On the Three P's of Heterogeneous Computing *with Accelerators*

Wu FENG

Dept. of Elec. & Comp. Engg., Dept. of Computer Science, Virginia Bioinformatics Institute

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

VirginiaTech
Invent the Future

SyNeRG
synergy.cs.vt.edu

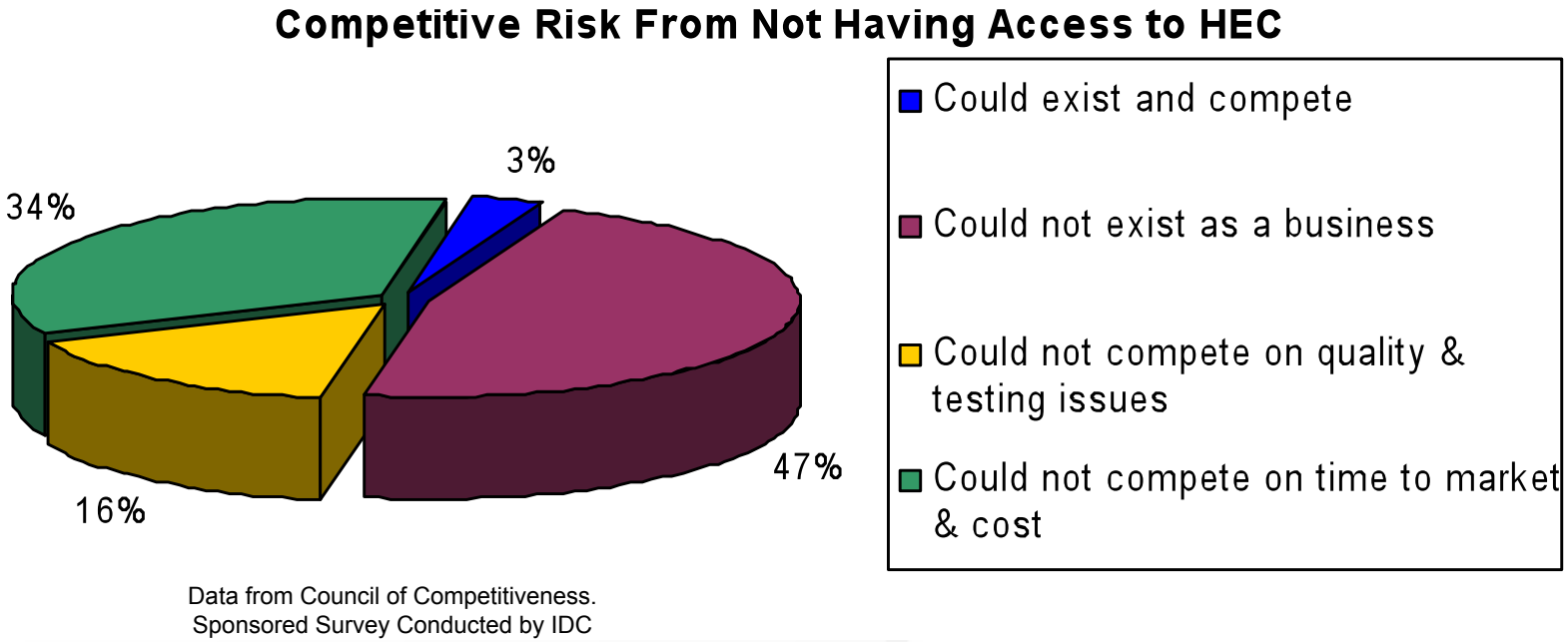
The Three P's of Heterogeneous Computing?

- Performance
  - How fast does your code run?
- Power
  - How much power (or energy) does your code consume?
- Programmability
  - How easy was your code to program?
- Price
  - How much did the system cost to acquire?
  - How much does the system cost to operate and maintain?
- Portability
  - How many different systems can your code run on?

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-End Computing (HEC)



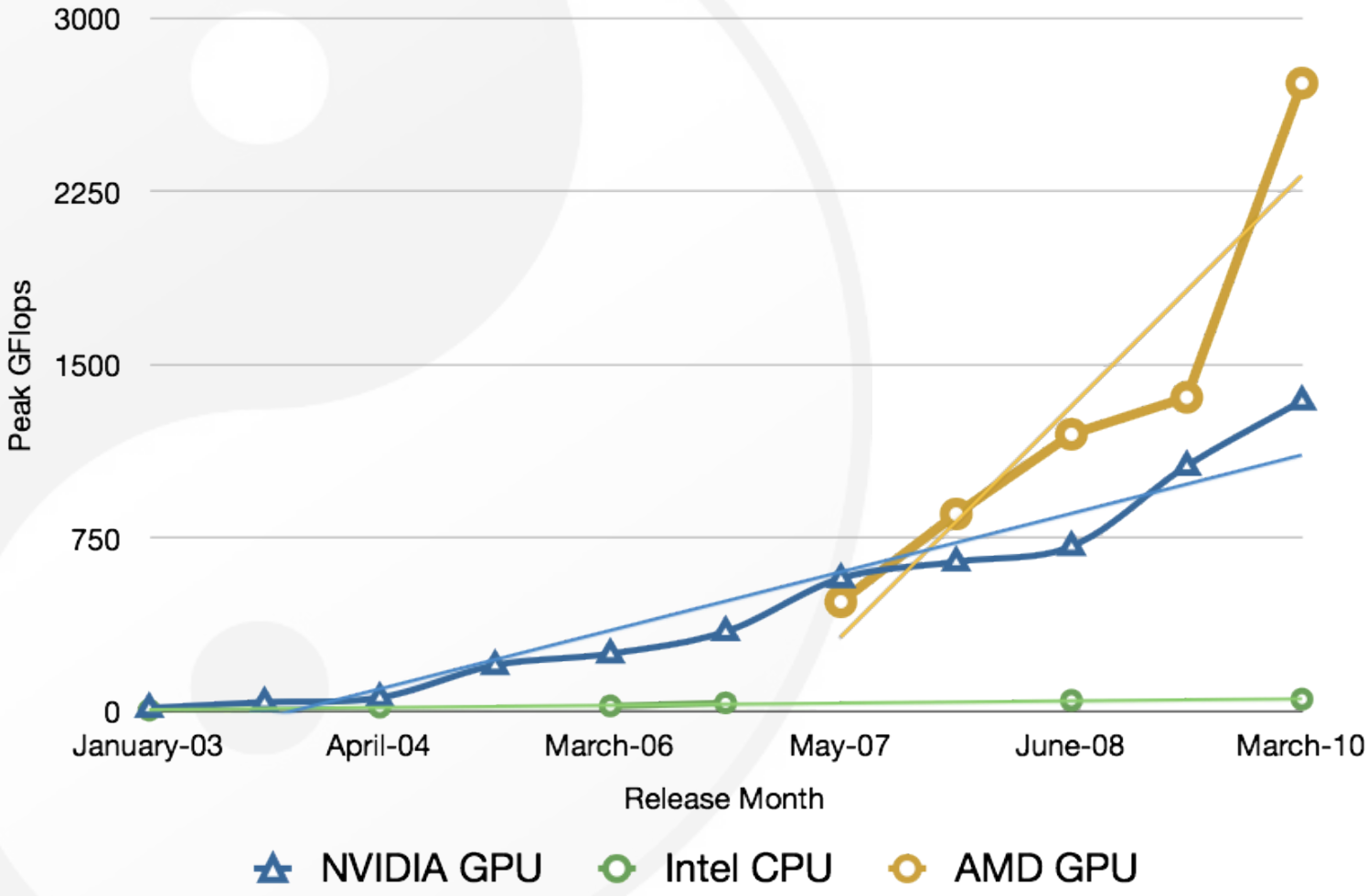
Data from Council of Competitiveness.
Sponsored Survey Conducted by IDC

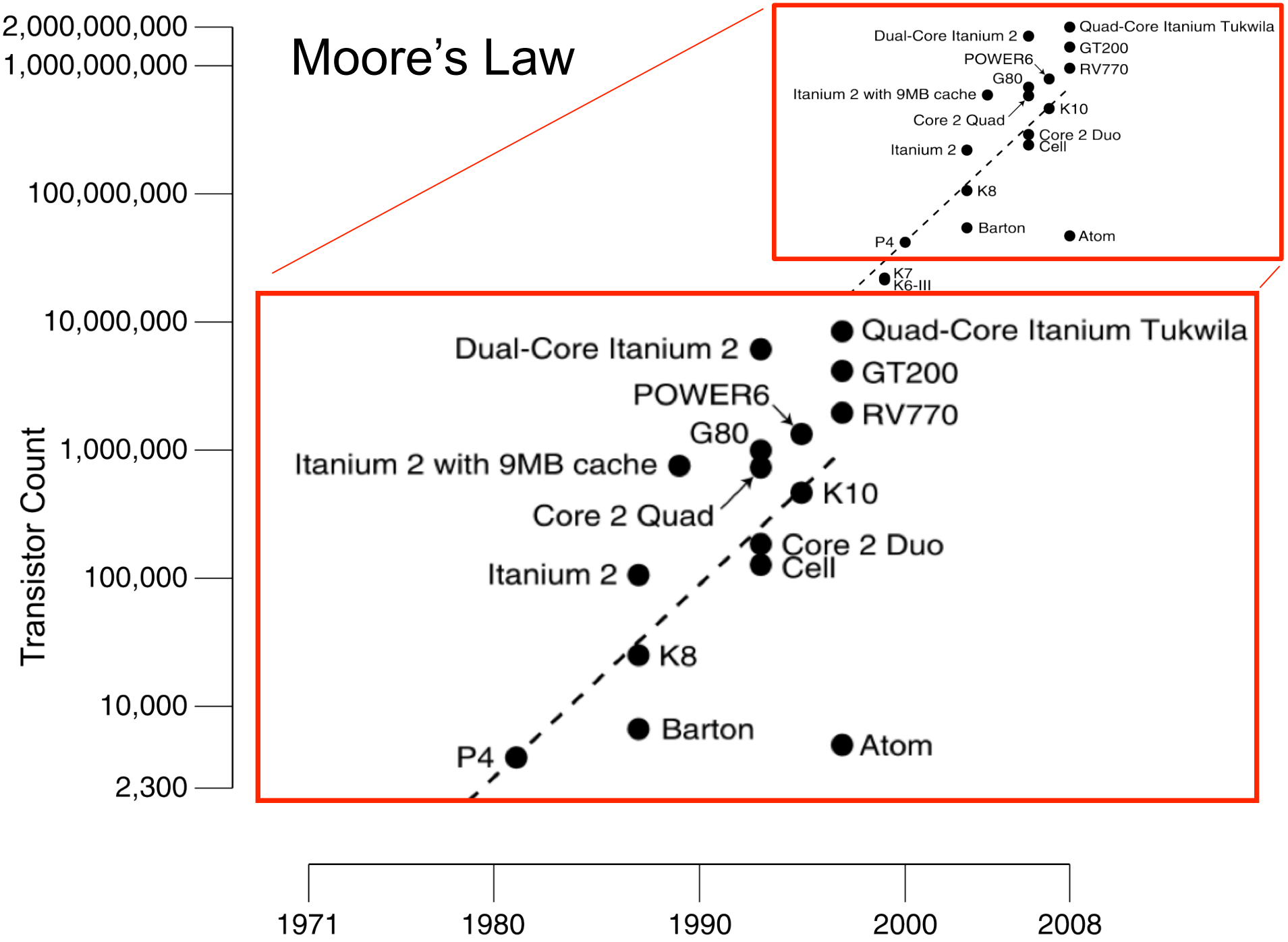
- Trend
  - Heterogeneous HEC dominates the top 10 of the

THE GREEN 500 November 2010

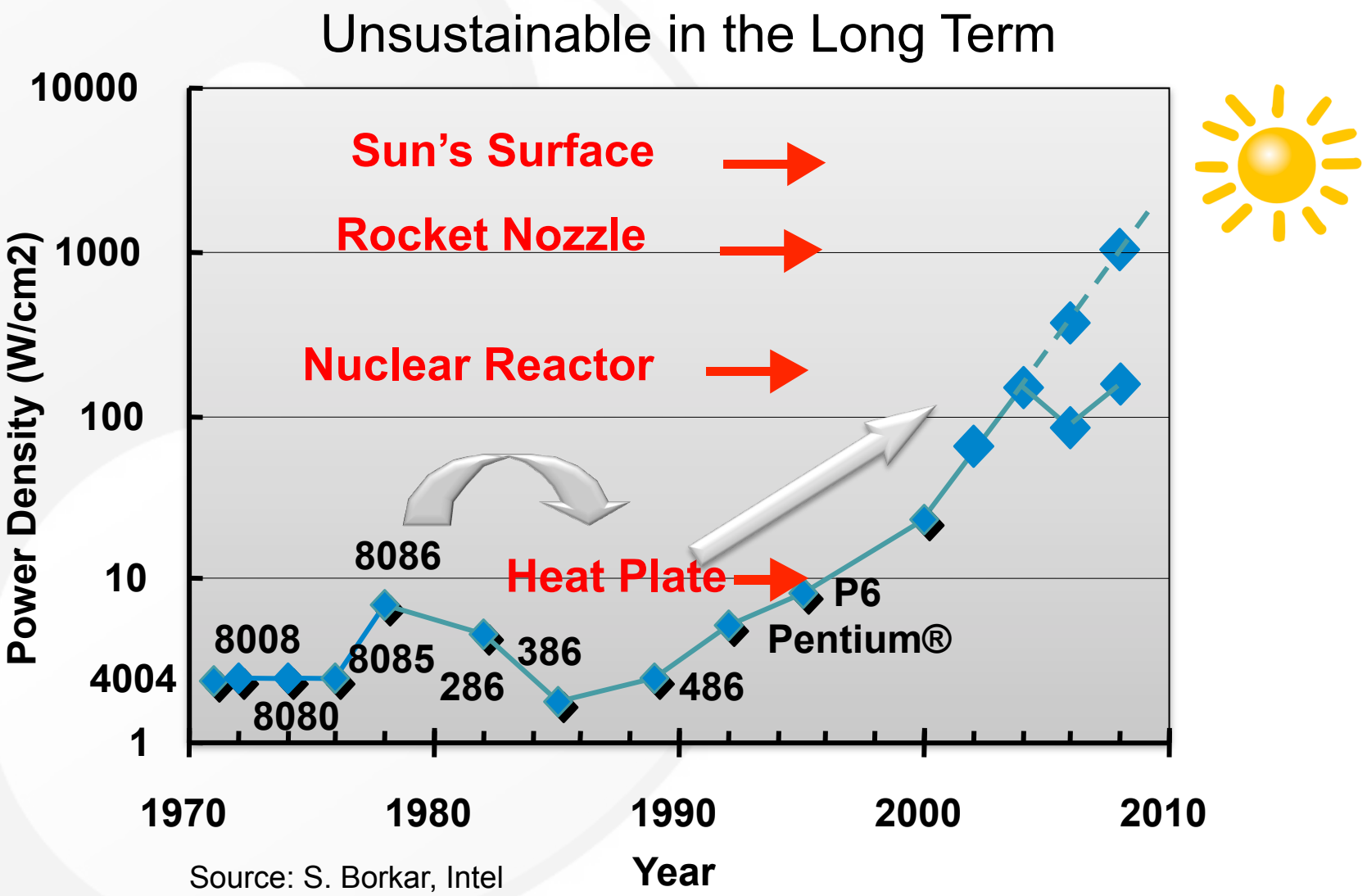
Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	1684.20	IBM Thomas J. Watson Research Center	NNSA/SC Blue Gene/Q Prototype	38.80
2	1448.03	National Astronomical Observatory of Japan	GRAPE-DR accelerator Cluster, Infiniband	24.59
3	958.35	GSIC Center, Tokyo Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows	1243.80
4	933.06	NCSA	Hybrid Cluster Core i3 2.93Ghz Dual Core, NVIDIA C2050, Infiniband	36.00
5	828.67	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect	57.96
6	773.38	Universitaet Wuppertal	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
7	773.38	Universitaet Regensburg	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
8	773.38	Forschungszentrum Juelich (FZJ)	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
9	740.78	Universitaet Frankfurt	Supermicro Cluster, QC Opteron 2.1 GHz, ATI Radeon GPU, Infiniband	385.00
10	677.12	Georgia Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5660 2.8Ghz, nVidia Fermi, Infiniband QDR	94.40
11	636.36	National Institute for Environmental Studies	GOSAT Research Computation Facility, nvidia	117.15

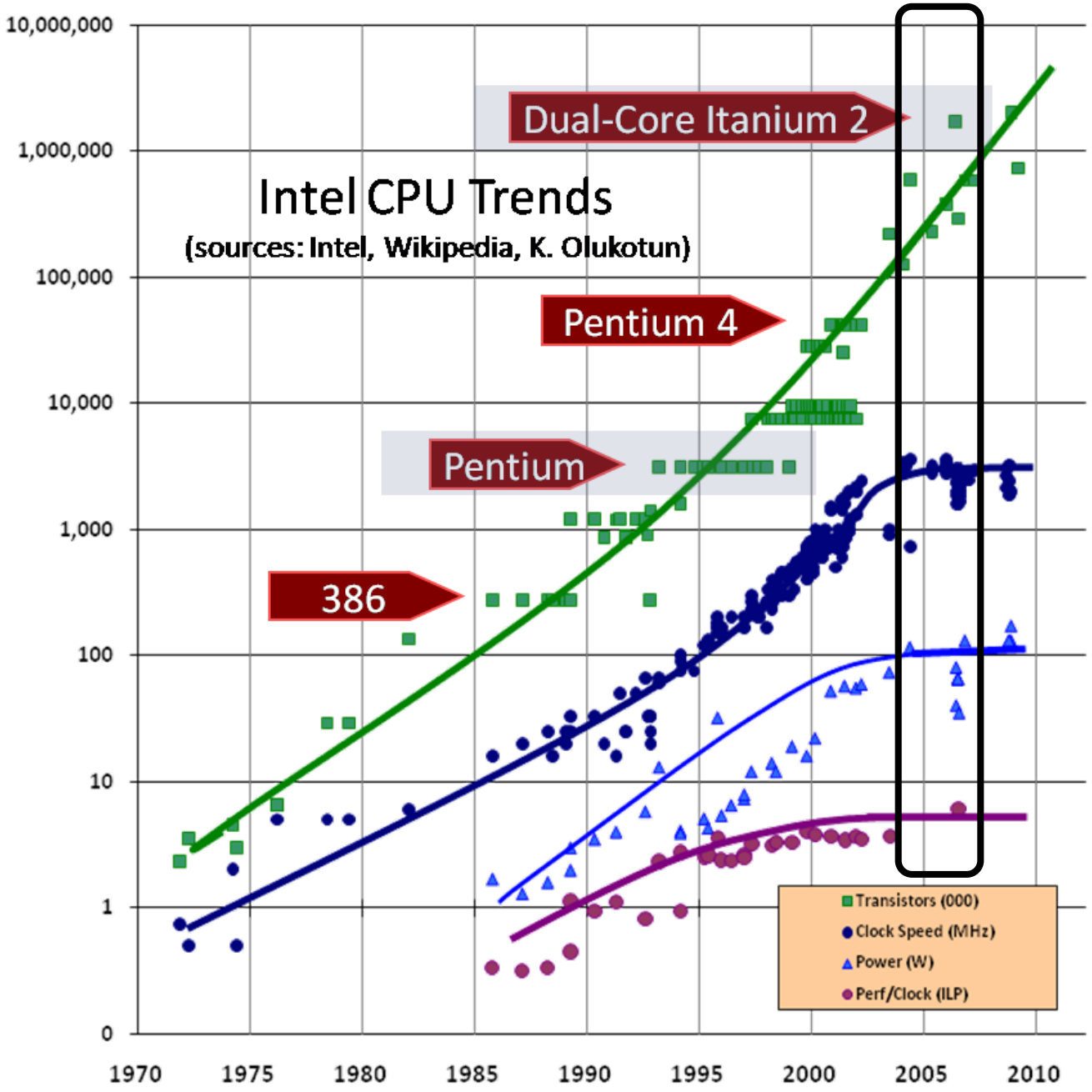
Peak Single Precision




Moore's Law
2,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
Transistor Count
Dual-Core Itanium 2
Quad-Core Itanium Tukwila
GT200
POWER6
RV770
G80
Itanium 2 with 9MB cache
K10
Core 2 Quad
Core 2 Duo
Cell
Itanium 2
K8
Barton
Atom
P4
K7
K6-III
1971
1980
1990
2000
2008

Moore's Law for Power Density




Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Dual-Core Itanium 2
Pentium 4
Pentium
386
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
0
1970
1975
1980
1985
1990
1995
2000
2005
2010
Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)

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 "cluster-on-a-chip" 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

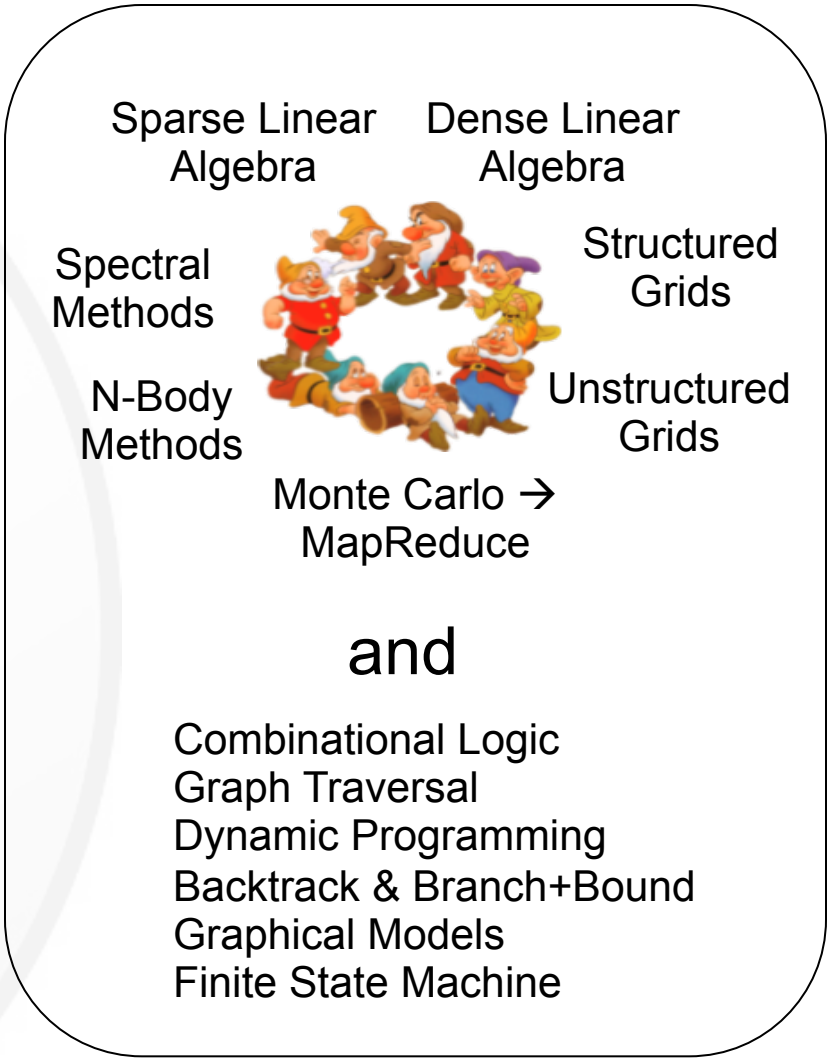
- "The free lunch is over..." †
  - Programmers can no longer expect substantial increases in single-threaded performance.
  - The burden falls on developers to exploit parallel hardware for performance gains.

- How do we lower the cost of concurrency?

† H. Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software," *Dr. 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 dwarfs
    - 7 original by P. Colella
    - 6 additional by Asanovic et al.



† 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 Computing

- Deliver personalized heterogeneous 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

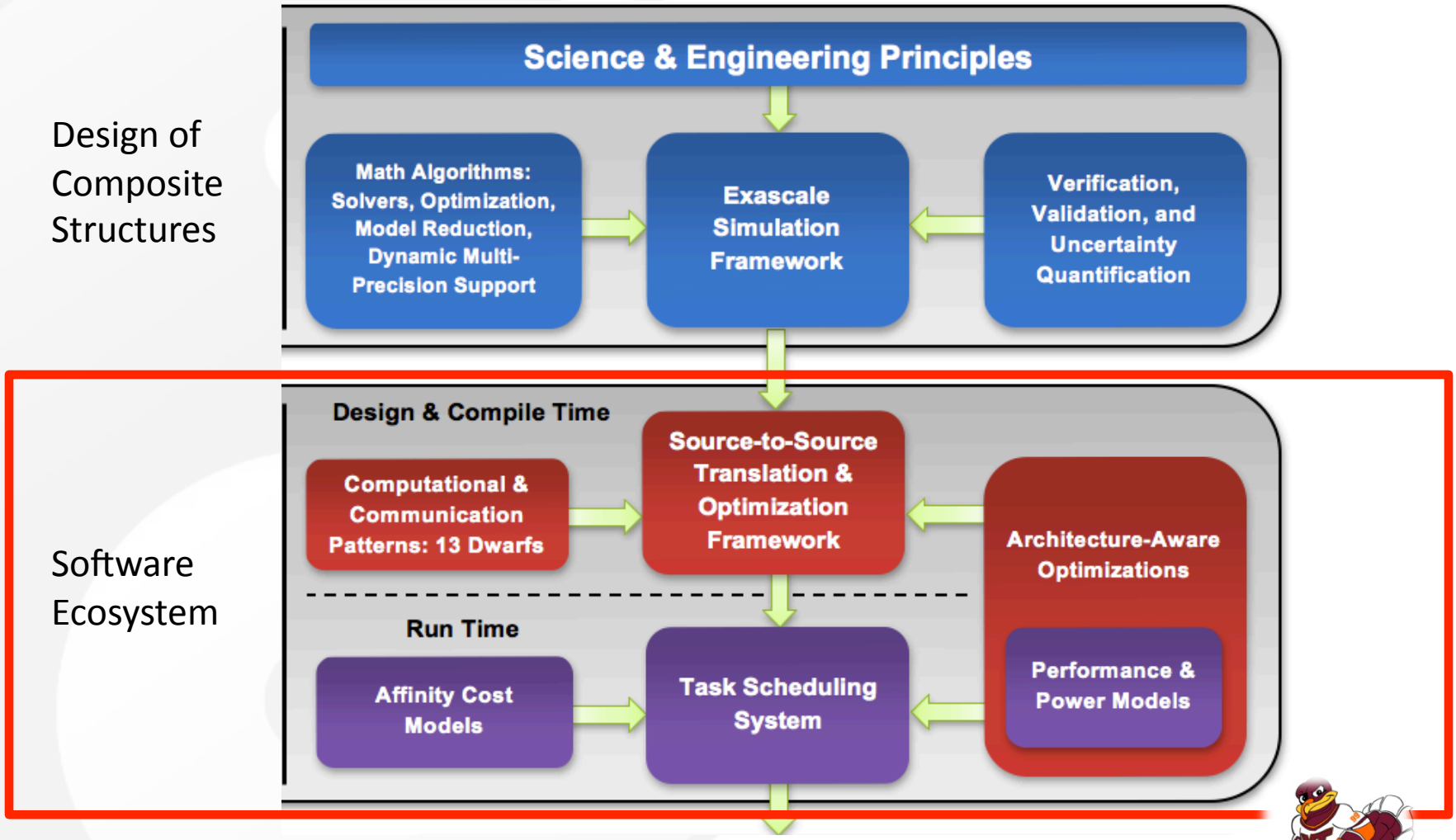
Project Outcome

An Ecosystem for Heterogeneous Computing

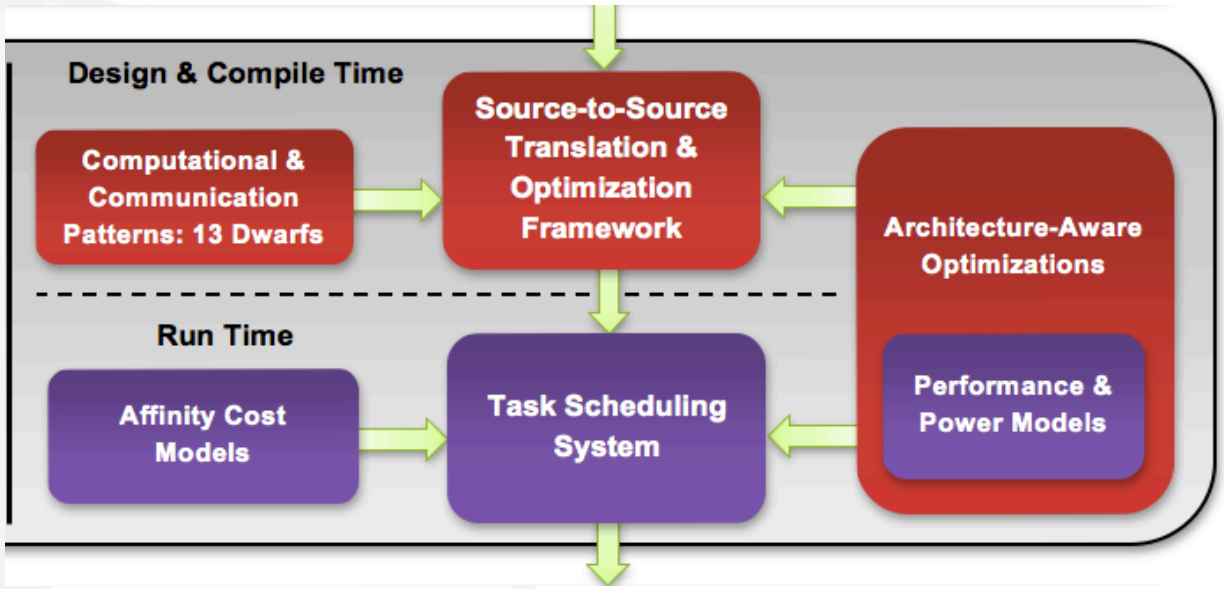
- A multi-dimensional understanding of how to optimize ***performance, power, programmability***, or some combination thereof
  - Performance (under Resource Constraints)
    - # threads/block; # blocks/grid; configurable memory; mixed-mode arithmetic; and so on
  - Power
    - Device vs. system power
    - 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

An Ecosystem for Heterogeneous Parallel Computing



Roadmap



- OpenCL and the 13 Dwarfs — All 3 P's
- Source-to-Source Translation — "Programmability"
- Architecture-Aware Optimizations — Performance
- Heterogeneous Task Scheduling — All 3 P's

Project Goal

An Ecosystem for Heterogeneous Computing

- Deliver personalized heterogeneous 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

What Is "OpenCL and 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	

Selected Applications and their Dwarfs …

- FFT: Fast Fourier Transform
  - A ***spectral method*** that is used for a myriad of signal processing applications, e.g., video surveillance, etc.
- Electrostatic Surface Potential (ESP) of Molecules
  - An ***n-body method*** calculation to support molecular dynamics
  - Popular packages: AMBER, GROMACS, LAMPPS
- Smith-Waterman: Local Sequence Alignment
  - A ***dynamic programming method***
    - The ***full computation***, including matrix filling and storing, affine gap-penalty calculations, and *backtracing*.
- N-Queens
  - A ***backtrack method*** … meant to highlight benefits of FPGA

Initial Performance: Actual & Estimated (Nov. 2010)

	CPU	FPGA Convey		GPU				
	Serial	HC-1 ex 1 V6 LX760	HC-1 1 V5 LX330	AMD Radeon HD 5450	AMD Radeon HD 5870	NVIDIA ION (Zotac)	NVIDIA 320M (Apple)	NVIDIA GeForce GTX280
ESP (MPPS)	4.6	4800.0	2400.0	41.5	2119.0 *6636.6*	141.4	393.7	2387.0 *5233.0*
FFT 1D-Float (GFLOPS)	3.3	240.3	38.4	-	379.2	3.8	-	129.8
FFT 2D-Float (GFLOPS)	1.6	144.0	26.5	-	111.6	-	-	83.7
FFT 1D-Int (GOPS)	-	-	10,137.0	-	-	-	-	-
Smith-Waterman (MCUPS)	14.2	-	688,000	-	7.6	-	-	35.5

MPPS: Million Pairs Per Second
GFLOPS: Giga Floating-Point Operations per Second
MCUPS: Million Cell Updates Per Second

Numbers in *italics* are architecture-optimized.

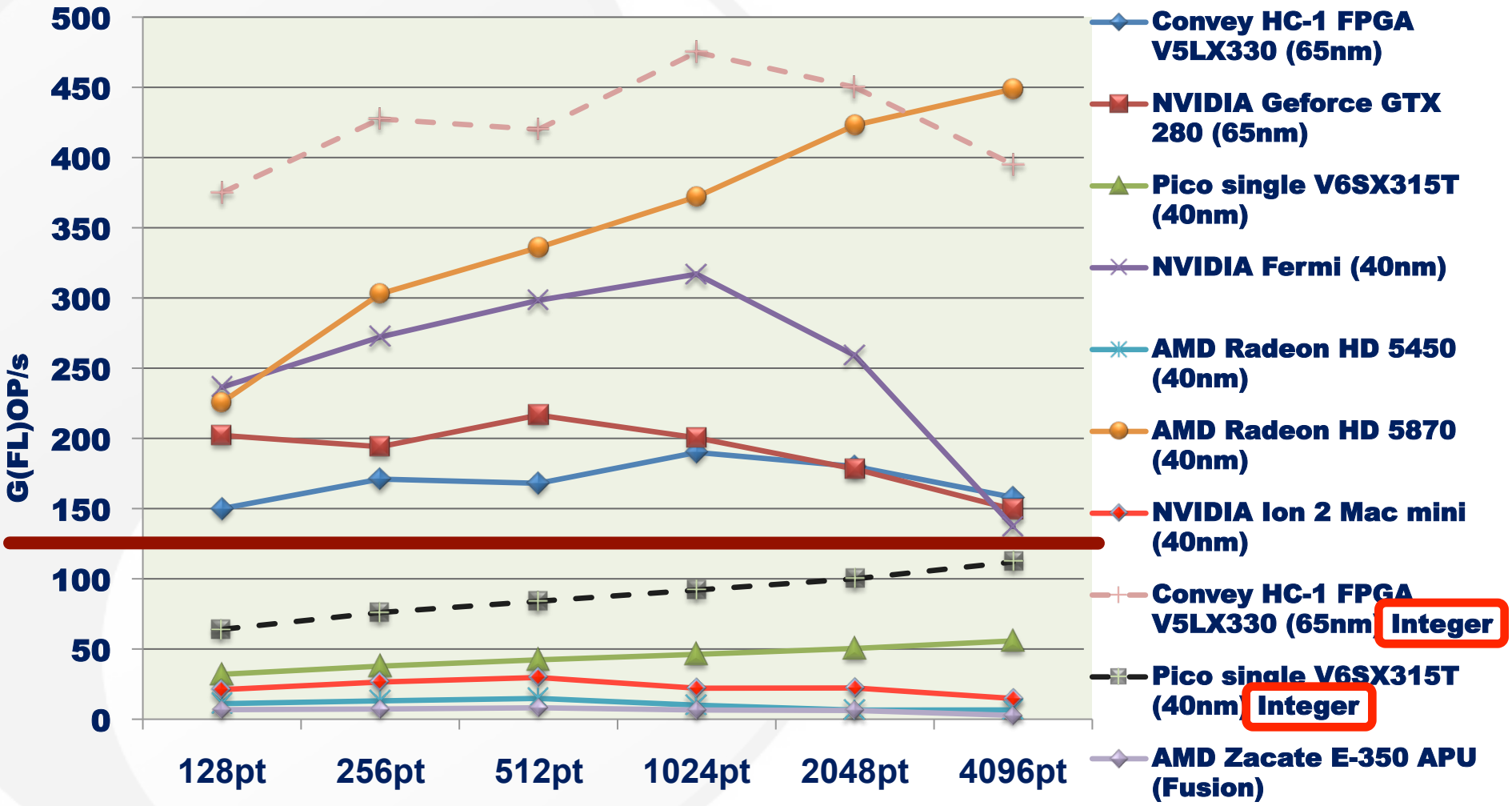
Estimated (or from Convey) in gray

IMPORTANT! See "Caveats" slide.

Caveats and Notes

- GPU Implementation of Smith-Waterman: A complete implementation with matrix filling, matrix storage, affine gap penalties, ***and*** backtrace, and thus, not directly comparable to other implementations … many of which ignore the backtrace.
- FPGA performance #s for ESP based on area & speed data of Xilinx FP operators in actual computation pipeline @ 300MHz. Actual implementation is in place but facing some issues.
- Floating-point FFT for GPU & FPGA: 1D-1024pt,32-bit and 2D-512*512 pts,32-bit. (FPGA @ 300MHz, GPU @ 800+MHz.)
- Integer FFT for Convey HC-1: 64pt,16-bit @ 150MHz.
- Device costs are **not** considered. High-end FPGA: 50x-80x more $$$ than a high-end GPU. Data transfer costs are ignored for "estimated" FPGA performance #s.
- FPGA assumption for FFT: Overhead to do a transpose for 2D FFT is negligible.
- Projected performance with all four devices active for Convey would be ~4X for both 1D and 2D.
- FFT numbers in **grey** are estimates. FFT IP cores are configured in streaming/pipelined mode. In V6 all 8 memory ports are used, thus performance is memory-bound.
- FFT numbers in **green** are actual numbers from device. All 8 memory ports are used. Streaming FFTs replaced by Radix-2 FFTs containing fewer DSP48E slices to have a balanced implementation across all memory ports. Performance is **limited** by the number of DSP48E slices (Total of 192 on V5LX330T). Numbers are low due to reduced real read/write memory BW obtained (~15.5 GB/s) mainly due to memory stalls on interfaces (expected 20GB/s per FPGA).

Performance Update: 1-D FFT (Jun. 2011)



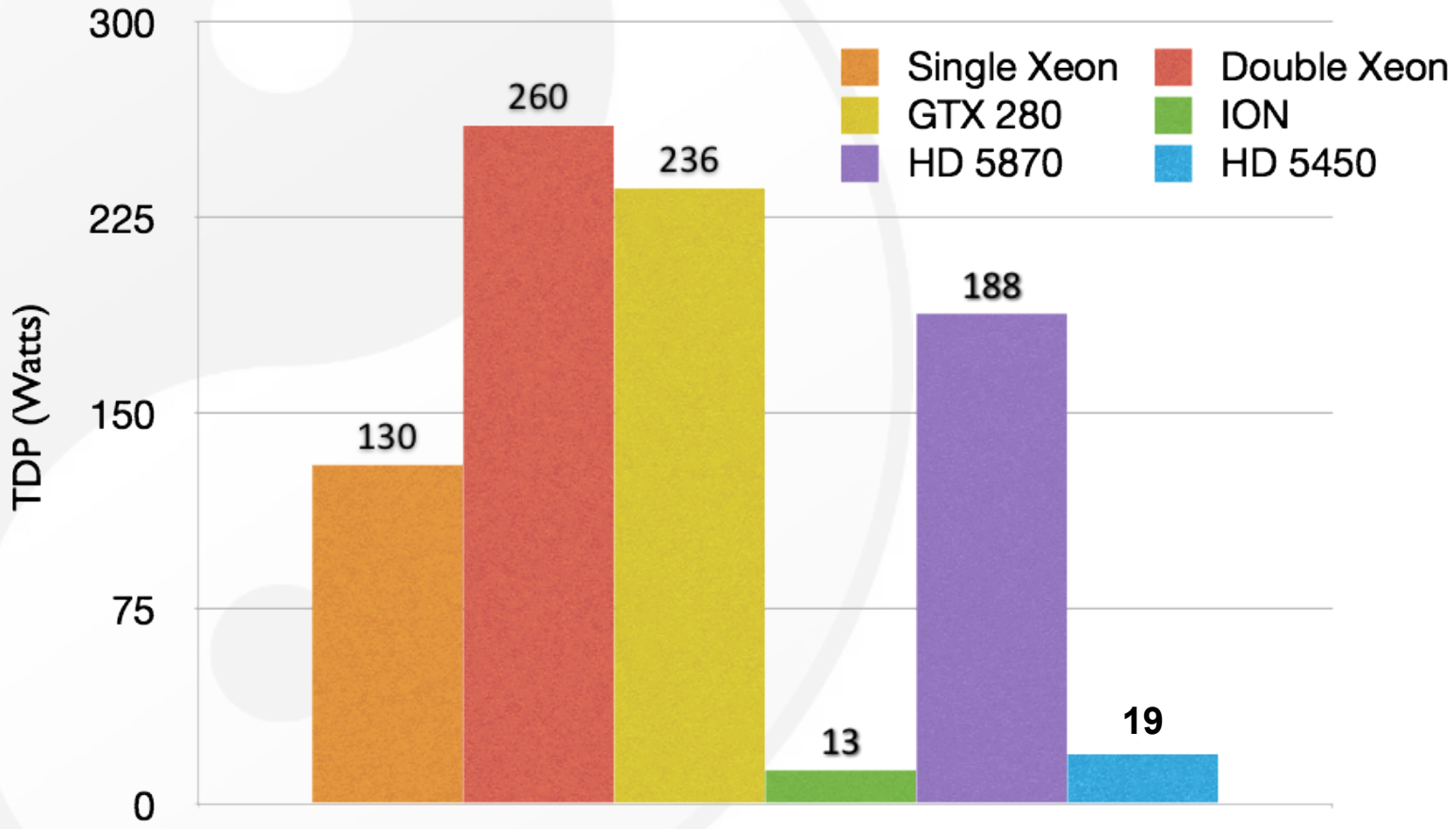
Note: Host-to-device transfer overhead ignored in all cases.

Performance: N-Queens (Backtrack)

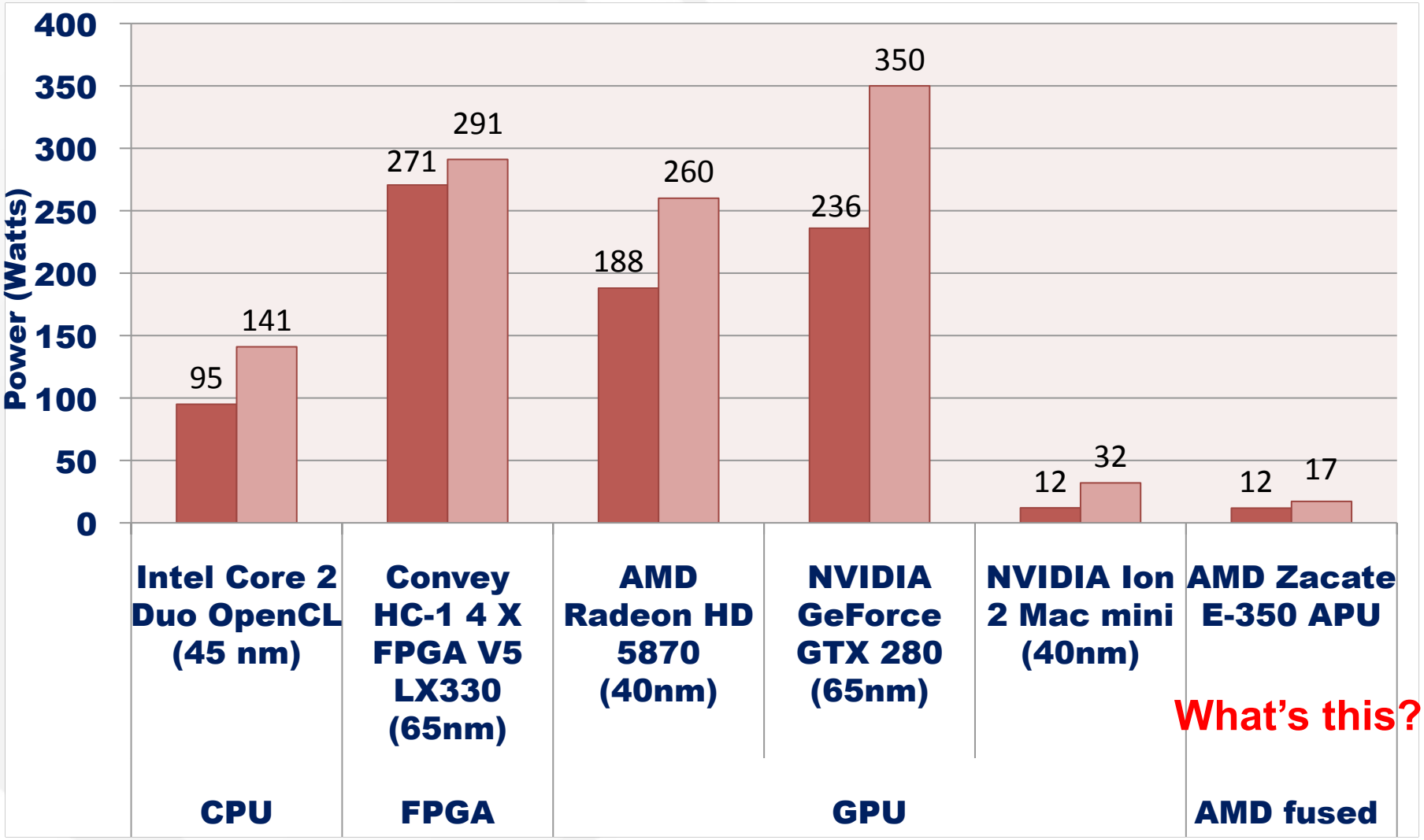
N	Execution time in seconds		
	AMD Radeon HD 5870 GPU **	NVidia Fermi GPU **	Convey HC - 1 FPGA *
8	0.04	0.13	0.000017
9	0.04	0.14	0.00004
10	0.04	0.16	0.00015
11	0.05	0.16	0.00071
12	0.05	0.18	0.0036
13	0.06	0.19	0.02
14	0.04	0.22	0.119
15	0.05	0.24	0.746
16	0.04	0.25	5.02
17	0.09	0.29	35.6
18	0.62	1.16	266.41

* Estimated execution time based on implementation on a single Virtex 5 FPGA
** Code downloaded from http://forum.beyond3d.com/showthread.php?t=56105

Power: Thermal Design Power (TDP)



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



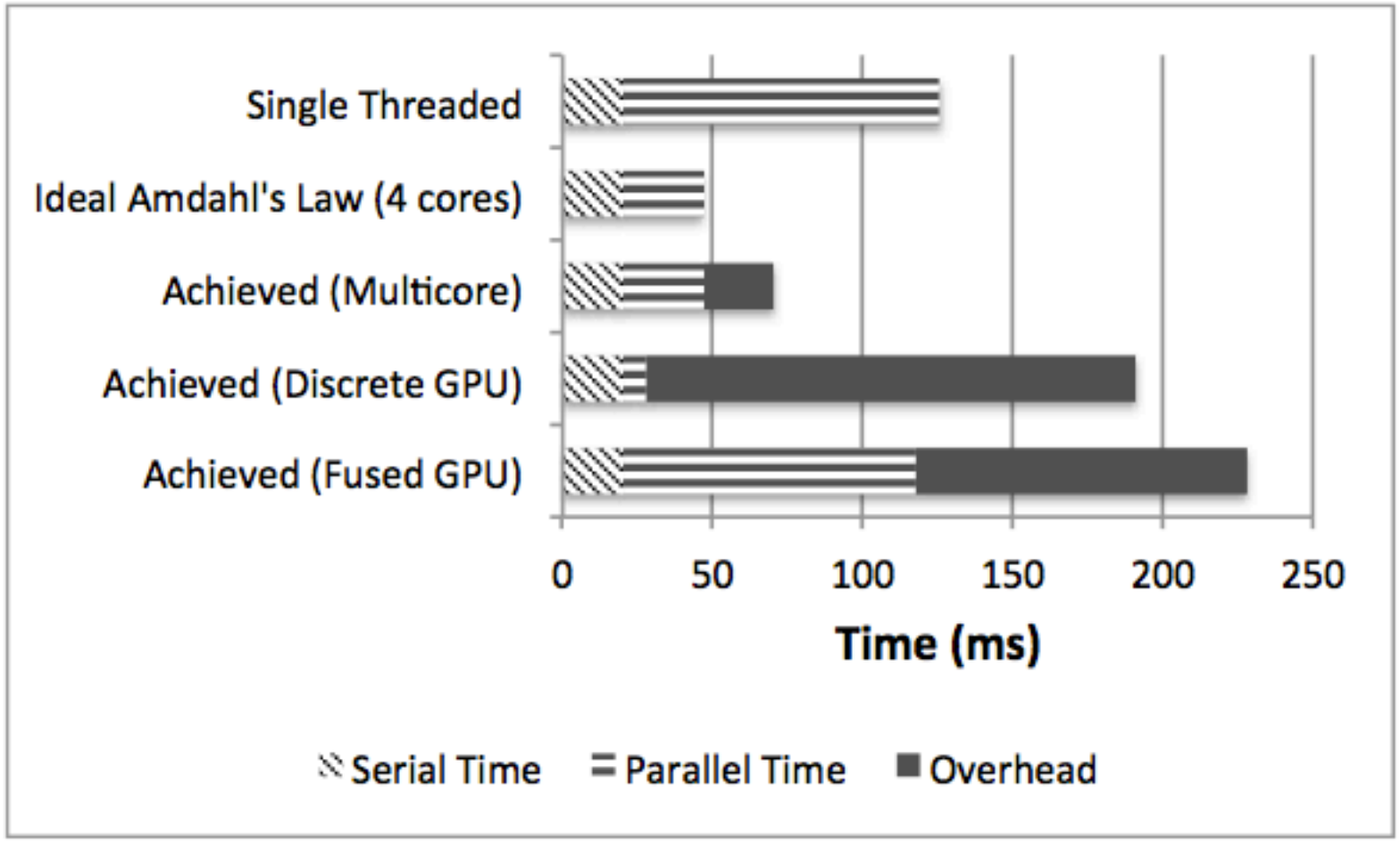
Experimental Set-Up: Machines and Workload

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

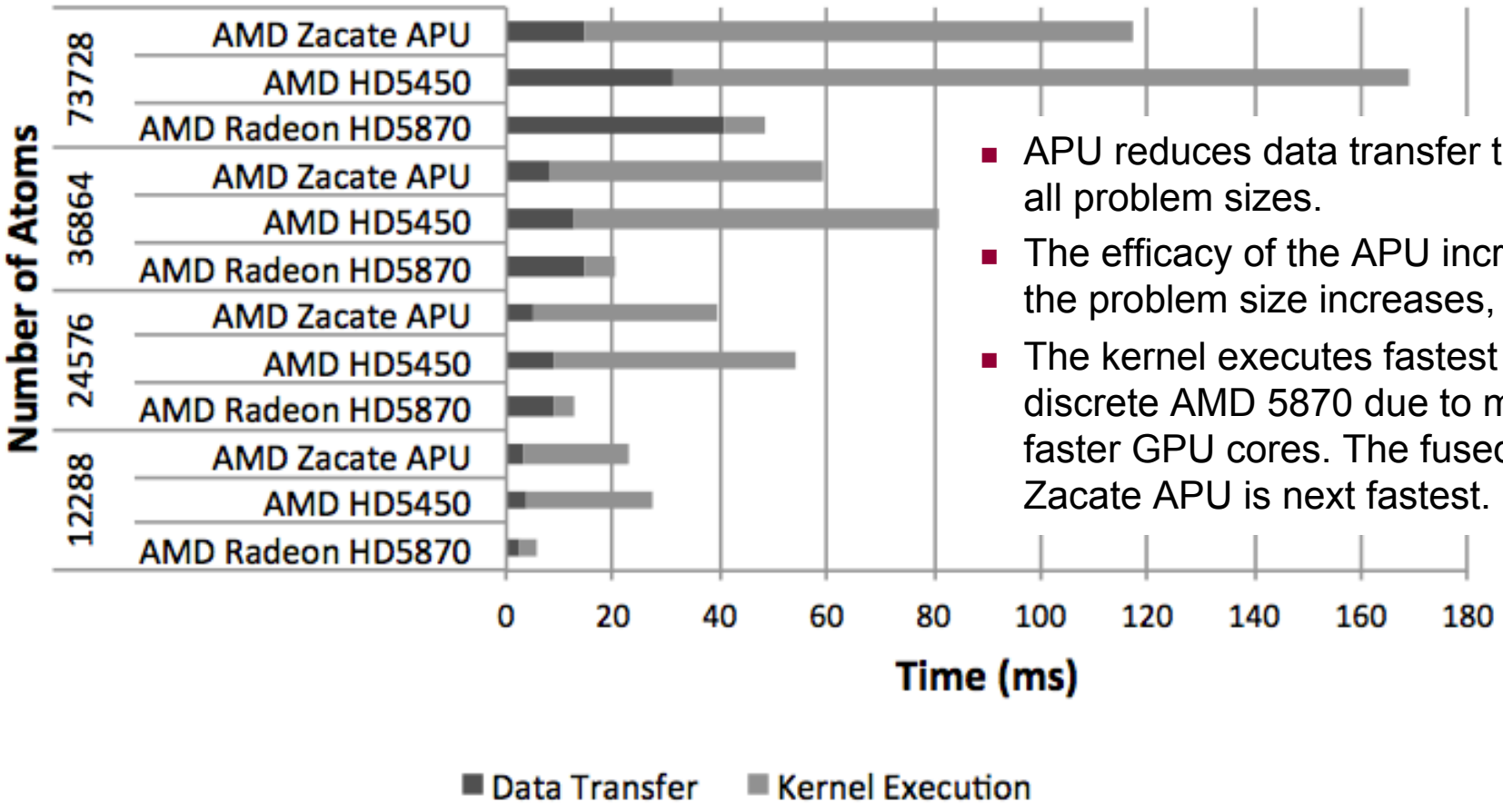
- OpenCL and the 13 Dwarfs
  - Sparse Linear Algebra: SpMV
  - N-body: Molecular Modeling
  - Spectral: FFT
  - Dense Linear Algebra: Scan and Reduce (SHOC @ ORNL)

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.

Amdahl's Law for Different Parallel Devices

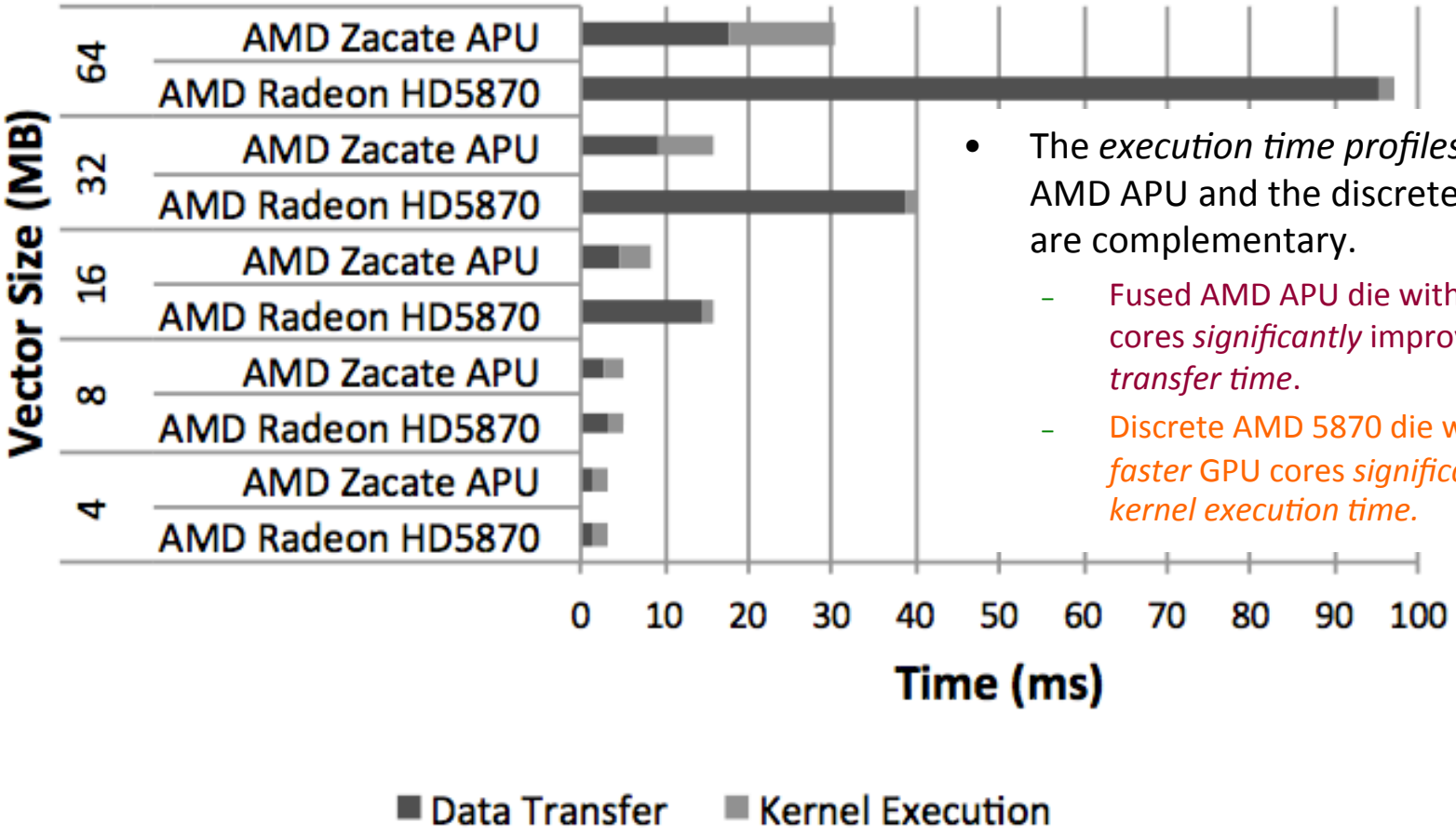


Performance: Molecular Modeling (N-Body)



- APU reduces data transfer times for all problem sizes.
- The efficacy of the APU increases as the problem size increases, but …
- The kernel executes fastest on discrete AMD 5870 due to more and faster GPU cores. The fused AMD Zacate APU is next fastest.

Performance: Reduction (Dense Linear Algebra)

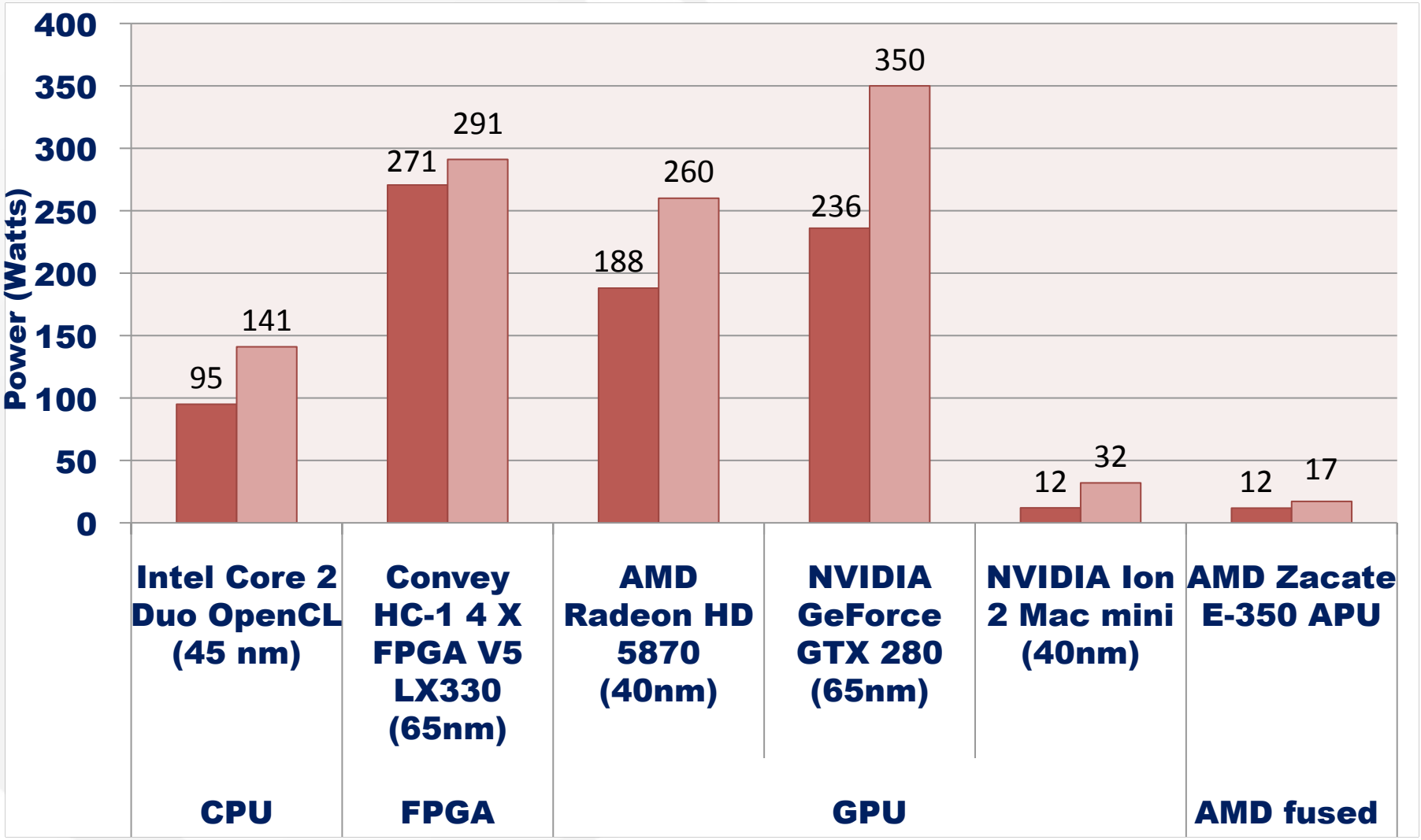


- The *execution time profiles* of the fused AMD APU and the discrete AMD 5870 are complementary.
  - Fused AMD APU die with only 80 GPU cores *significantly* improves *data transfer time*.
  - Discrete AMD 5870 die with 1600 *faster* GPU cores *significantly* improves *kernel execution time.*

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)

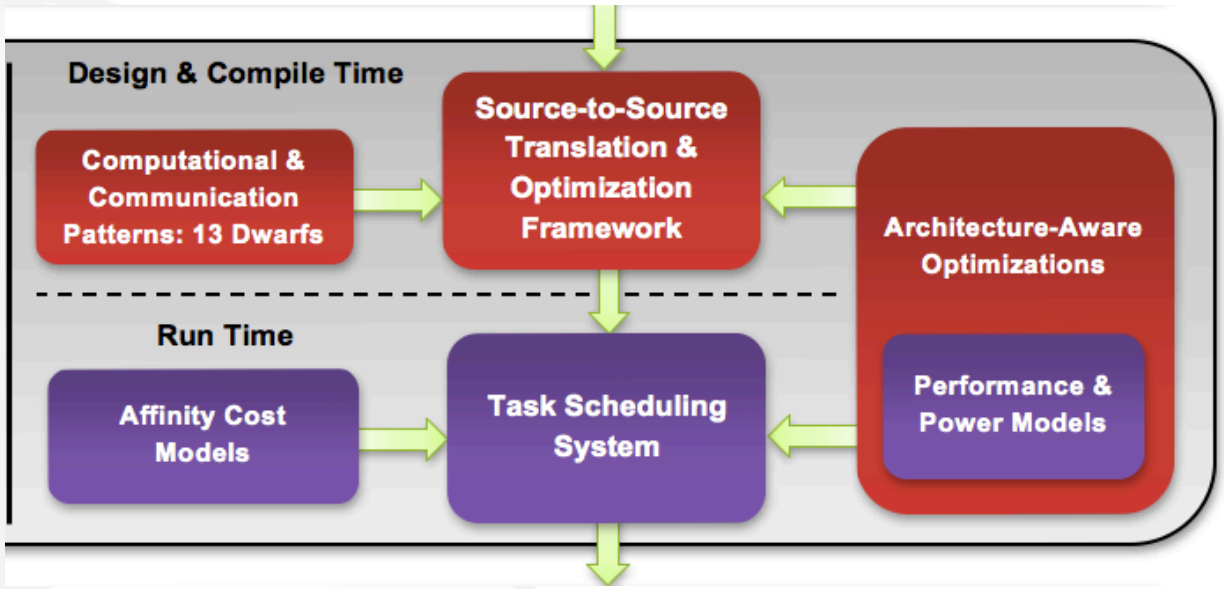


Status of OpenCL and the 13 Dwarfs

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

Roadmap



- OpenCL and the 13 Dwarfs — All 3 P's
- Source-to-Source Translation — "Programmability"
- Architecture-Aware Optimizations — Performance
- Heterogeneous Task Scheduling — All 3 P's

Creating an Army of Dwarfs via "CU2CL" †

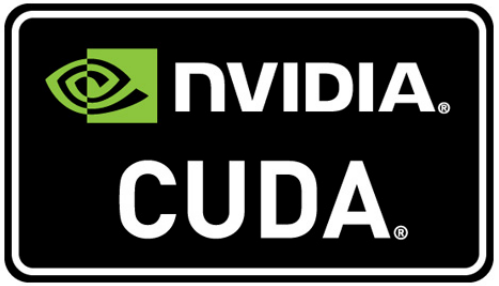
- CU2CL: *CU*DA-*to*-Open*CL* Source-to-Source Translator
  - Implemented as a Clang plugin, allowing us to leverage its production-quality compiler framework and target LLVM bytecode.
  - Covers the primary CUDA constructs found in CUDA C and the CUDA run-time API.
  - Successfully translates CUDA applications from the CUDA SDK and Rodinia benchmark suite.
  - 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***

Initialization Code: CUDA



None!

Initialization Code: OpenCL

OpenCL

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

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

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

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

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 ...

Goals of CU2CL

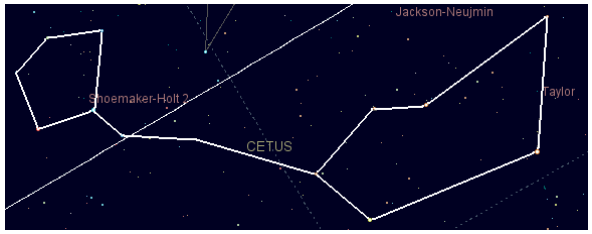
- Automatically translate (or support) CUDA applications
- Generate maintainable OpenCL code for future development

GCC

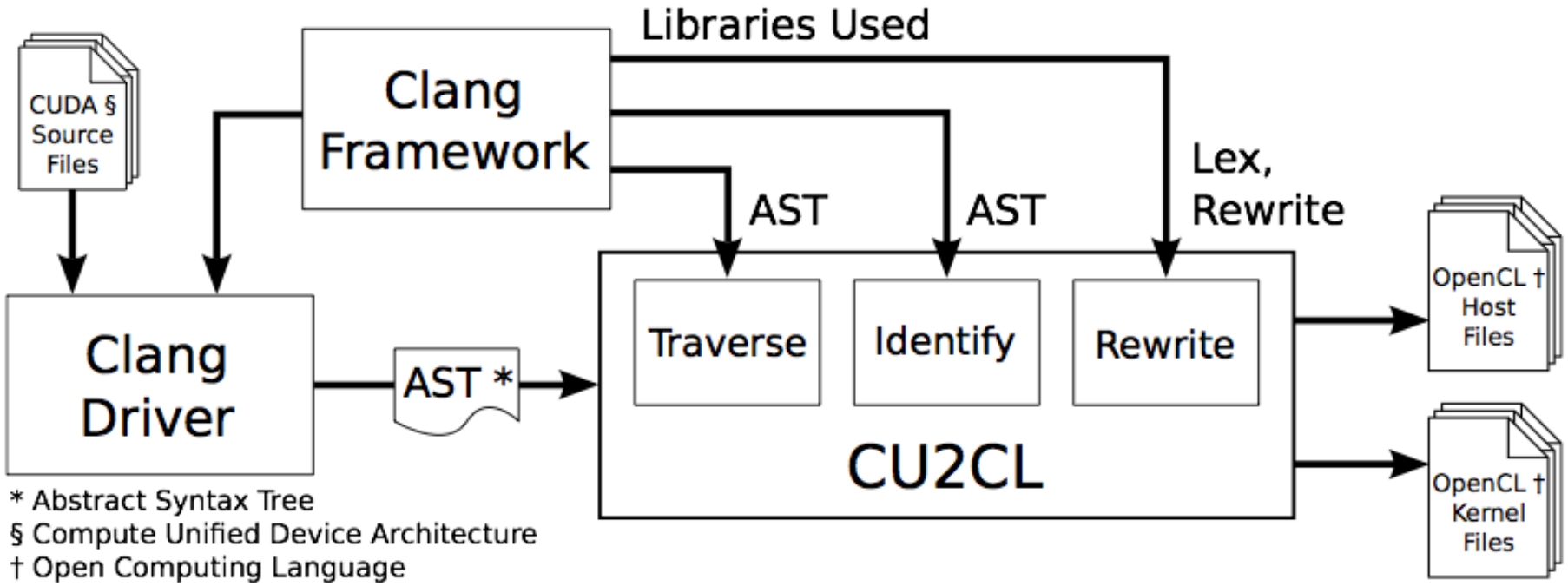
Clang

Open64

Cetus



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.

Experimental Set-Up

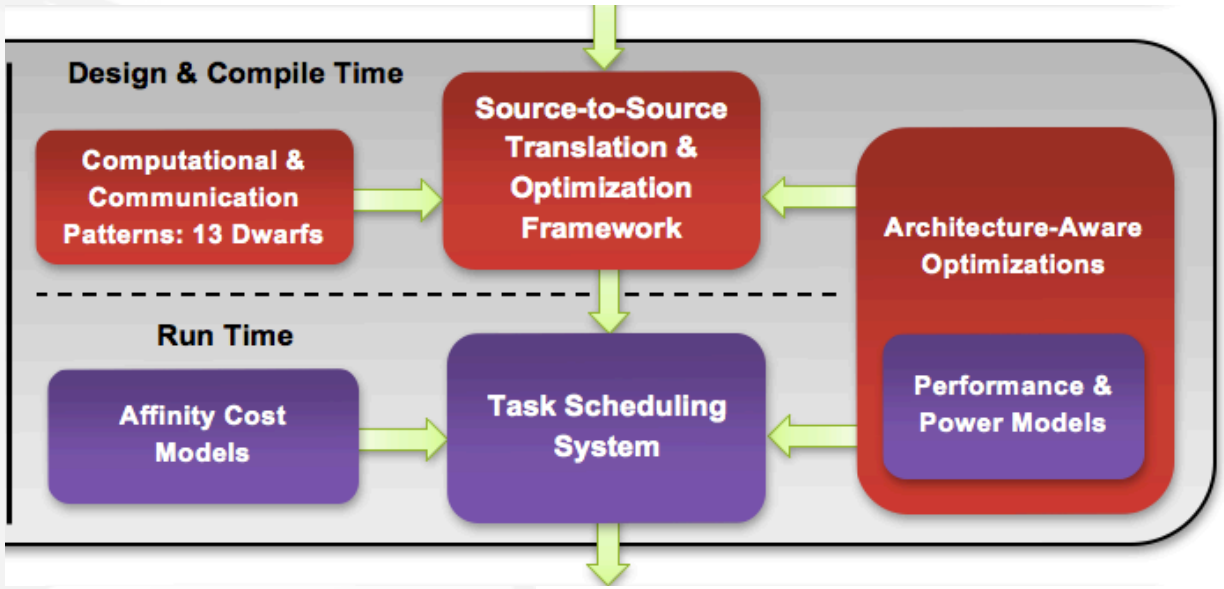
- CPU
  - 2 x 2.0-GHz Intel Xeon E5405 quad-core CPUs
- GPU
  - NVIDIA GTX 280 with 1 GB of graphics memory
- Applications
  - CUDA SDK: *asyncAPI, bandwidthTest, BlackScholes, matrixMul, scalarProd, vectorAdd*
  - Rodinia: *Back Propagation, Breadth-First Search, HotSpot, Needleman-Wunsch, SRAD*

Translated Application Performance (sec)

Application	CUDA	Automatic OpenCL	Manual OpenCL
vectorAdd	0.0499	0.0516	0.0521
Hotspot	0.0177	0.0565	0.0561
Needleman-Wunsch	6.65	8.77	8.77
SRAD	1.25	1.55	1.54

- Less optimized kernels account for longer runtimes in OpenCL

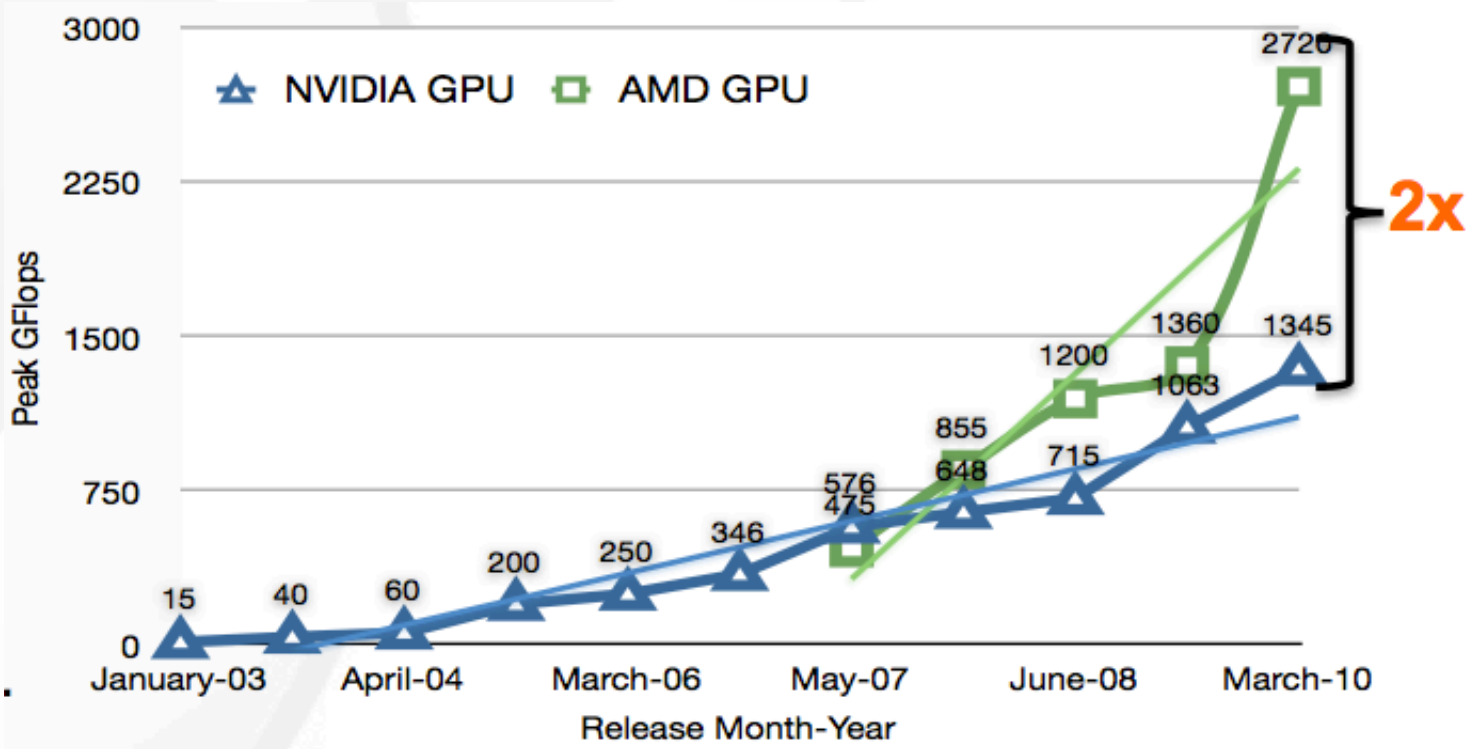
Roadmap



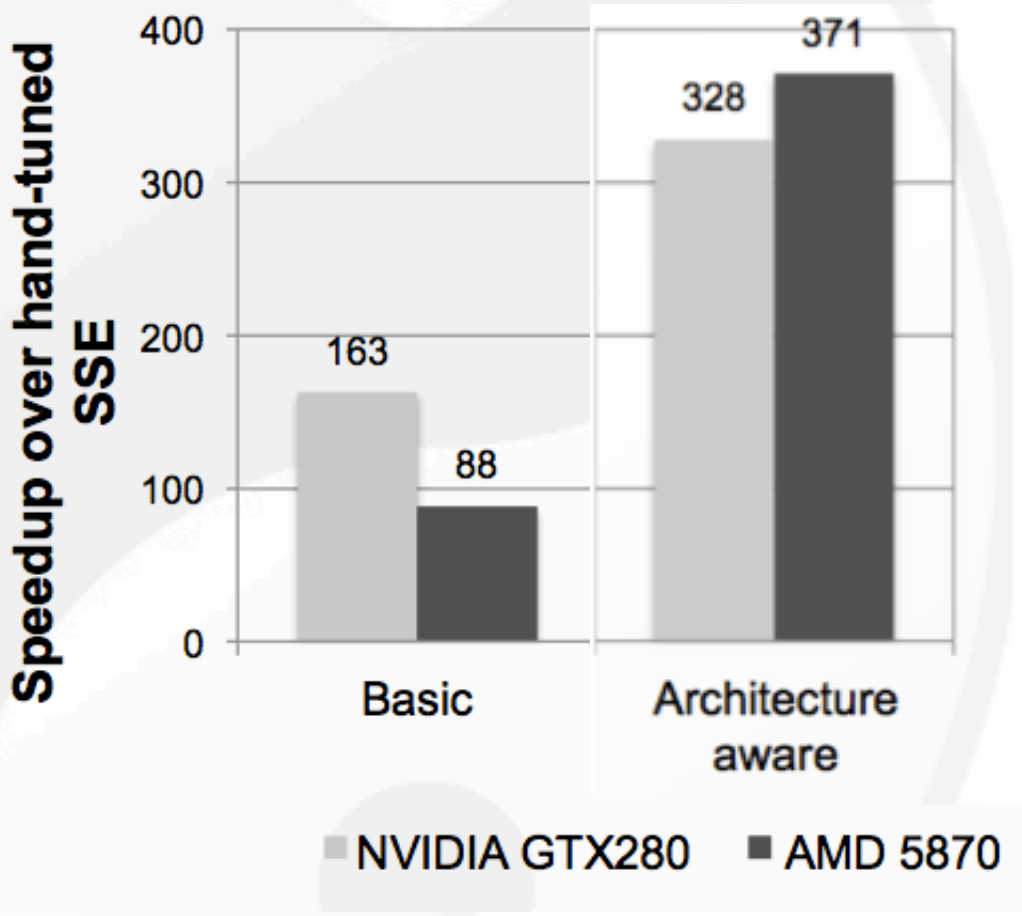
- OpenCL and the 13 Dwarfs — All 3 P's
- Source-to-Source Translation — "Programmability"
- Architecture-Aware Optimizations — Performance
- Heterogeneous Task Scheduling — All 3 P's

Computational Units *Not* Created Equal

- "AMD CPU != Intel CPU" and "AMD GPU != NVIDIA GPU"

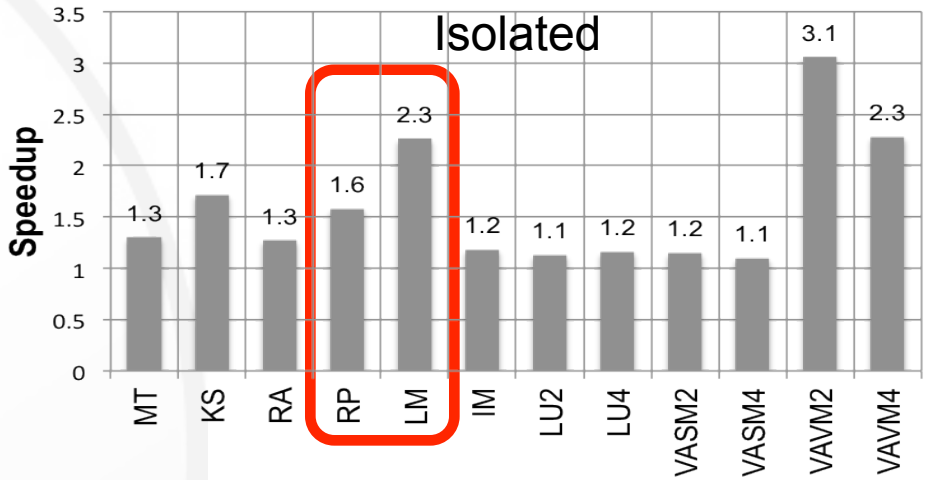


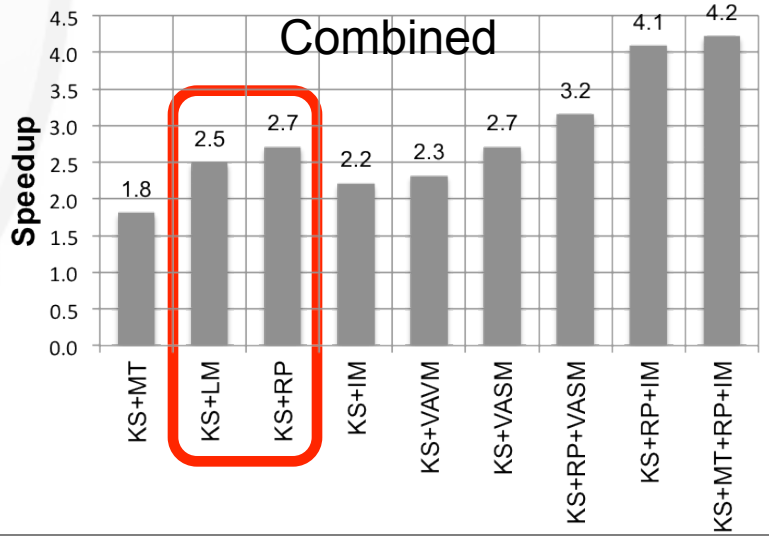
Summary: Architecture-Aware Optimization



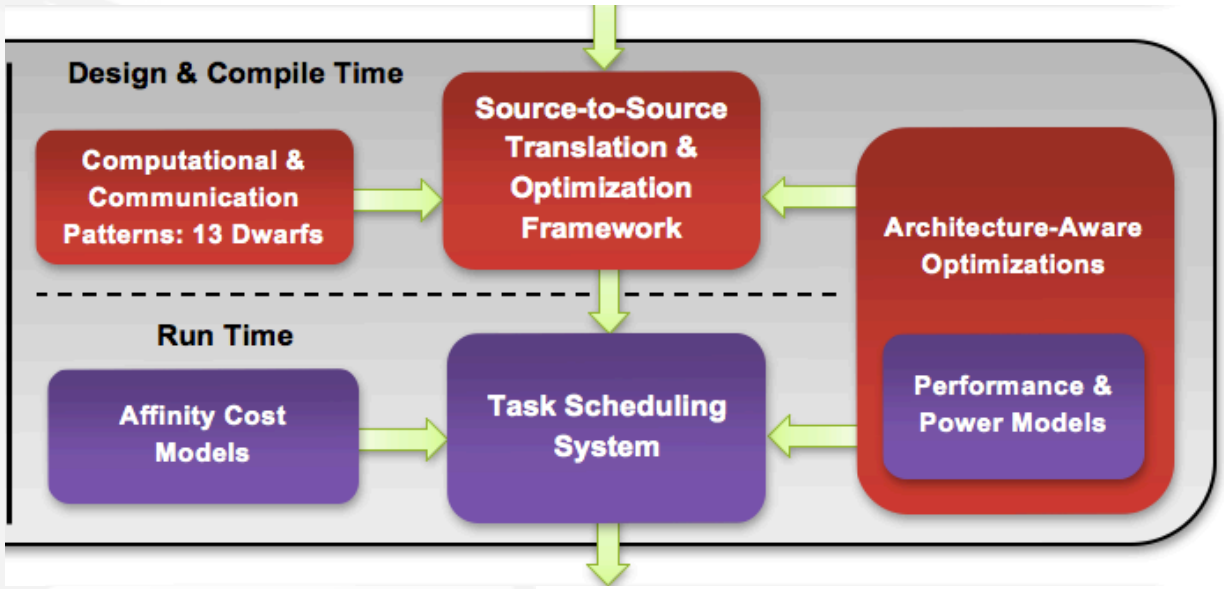
Architecture-Aware Optimization
(N-body Code for Molecular Modeling)

- 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





Roadmap



- OpenCL and the 13 Dwarfs — All 3 P's
- Source-to-Source Translation — "Programmability"
- Architecture-Aware Optimizations — Performance
- Heterogeneous Task Scheduling — All 3 P's

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.

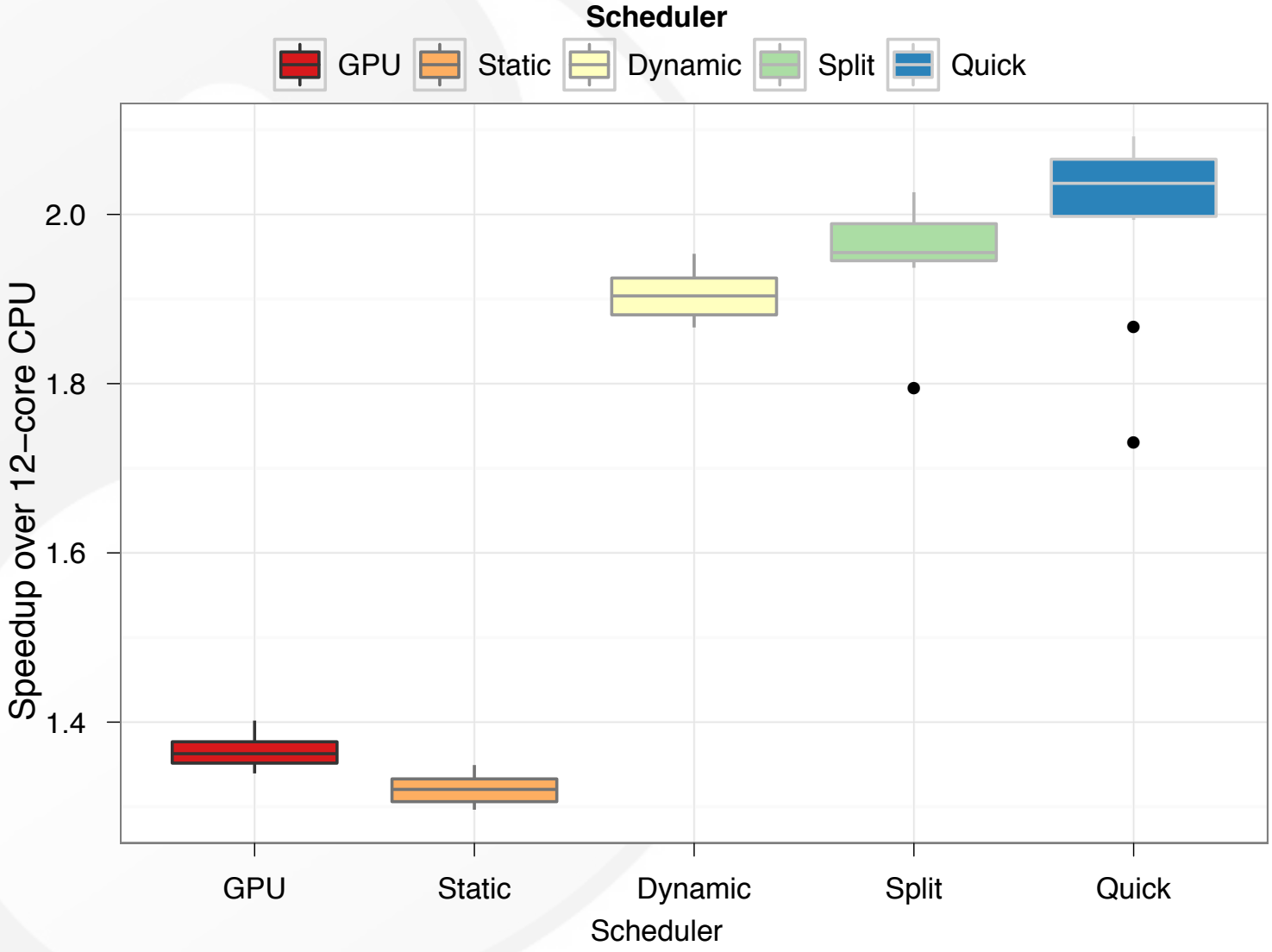
Goal

- 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!

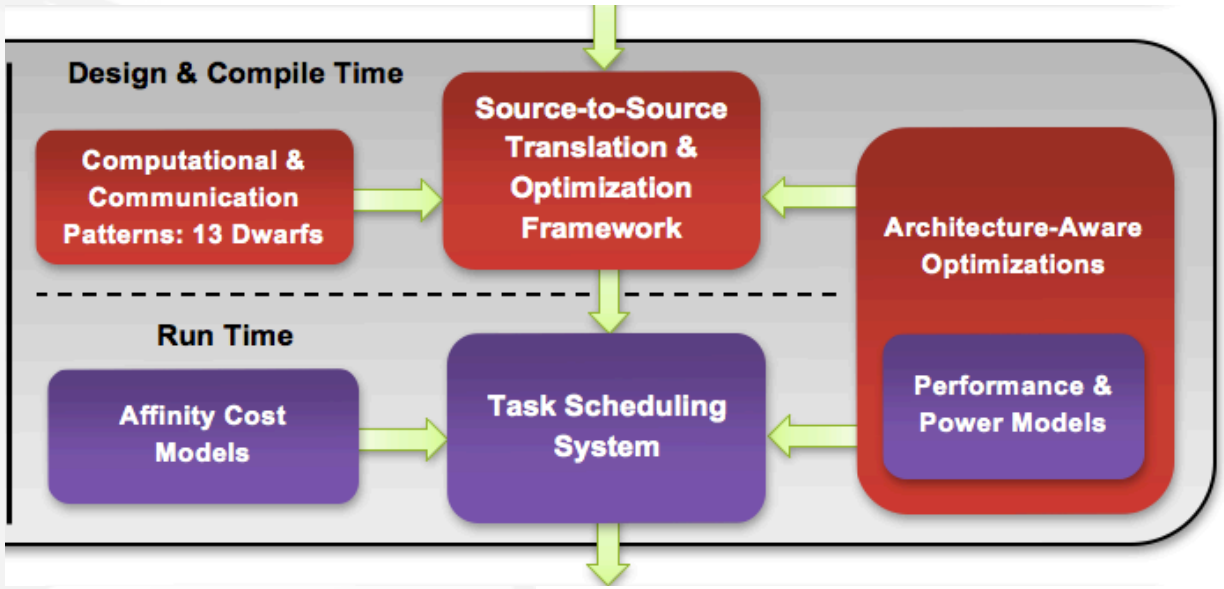
Experimental Testbed

- 12-core AMD Opteron 6174 CPU
- 1 NVIDIA Tesla C2050 GPU
- Linux 2.6.27.19
- CCE compiler with OpenMP accelerator extensions

KMeans with Heterogeneous Task Scheduling



Roadmap



- OpenCL and the 13 Dwarfs — All 3 P's
- Source-to-Source Translation — "Programmability"
- Architecture-Aware Optimizations — Performance
- Heterogeneous Task Scheduling — All 3 P's
  - Stay tuned for more on this soon …

Recent Publications on Heterogeneous Computing

- M. Daga, T. Scogland, and W. Feng, "Architecture-Aware Mapping and Optimization on a 1600-Core GPU," *17th IEEE Int'l Conf. on Parallel & Distributed Systems*, December 2011.
- M. Elteir, H. Lin, and W. Feng, "StreamMR: An Optimized MapReduce Framework for AMD GPUs," *17th IEEE Int'l Conf. on Parallel & Distributed Systems*, December 2011.
- W. Feng, Y. Cao, D. Patnaik, and N. Ramakrishnan, "Temporal Data Mining for Neuroscience," *GPU Computing Gems,* Editor: W. Hwu, Elsevier/Morgan-Kaufmann, February 2011.
- K. Bisset, A. Aji, M. Marathe, and W. Feng, "High-Performance Biocomputing for Simulating the Spread of Contagion over Large Contact Networks," *BMC Genomics*, 2011.
- M. Elteir, H. Lin, and W. Feng, "Performance Characterization and Optimization of Atomic Operations on AMD GPUs," *IEEE Cluster*, Sept. 2011.
- M. Daga, A. Aji, and W. Feng, "On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing," *Symposium 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

People

- Research Staff
  - Mark K. Gardner, Ph.D.
  - Heshan Lin, Ph.D.
- Ph.D. Students
  - Ashwin Aji
  - Tom Scogland
  - Balaji Subramaniam
  - Shucai Xiao (graduating)
  - Jing Zhang
- M.S. Students
  - Mayank Daga (now @ AMD)
  - Gabriel Martinez (now @ Intel)

SyNeRG

synergy.cs.vt.edu

THE GREEN 500

www.green500.org

CHREC
NSF Center for High-Performance Reconfigurable Computing

www.chrec.org

- B.S./M.S. Students
  - Carlo del Mundo
  - Tyler Kahn
  - Kenneth Lee
  - Paul Sathre

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

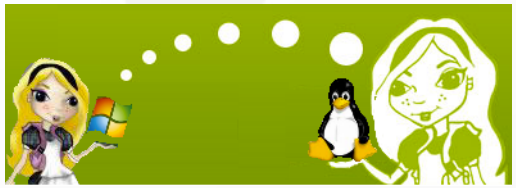
http://www.mpiblast.org/

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/