



An Ecosystem for the New HPC – Heterogeneous Parallel Computing

Wu Feng

Dept. of Computer Science and Electrical & Computer Engineering

Virginia Bioinformatics Institute

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?

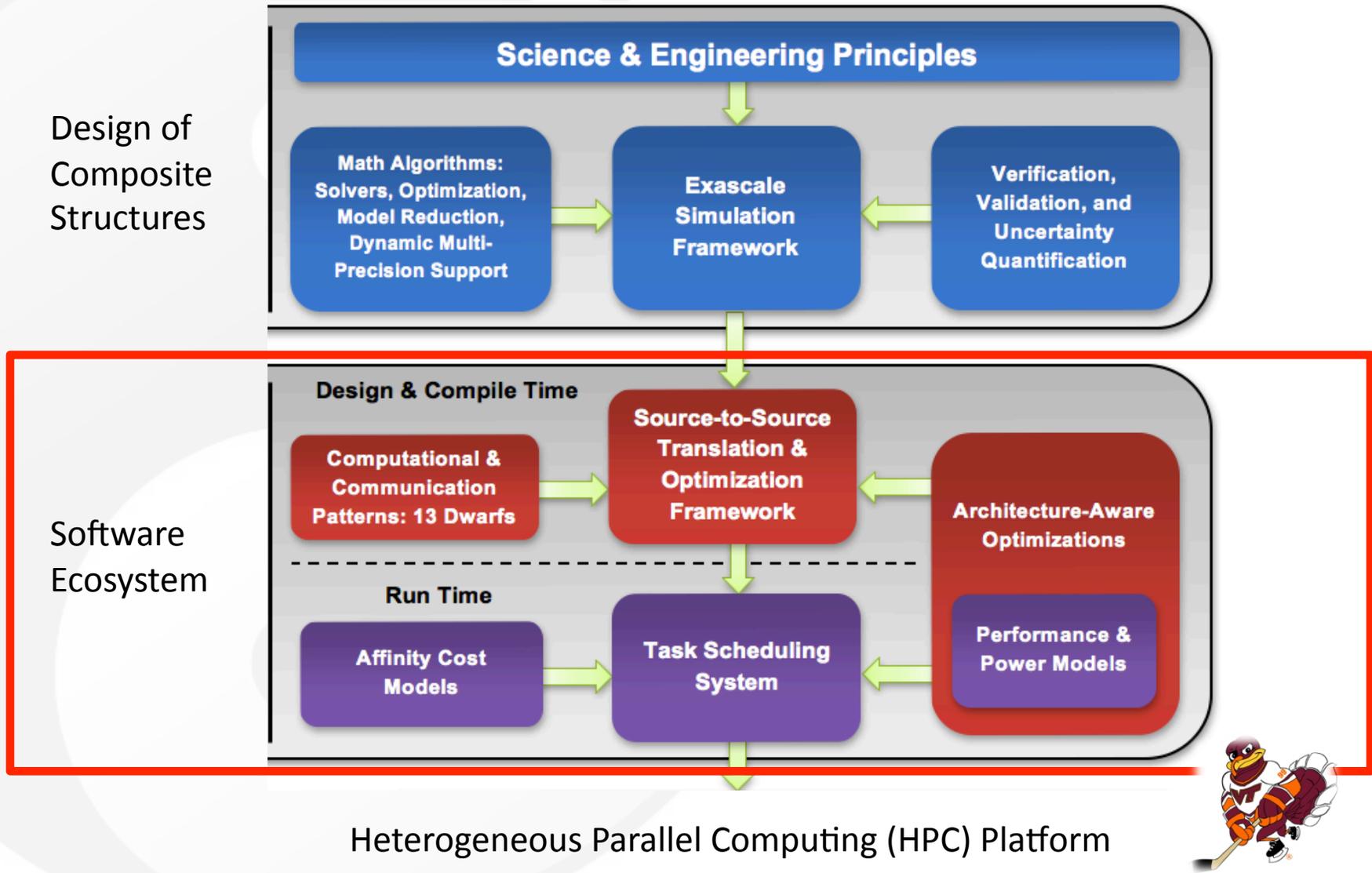
† H. Sutter, “The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software,” *Dr. Dobbs’s Journal*, 30(3), March 2005. (Updated August 2009.)

The Berkeley View *

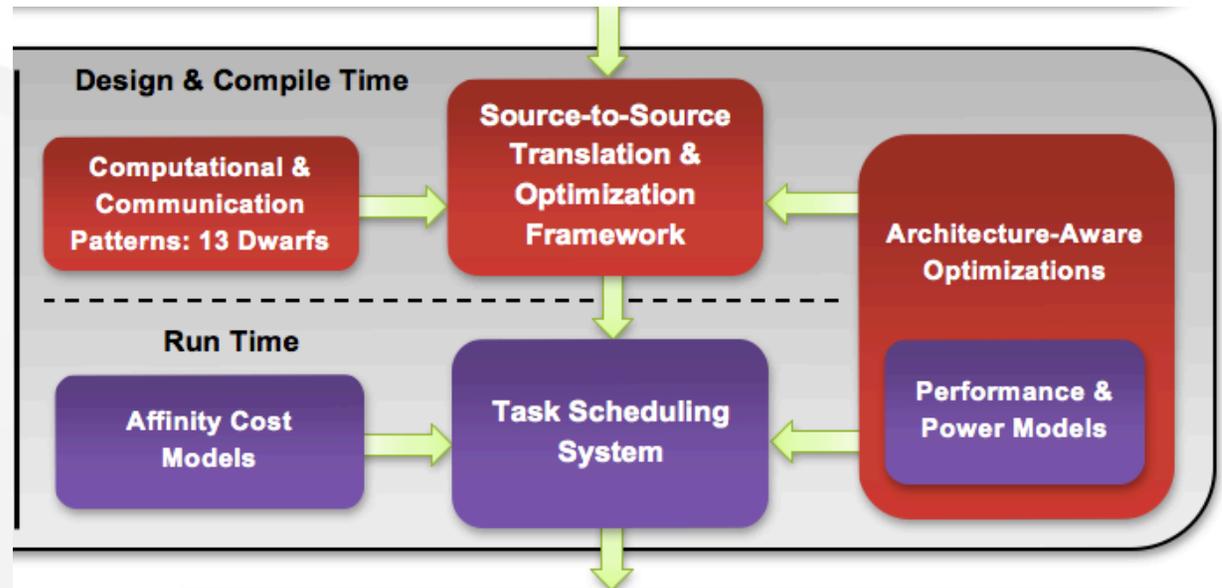
- Traditionally, applications target existing hardware and programming models
- Instead, design hardware keeping future applications in mind
 - ... Software of the future?

* 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.

An Ecosystem for Heterogeneous Parallel Computing



Roadmap

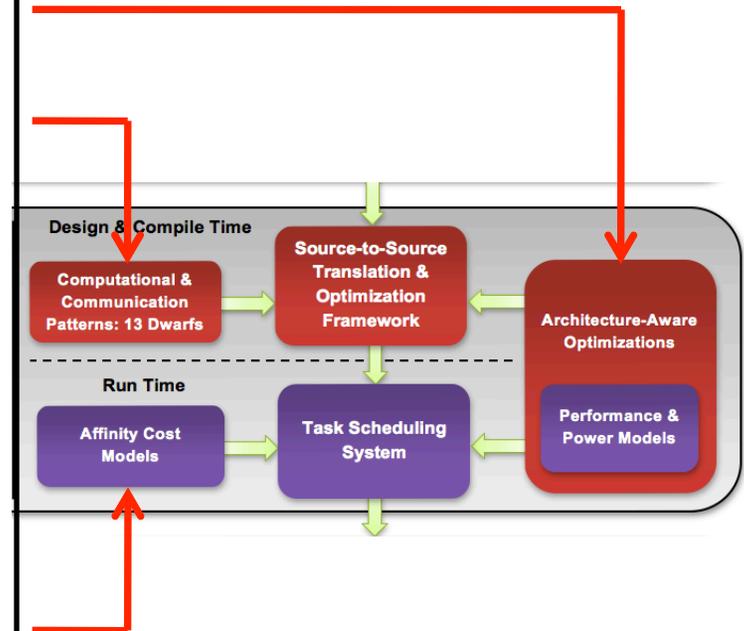


Focus on extracting node-level parallelism to support exascale

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

How Did the Roadmap Come About?

NSF Center for High-Performance Reconfigurable Computing



Project Goal

- Deliver personalized heterogeneous supercomputing to the masses
 - Heterogeneity of hardware devices plus ...
 - Enabling software that tunes the parameters of the hardware devices with respect to performance, power, portability, and programmability

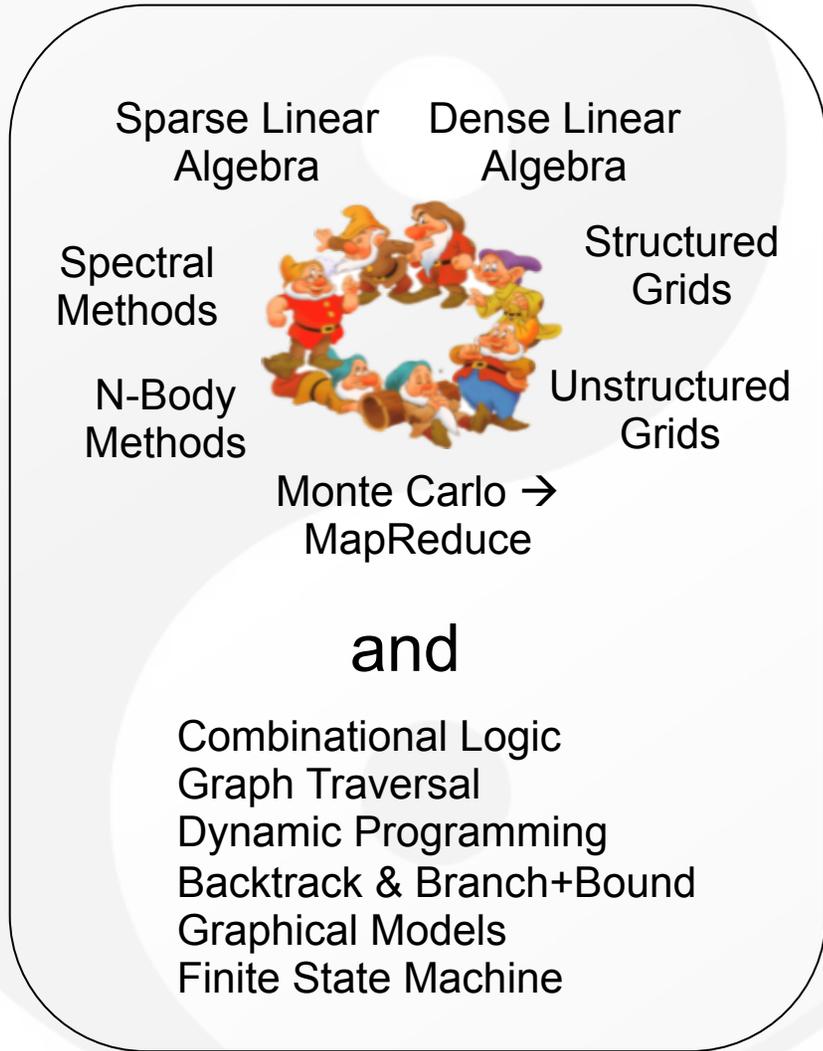
via a benchmark suite of computational dwarfs and apps



Project Outcome

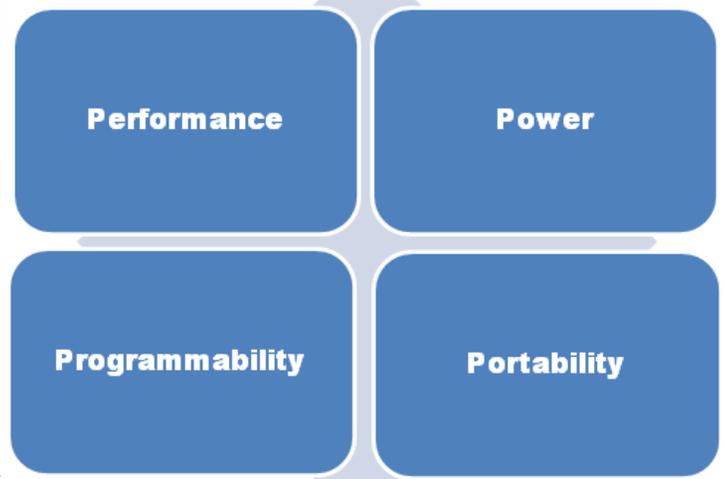
- A multi-dimensional understanding of how to optimize **performance**, **programmability**, **power**, **portability**, or some combination thereof.
 - **Performance** (under Resource Constraints): # threads/block; # blocks/grid; configurable memory; mixed-mode arithmetic; and so on.
 - **Power**: Device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

Step 1: OpenCL and the 13 Dwarfs (+ 4 P's + Accelerator Platforms)



+

+



Status of OpenCL Dwarfs

| Dwarf | Application(s) | OpenCL |
|--------------------------|---------------------------------|------------|
| Dense Linear Algebra | LU Decomposition | Completed |
| Sparse Linear Algebra | Matrix Multiplication | Completed |
| Spectral Methods | FFT | Completed |
| N-Body Methods | GEM RRU(LANL) | Completed |
| Structured Grids | SRAD | Completed |
| Unstructured Grids | CFD Solver | Completed |
| MapReduce | PSICL-BLAST | Prototyped |
| Combinational Logic | CRC | Completed |
| Graph Traversal | BFS Bitonic Sort | Completed |
| Dynamic Programming | Needleman-Wunsch | Completed |
| Backtrack & Branch+Bound | N-Queens Travelling Salesman | Completed |
| Graphical Models | HMM | Completed |
| Finite State Machine | Temporal Data Mining | Completed |

Caveat: “Write once, run anywhere” but *not* necessarily well.

Completed
 Prototyped



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 | 6636.6 (2119.0) | 141.4 | 393.7 | 5233.0 (2387.0) |
| 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 parentheses are from June 2010.)

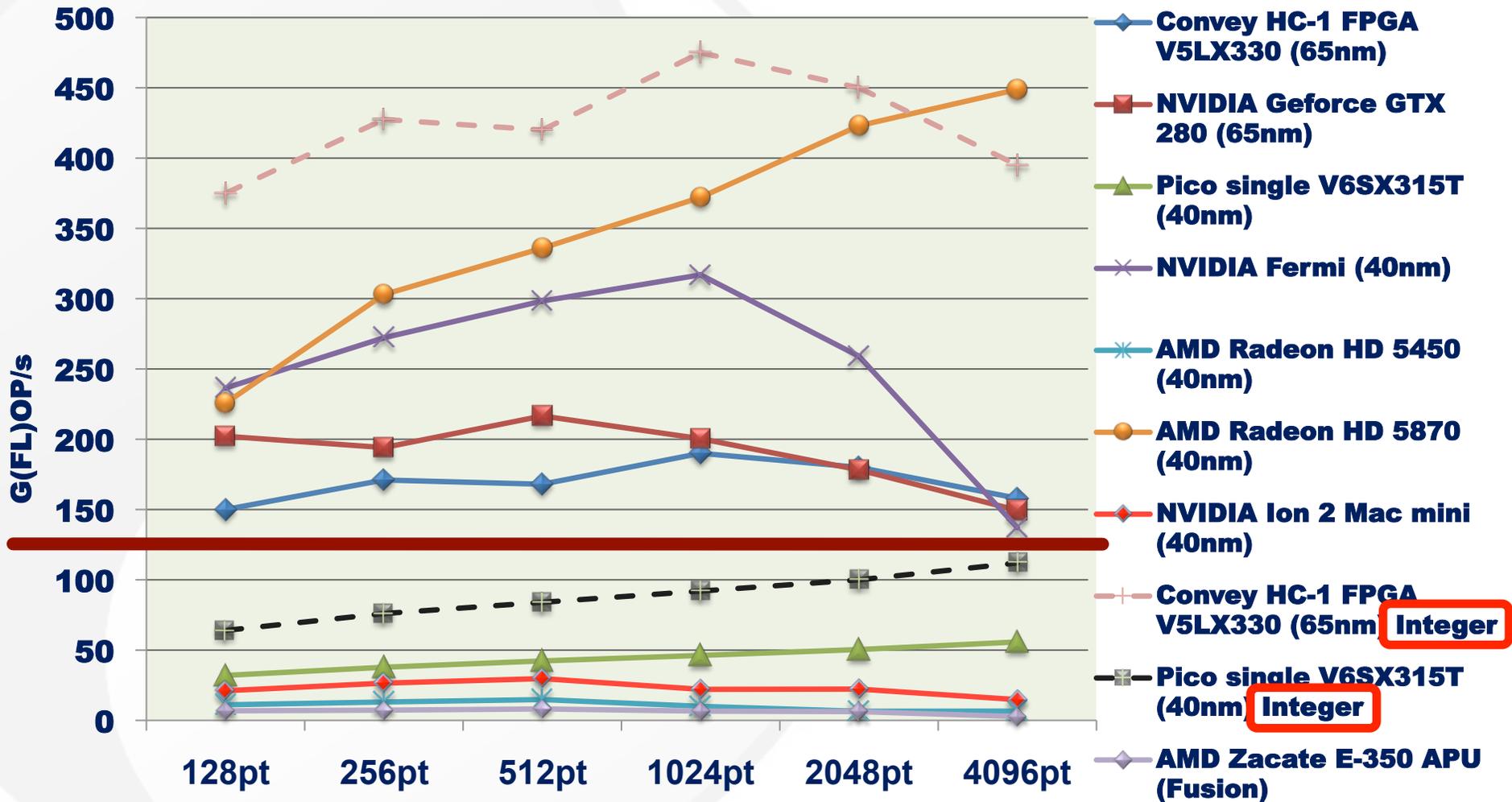
Estimated (or from Convey) in gray

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



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

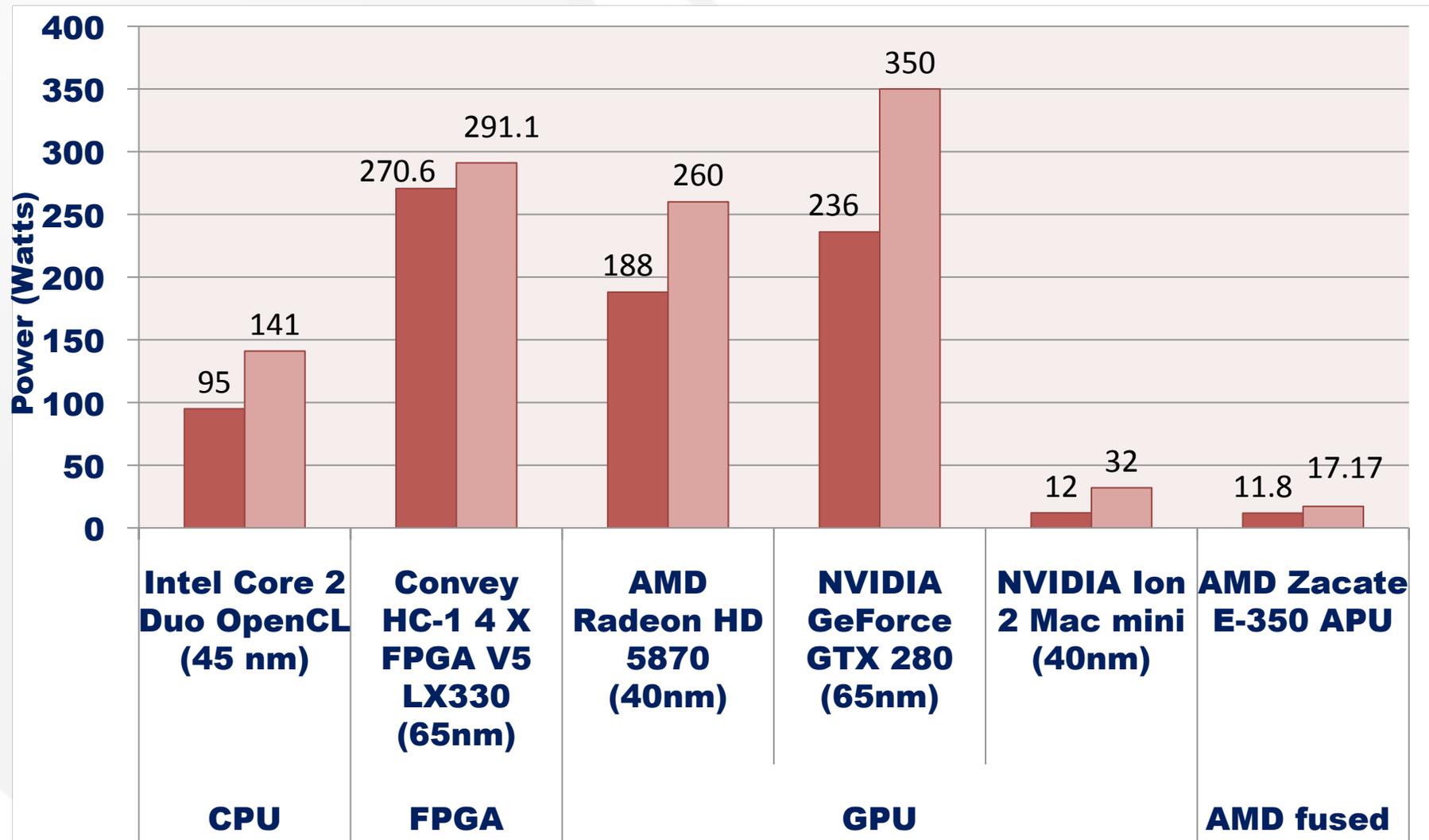
Performance: N-Queens (Backtrack)

| 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 |

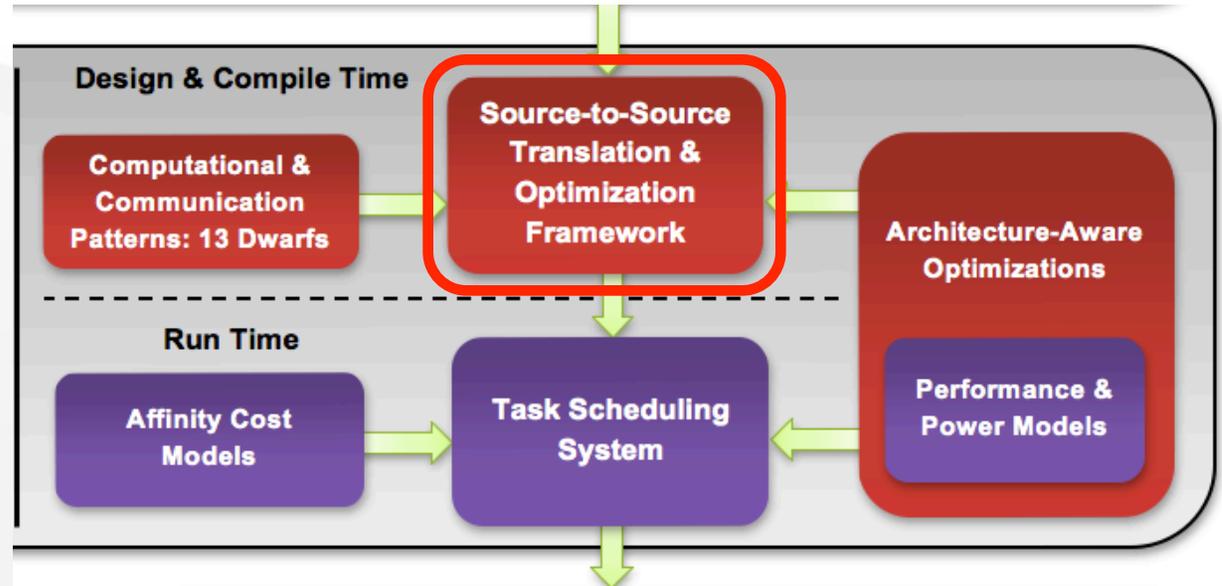
* Estimated execution time based on implementation on a single Virtex 5 FPGA

** Code downloaded from <http://forum.beyond3d.com/showthread.php?t=56105>

Total System Power: Idle vs. At Load



Roadmap



Focus on extracting node-level parallelism to support exascale

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

Step 2: Creating an Army of Dwarfs via “CU2CL” †

- CU2CL: CUDA-to-OpenCL 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 VT technical report.

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
- “Write once, run anywhere ...”

Why *Not* CU2CL?

- CU2CL only does *source-to-source translation* at present
 - No architecture-aware optimization
 - CUDA and OpenCL version compatibility

CU2CL Goals

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

GCC



Clang



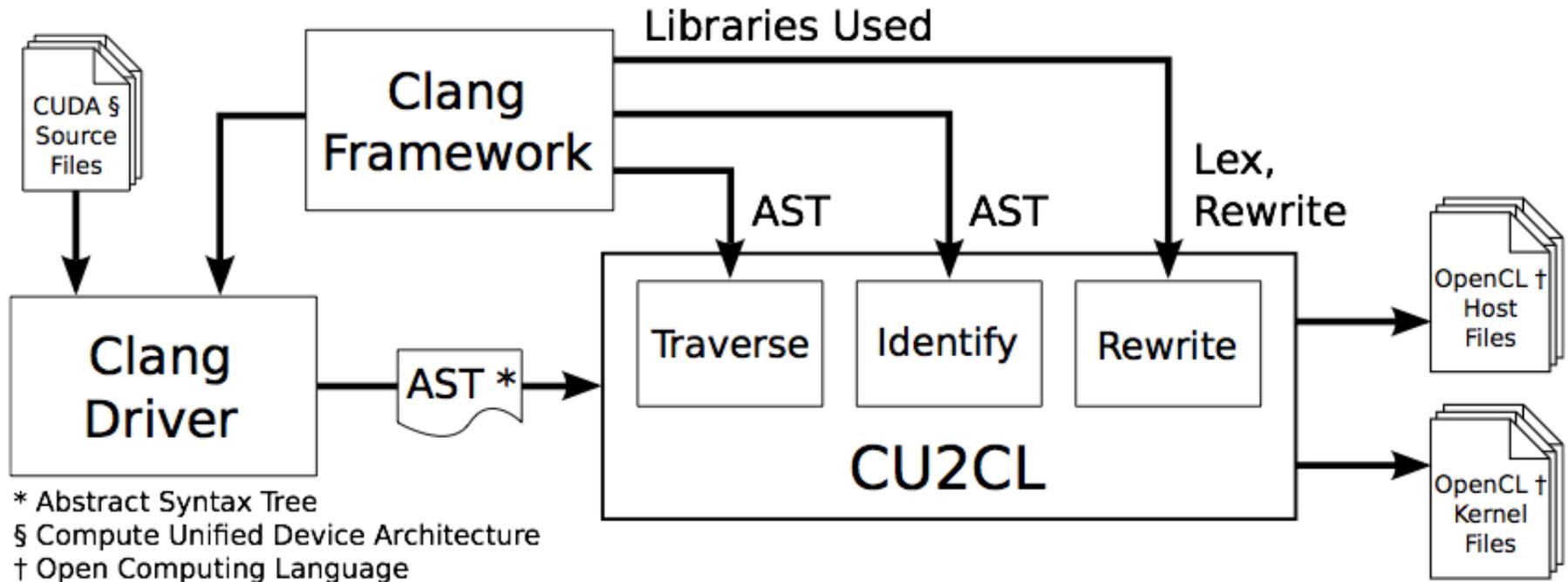
Open64



Cetus



Overview of CU2CL Translation



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

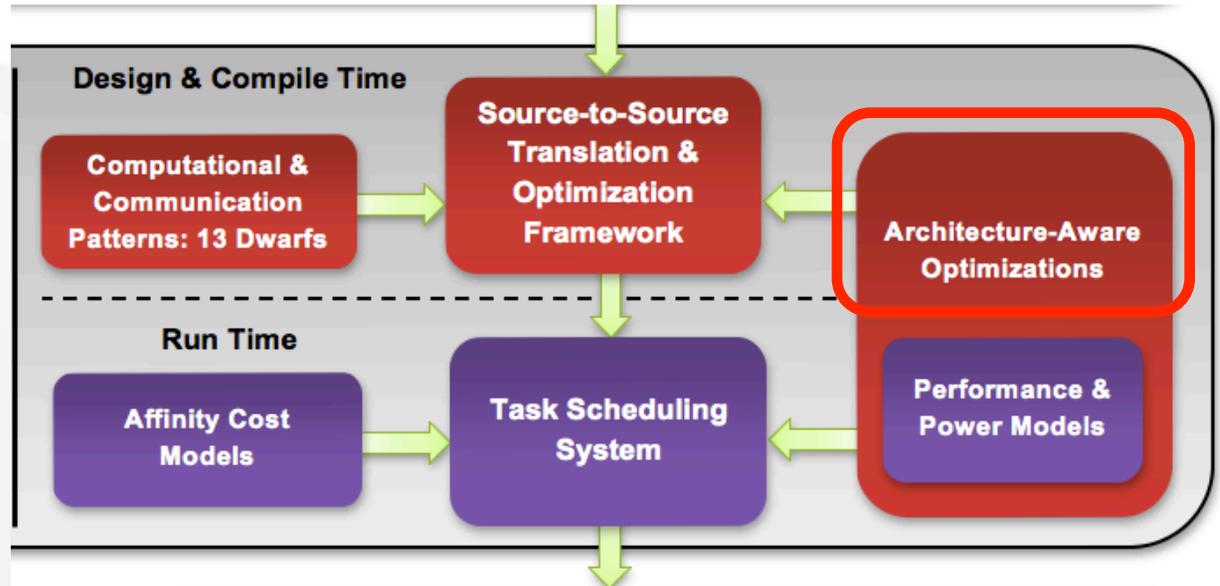
Breakdown of CU2CL Translator Performance

| Source | Application | Total Translation Time (s) | CU2CL Time (ms) | CUDA Lines | Preprocessed Lines |
|----------|----------------------|----------------------------|-----------------|------------|--------------------|
| CUDA SDK | asyncAPI | 0.331 | 3.35 | 136 | 13743 |
| | bandwidthTest | 0.623 | 7.98 | 891 | 34766 |
| | BlackScholes | 0.606 | 5.24 | 347 | 34222 |
| | matrixMul | 0.607 | 5.47 | 351 | 34209 |
| | scalarProd | 0.327 | 3.75 | 171 | 13835 |
| | vectorAdd | 0.287 | 3.11 | 147 | 11605 |
| Rodinia | Back Propagation | 0.300 | 4.46 | 313 | 12765 |
| | Breadth-First Search | 0.301 | 4.51 | 306 | 12594 |
| | Hotspot | 0.297 | 4.90 | 328 | 11810 |
| | Needleman-Wunsch | 0.303 | 5.46 | 418 | 12815 |
| | SRAD | 0.303 | 6.56 | 541 | 12778 |

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% |

Roadmap

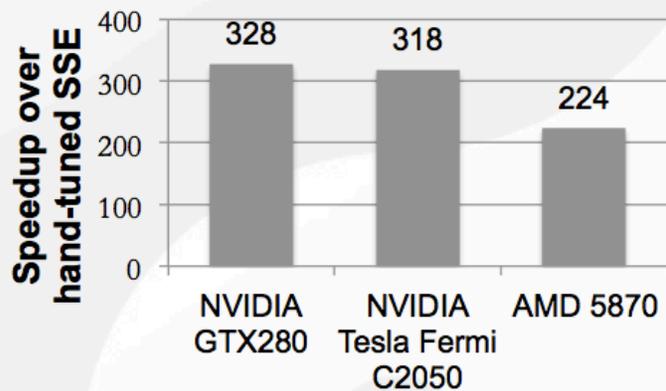


Focus on extracting node-level parallelism to support exascale

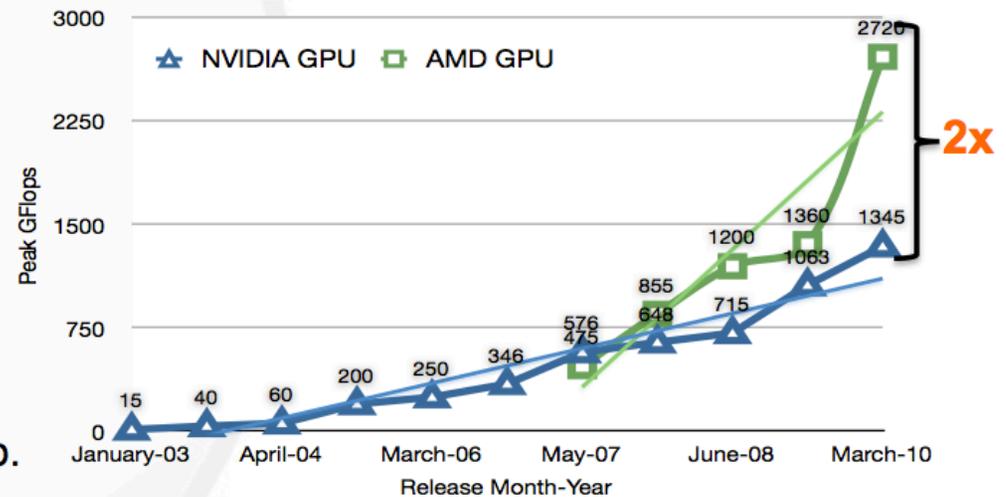
- OpenCL and the 13 Dwarfs
- Source-to-Source Translation (and Optimization)
- **Architecture-Aware Optimizations**
- Performance & Power Modeling
- Affinity-Based Cost Modeling
- Heterogeneous Task Scheduling

Computational Units *Not* Created Equal

- “AMD CPU != Intel CPU” and “AMD GPU != NVIDIA GPU”
- Initial Performance Results for N-body Dwarf



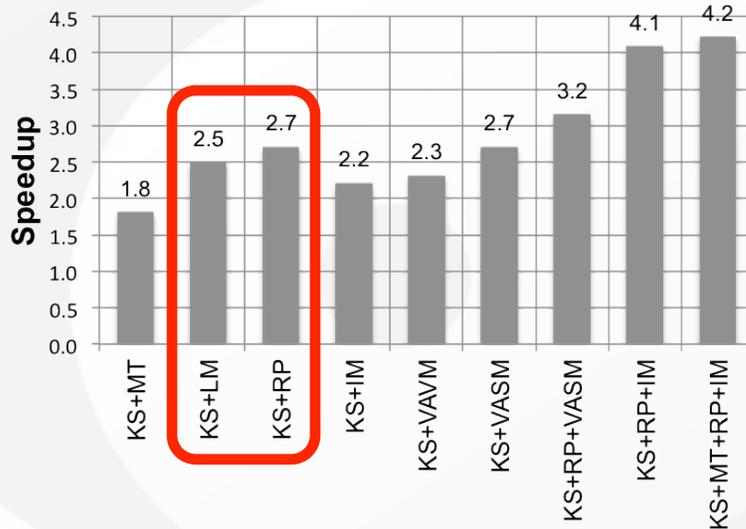
Performance of a Molecular Modeling App.



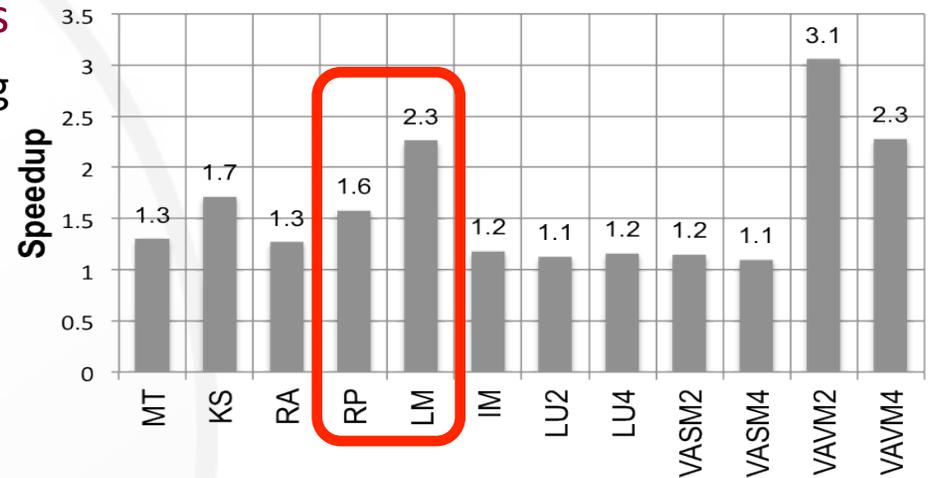
Step 3: Architecture-Aware Optimizations (N-body Code for Molecular Modeling)

- Optimization techniques on AMD GPUs
 - Removing conditions → kernel splitting
 - Local staging
 - Using vector types
 - Using image memory

Combined Optimizations

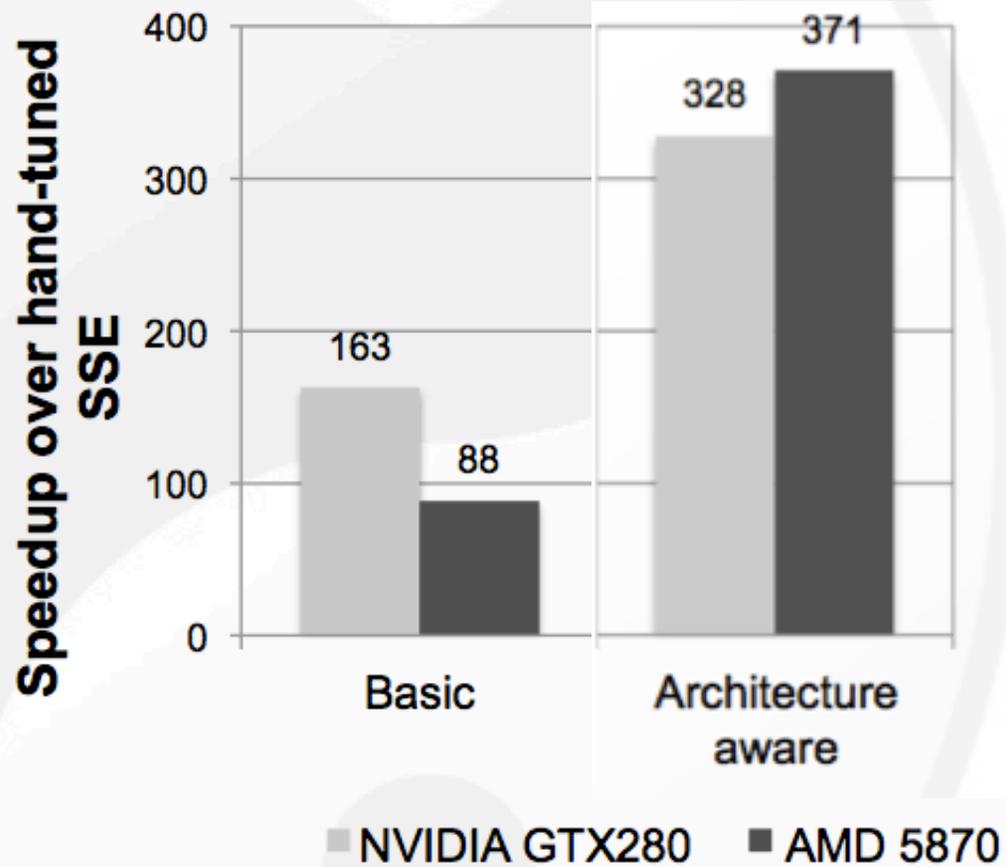


Isolated Optimizations

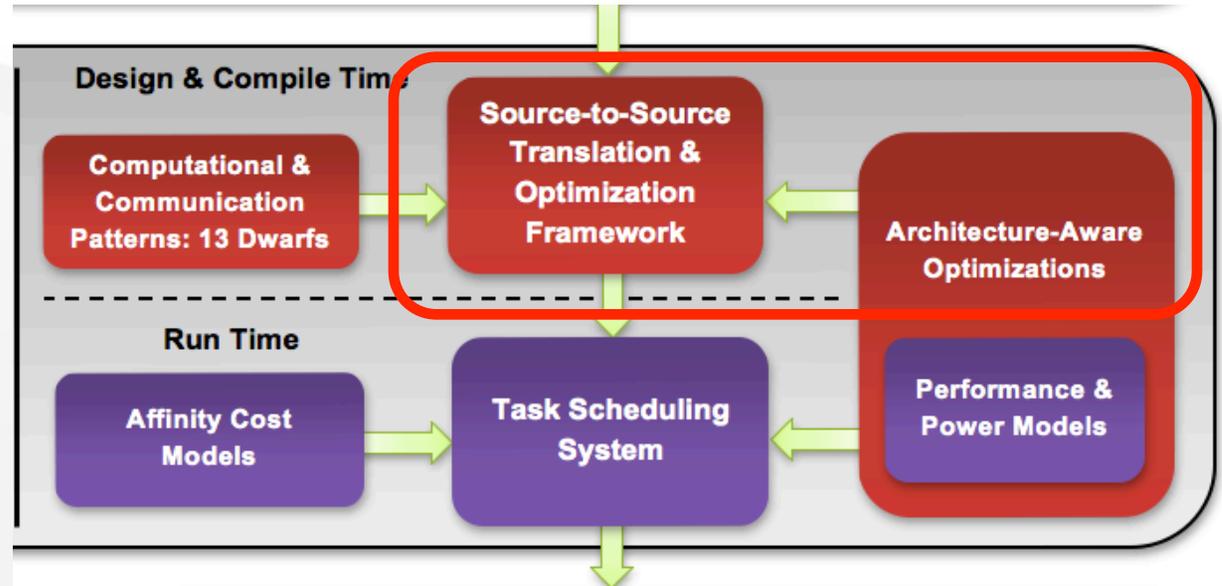


- Speedup over basic OpenCL GPU implementation
 - Isolated optimization
 - Combined optimizations (32 combinations)

Summary: Architecture-Aware Optimization



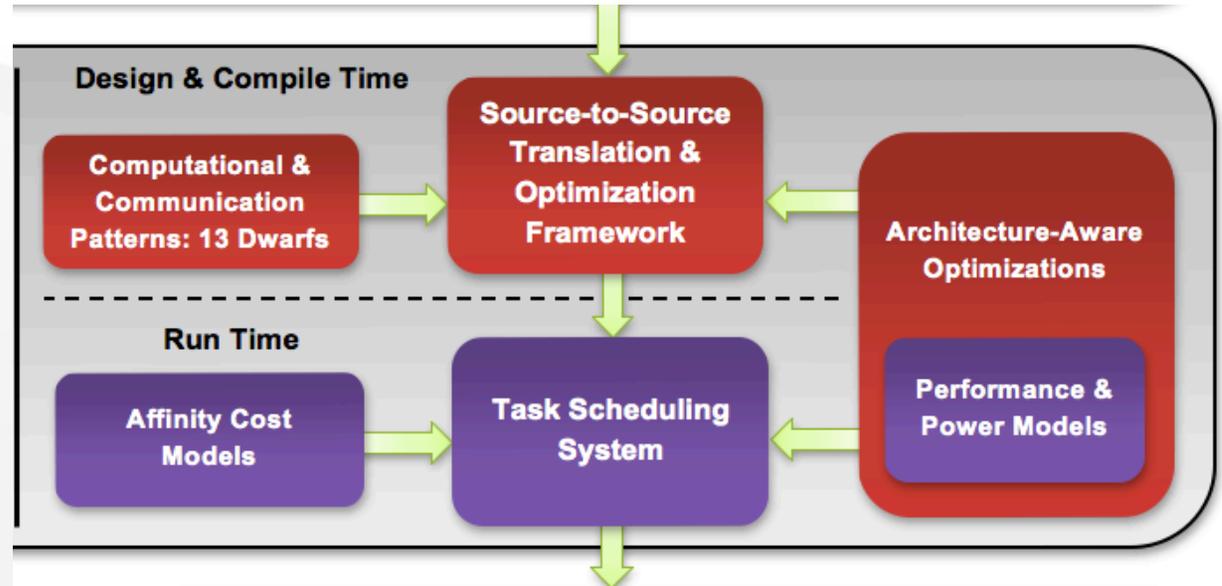
Future Work



Source-to-Source Translation + Architecture-Aware Optimizations
= Source-to-Source Translation and Optimization Framework

- Auto-tuning framework to find the *best combination of optimizations* for a particular node architecture

Roadmap



Focus on extracting node-level parallelism to support exascale

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

What platforms are we doing this on?

Infrastructure

- A farm of personal desktop supercomputers, accelerated by GPUs or FPGAs
 - CPUs: AMD Magny Cours and Intel Nehalem
 - GPUs: AMD Radeon HD 5870, NVIDIA GTX 280/480/580, NVIDIA Tesla Fermi C2050 and C2070
 - Integrated or Fused CPU+GPU: Apple Mac Mini, AMD Zacate E-350 (i.e., Fusion)
- Fire: A Dense 20-Tflop CPU+GPU Cluster
- HokieSpeed: A 425-Tflop CPU+GPU Cluster for Computing and In-Situ Visualization
 - 200+ dual-socket CPU nodes → 400+ sockets
 - 400+ GPUs

Selected Publications ...

- 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.

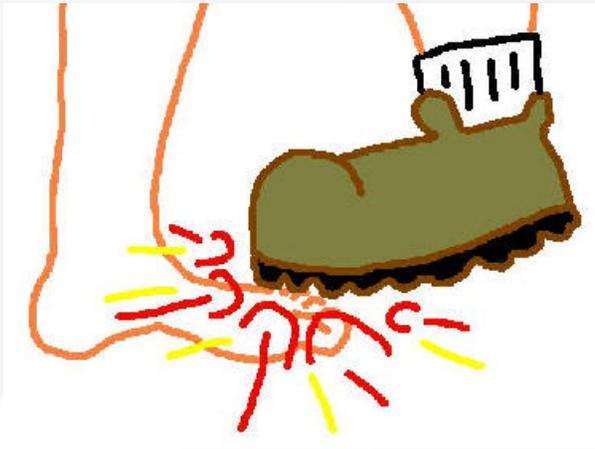
Lessons Learned

- 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 a potential way to get both

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

Lessons Learned



Funding Acknowledgements



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



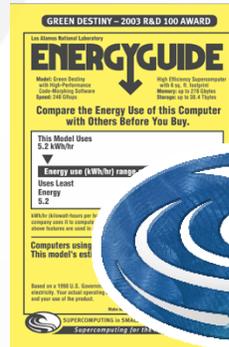
<http://synergy.cs.vt.edu/>



<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/>