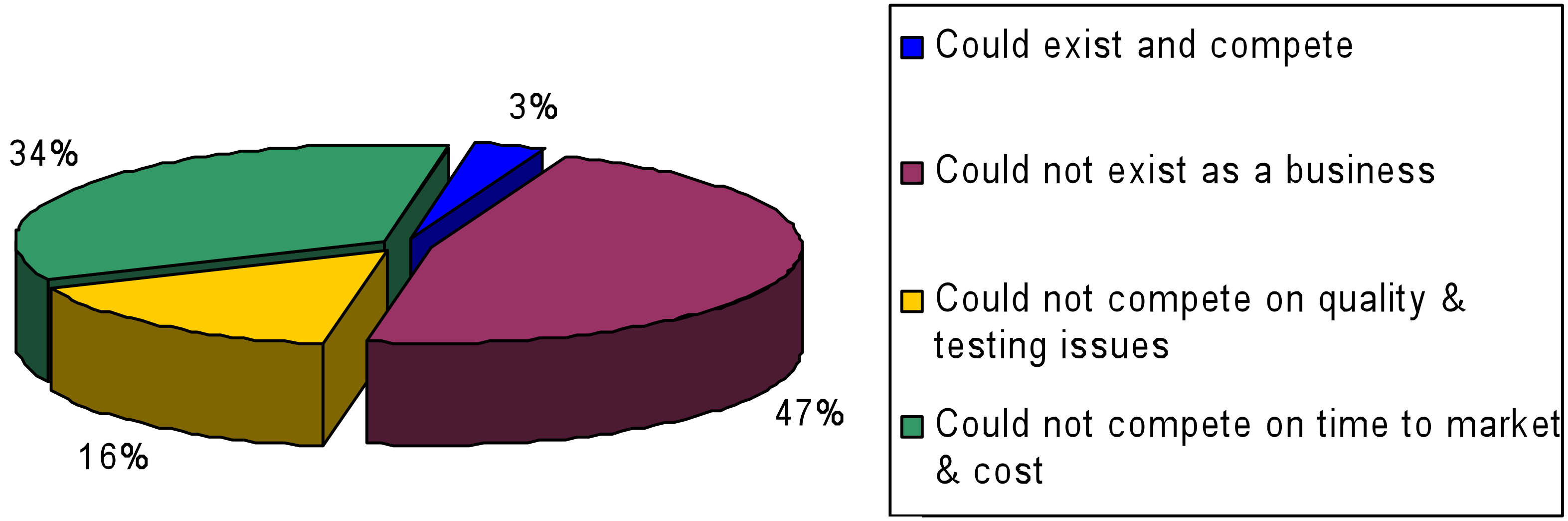
Japanese 'Computnik' Earth Simulator Shatters U.S. Supercomputer Hegemony



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 Asci 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

POC: wfeng@vt.edu, 540.231.1192

Computnik 2.0?


HOME PAGE | TODAY'S PAPER | VIDEO | MOST POPULAR | TIMES TOPICS

The New York Times

Business Day

Technology

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH

Search Technology

Go

Inside Technology

Internet | Start-Ups | Business Computin

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

- 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

VirginiaTech
Invent the Future

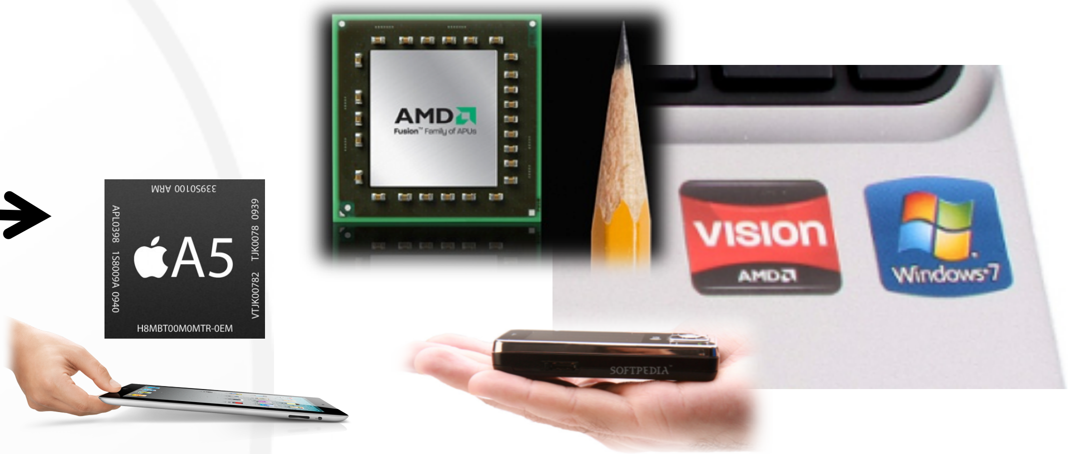
SyNeRG
synergy.cs.vt.edu

Talk Forecast

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

Tianhe-1

→



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

iPad2: 1.5 x 10^9 flops = 1.5 gigaflops
1.5 gigaflops/iPad2 * 9.3M 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

POC: wfeng@vt.edu, 540.231.1192

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. Dobb'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.

Sparse Linear Algebra

Dense Linear Algebra

Spectral Methods

Structured Grids

N-Body Methods

Unstructured Grids

Monte Carlo → MapReduce

and

Combinational Logic
Graph Traversal
Dynamic Programming
Backtrack & Branch+Bound
Graphical Models
Finite State Machine

† 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.

POC: wfeng@vt.edu, 540.231.1192

Project Goal

An Ecosystem for Heterogeneous 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 programmability

via a benchmark suite of computational dwarfs and apps

CHREC
NSF Center for High-Performance Reconfigurable Computing

POC: wfeng@vt.edu, 540.231.1192

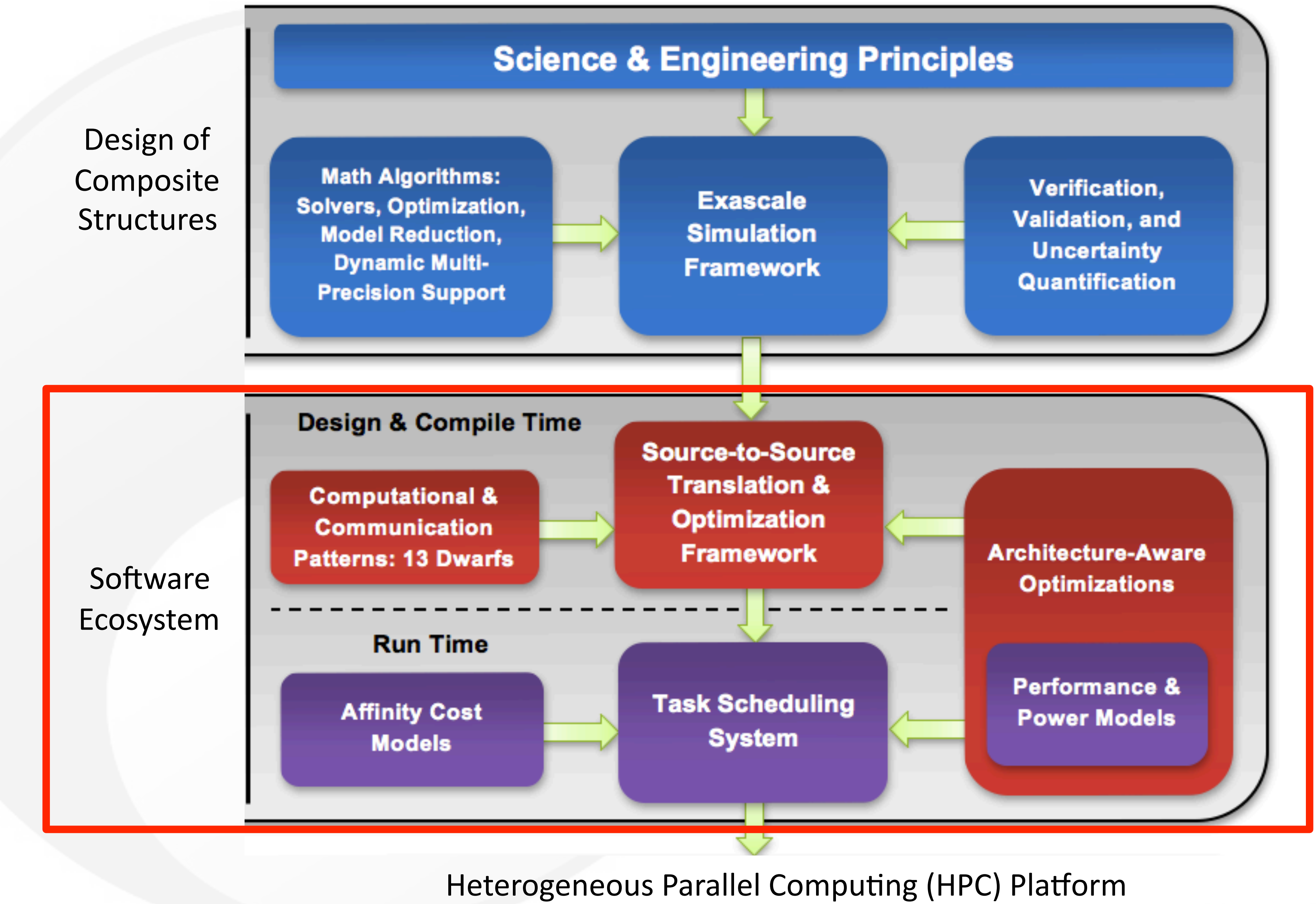
Project Outcome

An Ecosystem for Heterogeneous Computing

- A multi-dimensional understanding of how to optimize ***performance, power, programmability***, or some combination thereof
  - Performance (under Resource Constraints)
    - # threads/block; # blocks/grid; configurable memory; mixed-mode arithmetic; and so on
  - Power
    - Device vs. system power
    - Instantaneous vs. average power consumption
  - Programmability
    - OpenCL vs. CUDA (NVIDIA) vs. Verilog/ImpulseC (Convey)
- What about *portability*?

CHREC
NSF Center for High-Performance Reconfigurable Computing

POC: wfeng@vt.edu, 540.231.1192



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

Software Ecosystem

Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

Architecture-Aware Optimizations

Run Time

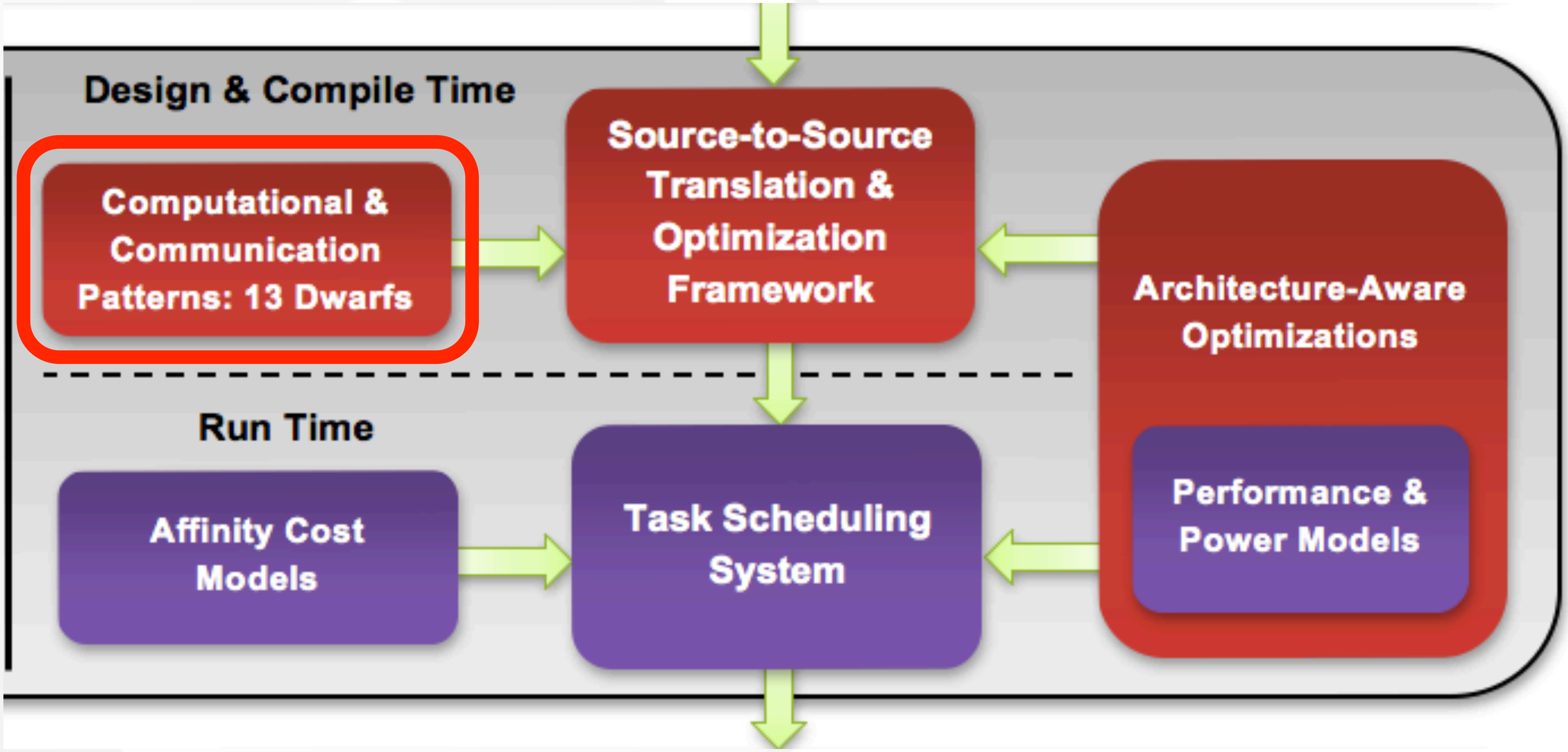
Affinity Cost Models

Task Scheduling System

Performance & Power Models

Heterogeneous Parallel Computing (HPC) Platform

Roadmap



POC: wfeng@vt.edu, 540.231.1192

Project Goal

An Ecosystem for Heterogeneous 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 programmability

  via a benchmark suite of computational dwarfs and apps

- Recall what a **computational dwarf** is
  - ***A pattern of communication & computation***
    … that is common across a set of applications.

CHREC
NSF Center for High-Performance Reconfigurable Computing

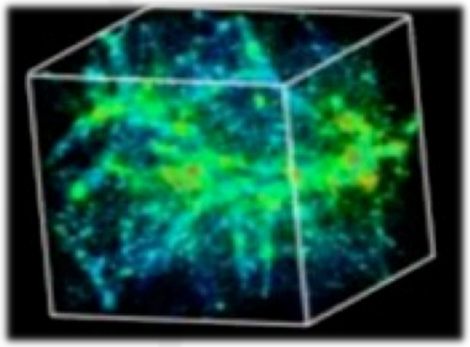
VirginiaTech
Invent the Future

POC: wfeng@vt.edu, 540.231.1192

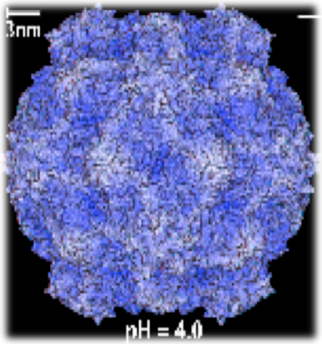
SyNeRG
synergy.cs.vt.edu

Example: N-Body

- N-Body problems are studied in
  - Cosmology
  - Particle physics
  - Biology
  - 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

POC: wfeng@vt.edu, 540.231.1192

What Is "OpenCL & the 13 Dwarfs"?

OpenCL: Open Computing Language

- A framework for writing programs that execute
  … across heterogeneous computing platforms,
  … consisting of CPUs, GPUs, or other processors.

KHRONOS GROUP

The 13 Dwarfs

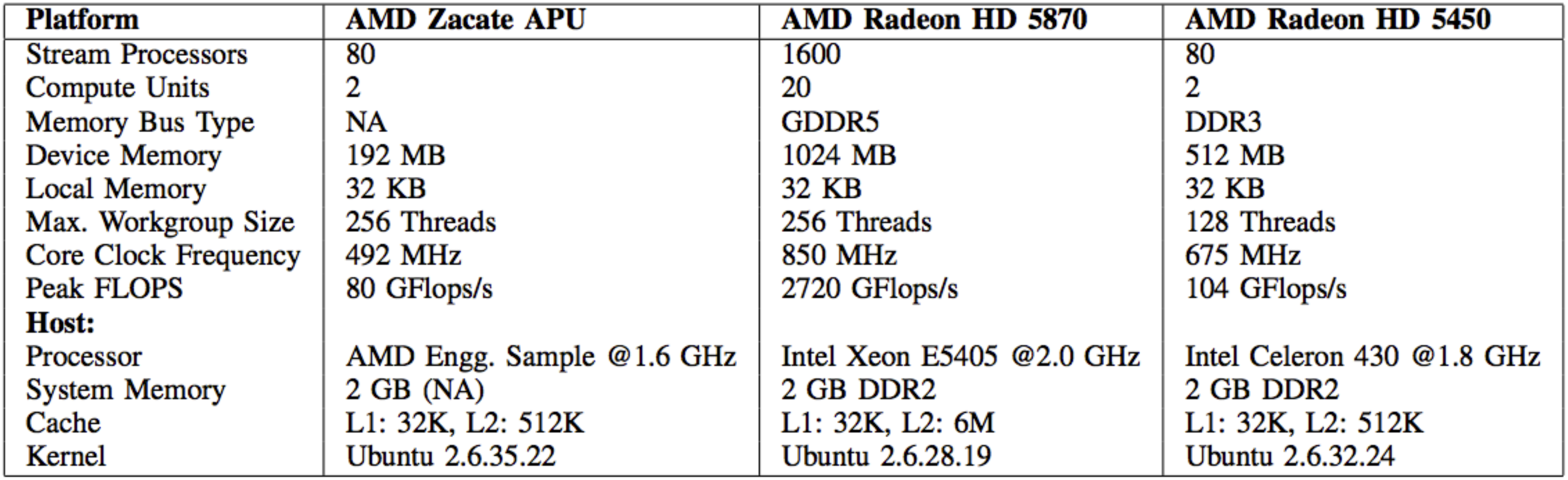
- Original 7
  - Dense linear algebra (e.g. dense matrix multiply)
  - Sparse linear algebra (e.g. sparse matrix solvers)
  - Spectral methods (e.g. FFT)
  - N-Body methods (e.g. gravity simulations)
  - Structured grids (e.g. PDEs)
  - Unstructured grids (e.g., irregular grids)
  - MapReduce (e.g., data parallelism & combination)
- 6 More Dwarfs
  - Combinational logic
  - Graph traversal
  - Dynamic programming
  - Backtrack & branch-and-bound
  - Graphical models
  - Finite state machines

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	RRU (RoadRunner Universe)
Structured grids	SRAD	
Unstructured grids	CFD Solver	
MapReduce		PSICL-BLAST, StreamMR
Combinational logic	CRC	
Graph traversal	BFS, Bitonic Sort	
Dynamic programming	Needleman-Wunsch	
Backtrack and Branch-and-Bound		N-Queens, Traveling Salesman
Graphical models	Hidden Markov Model	
Finite state machines	Temporal Data Mining	

The Future is Fusion

Experimental Set-Up: Machines and Workload

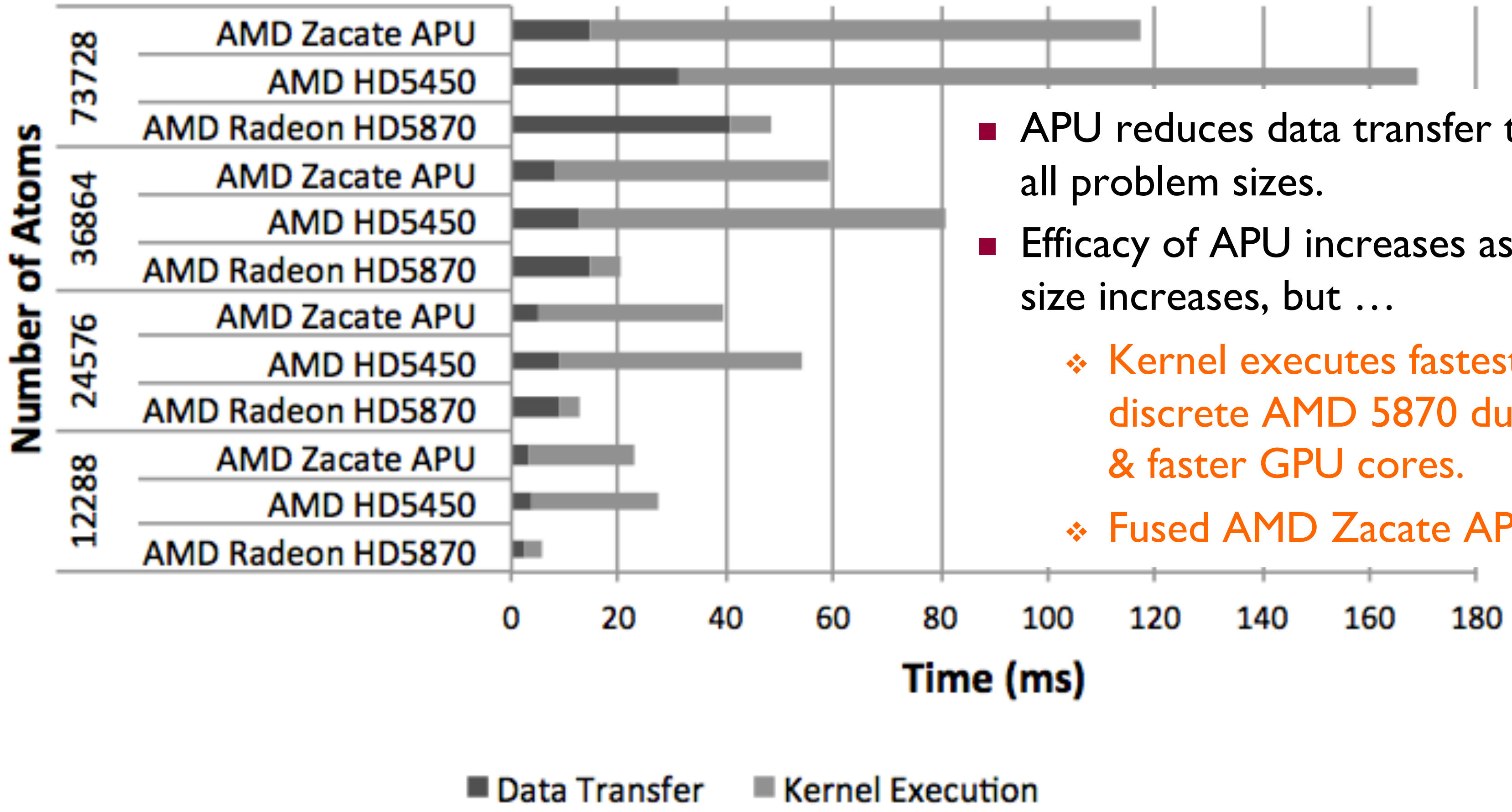


Platform	AMD Zacate APU	AMD Radeon HD 5870	AMD Radeon HD 5450
Stream Processors	80	1600	80
Compute Units	2	20	2
Memory Bus Type	NA	GDDR5	DDR3
Device Memory	192 MB	1024 MB	512 MB
Local Memory	32 KB	32 KB	32 KB
Max. Workgroup Size	256 Threads	256 Threads	128 Threads
Core Clock Frequency	492 MHz	850 MHz	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.

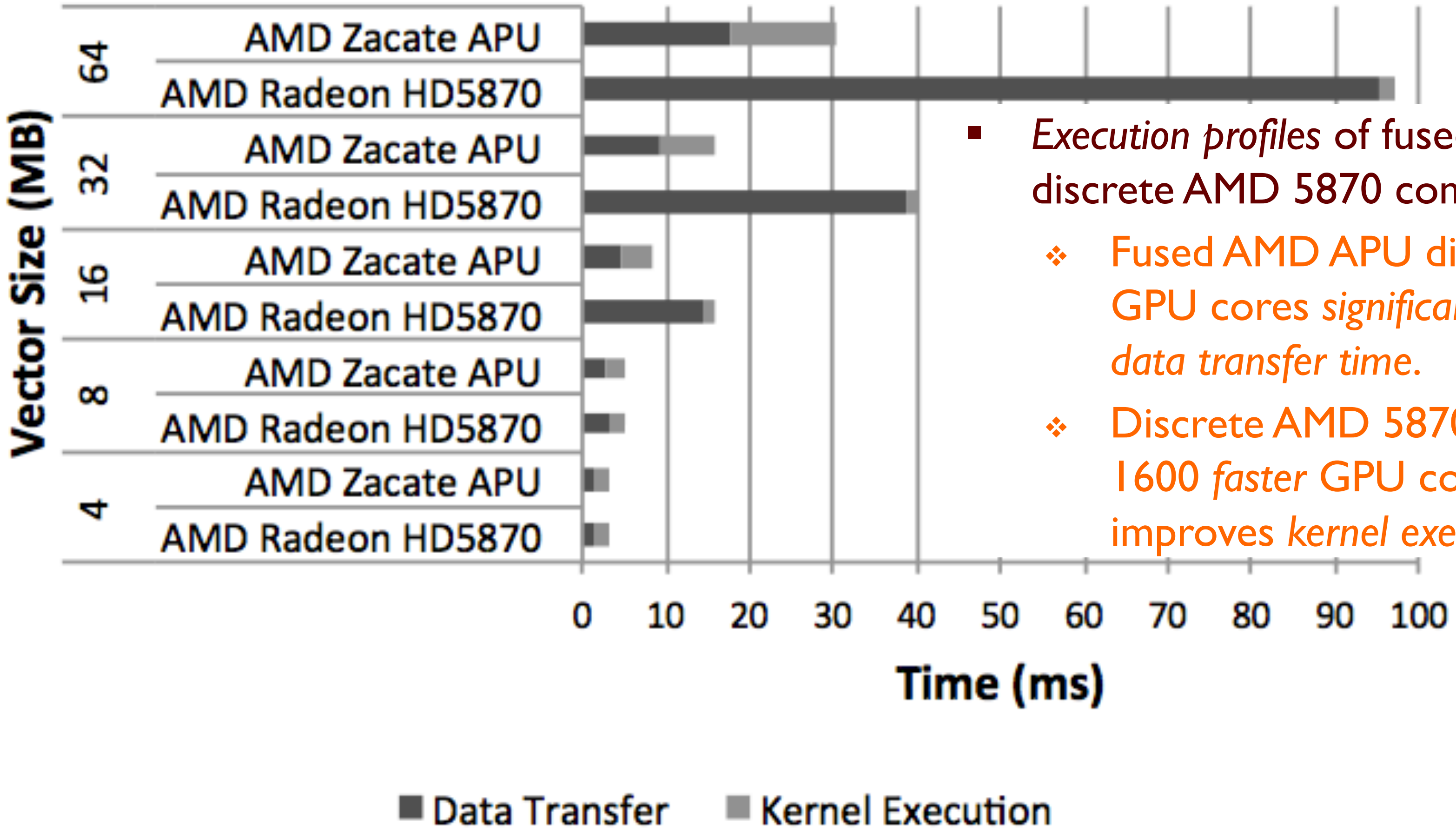
Performance: Molecular Modeling (N-Body)



- APU reduces data transfer times for all problem sizes.
- Efficacy of APU increases as problem size increases, but …
  - Kernel executes fastest on discrete AMD 5870 due to more & faster GPU cores.
  - Fused AMD Zacate APU next.

POC: wfeng@vt.edu, 540.231.1192

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.*

VirginiaTech
Invent the Future

POC: wfeng@vt.edu, 540.231.1192

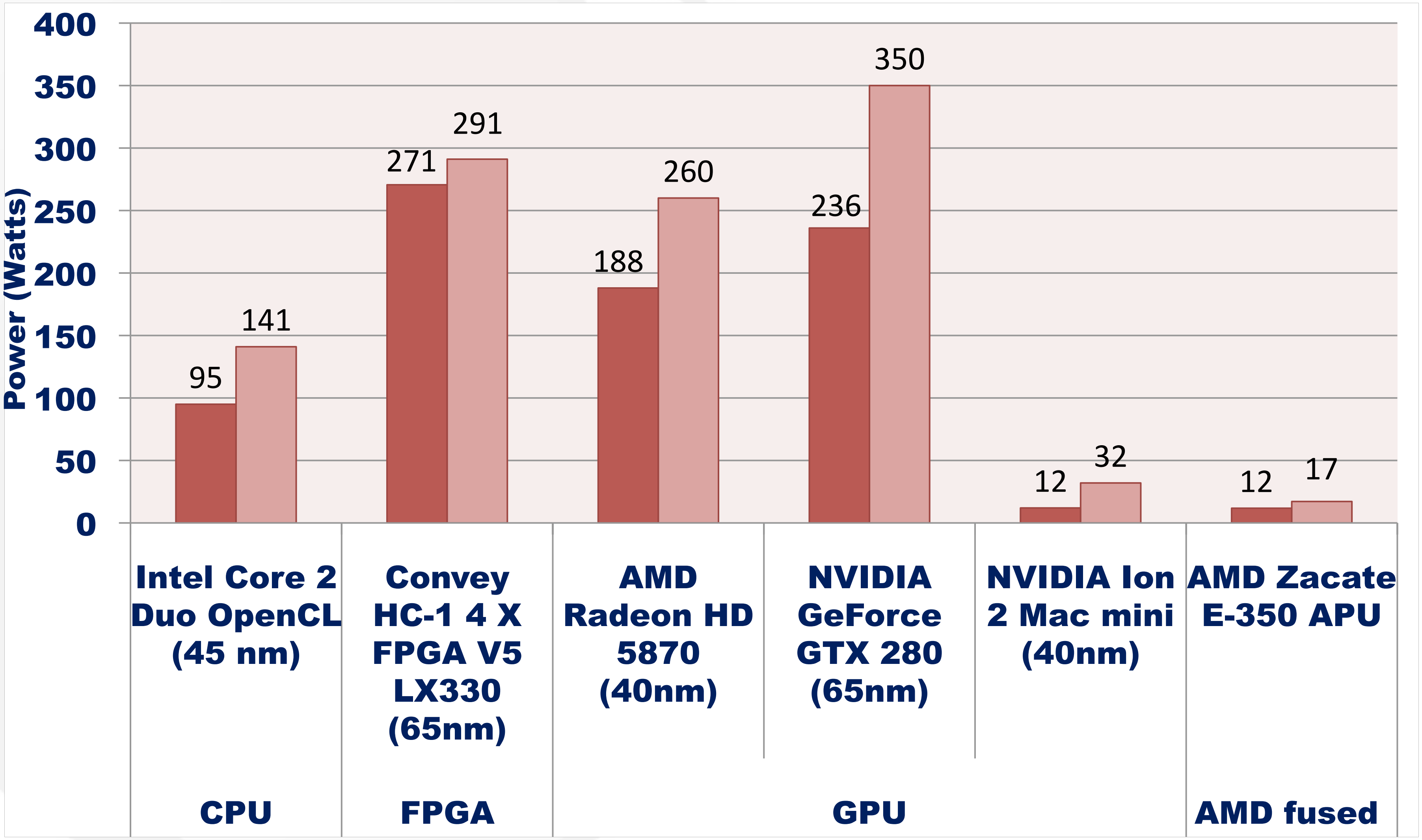
SyNeRGy
synergy.cs.vt.edu

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

```
mdaga@Fusion ~/shoc-1.0.2/bin/Serial/OpenCL $ ./MD -s 3 -n 2
Chose device: name='Loveland' index=0 id=0x859fa0
Initializing test problem (this can take several minutes for large pro
blems).
```

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



POC: wfeng@vt.edu, 540.231.1192

Status of OpenCL & the 13 Dwarfs

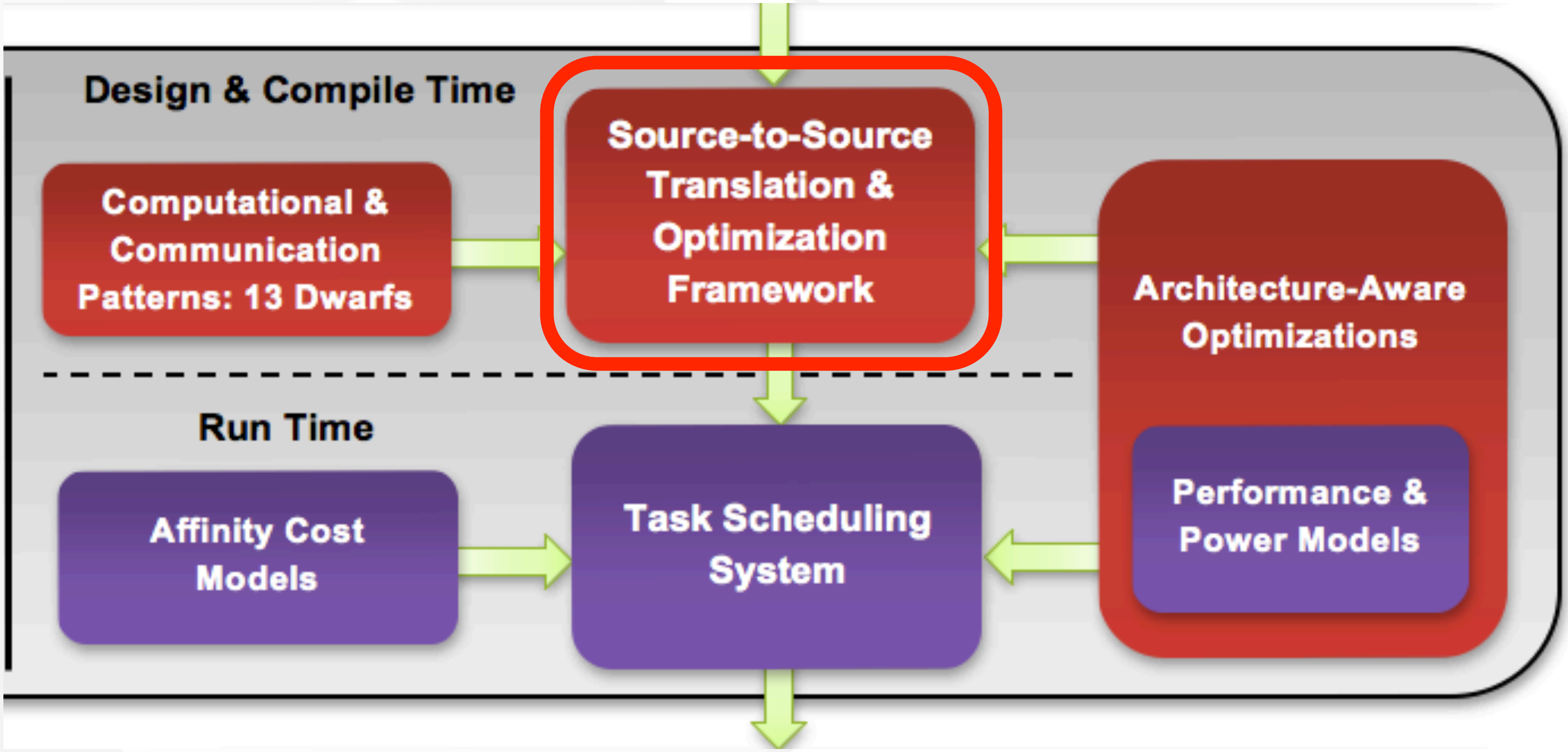
2009 – 2011

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	BFS, Bitonic sort
Dynamic programming	Needleman-Wunsch
Backtrack and Branch-and-Bound	
Graphical models	Hidden Markov Model
Finite state machines	Temporal Data Mining

88x → 371x

POC: wfeng@vt.edu, 540.231.1192

Roadmap



POC: wfeng@vt.edu, 540.231.1192

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*, Dec. 2011 (to appear). Also available as a technical report from CS at Virginia Tech: TR-11-13.

Why CU2CL?

- Much larger # of apps implemented in CUDA than in OpenCL
  - Idea
    - Leverage scientists' investment in CUDA to drive OpenCL adoption
  - Issues (from the perspective of *domain* scientists)
    - Writing from Scratch: Learning Curve
      (OpenCL is too low-level an API compared to CUDA. CUDA also low level.)
    - ***Porting from CUDA: Tedious and Error-Prone***

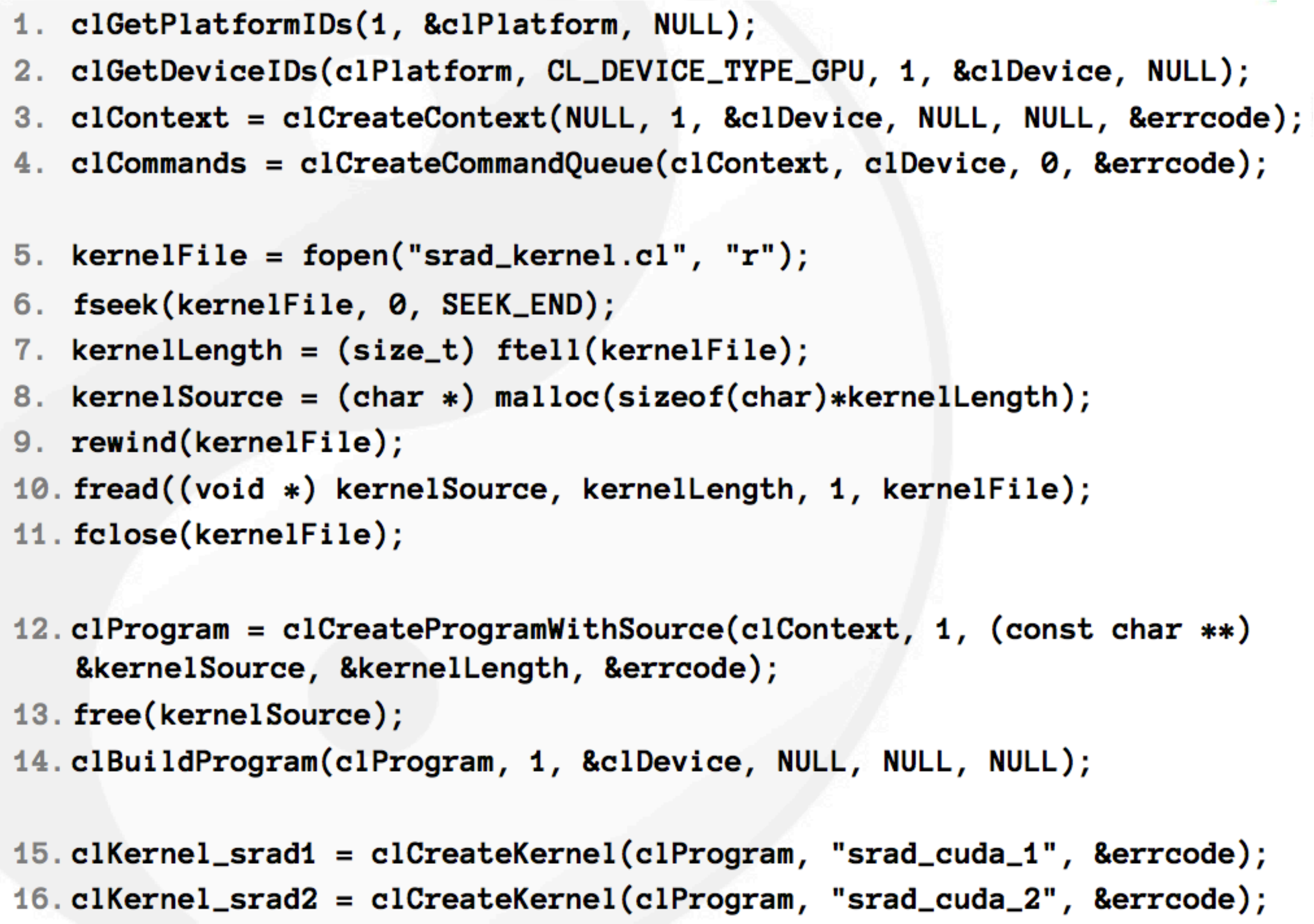
POC: wfeng@vt.edu, 540.231.1192

Initialization Code: CUDA

None!

Initialization Code: OpenCL

OpenCL



```
clGetPlatformIDs(1, &clPlatform, NULL);
clGetDeviceIDs(clPlatform, CL_DEVICE_TYPE_GPU, 1, &clDevice, NULL);
clContext = clCreateContext(NULL, 1, &clDevice, NULL, NULL, &errcode);
clCommands = clCreateCommandQueue(clContext, clDevice, 0, &errcode);

kernelFile = fopen("srad_kernel.cl", "r");
fseek(kernelFile, 0, SEEK_END);
kernelLength = (size_t) ftell(kernelFile);
kernelSource = (char *) malloc(sizeof(char)*kernelLength);
rewind(kernelFile);
fread((void *) kernelSource, kernelLength, 1, kernelFile);
fclose(kernelFile);

clProgram = clCreateProgramWithSource(clContext, 1, (const char **)
 &kernelSource, &kernelLength, &errcode);
free(kernelSource);
clBuildProgram(clProgram, 1, &clDevice, NULL, NULL, NULL);

clKernel_srad1 = clCreateKernel(clProgram, "srad_cuda_1", &errcode);
clKernel_srad2 = clCreateKernel(clProgram, "srad_cuda_2", &errcode);
```

Executing Device Code: CUDA

```
dim3 dimBlock(BLOCK_SIZE, BLOCK_SIZE);

for (int i=0; i < matrix_dim-BLOCK_SIZE; i += BLOCK_SIZE)
{
   lud_diagonal<<<1, BLOCK_SIZE>>>(m, matrix_dim, i);
   lud_perimeter<<<(matrix_dim-i)/BLOCK_SIZE-1, BLOCK_SIZE*2>>>(m, matrix_dim, i);
   dim3 dimGrid((matrix_dim-i)/BLOCK_SIZE-1, (matrix_dim-i)/BLOCK_SIZE-1);
   lud_internal<<<dimGrid, dimBlock>>>(m, matrix_dim, i);
}
lud_diagonal<<<1,BLOCK_SIZE>>>(m, matrix_dim, i);
```

Executing Device Code: OpenCL

OpenCL

```
size_t localWorkSize[2];
size_t globalWorkSize[2];

for (int i=0; i < matrix_dim-BLOCK_SIZE; i += BLOCK_SIZE)
{
   clSetKernelArg(clKernel_diagonal, 0, sizeof(cl_mem), (void *) &d_m);
   clSetKernelArg(clKernel_diagonal, 1, sizeof(int), (void *) &matrix_dim);
   clSetKernelArg(clKernel_diagonal, 2, sizeof(int), (void *) &i);
   localWorkSize[0] = BLOCK_SIZE;
   globalWorkSize[0] = BLOCK_SIZE;
   clEnqueueNDRangeKernel(clCommands, clKernel_diagonal, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, NULL);
... (14 more lines)
}
clSetKernelArg(clKernel_diagonal, 0, sizeof(cl_mem), (void *) &d_m);
clSetKernelArg(clKernel_diagonal, 1, sizeof(int), (void *) &matrix_dim);
clSetKernelArg(clKernel_diagonal, 2, sizeof(int), (void *) &i);
localWorkSize[0] = BLOCK_SIZE;
globalWorkSize[0] = BLOCK_SIZE;
clEnqueueNDRangeKernel(clCommands, clKernel_diagonal, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, NULL);
```

Why CU2CL?

- Much larger # of apps implemented in CUDA than in OpenCL
  - Idea
    - Leverage scientists' investment in CUDA to drive OpenCL adoption
  - Issues (from the perspective of *domain* scientists)
    - Writing from Scratch: Learning Curve (OpenCL is too low-level an API compared to CUDA. CUDA also low level.)
    - Porting from CUDA: Tedious and Error-Prone
- Significant demand from major stakeholders

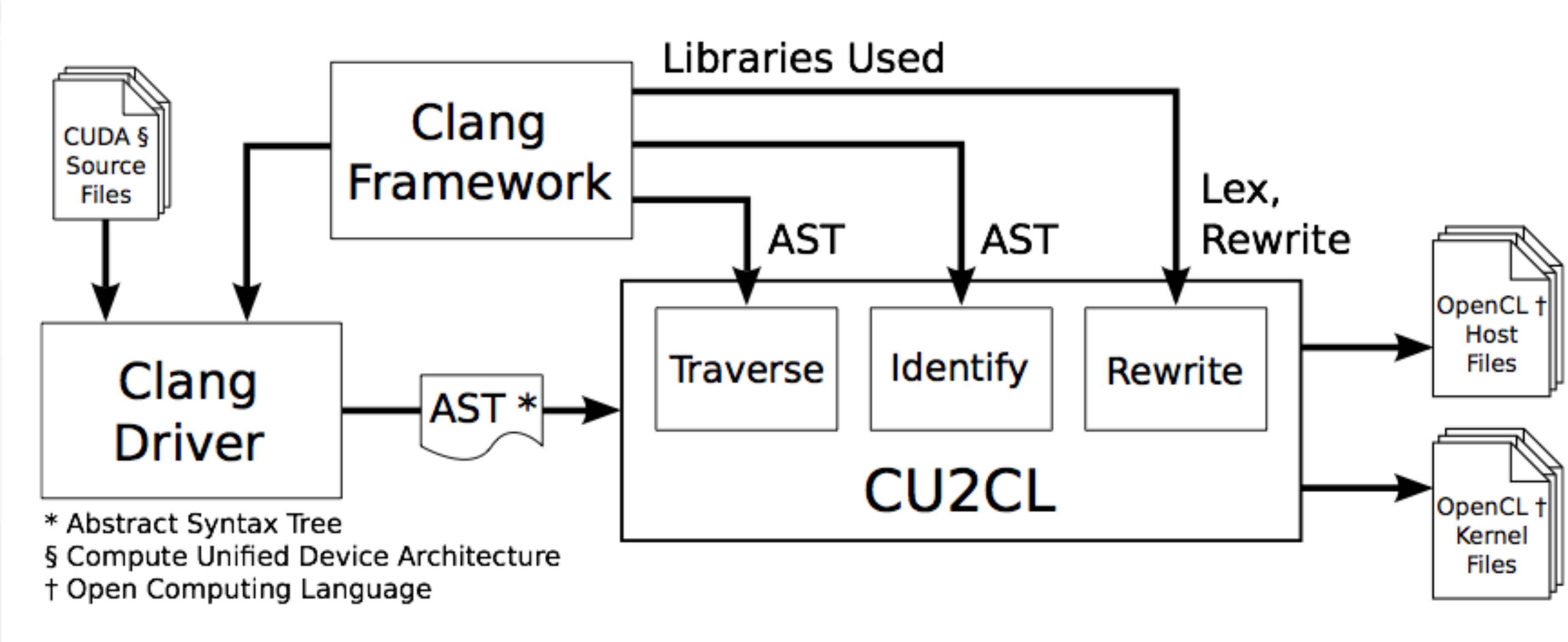
Why *Not* CU2CL?

- Just start with OpenCL?!
- CU2CL only does *source-to-source translation* at present
  - No architecture-aware optimization
  - CUDA and OpenCL version compatibility

At odds …

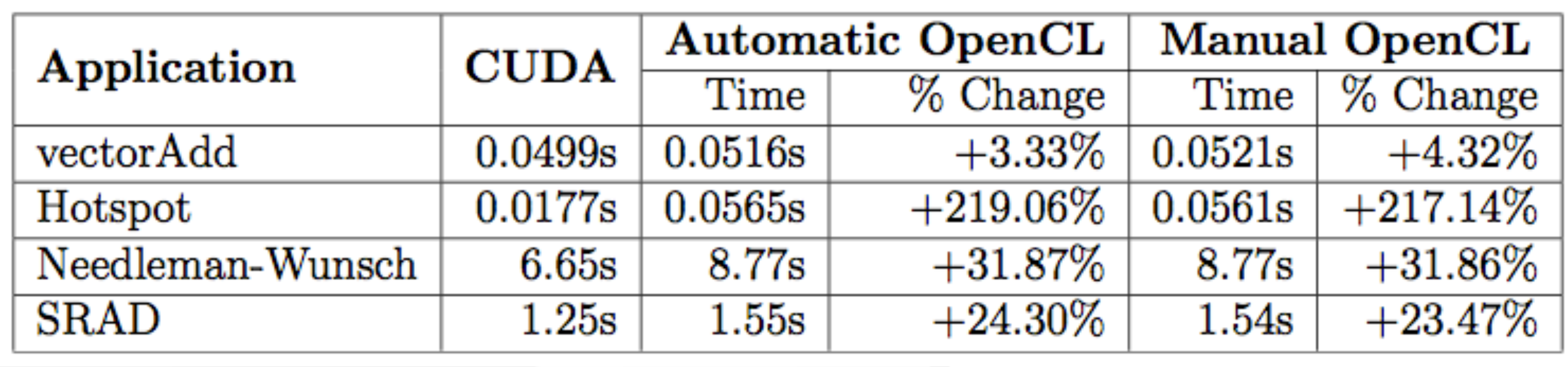
POC: wfeng@vt.edu, 540.231.1192

Overview of CU2CL Translation



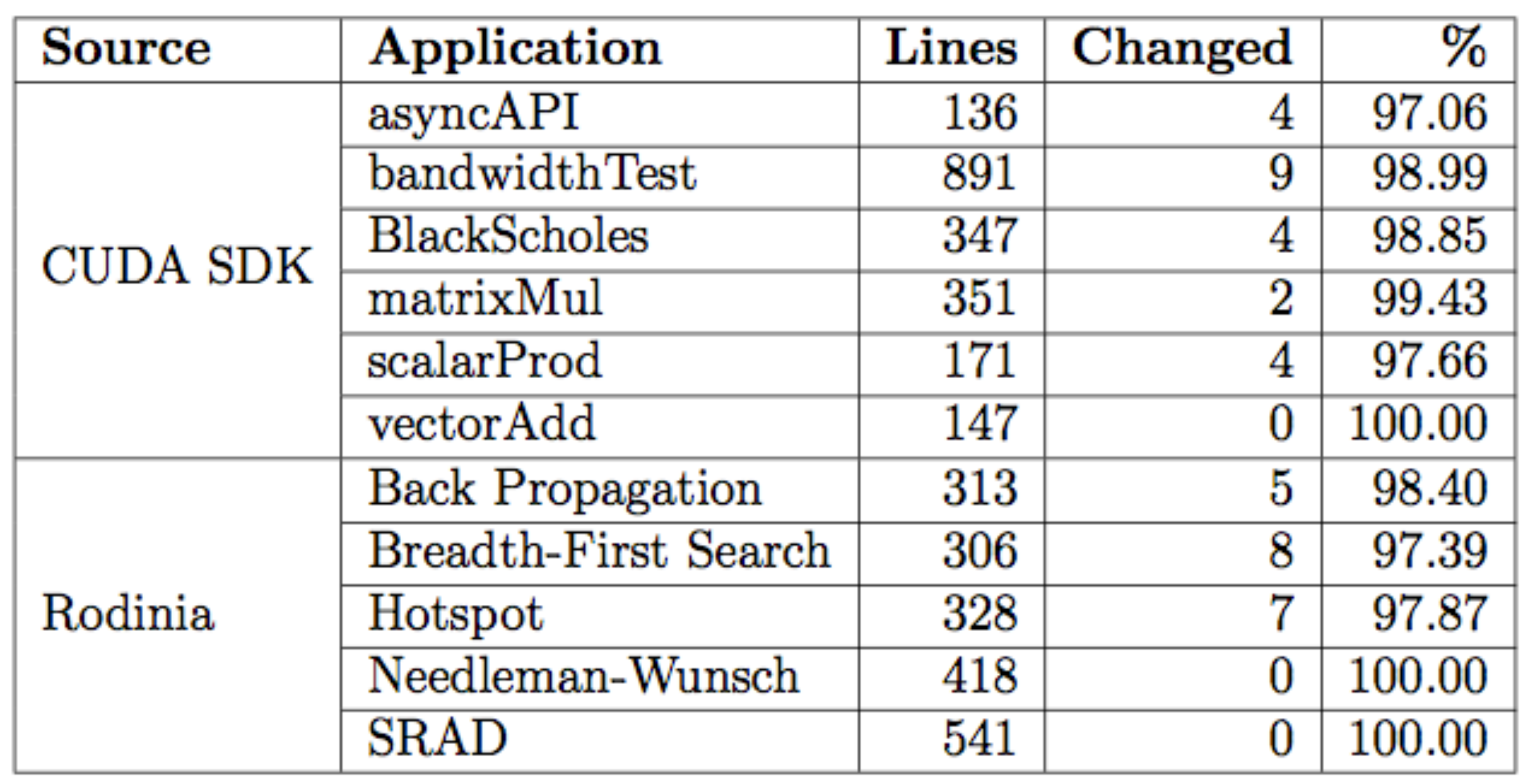
"CU2CL: A CUDA-to-OpenCL Translator for Multi- and Many-core Architectures," *17th IEEE Int'l Conf. on Parallel & Distributed Systems*, Dec. 2011 (to appear). Also available as a technical report from CS at Virginia Tech: TR-11-13.

Performance of CU2CL-Translated Apps



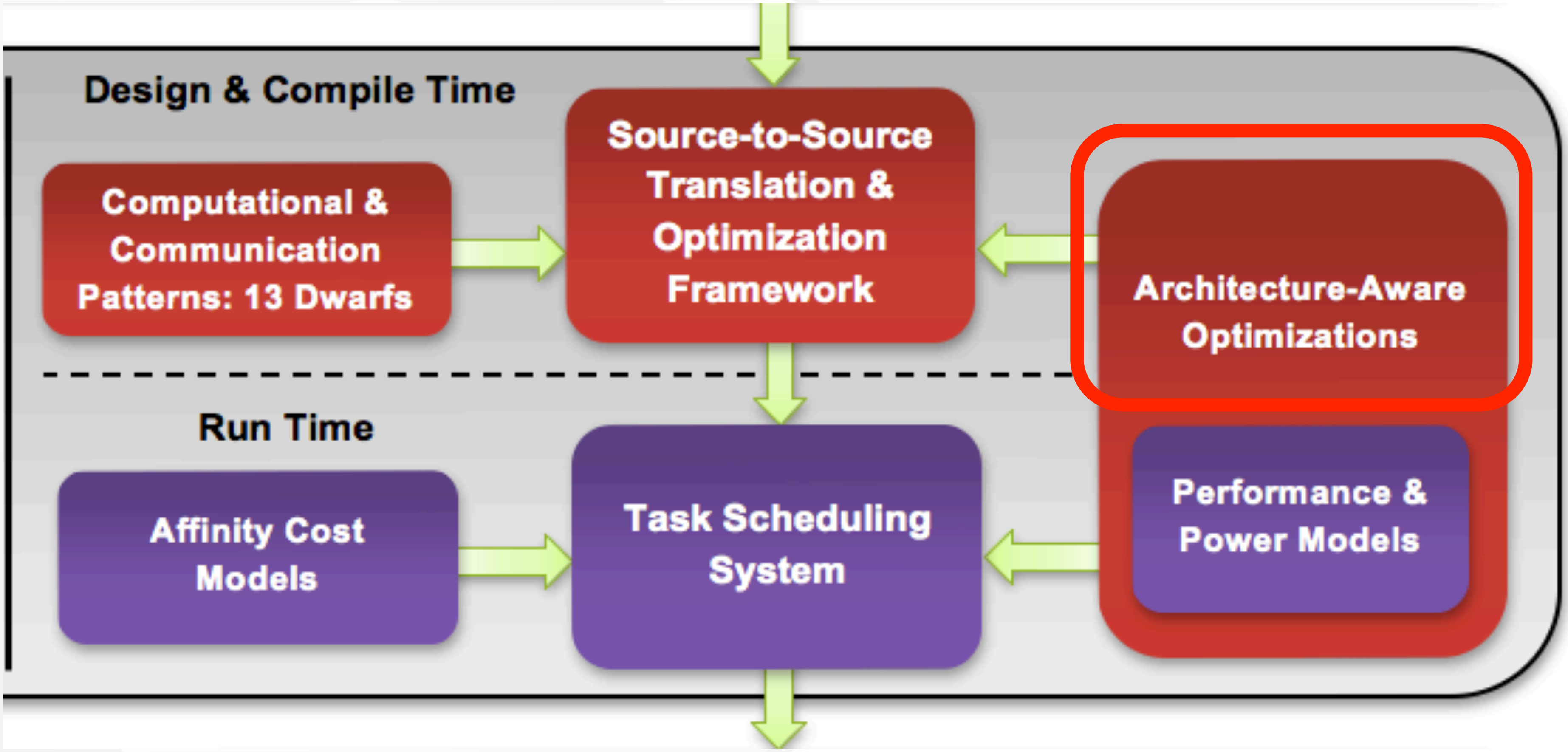
Application	**CUDA**	**Automatic OpenCL**		**Manual OpenCL**	
		Time	% Change	Time	% Change
vectorAdd	0.0499s	0.0516s	+3.33%	0.0521s	+4.32%
Hotspot	0.0177s	0.0565s	+219.06%	0.0561s	+217.14%
Needleman-Wunsch	6.65s	8.77s	+31.87%	8.77s	+31.86%
SRAD	1.25s	1.55s	+24.30%	1.54s	+23.47%

CU2CL Coverage



Source	Application	Lines	Changed	%
CUDA SDK	asyncAPI	136	4	97.06
	bandwidthTest	891	9	98.99
	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
	Breadth-First Search	306	8	97.39
	Hotspot	328	7	97.87
	Needleman-Wunsch	418	0	100.00
	SRAD	541	0	100.00

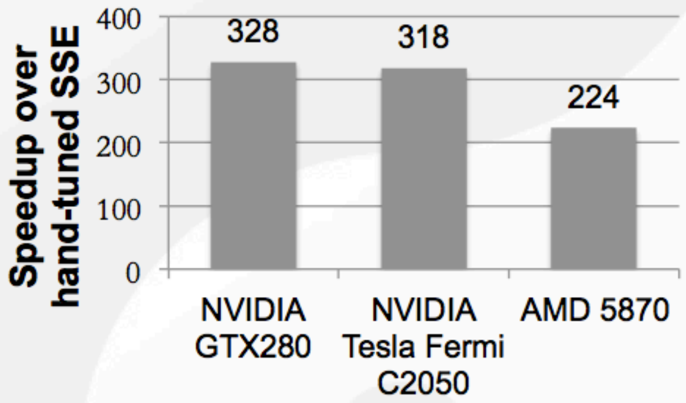
Roadmap



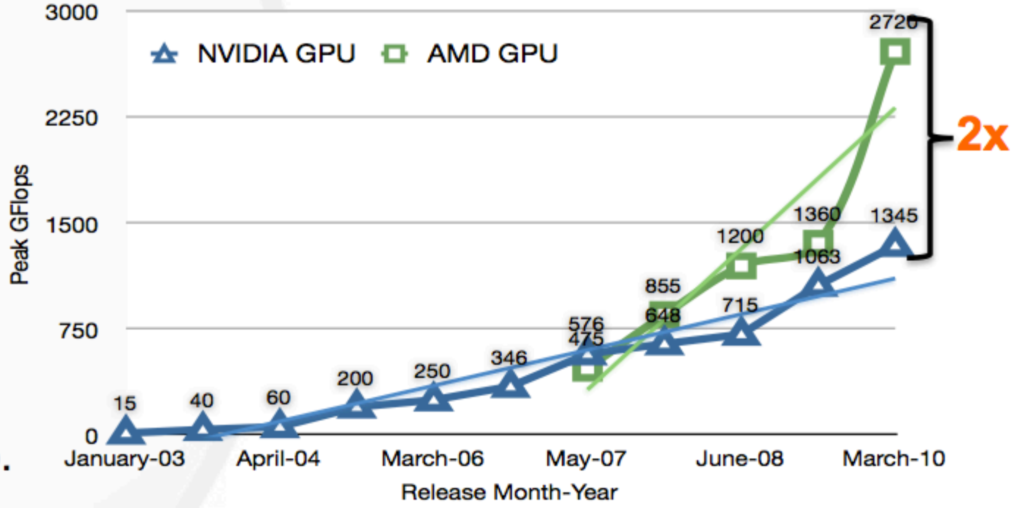
POC: wfeng@vt.edu, 540.231.1192

Computational Units *Not* Created Equal

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



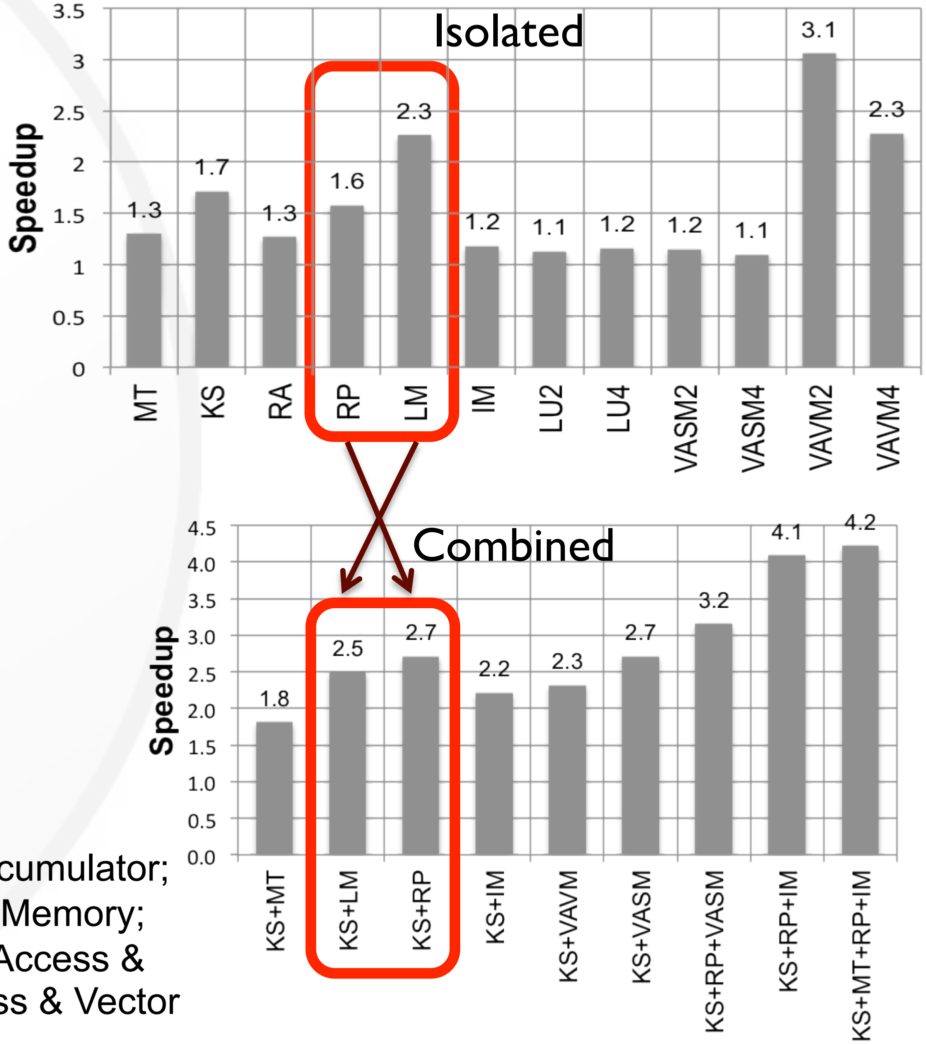
Performance of a Molecular Modeling App.



Architecture-Aware Optimization

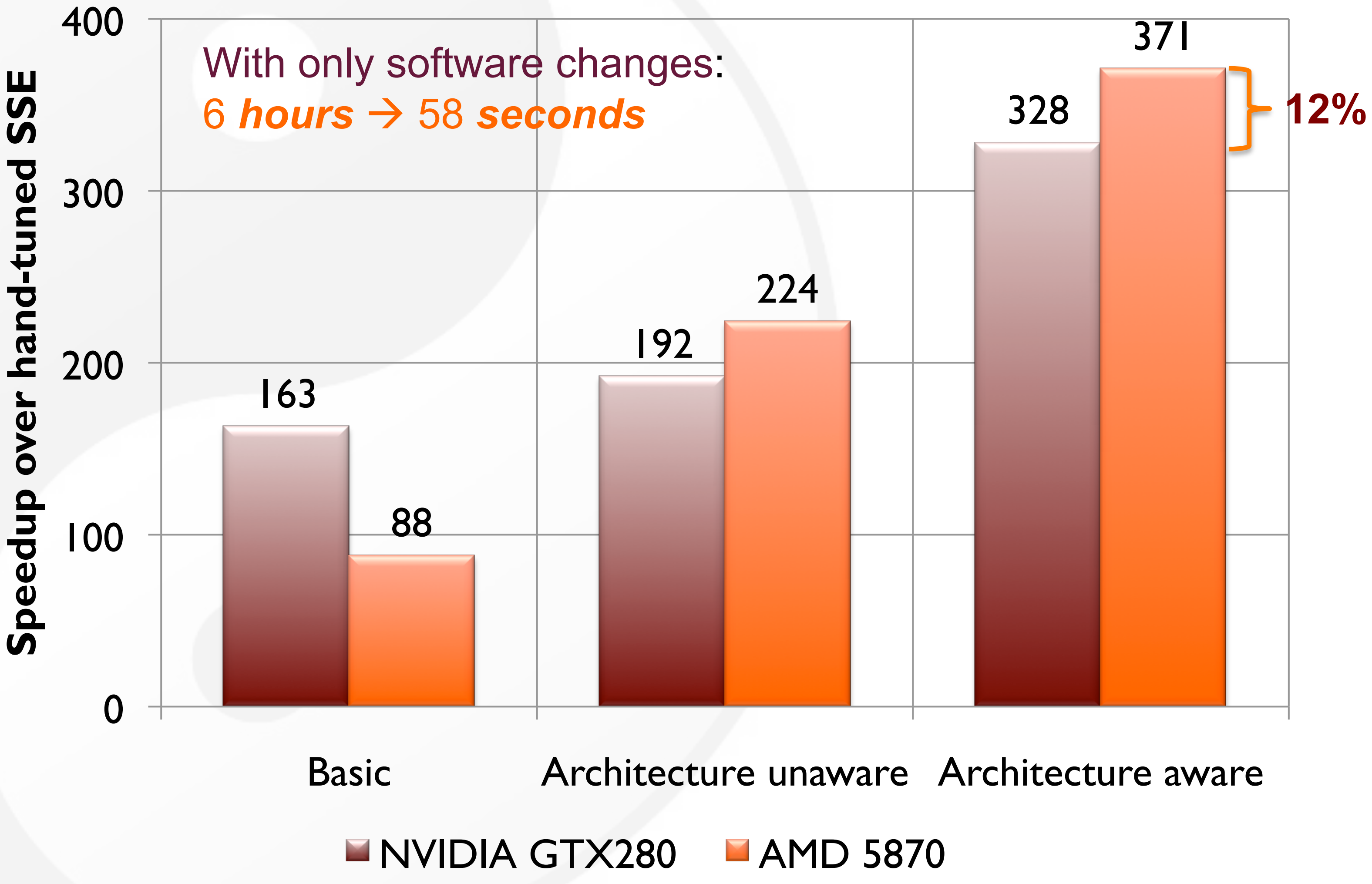
(N-body Code for Molecular Modeling)

- 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}

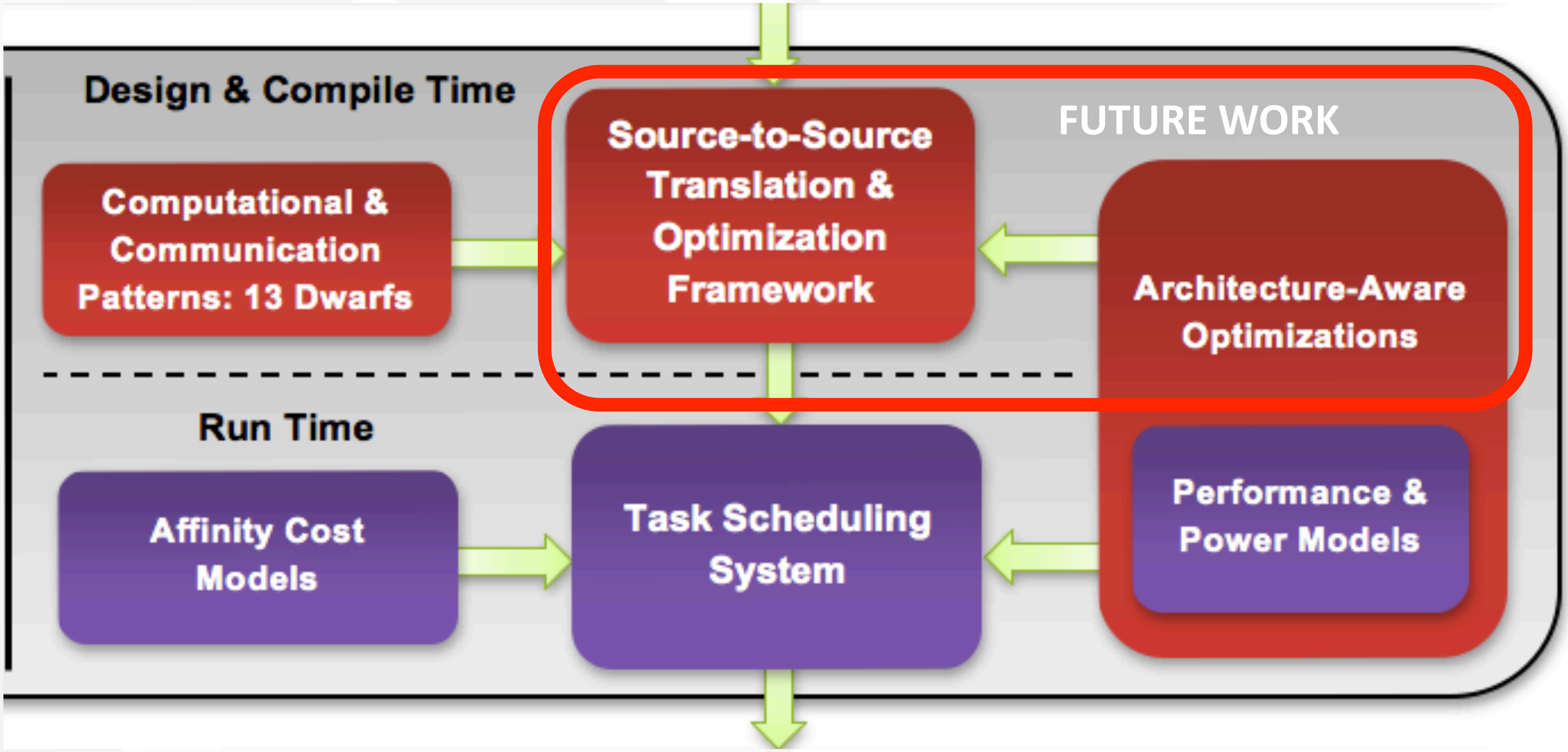
Summary: Architecture-Aware Optimization



With only software changes:
6 ***hours*** → 58 ***seconds***

POC: wfeng@vt.edu, 540.231.1192

Roadmap



Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

FUTURE WORK

Architecture-Aware Optimizations

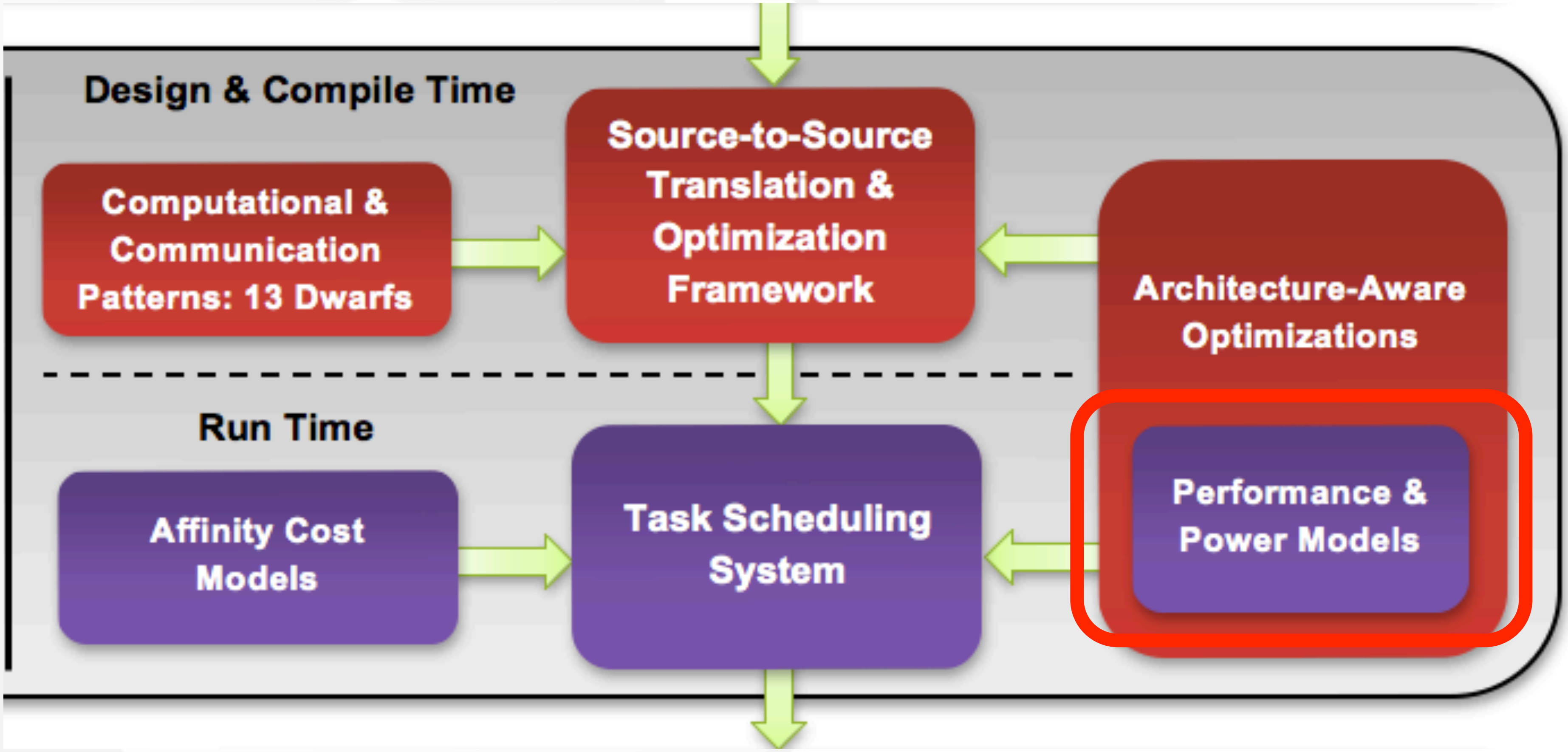
Run Time

Affinity Cost Models

Task Scheduling System

Performance & Power Models

Roadmap



Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

Architecture-Aware Optimizations

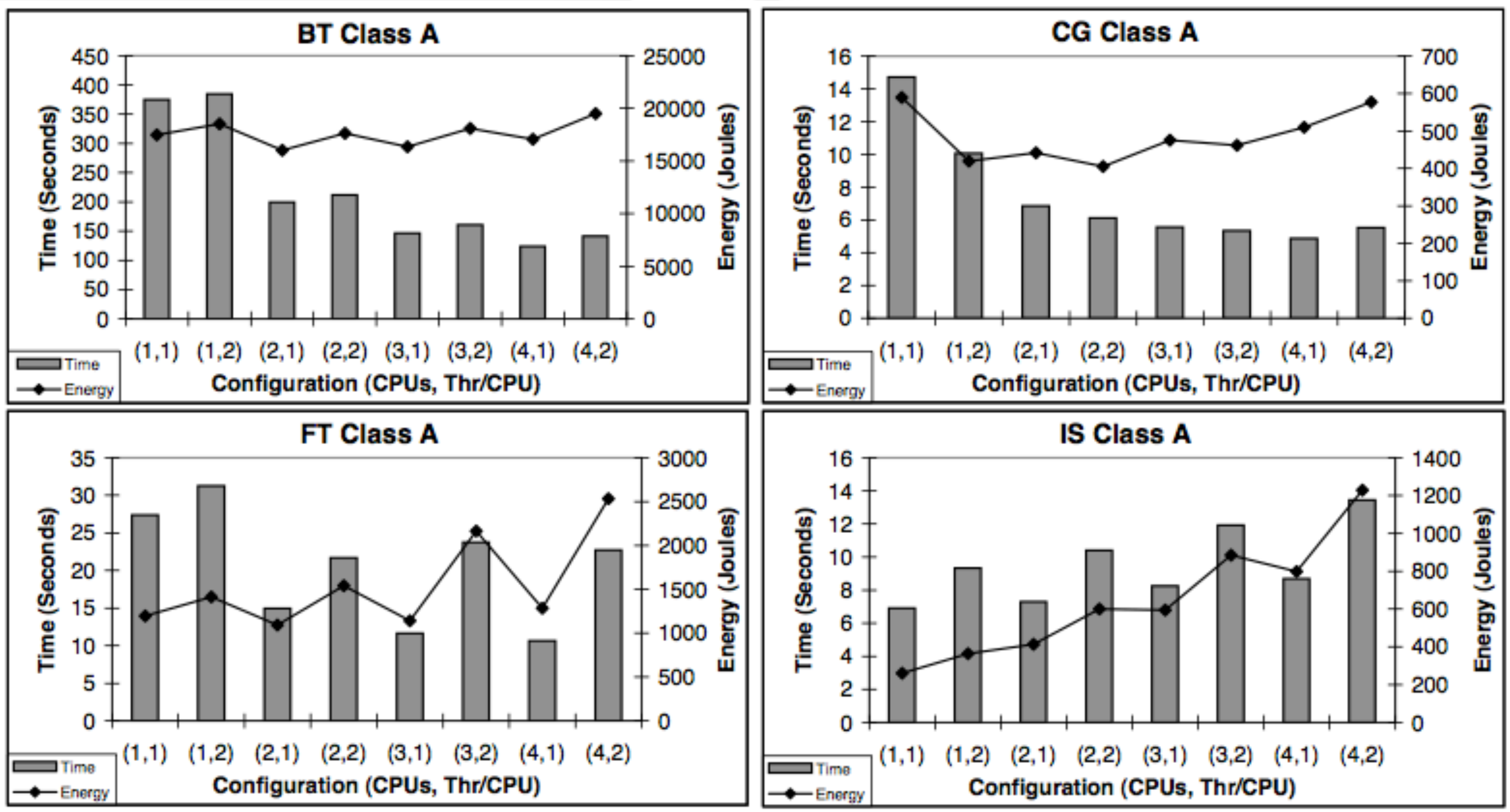
Run Time

Affinity Cost Models

Task Scheduling System

Performance & Power Models

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

POC: wfeng@vt.edu, 540.231.1192

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 $(t_l^*, \ldots, t_n^*)$ 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

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

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

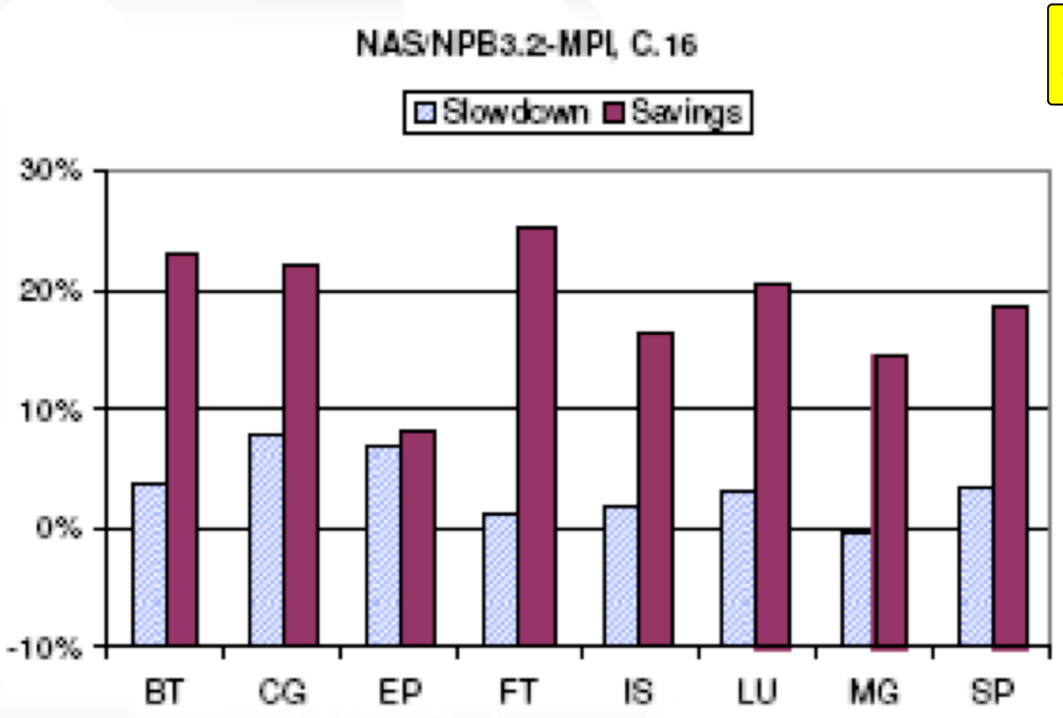
$$\beta = \frac{W_{cpu}}{W_{cpu} + T_{mem} \cdot f_{max}}$$

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

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

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

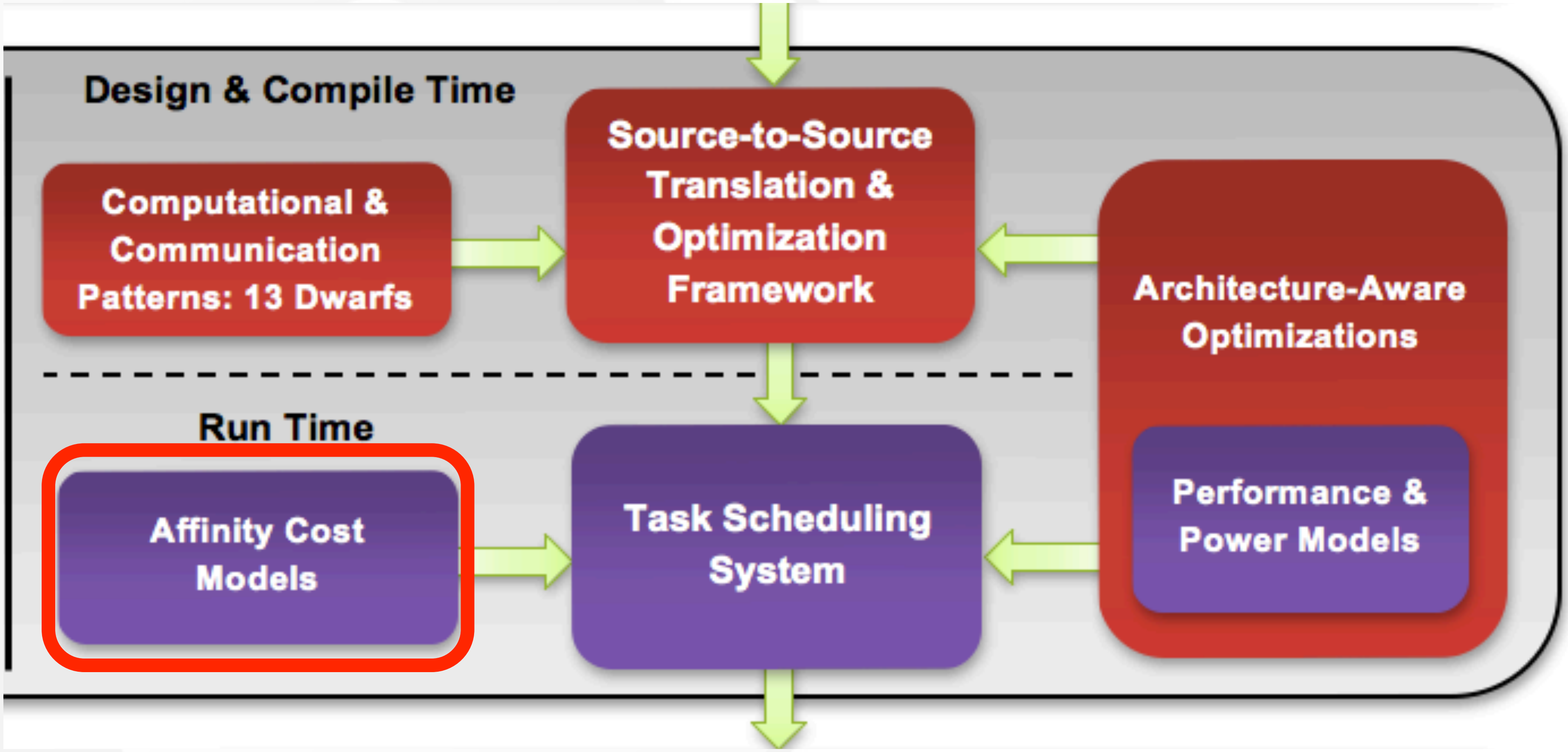
NAS Parallel on an AMD Opteron Cluster



Energy reduction (15%)
Performance improvement

"A Power-Aware Run-Time System for High-Performance Computing," *SC|05,* Nov. 2005

Roadmap



Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

Architecture-Aware Optimizations

Run Time

Affinity Cost Models

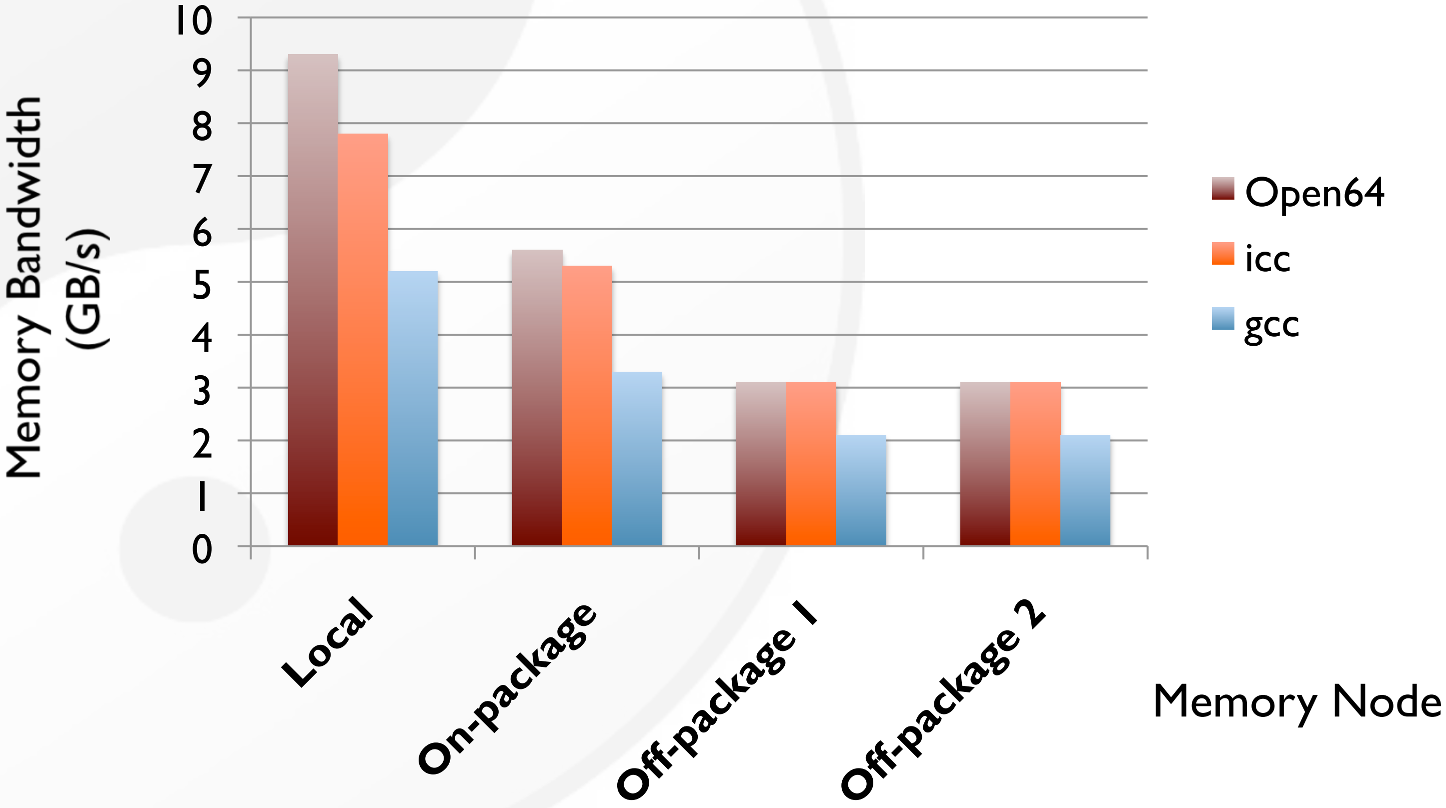
Task Scheduling System

Performance & Power Models

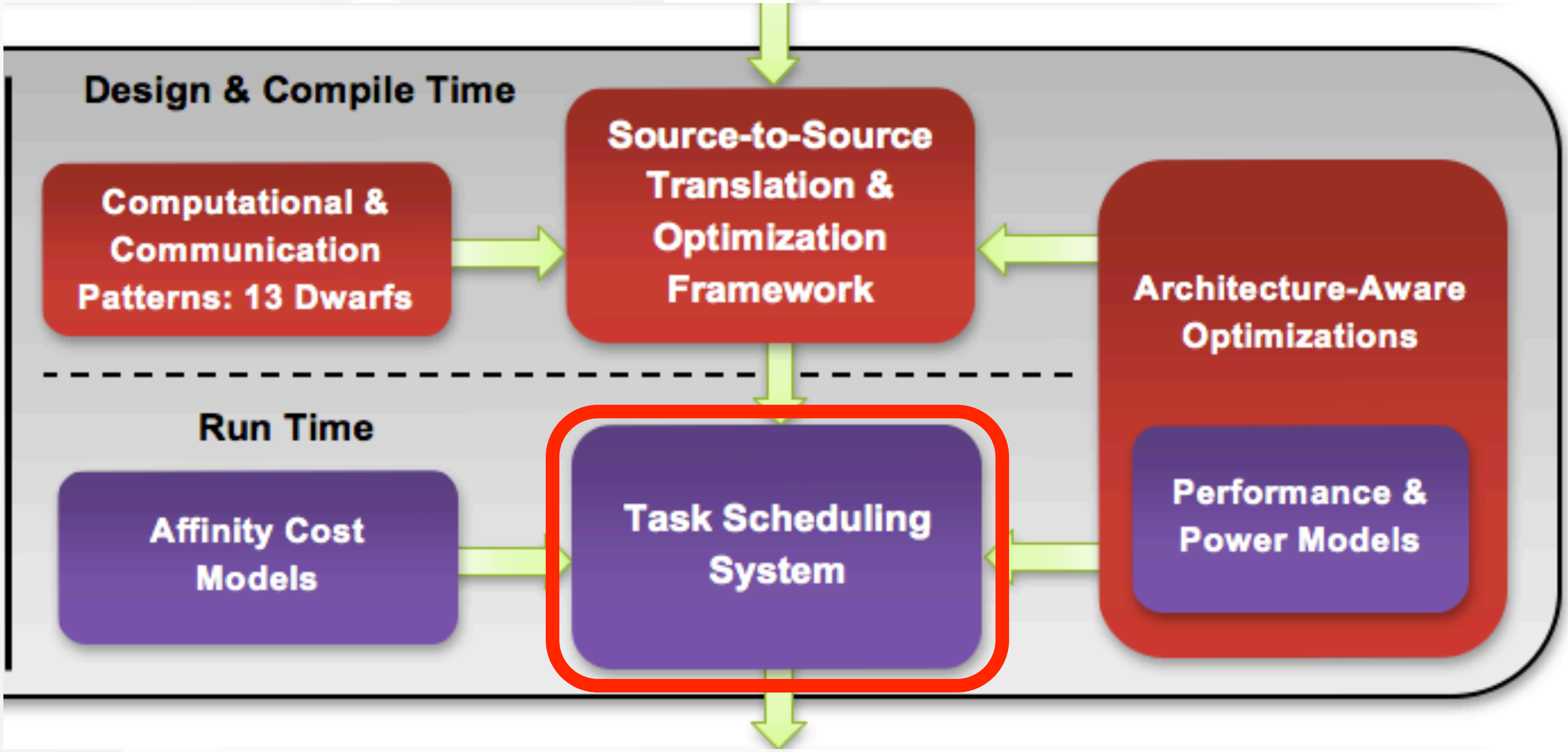
POC: wfeng@vt.edu, 540.231.1192

Affinity Results → Cost Modeling

- Goal
  - Performance cost modeling to drive heterogeneous task scheduling



Roadmap



Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

Architecture-Aware Optimizations

Run Time

Affinity Cost Models

Task Scheduling System

Performance & Power Models

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

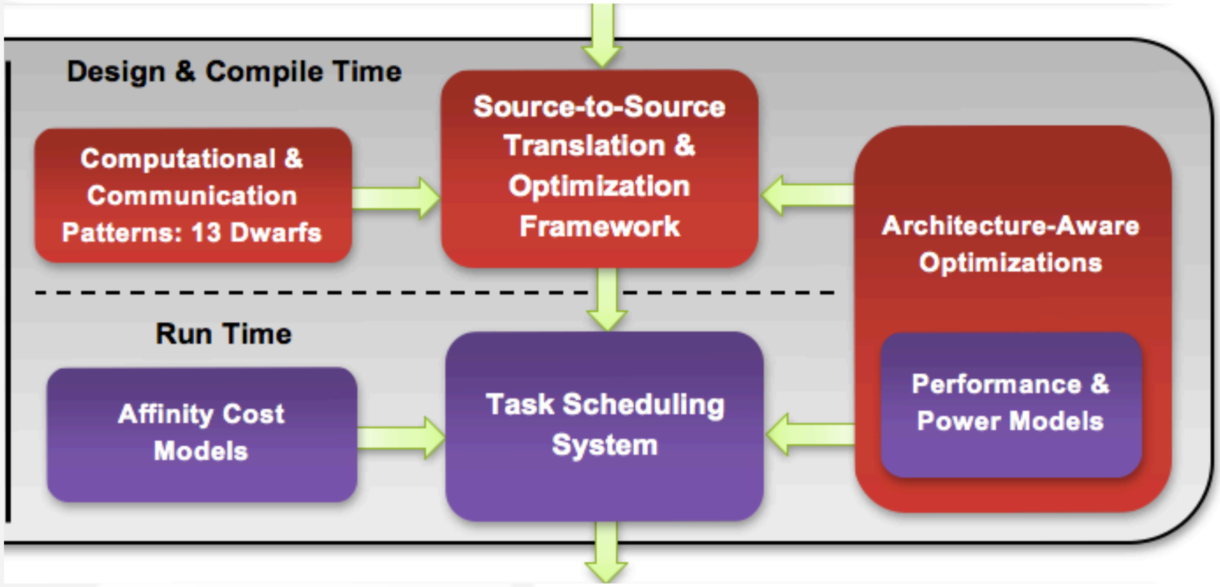
Why Heterogeneous Task Scheduling?

- OpenCL is portable, running on
  - CPUs, CellBE, GPUs, APUs, and Intel MIC (future)
- Heterogeneous resource usage cannot be predicted at design or compile time
- "Architecture-Aware Optimizations" show that different systems cannot be optimized the same way.

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!

Roadmap Status



Design & Compile Time

Computational & Communication Patterns: 13 Dwarfs

Source-to-Source Translation & Optimization Framework

Architecture-Aware Optimizations

Run Time

Affinity Cost Models

Task Scheduling System

Performance & Power Models

Much work still to be done.

OpenCL and the 13 Dwarfs	Beta release pending
Source-to-Source Translation	CU2CL only & no optimization
Architecture-Aware Optimization	Only on one dwarf addressed
Performance & Power Modeling	Preliminary & pre-multicore
Affinity-Based Cost Modeling	Empirical results; modeling in progress
Heterogeneous Task Scheduling	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.

Lessons Learned

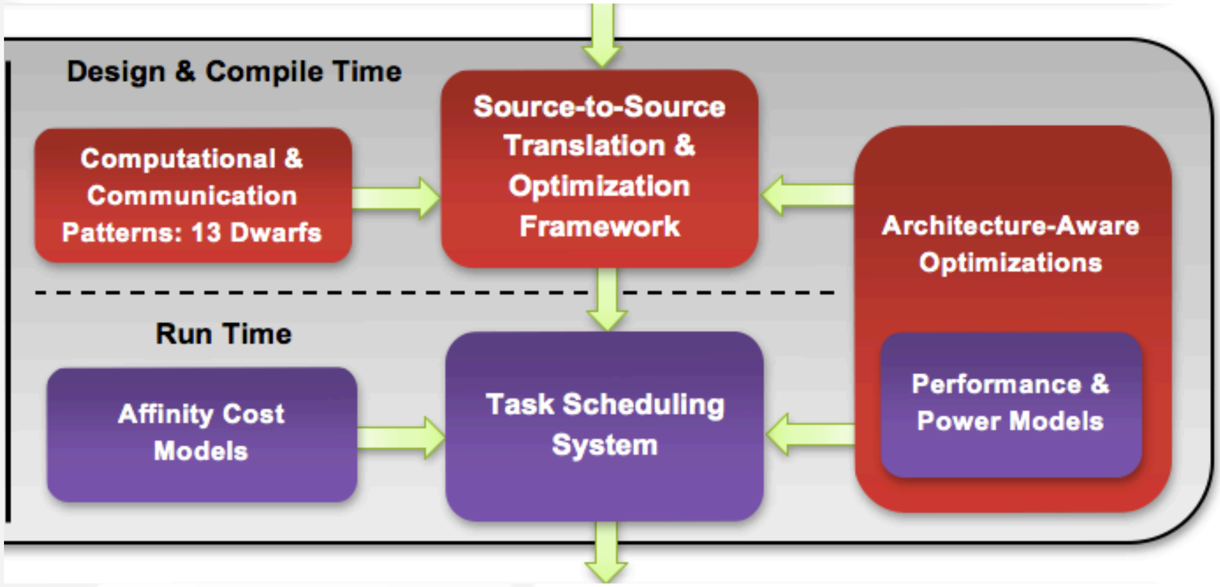
- Scientists with interest in heterogeneous computing want …
  - The performance of CUDA on NVIDIA GPUs
  - The portability of OpenCL to "write once, run anywhere"

We have shown potential paths to achieve both

- Domain scientists (at least in academia) want to write heterogeneous-accelerated applications only once.
  - Two complementary efforts
    - Just educate and have folks write in OpenCL
    - Many have invested in CUDA. Desire to run their CUDA-accelerated applications anywhere
- Architecture-aware optimizations matter … ***a lot*** …
- Heterogeneous task scheduling at run time can make a ***big*** difference …

POC: wfeng@vt.edu, 540.231.1192

Roadmap for Today's Talks



- Performance Portability via Architecture-Aware Optimizations and Heterogeneous Task Scheduling (Tom Scogland)
- CU2CL: CUDA-to-OpenCL Source-to-Source Translator (Mark Gardner)
- Performance Characterization and Optimization of Atomic Operations on AMD GPUs (Heshan Lin)
- Towards a Robust and Accurate Performance and Power Prediction Framework (Balaji Subramaniam)
- Towards Discrete GPUs as First-Class Citizens (Ashwin Aji)

VirginiaTech
Invent the Future

Funding Acknowledgements

NSF

CHREC
NSF Center for High-Performance Reconfigurable Computing

NVIDIA

1872 VirginiaTech

AMD

IBM

Department of Energy, United States of America

AMD

Los Alamos
NATIONAL LABORATORY

HARRIS

XILINX

Air Force Research Laboratory

POC: wfeng@vt.edu, 540.231.1192

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

SyNeRG

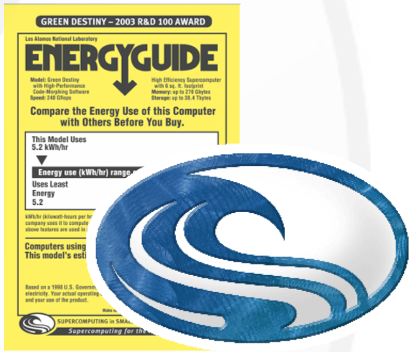
http://synergy.cs.vt.edu/

CHREC
NSF Center for High-Performance Reconfigurable Computing

http://www.chrec.org/

mpiBLAST

http://www.mpiblast.org/



GREEN DESTINY – 2003 R&D 100 AWARD

Los Alamos National Laboratory

ENERGYGUIDE

Model: Green Destiny with High-Performance Code-Morphing Software
Speed: 240 Gflops

High Efficiency Supercomputer with 6 sq. ft. footprint
Memory: up to 270 Gbytes
Storage: up to 38.4 Tbytes

Compare the Energy Use of this Computer with Others Before You Buy.

This Model Uses
5.2 kWh/hr

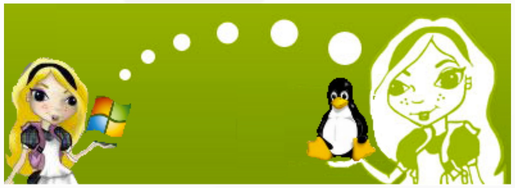
Uses Least
Energy
5.2

SUPERCOMPUTING in SMALL SPACES

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

THE GREEN 500

http://www.green500.org/



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

"Accelerators 'R Us"

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

VirginiaTech
Invent the Future

POC: wfeng@vt.edu, 540.231.1192

SyNeRG
synergy.cs.vt.edu