OpenCL and the 13 Dwarfs

Wu FENG

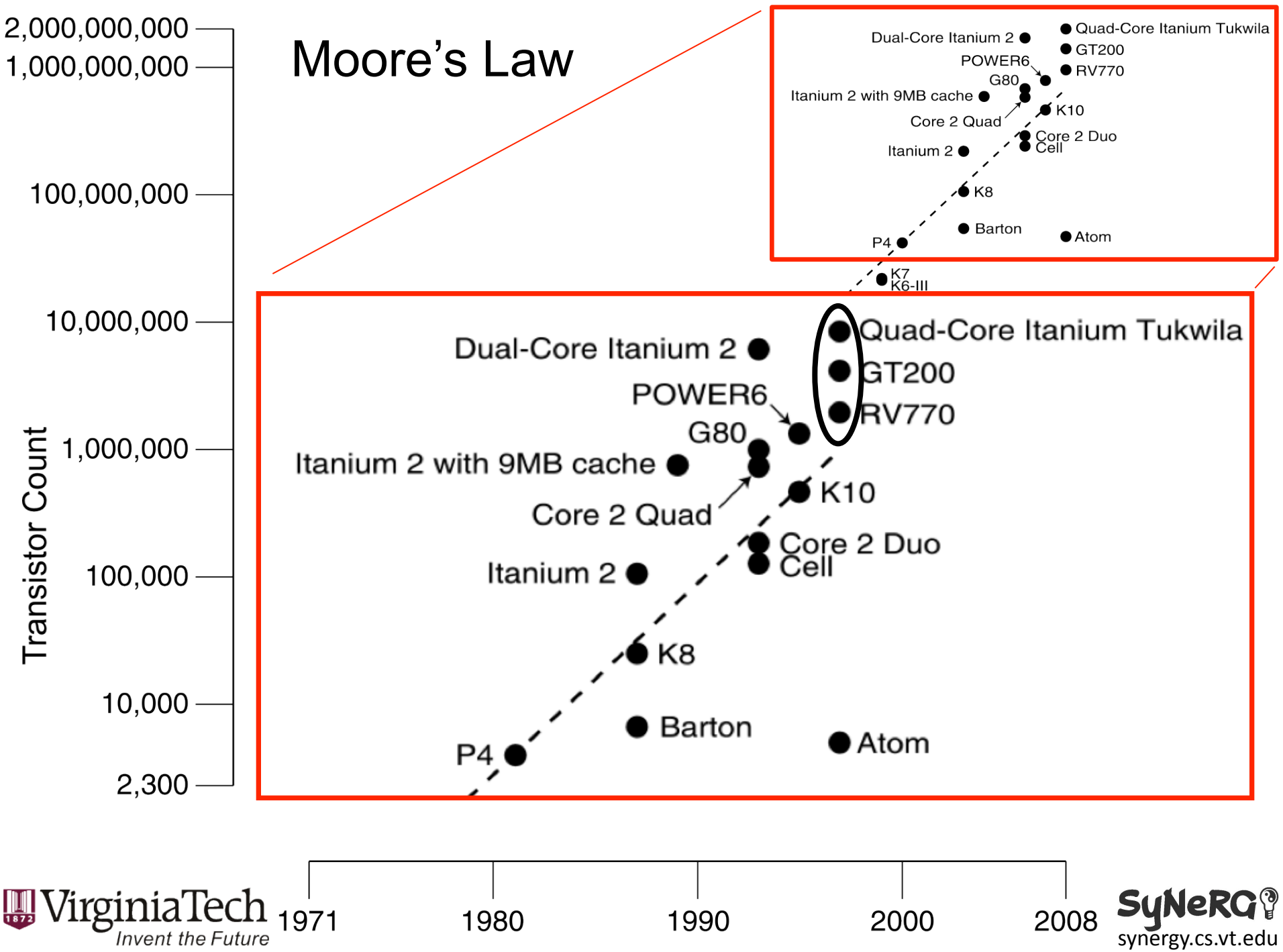
Dept. of Computer Science and Dept. of Electrical & Computer Engineering

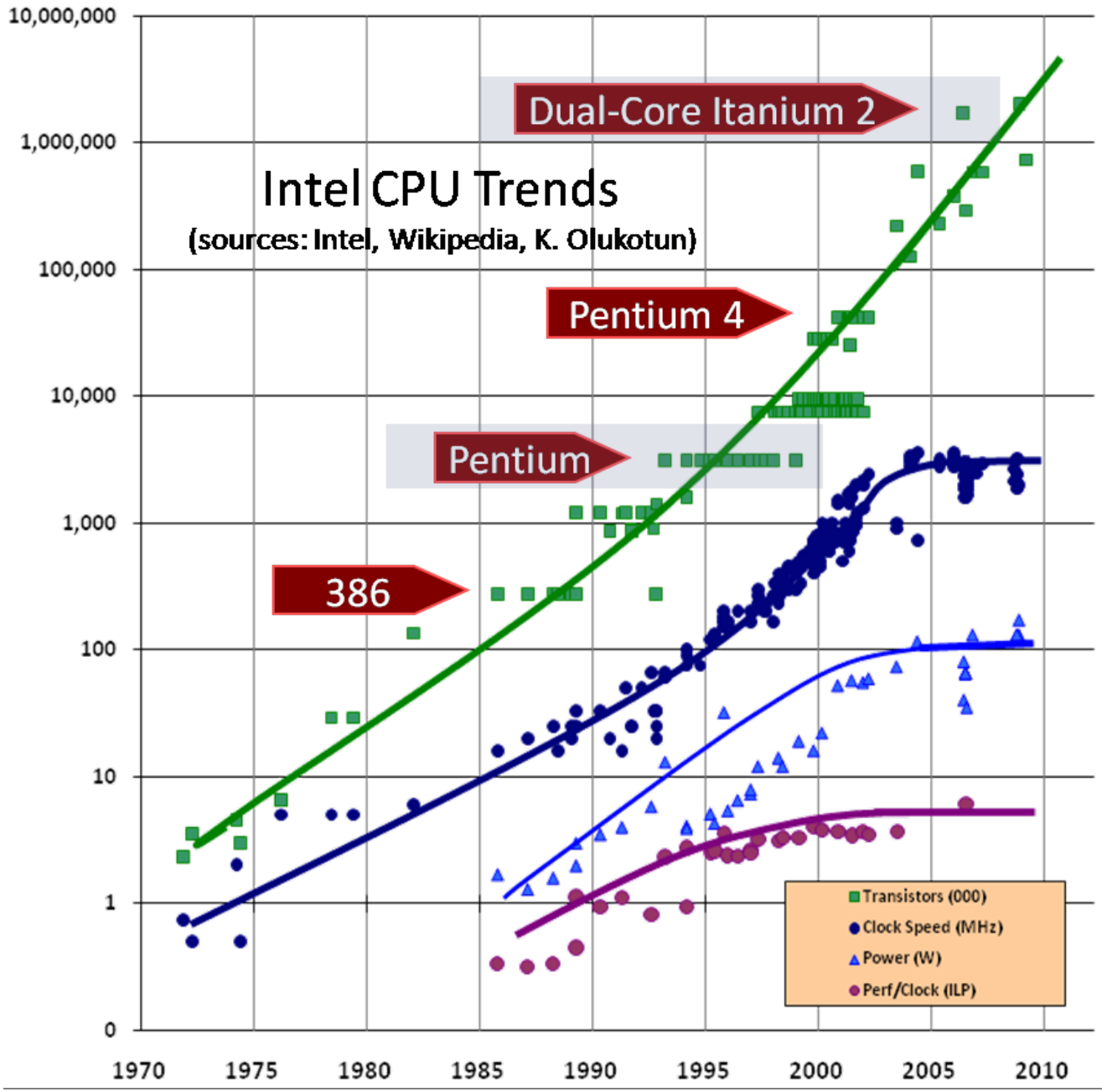
Virginia Bioinformatics Institute

(Dept. of Cancer Biology and Translational Science Institute, Wake Forest U.)

VirginiaTech
Invent the Future

SyNeRG
synergy.cs.vt.edu


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



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.)

© W. Feng, 2011

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

© W. Feng, 2011

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.)

© W. Feng, 2011

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.

© W. Feng, 2011

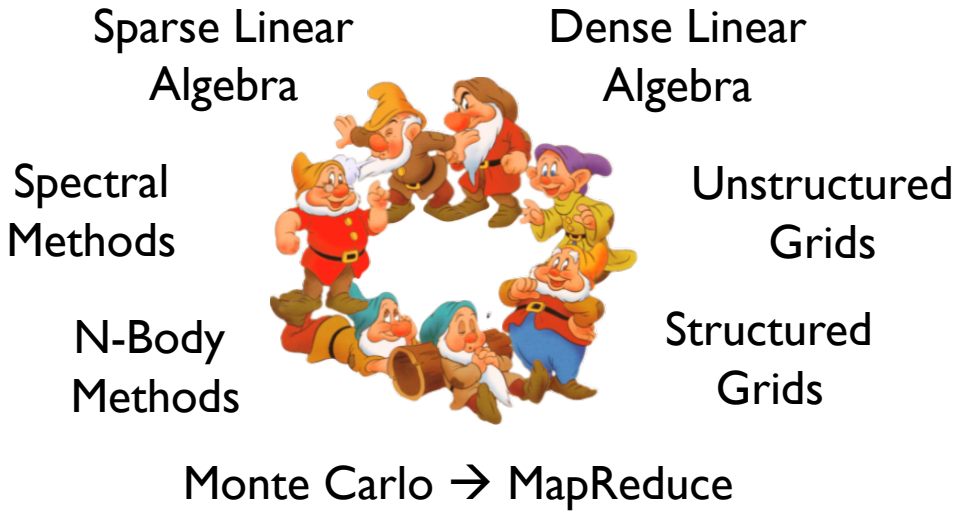
What is the Software of the Future?

- Focus on consumer applications
  - Desktop
  - Server
  - Enterprise
- How to parallelize?
  - Draw inspiration from more challenging HPC applications

© W. Feng, 2011

The Seven Dwarfs of High-Performance Computing

- A dwarf is an algorithmic method that
  … captures a pattern of computation and communication
  - Inspired by Phil Colella, who identified *seven* numerical methods important for science and engineering.

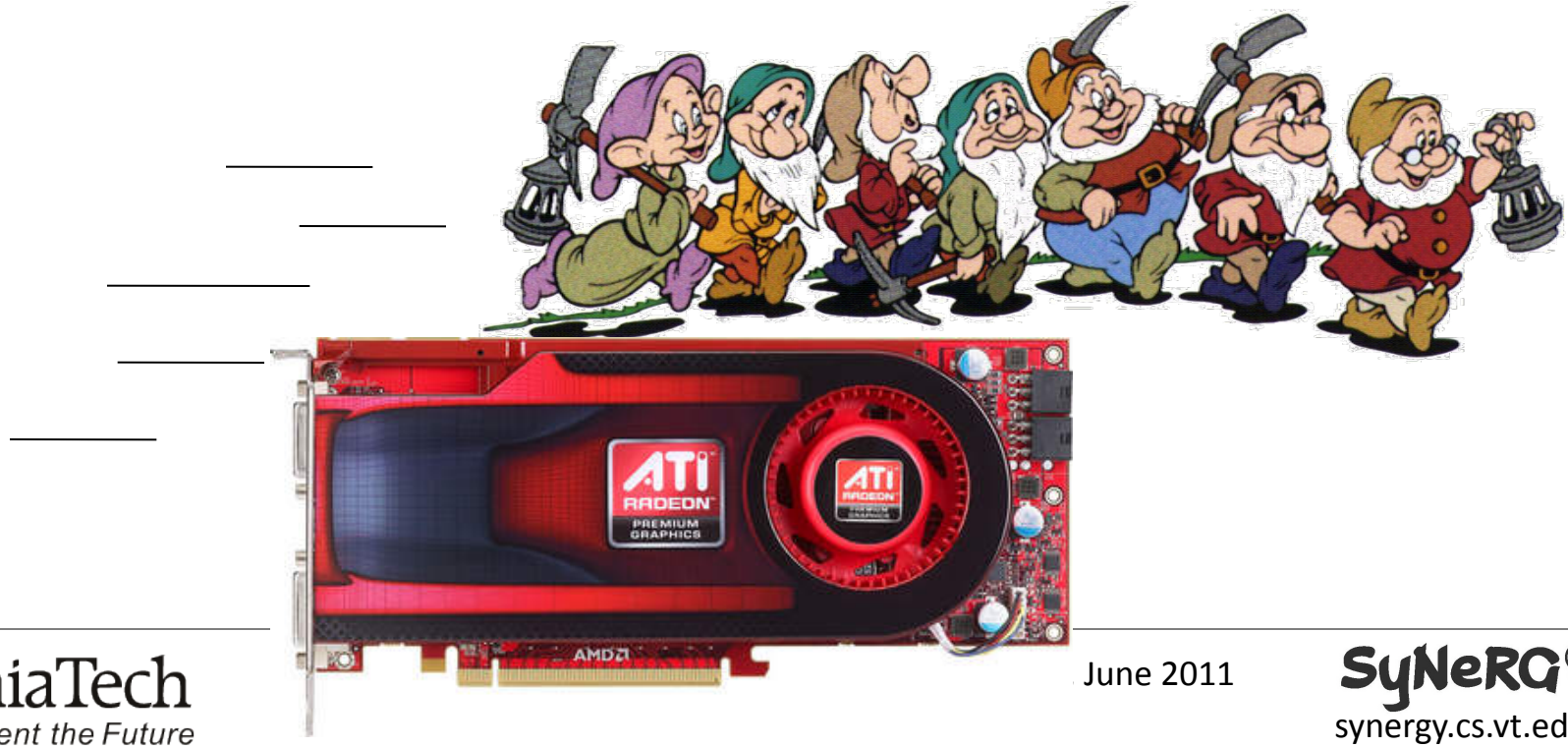


  - Wisdom: These will be common computational patterns of current and *future* interest.

© W. Feng, 2011

Outline

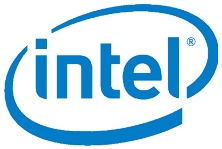
- Overview of *OpenCL and the 13 Dwarfs*
- Status of *OpenCL and the 13 Dwarfs*
- Creating an Army of Dwarfs with "CU2CL" (pronounce as "cuticle")
- Performance and Power of *OpenCL and the 13 Dwarfs*



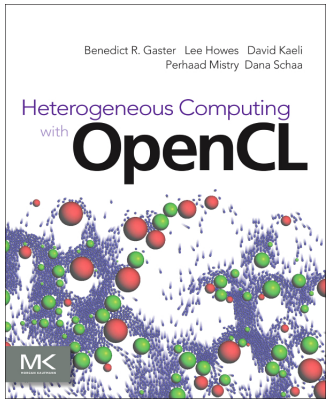
What is OpenCL?

OpenCL: Open Computing Language

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



  - Managed by non-profit consortium: Khronos Group



For more information,

- Attend an OpenCL tutorial at AFDS or IEEE/ACM SC
- Check out *Heterogeneous Computing with OpenCL* by Gaster et al.
- Visit *OpenCL Zone* (http://developer.amd.com/zones/openclzone/pages/default.aspx)

© W. Feng, 2011

What are the 13 Dwarfs?

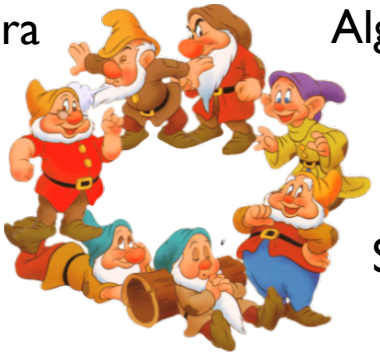
- A (computational) dwarf is an algorithmic method that *captures a pattern of computation and communication*
  - Inspired by Phil Colella, who identified *seven* numerical methods important for science and engineering

- Benchmark Suite of 13 Computational Dwarfs & Apps

Sparse Linear Algebra

Dense Linear Algebra

Spectral Methods



Unstructured Grids

N-Body Methods

Structured Grids

Monte Carlo → MapReduce

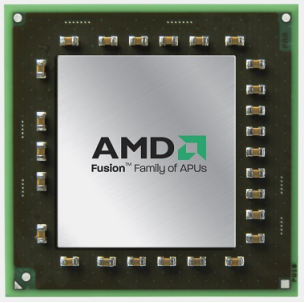
+

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

© W. Feng, 2011

What is "OpenCL and the 13 Dwarfs"?

- Goal
  - Provide common algorithmic methods, i.e., dwarfs, in a language that can "run anywhere" (CPU, GPU, or even FPGA), i.e., OpenCL



- Part of a larger umbrella project (2008-2011) funded by the NSF Center for High-Performance Reconfigurable Computing (CHREC)*

CHREC
NSF Center for High-Performance Reconfigurable Computing

* Talk with Wu after the talk if you have interest in joining.

© W. Feng, 2011

Why "OpenCL and the 13 Dwarfs"?

Evaluate and guide architectural innovation

- Conventional Approach
  - Study a benchmark suite based on existing programs, e.g., SPEC.
- Obstacle to Innovation in Parallel Computing?
  - Unclear how to express a parallel computation best.
- Opportunity
  - Delineate application requirements to not be overly specific to individual applications or the optimizations used for certain hardware platforms.

    → draw broader conclusions about hardware requirements

Refer to http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

© W. Feng, 2011

N-Body Methods

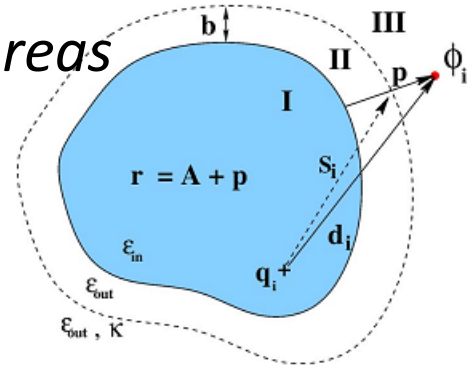
Calculations that depend on interactions between discrete points

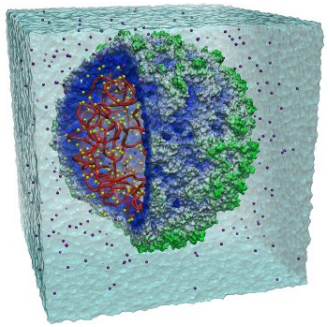
Approaches

- In particle-particle methods
  - Every point depends on all others
    - O(N2): Brute-force method
- Hierarchical particle methods
  - Combine forces or potentials from multiple points
    - O(N log N): Barnes-Hutt
    - O(N log N): Hierarchical charge partitioning
    - O(N): Fast multipole

Application Areas

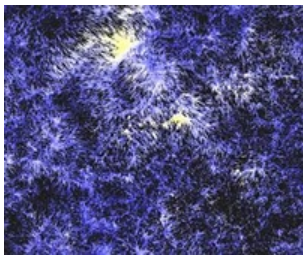
Molecular Modeling





Molecular Dynamics

Cosmology (Roadrunner Universe @ LANL)



Recent GPU References from our Literature Cache

- "Accelerating Electrostatic Surface Potential Calculation with Multiscale Approximation on Graphics Processing Units" *J. Molecular Graphics & Modelling*, 2010.
- "Multi-Dimensional Characterization of Electrostatic Surface Potential Computation on Graphics Processors," *BMC Bioinformatics*, 2011.

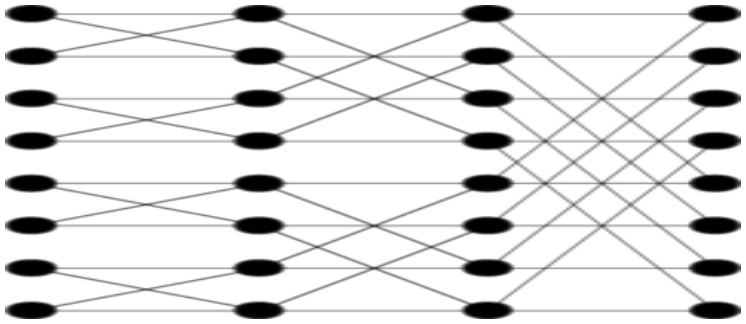
© W. Feng, 2011

Spectral Methods

Spectral domain computations transformed from temporal or spatial domains and solved numerically

Approach

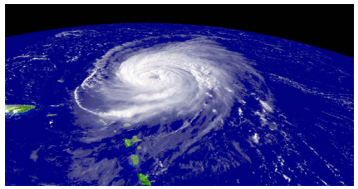
- 'Butterfly' pattern dependencies between stages of transformation
  - MAD operations on complex #
    - O(N log N): FFT



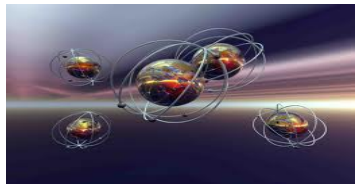
Computational Organization

Application Areas

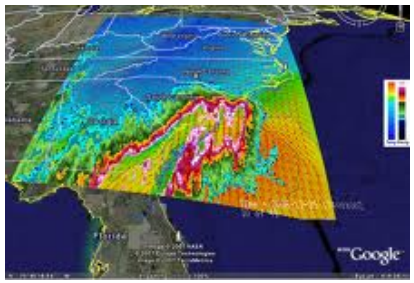
Fluid Dynamics



Quantum Mechanics



Weather Prediction



Recent GPU Reference from our Literature Cache

- "On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing" *Symp. on Application Accelerators for High-Performance Computing*, July 2011.

© W. Feng, 2011

Sparse Linear Algebra

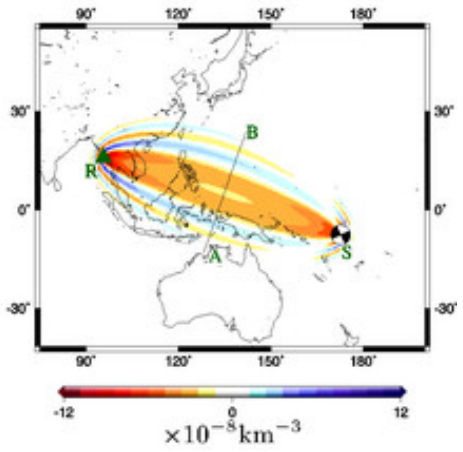
Multiplication involving matrices composed primarily of zeros

Approaches

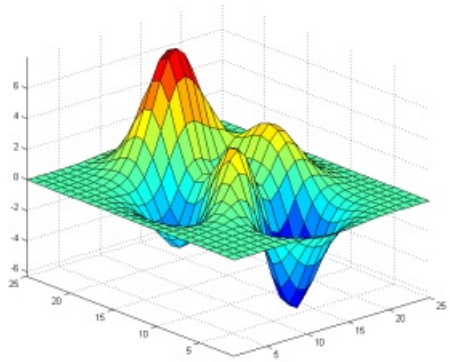
- Direct solving
  - Solve the problem with a series of operations
    - Large overhead
- Iterative methods
  - Use a series of successive approximations
    - Begins with a guess
    - Repeats until error is not significant

$$\begin{pmatrix} 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 8 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 4 & 0 \\ 0 & 1 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 7 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \end{pmatrix}$$

Application Areas & Kernels



Finite element analysis



Partial differential equations

© W. Feng, 2011

Dense Linear Algebra

Classic vector and matrix operations

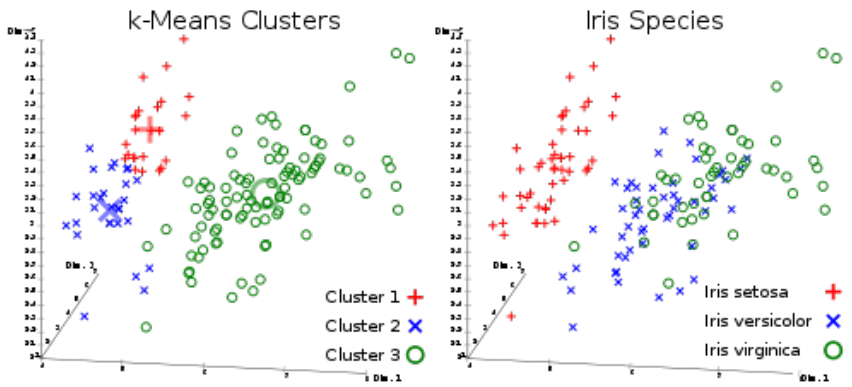
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \\ 0 & 1 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 30 \\ 40 \\ 80 \\ 20 \end{bmatrix}$$

Approaches

- Uniprocessor Mapping
  - Naïve Method
    - Loops down rows & columns
  - Blocked Method
    - Makes use of spatial locality
- Parallel Mapping
  - 2-D Block Cyclic Distribution
    - Provides better data distribution and load balancing

Application Areas

- Linear Algebra
  - LAPACK
  - ATLAS
- Data Mining
  - Streamcluster
  - K-means



Source: http://view.eecs.berkeley.edu

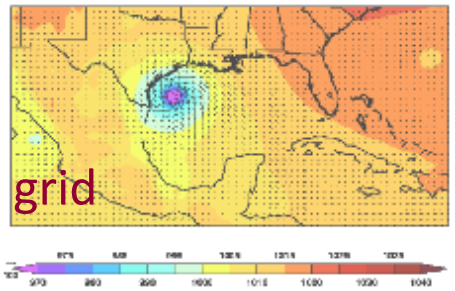
Recent GPU References from our Literature Cache

- "A First Look at Integrated GPUs for Green High-Performance Computing," *EnA-HPC*, Sept. 2010.
- "On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing" *Symp. on Application Accelerators for High-Performance Computing*, July 2011.

© W. Feng, 2011

Structured Grids

Computation steps update data in a regular multi-dimensional grid

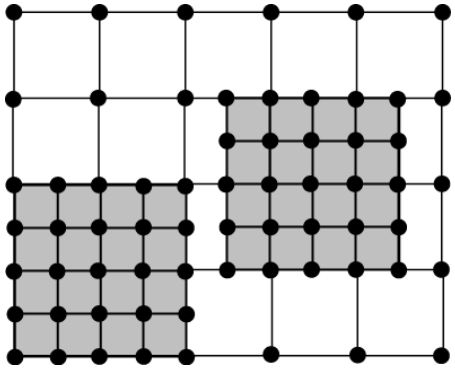


Approaches

- Regular Static Grid
  - Grid dimensions remain as they are from the start
    - High spatial locality, limited temporal locality
    - Parallel processors can be assigned subgrids
- Adaptive Mesh Refinement
  - Overlay higher resolution grids on areas of interest
    - Expensive boundary computations

Application Areas

- Image Processing
  - SRAD
- Physics Simulations
  - HotSpot



Recent GPU References from our Literature Cache

- "A First Look at Integrated GPUs for Green High-Performance Computing," *EnA-HPC*, Sept. 2010.
- "On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing" *Symp. on Application Accelerators for High-Performance Computing*, July 2011.

© W. Feng, 2011

Unstructured Grids

Computations that depend on neighbors in an irregular grid

- *Approaches*
  - In particle-particle methods
    - Every point depends on its neighbors
      - O(N): Brute-force method
- *Application Areas*
  - Computational fluid dynamics
  - Belief propagation

© W. Feng, 2011

MapReduce

Process subsets of data independently and merge results

Approaches

- Manual MapReduce
  - Application-specific "distribute" and "merge" functionality
    - mpiBLAST
- MapReduce Frameworks
  - Ease scaling of embarrassingly parallel applications
    - Hadoop and BOINC

Application Areas

- Distributed Searching
- Sequence Alignment
- Parallel Monte Carlo Simulations

Recent GPU References from our Literature Cache

- "Enhancing MapReduce via Asynchronous Data Processing," *16th IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS),* Dec. 2010.
- "Accelerating Protein Sequence Search in a Heterogeneous Computing System, " *25th Int'l Parallel and Distributed Processing Symp.,* May 2011.
- "On the Efficacy of a Fused CPU+GPU Processor for Parallel Computing" *Symp. on Application Accelerators for High-Performance Computing*, July 2011.

© W. Feng, 2011

Combinational Logic

Simple computation on large data sets, exhibiting bit-level parallelism

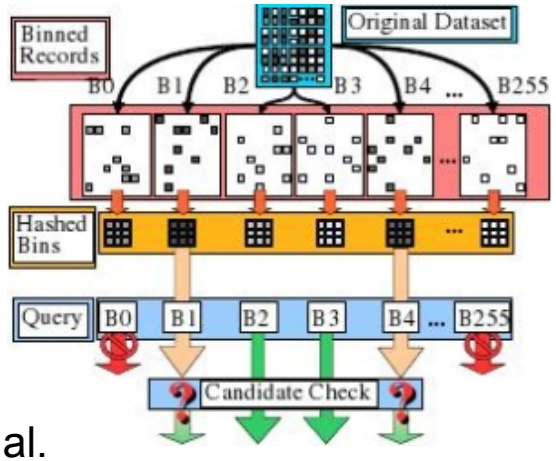
Approaches

- Often naturally parallel
  - Can use pipelining techniques to increase throughput
    - O(N): Naive

Application Areas

- Encryption & Decryption

- Hashing



Source:
John Owens et al.

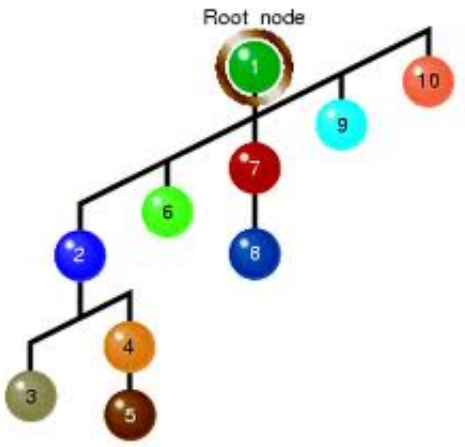
© W. Feng, 2011

Graph Traversal

Traverse objects and examine them as they are traversed

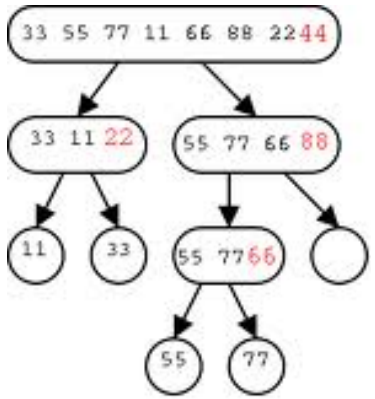
Approaches

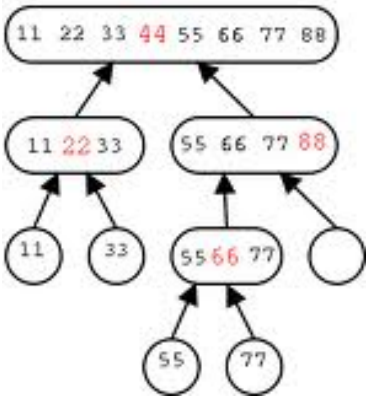
- Wavefront Movement
  - Traversal from single source
    - O(N): Brute-force method
- Divide and Conquer
  - Sorting Algorithms
    - O(Nlog(N)): Quicksort



Application Areas

- Searching
- Sorting
- Collision Detection





© W. Feng, 2011

Dynamic Programming

Compute solutions by solving simpler overlapping subproblems

Approaches

- Uniprocessor Mapping
  - Solve subproblem once & store
    - Assumes fast store and lookup times of solutions
- Parallel Mapping
  - Solutions to overlapping subproblems communicated or re-computed
    - Trade-off between compute and communicate costs

Recent GPU References from our Literature Cache
• Visit http://synergy.cs.vt.edu/.

Application Areas

- Graph problems
  - Floyd's All-Pairs shortest path
  - Bellman-Ford algorithm
- Sequence alignment
  - Needleman-Wunsch
  - Smith-Waterman

© W. Feng, 2011

Backtracking and Branch & Bound

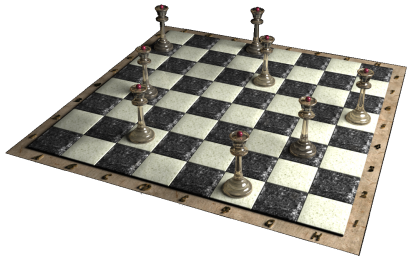
Branch-and-bound algorithms used to solve search and global optimization

Approaches

- Heuristic
- Dynamic Load Balance
  - Divide search space (fairly) among available processors.

Application Areas

- Artificial Intelligence: N-Queens



- Integer Linear Programming
- Boolean Satisfiability
- Combinatorial Optimization

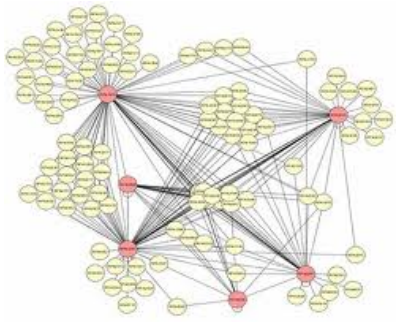
© W. Feng, 2011

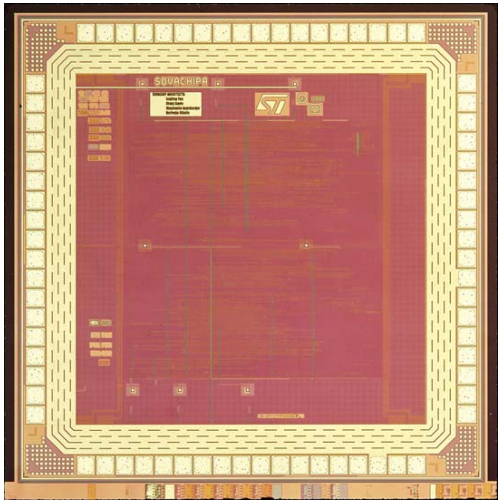
Graphical Models

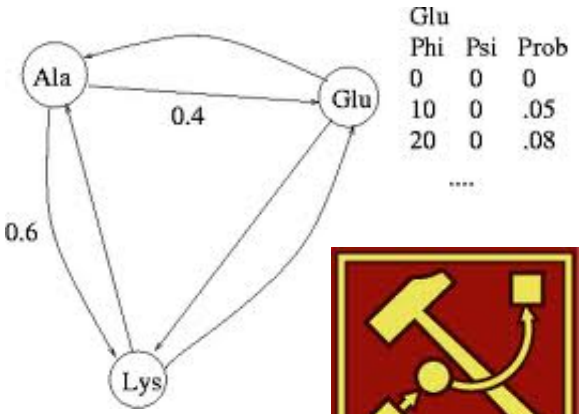
Construct graphs that represent random variables as nodes and conditional dependencies as edges

Application Areas

- Computational Biology
  - Sequence homology search
- Machine Learning
  - Hidden Markov models
- Embedded Computing
  - Viterbi decode







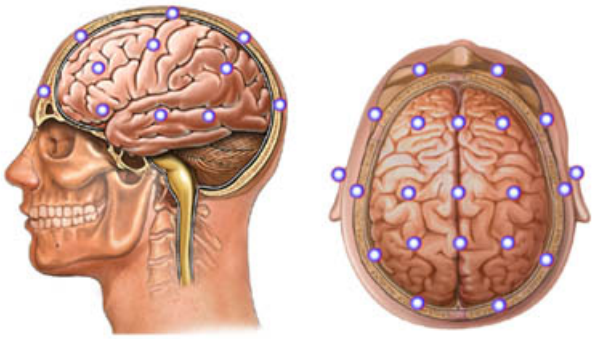
© W. Feng, 2011

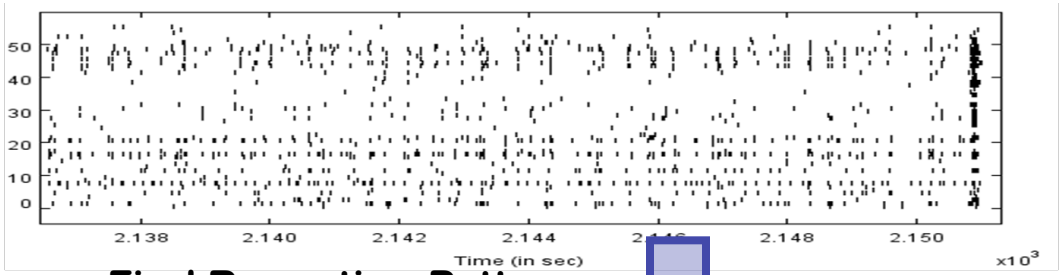
Finite State Machine

Interconnected states which transition between one another

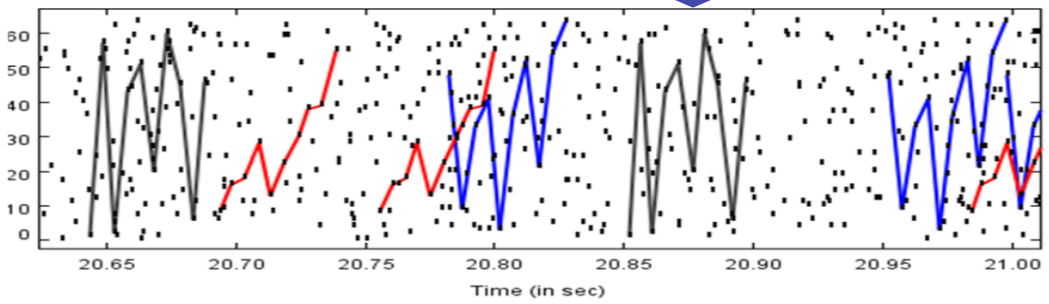
Application Areas

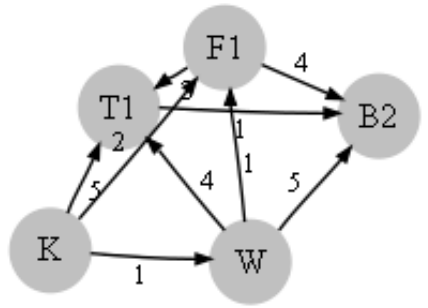
- Video Decoding, Parsing, Compression
- Data Mining → Reverse Engineering the Brain





Find Repeating Patterns





Infer
Network Connectivity

"Temporal Data Mining for Neuroscience," *GPU Computing Gems*, Morgan Kaufmann, Feb. 2011.
"High-Performance Biocomputing for Simulating the Spread of Contagion over Large Contact Networks," *BMC Genomics*, 2011.

© W. Feng, 2011

Sampling of Recent (Dwarf) Publications

- W. Feng et al., "Temporal Data Mining for Neuroscience," *GPU Computing Gems*, Chapter 15, Morgan Kaufmann, February 2011. ***Dwarf: Finite State Machine.***
  http://mkp.com/news/gpu-computing-gems-edited-by-wen-mei-w-hwu (or go to Amazon)
  - Book chapter that is working towards reverse engineering the brain on a GPU.
- B. Subramaniam et al., "Statistical Power and Performance Modeling for Optimizing the Energy Efficiency of Scientific Computing," *IEEE/ACM Int'l Conf. on Green Computing & Communications*, December 2010. ***Dwarf: Dense Linear Algebra → LINPACK.***
- Y. Jiao et al., "Power and Performance Characterization of Computational Kernels on the GPU," *IEEE/ACM Int'l Conf. on Green Computing & Communications*, December 2010. ***Dwarf: Dense Linear Algebra, Spectral Method.***
- T. Scogland et al., "A First Look at Integrated GPUs for Green High-Performance Computing," *Computer Science – Research & Development Journal*, September 2010. ***Dwarfs: Dense Linear Algebra, Dynamic Programming, N-body, Structured Grid.***
  - Embedded supercomputing with low-power CPU+GPU.
- R. Anandakrishnan et al., "Accelerating Electrostatic Surface Potential Calculation with Multiscale Approximation on Graphics Processing Units," *Journal of Molecular Graphics and Modelling*, June 2010. ***Dwarf: N-body.***
  - Application-to-HW mapping to an AMD CPU+GPU heterogeneous computing system.

© W. Feng, 2011

Outline

- Overview of *OpenCL and the 13 Dwarfs*
- Status of *OpenCL and the 13 Dwarfs*
- Creating an Army of Dwarfs with "CU2CL" (pronounce as "cuticle")
- Performance and Power of *OpenCL and the 13 Dwarfs*

© W. Feng, 2011

Status of OpenCL Dwarfs

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

Completed Prototyped

Next Steps?

1. Finalize OpenCL dwarfs and its build system
2. Report and document the dwarfs

Projected official release?
… soon :-)

© W. Feng, 2011

Outline

- Overview of *OpenCL and the 13 Dwarfs*
- Status of *OpenCL and the 13 Dwarfs*
- Creating an Army of Dwarfs with "CU2CL" (pronounce as "cuticle")
- Performance and Power of *OpenCL and the 13 Dwarfs*

© W. Feng, 2011

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.

Source: "CU2CL: A CUDA-to-OpenCL Translator for Multi- and Many-core Architectures," *ACM/IEEE SC* (submitted), Nov. 2011. Available as technical report.

© W. Feng, 2011

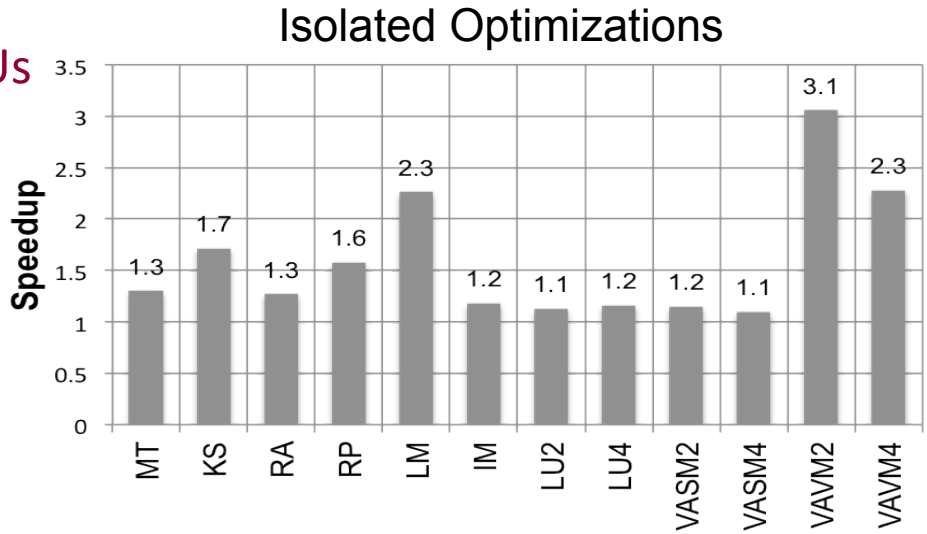
Motivation for CU2CL

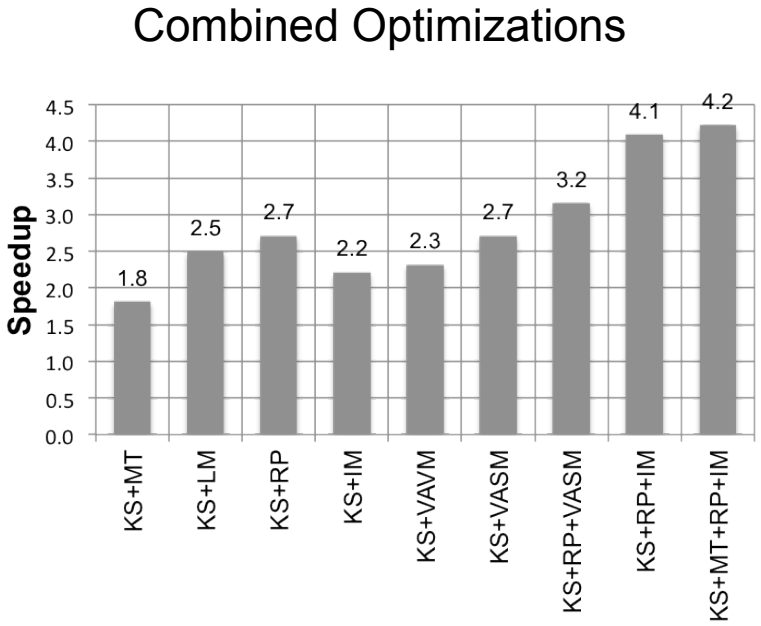
- OpenCL is a lower-level API than CUDA.
  - Longer term? A high-level library or domain-specific language
- Significantly larger number of applications implemented in CUDA than in OpenCL.
  - Idea: Leverage domain scientists' investment in CUDA to drive OpenCL adoption
  - Issues (from the perspective 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.
- Caveat: "Double-edged" sword
  - CU2CL only does *translation* (source-to-source).
  - No architecture-aware optimization.

© W. Feng, 2011

Architecture-Aware Optimizations (N-body Code)

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



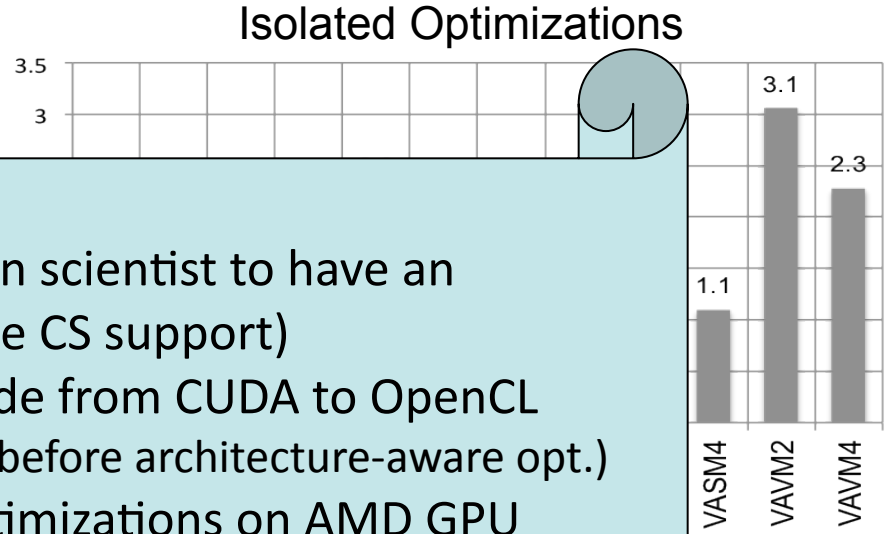


- GEM: molecular dynamic simulation app
  - Computes and visually depicts electrostatic-surface potential of a biomolecule
- Speedup over basic OpenCL GPU implementation
  - Isolated optimization
  - Optimization combination (32 combinations)

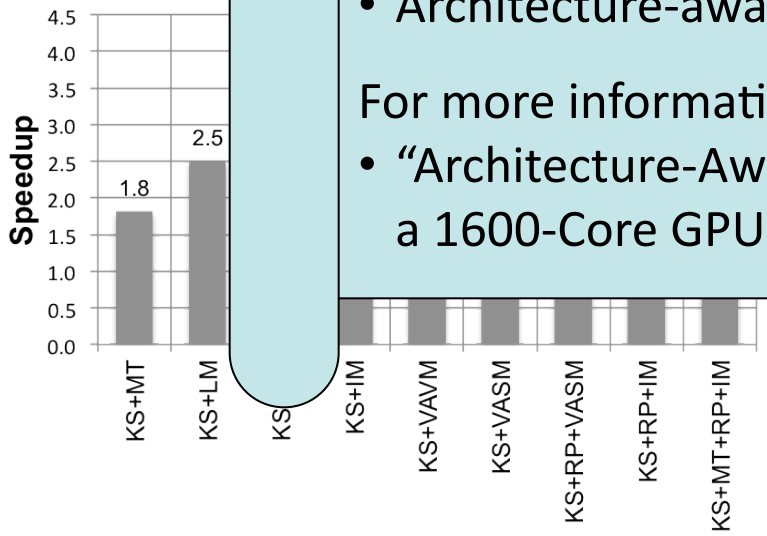
© W. Feng, 2011

Architecture-Aware Optimizations (N-body Code)

- Optimization techniques on AMD GPUs
  - Removing conditions
  - Local
  - Using
  - Using



Com



- Little interest by domain scientist to have an OpenCL version (despite CS support)
- Manually porting of code from CUDA to OpenCL
  - Issue: Performance (before architecture-aware opt.)
- Architecture-aware optimizations on AMD GPU

For more information, attend the following talk:

- "Architecture-Aware Mapping and Optimization of a 1600-Core GPU" on Wed., Jun. 15.

on app

trostatic-

implementation

- Isolated optimization
- Optimization combination (32 combinations)

© W. Feng, 2011

Degree of Positive Impact of Optimization Strategies

Optimization Strategy	AMD	NVIDIA
Algorithm Design:		
Recomputing instead of Transferring	+ + + + +	+ + + + +
Using Host Asynchronously	+	+
Execution Configuration:		
Running Numerous Threads	+ + + + +	+ + + + +
Ensuring #Threads to be a multiple of Wavefront/Warp Size	+ + + + +	+ + + + +
Ensuring #Workgroups/Blocks to be a multiple of #Cores	+ + + + +	+ + + + +
Reducing Per-Thread Register Usage	+ + +	+ + + + +
Control Flow:		
Removing Branches	+ + +	+
Removing Divergent Branches	+ + + + +	+ + + + +
Memory Types:		
Using Registers	+ + + + +	+ + +
Using Local/Shared Memory	+ + + + +	+ + + + +
Using Constant Memory	+ + +	+ + +
Using Image/Texture Memory	+ + + + +	+
Memory Access Pattern:		
Coalescing Global Memory Accesses	+ + + + +	+ + + + +
Avoiding Partition Camping	+ + +	+ + + + +
Avoiding Bank Conflicts	+ + + + +	+ + + + +
Instruction Count:		
Using Vector Types and Operations	+ + + + +	+
Prefetching of Data from Global Memory	+ + + + +	+ + + + +
Loop Unrolling	+ + + + +	+ + + + +

© W. Feng, 2011

Outline

- Overview of *OpenCL and the 13 Dwarfs*
- Status of *OpenCL and the 13 Dwarfs*
- Creating an Army of Dwarfs with "CU2CL" (pronounce as "cuticle")
- Performance and Power of *OpenCL and the 13 Dwarfs*

© W. Feng, 2011

Approach: 4 P's + Dwarfs + Accelerators

Sparse Linear Algebra

Dense Linear Algebra

Spectral Methods

Structured Grids

N-Body Methods

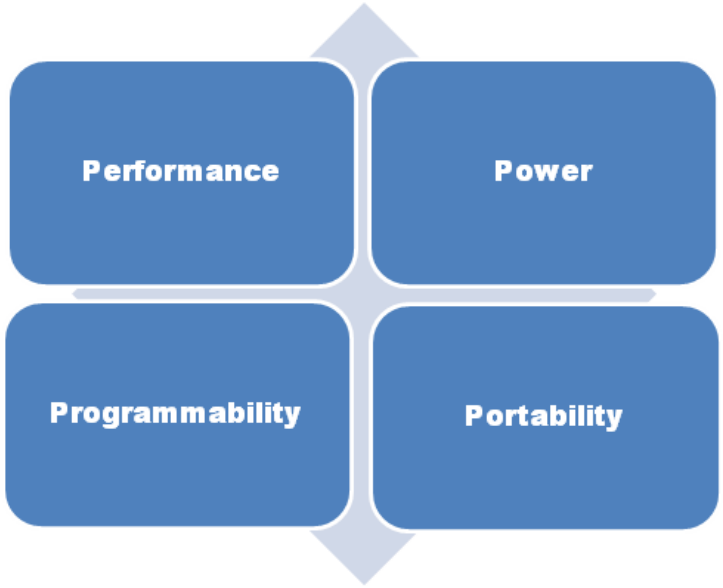
Unstructured Grids

Monte Carlo → MapReduce

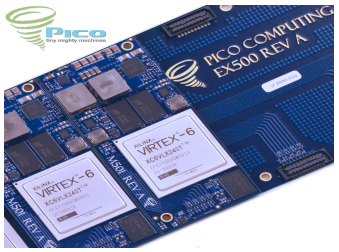
and

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

+

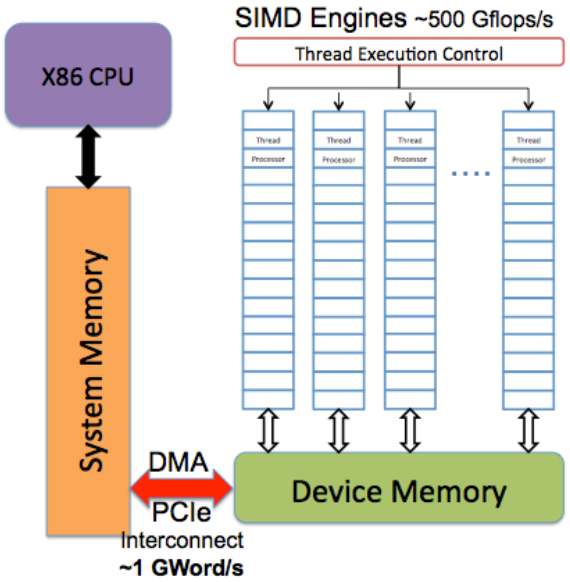


+



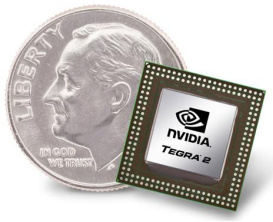
Approach: Heterogeneous Computing

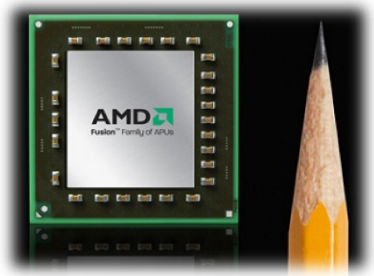
- For a given task, select the right core (i.e., "tool") at the right settings (e.g., degree of parallelism, voltage & frequency) at the right time.
  - Longer term: Each core could be designed for high performance and energy efficiency for each of the different computational idioms, e.g., Berkeley dwarfs.
- Hints of the above with CPU+GPU systems
  - General-purpose cores → CPU
  - Data-parallel/task-parallel cores → GPU
    - Reduced overhead
    - Explicitly hidden memory latency
    - Simplified control
    - Problem: Data movement between CPU & GPU



© W. Feng, 2011

Simpler CPU Cores & GPU Cores





- Simpler cores enable use of slower clock rates, resulting in cubic drop in power due to $V^2 * f$

- Simpler cores use less area and produce lower leakage power

- Simpler cores place more burden on the programmer
  - Need better languages and tools to *express* massive parallelism.
  - Need better system software and run-time systems to *manage* massive parallelism.

© W. Feng, 2011

Experimental Set-Up: Machines and Workload

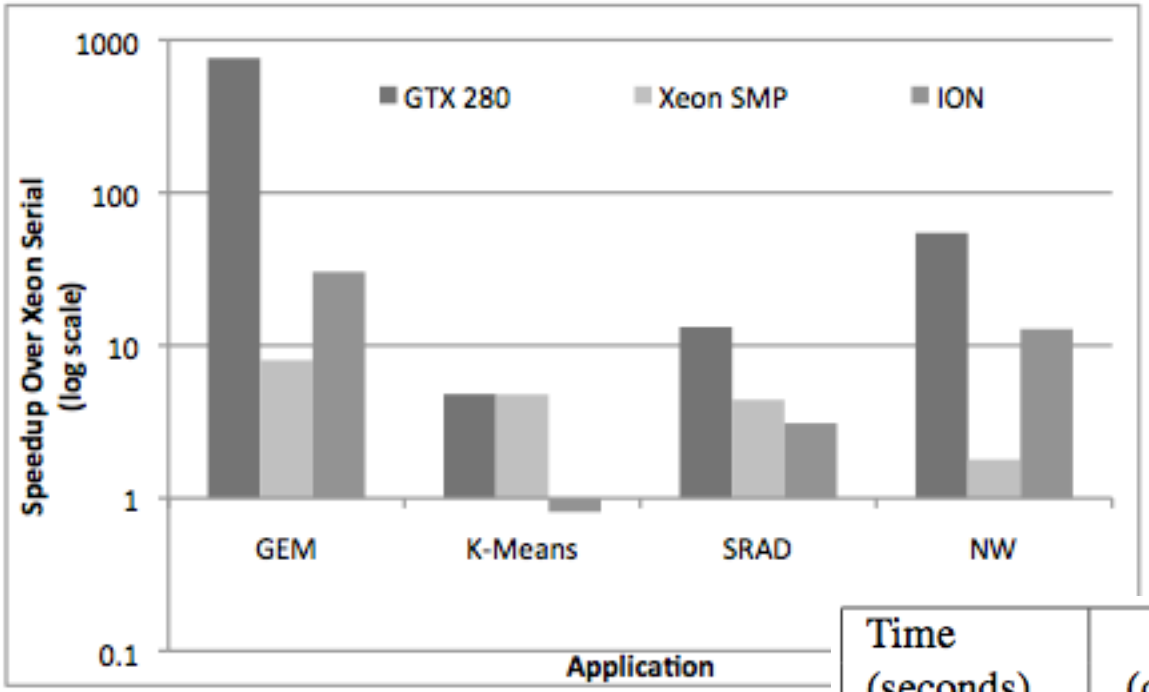
- Two 2.0-GHz Intel Xeon E5405 quad-core CPUs w/ 4GB RAM
  - … and a discrete NVIDIA GTX 280 GPU
- One 1.6-GHz Intel Atom 230 dual-core CPU w/ 3GB RAM
  - … and a NVIDIA MCP79 chipset w/ an *integrated* ION graphics chip with 256MB of graphics memory and 2 multiprocessors (16 stream cores).

	Kernel launches	Explicit synchronization between launches	Per launch data transfer
GEM	78	No	None
K-Means	37	Yes	Large
SRAD	4000	Yes	Small
NW	255	No	None

© W. Feng, 2011

Performance:

Integrated "Low-Power CPU + GPU" MCP

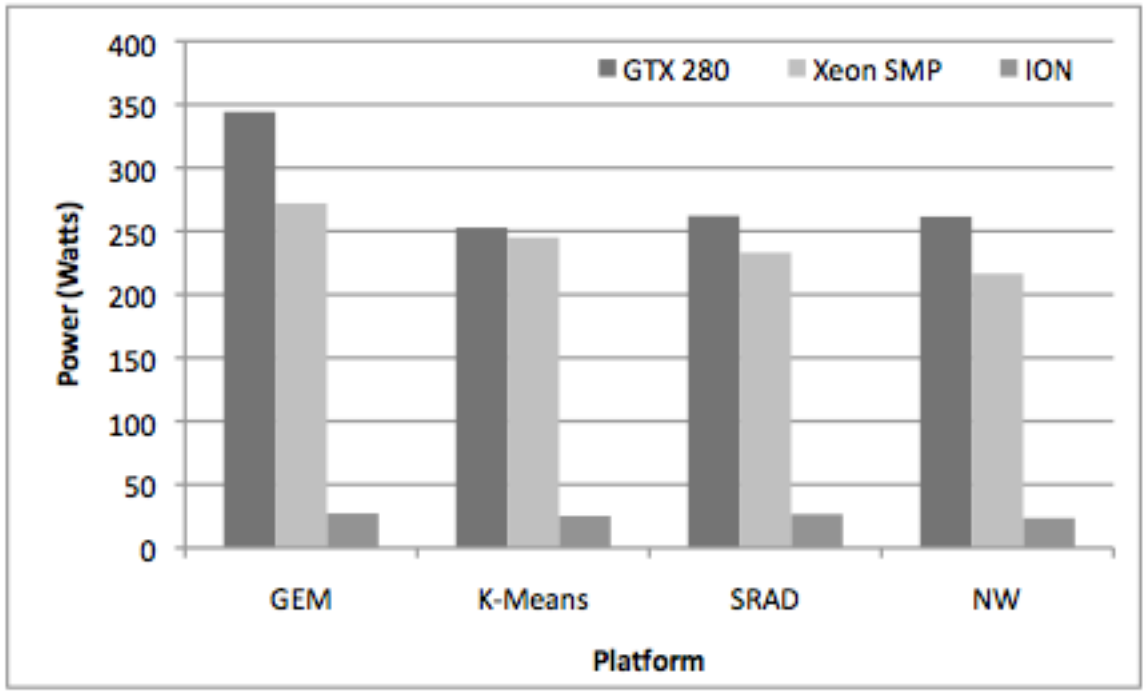


Time (seconds)	GEM (capsid)	K-Means	SRAD	NW
Xeon serial	63,029.5	7.9	788.5	377.0
Xeon SMP	7,878.7	1.7	179.0	210.5
GTX 280	82.9	1.7	59.8	6.9
ION	1,998.5	9.7	254.9	29.5

© W. Feng, 2011

Power:

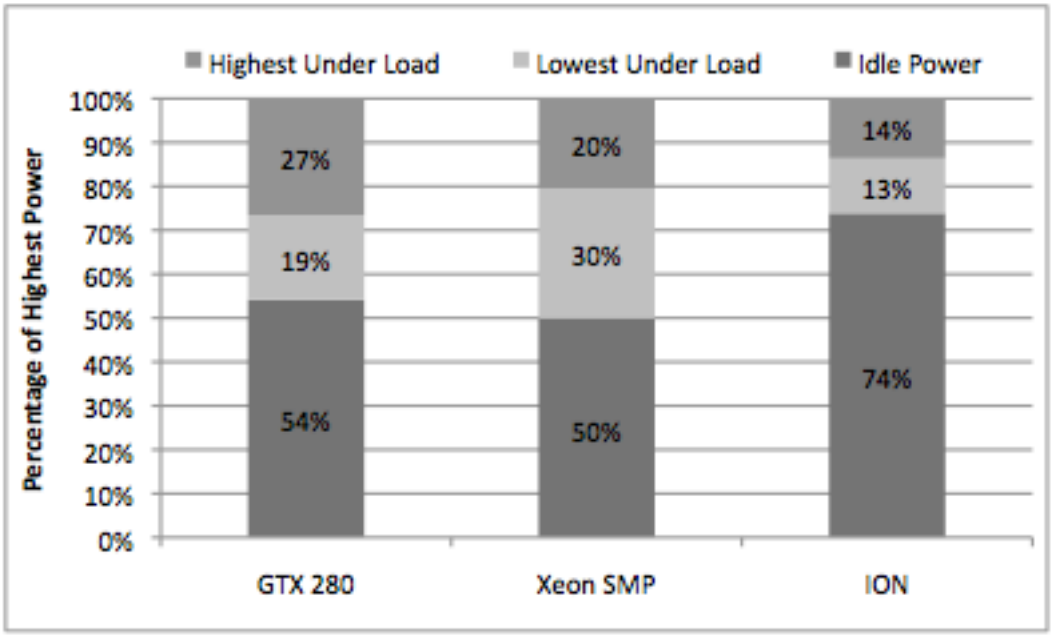
Integrated "Low-Power CPU + GPU" MCP



© W. Feng, 2011

Static vs. Dynamic Power:

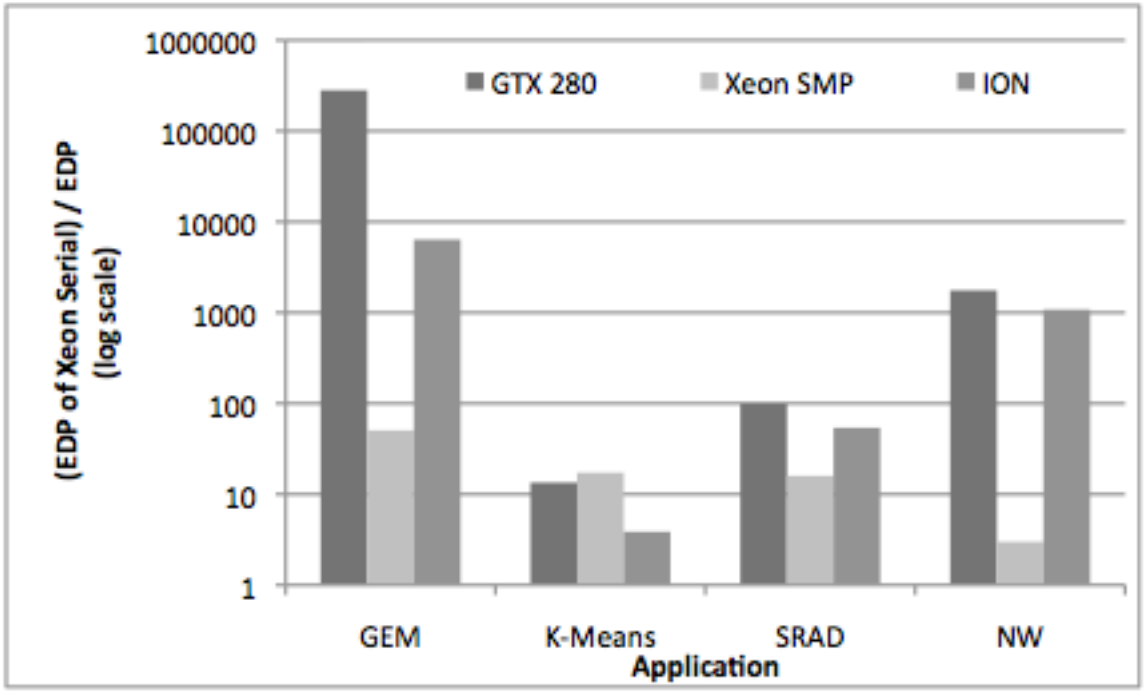
Integrated "Low-Power CPU + GPU" MCP



© W. Feng, 2011

Energy Efficiency:

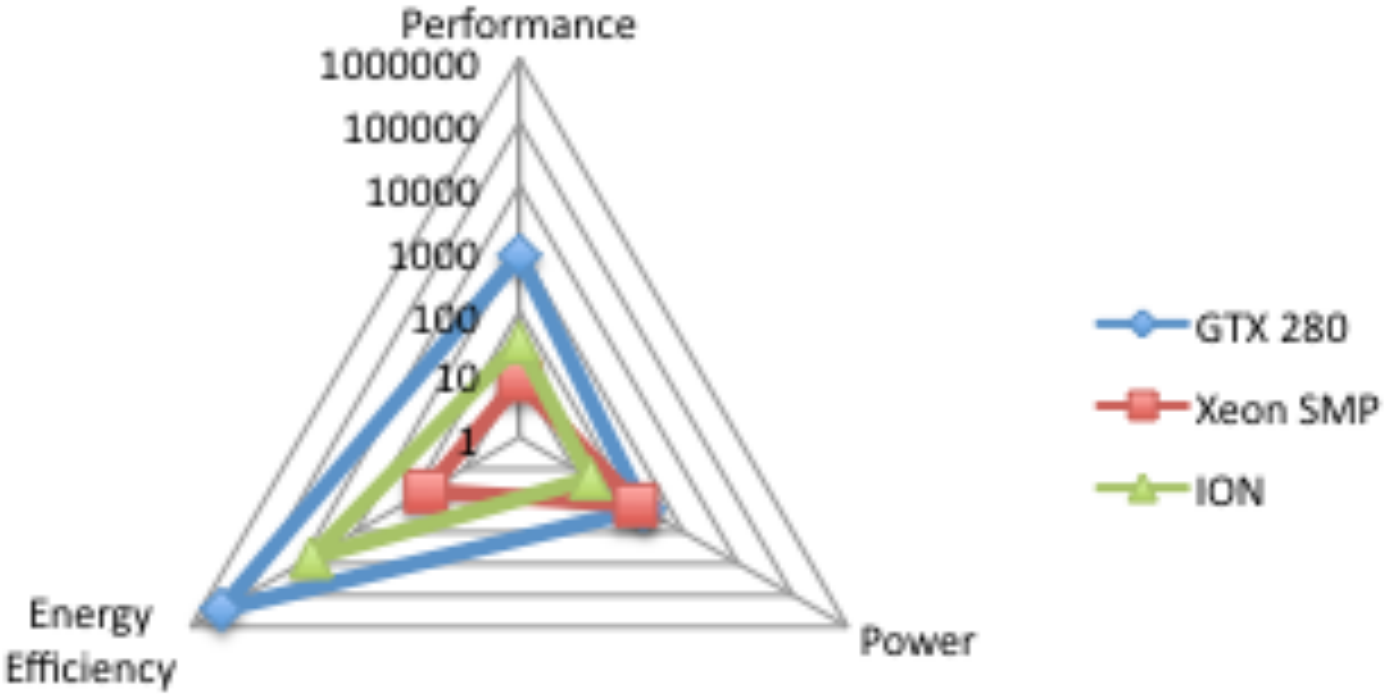
Integrated "Low-Power CPU + GPU"



© W. Feng, 2011

Kiviat Diagram:

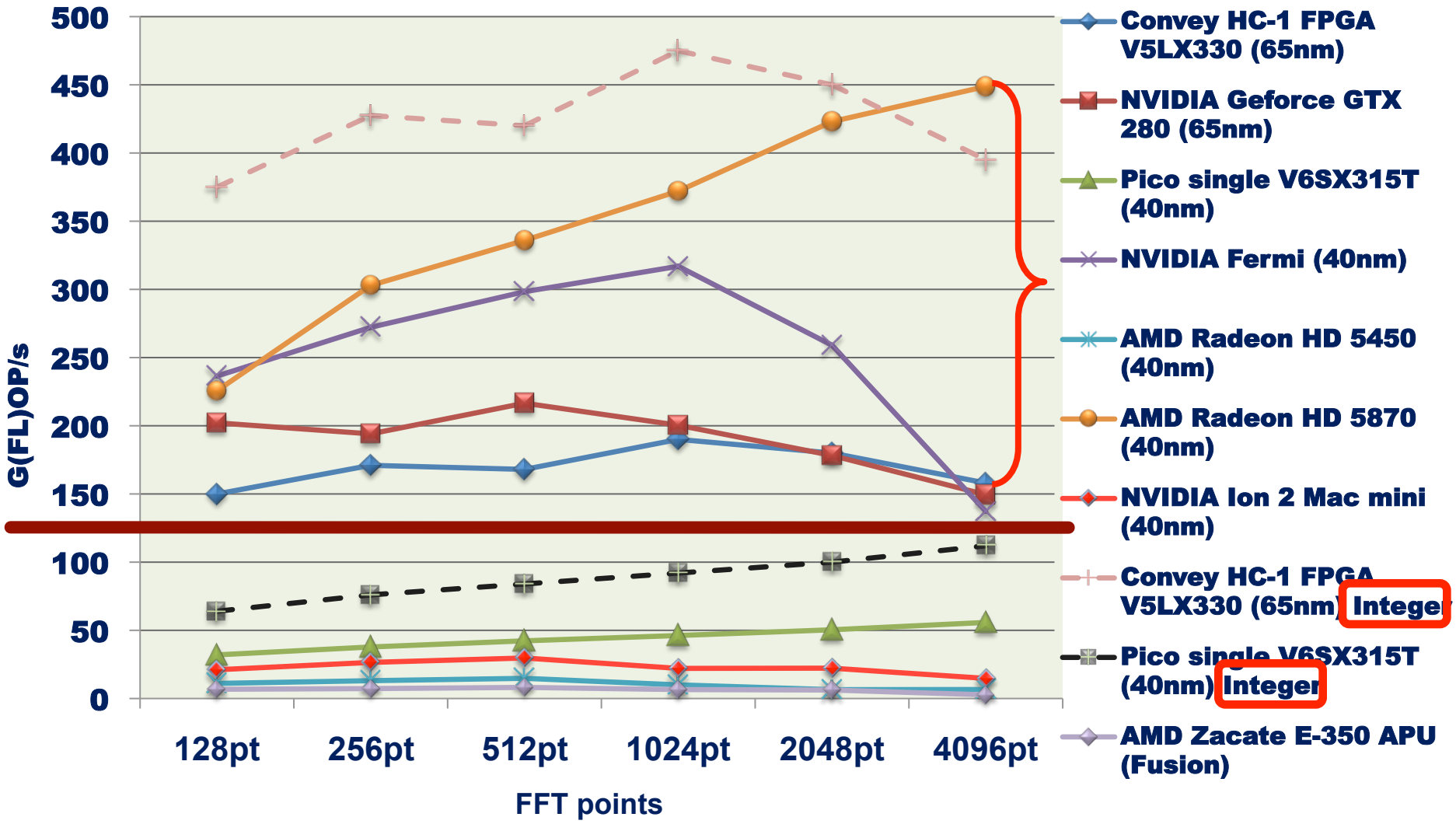
Performance, Power, and Energy Efficiency



© W. Feng, 2011

Performance: 1-D FFT (Spectral Method)

CHREC
NSF Center for High-Performance Reconfigurable Computing



© W. Feng, 2011

Hot off the presses …

Experimental Set-Up: Machines and Workload

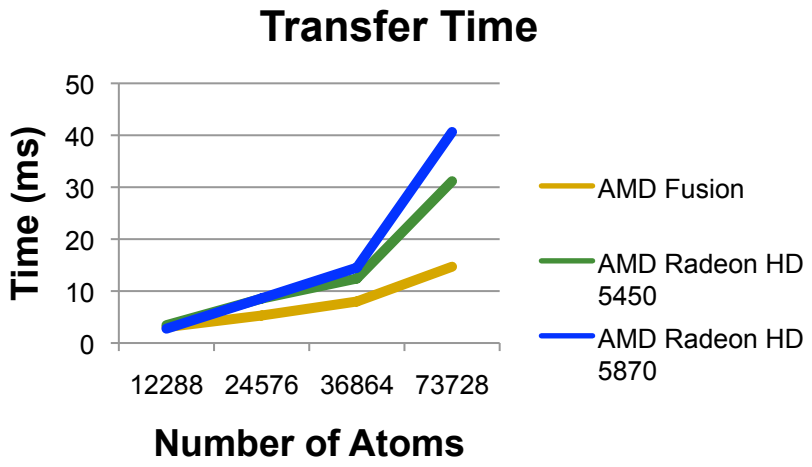
Platform	AMD Zacate APU	AMD Radeon HD 5870	AMD Radeon HD 5450
Stream Processors	80	1600	80
Compute Units	2	20	2
Memory Bus Type	NA	GDDR5	DDR3
Device Memory	192 MB	1024 MB	512 MB
Local Memory	32 KB	32 KB	32 KB
Max. Workgroup Size	256 Threads	256 Threads	128 Threads
Core Clock Frequency	492 MHz	850 MHz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

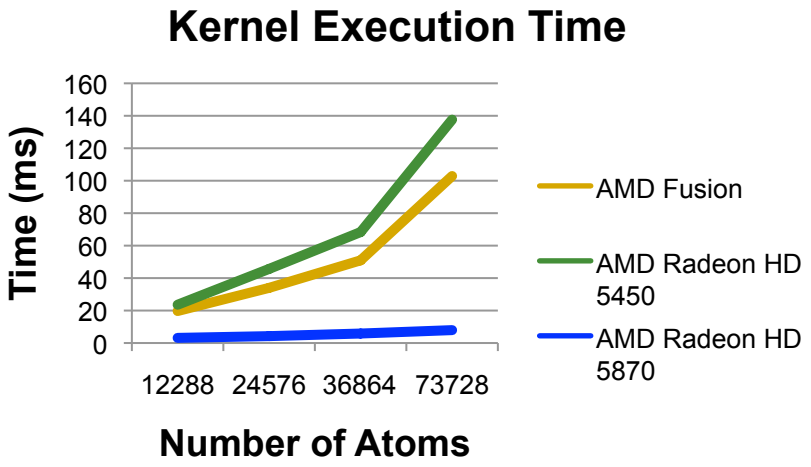
- OpenCL and the 13 Dwarfs
  - Sparse Linear Algebra: SpMV
  - N-body: Molecular Dynamics
  - Spectral: FFT
  - Dense Linear Algebra: Scan and Reduce

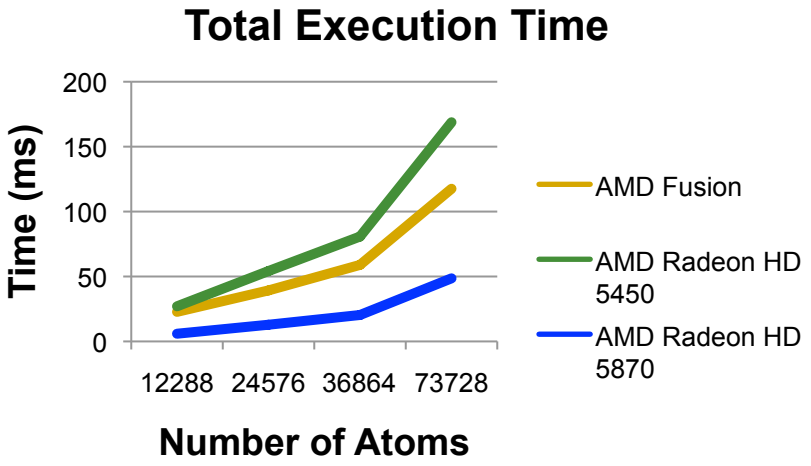
© W. Feng, 2011

Performance: Molecular Dynamics (N-Body)

Compute-bound





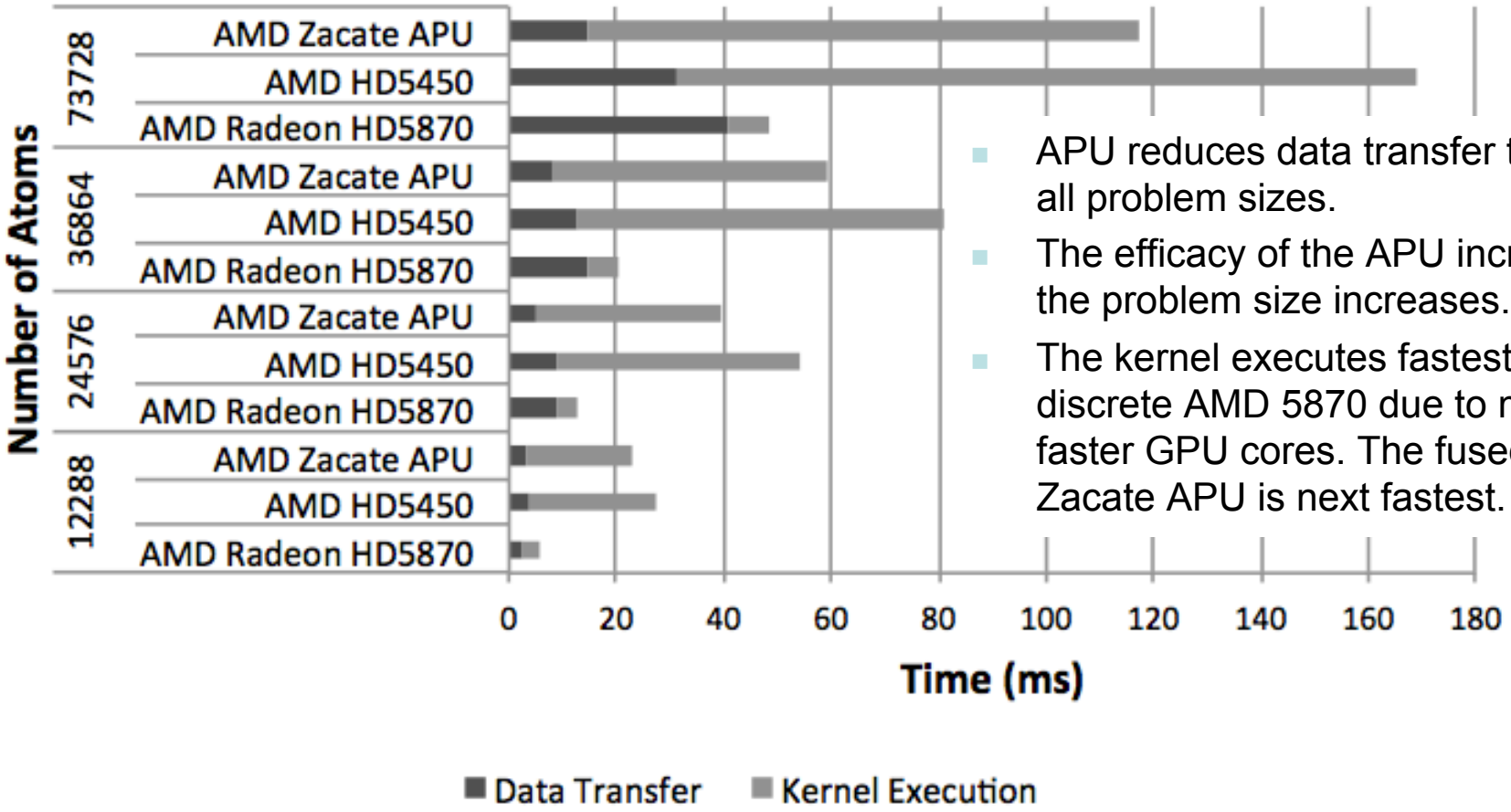


- APU reduces data transfer times for all problem sizes.
- The efficacy of the APU increases as the problem size increases.
- The kernel executes fastest on discrete AMD 5870 due to more and faster GPU cores. The fused Zacate APU is next fastest.

© W. Feng, 2011

Performance: Molecular Dynamics (N-Body)

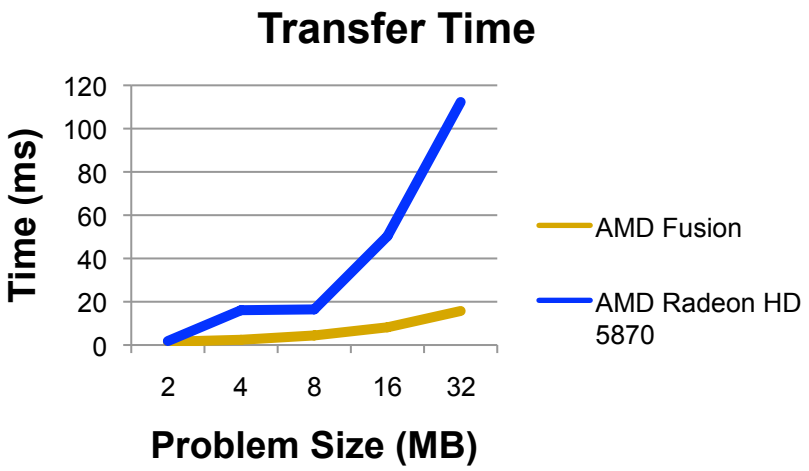
Compute-bound

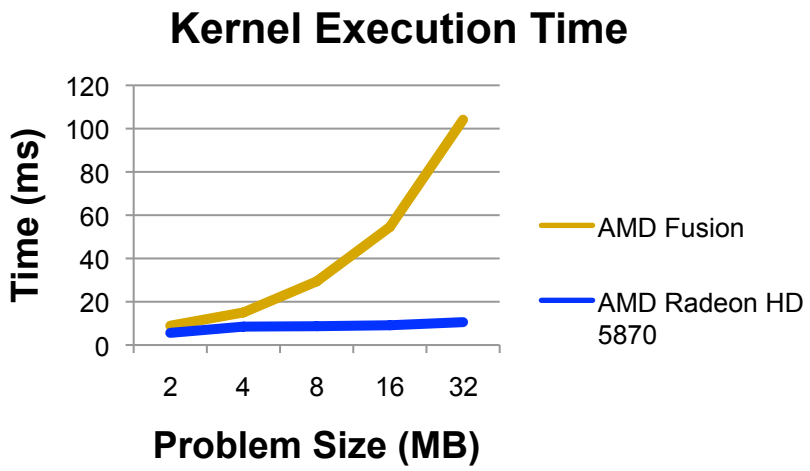


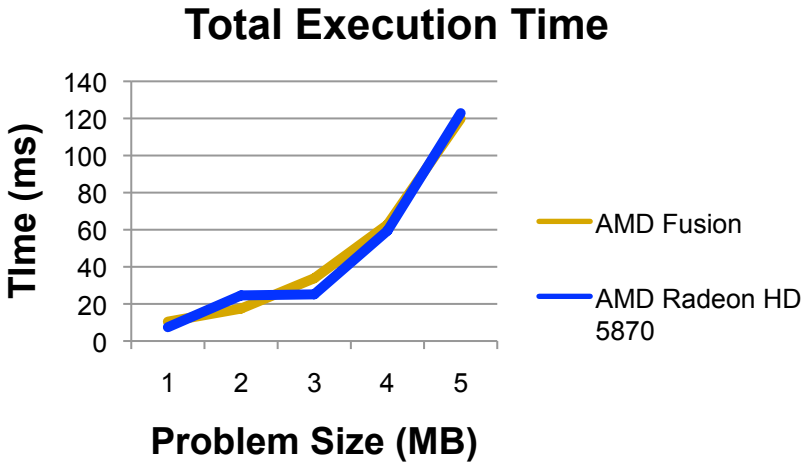
- APU reduces data transfer times for all problem sizes.
- The efficacy of the APU increases as the problem size increases.
- The kernel executes fastest on discrete AMD 5870 due to more and faster GPU cores. The fused AMD Zacate APU is next fastest.

© W. Feng, 2011

Performance: Scan (Dense Linear Algebra) I/O-bound





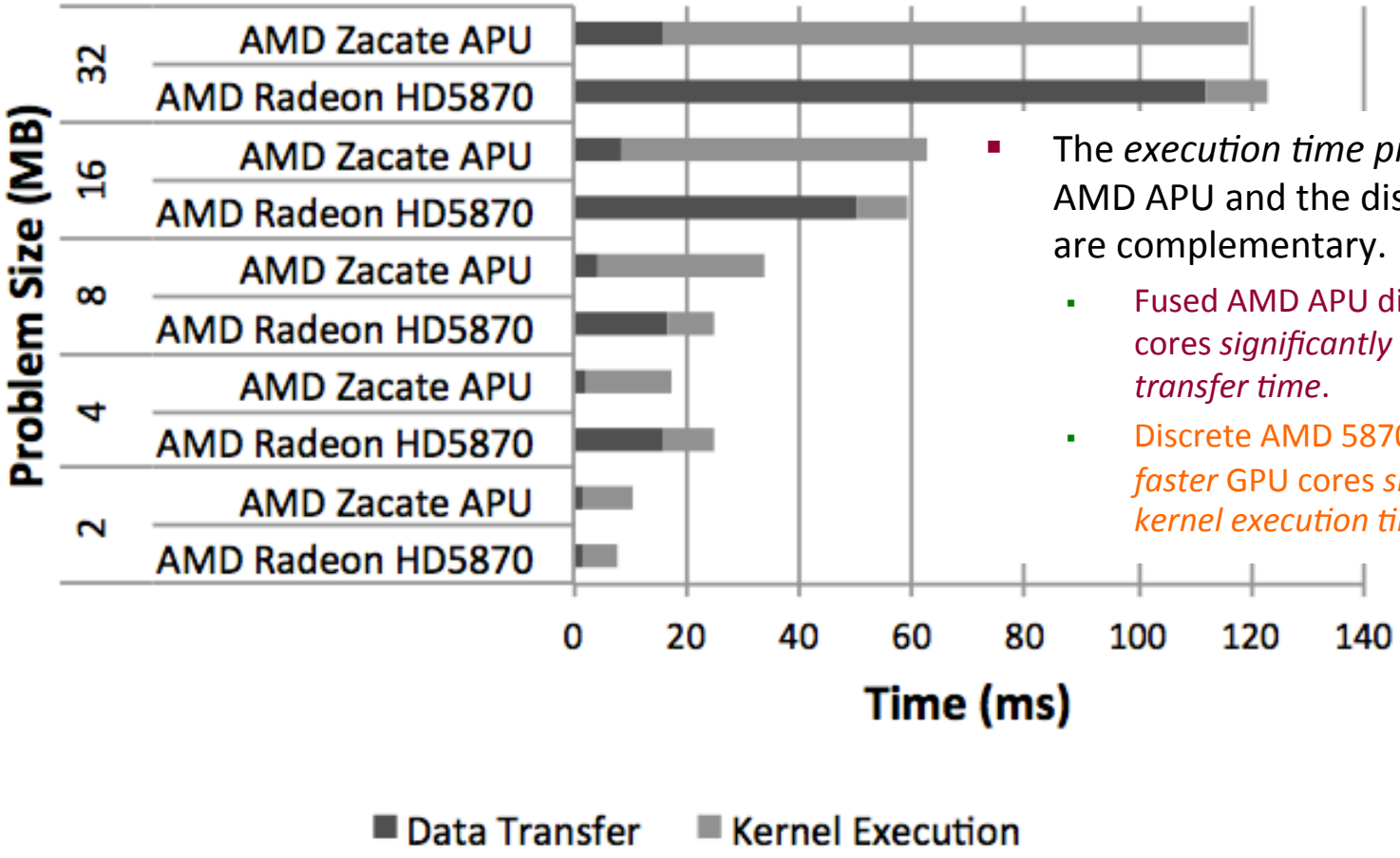


- The *execution time profiles* of the fused AMD APU and the discrete AMD 5870 are complementary.
  - Fused AMD APU die with only 80 GPU cores *significantly* improves *data transfer time*.
  - Discrete AMD 5870 die with 1600 *faster* GPU cores *significantly* improves *kernel execution time.*

© W. Feng, 2011

Performance: Scan (Dense Linear Algebra)

I/O-bound

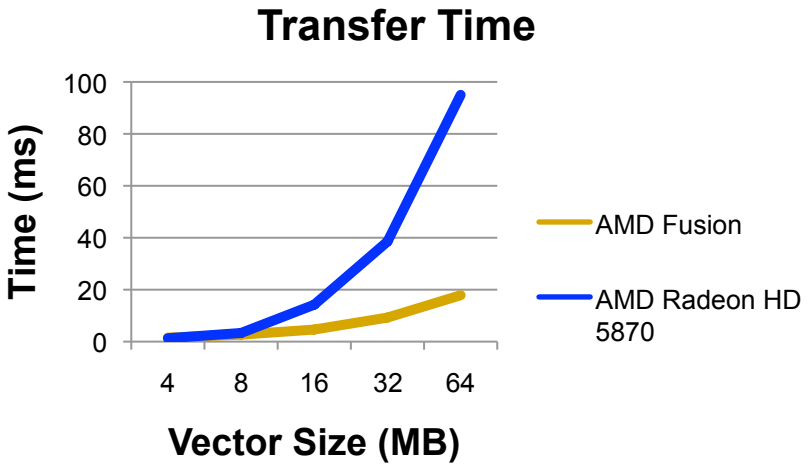


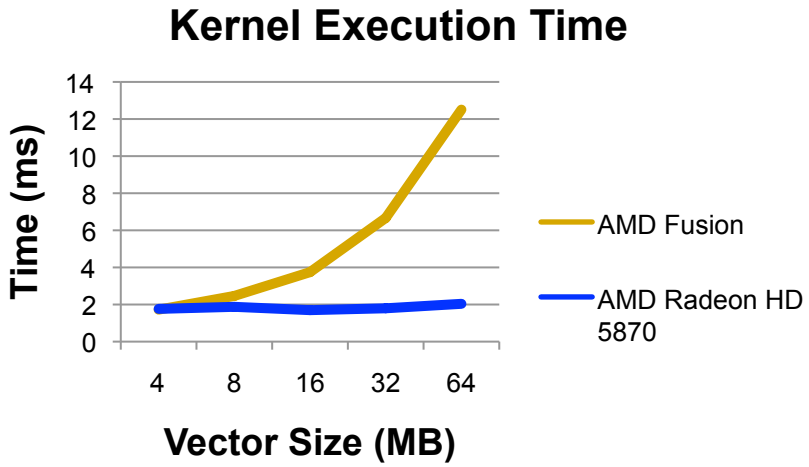
- The *execution time profiles* of the fused AMD APU and the discrete AMD 5870 are complementary.
  - Fused AMD APU die with only 80 GPU cores *significantly* improves *data transfer time*.
  - Discrete AMD 5870 die with 1600 *faster* GPU cores *significantly* improves *kernel execution time.*

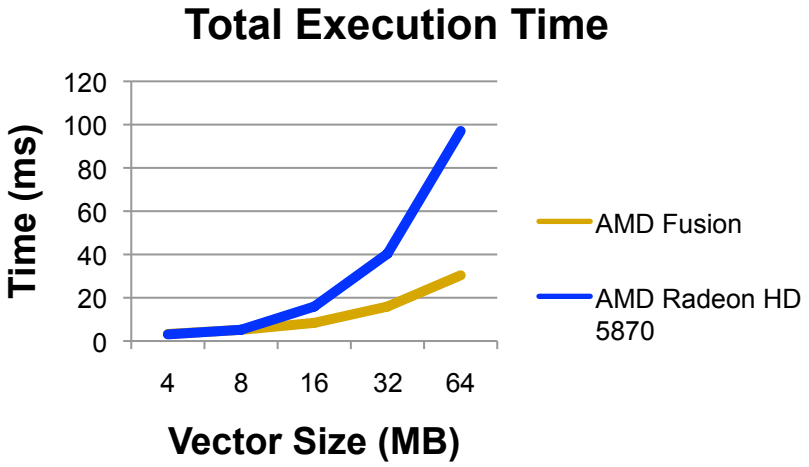
© W. Feng, 2011

Performance: Reduction (Dense Linear Algebra)

I/O-bound





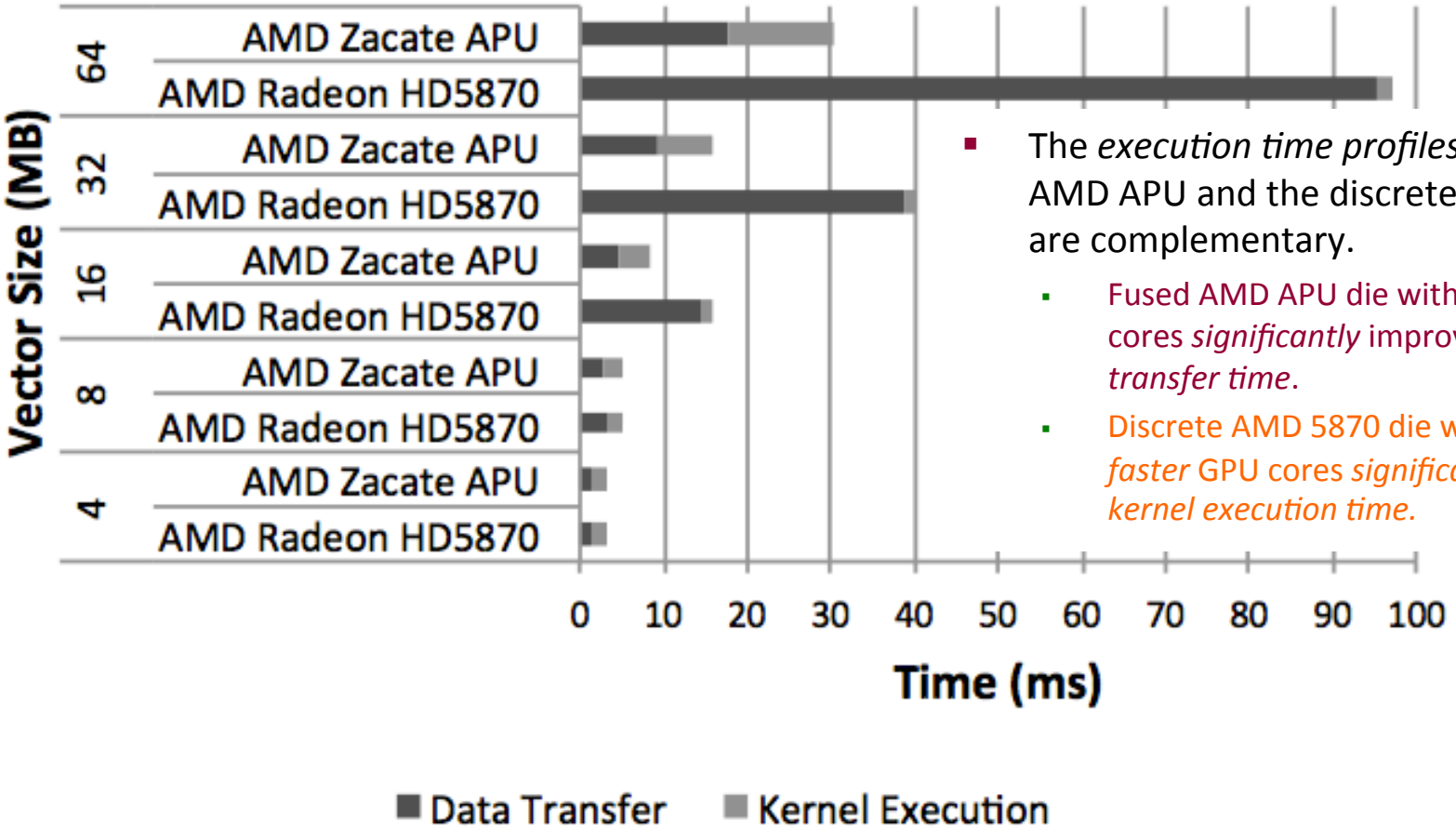


- The *execution time profiles* of the fused AMD APU and the discrete AMD 5870 are complementary.
  - Fused AMD APU die with only 80 GPU cores *significantly* improves *data transfer time*.
  - Discrete AMD 5870 die with 1600 *faster* GPU cores *significantly* improves *kernel execution time.*

© W. Feng, 2011

Performance: Reduction (Dense Linear Algebra)

I/O-bound



- 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.*

© W. Feng, 2011

Power Consumption of Fused vs. Discrete GPU

- AMD Zacate APU Machine
  - Idle: 11 watts
  - Load: 17-20 watts
- AMD Radeon HD 5870 Machine w/ 2-GHz Intel Xeon E5405
  - Idle: 188 watts
  - Load: 260 watts

© W. Feng, 2011

System Power

- AMD Fusion APU
  - At idle: 11.8 watts
  - At load: 17.2 watts
    (Spectral Method: FFT)
  - At load: 20.1 watts
    (N-body: Molecular Dynamics)
- AMD Radeon HD 5870 Machine w/ 2-GHz Intel Xeon E5405
  - At idle: 188 watts
  - At load: 260 watts

© W. Feng, 2011

Energy Efficiency of Fused vs. Discrete GPU

- AMD Zacate APU
  - SpMV: ~5x (on avg)
  - N-body: ~3x (on avg)
  - FFT: ~1.5x (on avg)
  - Scan: ~8x (on avg)
  - Reduce: ~25x (on avg)

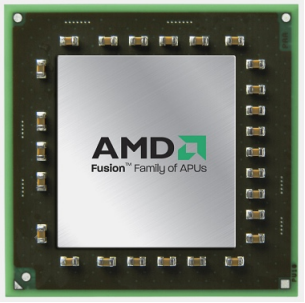
  better than the AMD Radeon HD 5870

The above results are highly dependent on problem size.

© W. Feng, 2011

Recap: OpenCL and the 13 Dwarfs

- Goal
  - Provide common algorithmic methods, i.e., dwarfs, in a language that can "run anywhere" (CPU, GPU, or even FPGA), i.e., OpenCL



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

CHREC
NSF Center for High-Performance Reconfigurable Computing

* Talk with Wu after the talk if you have interest in joining.

© W. Feng, 2011

Why "OpenCL and the 13 Dwarfs"?

Evaluate and guide architectural innovation

- Conventional Approach
  - Study a benchmark suite based on existing programs, e.g., SPEC.
- Obstacle to Innovation in Parallel Computing?
  - Unclear how to express a parallel computation best.
- Opportunity
  - Delineate application requirements to not be overly specific to individual applications or the optimizations used for certain hardware platforms.

    → draw broader conclusions about hardware requirements

Refer to http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

© W. Feng, 2011

Moving Forward ....

- GPU Dwarf Repository
  - Finalize base distribution and bridge to Rodinia.
  - Look at architecture-aware optimizations per platform.
- New GPU Dwarfs
  - Integer FFT (spectral method) and/or N-Queens (branch-and-bound).
- AMD Fusion (Fused CPU+GPU)
  - Leverage GPU dwarf repository to build-on the results presented here.

For additional related information, attend the following talks:

- "The SHOC Benchmark Suite" on Wed., Jun. 15
- "Architecture-Aware Mapping and Optimization of a 1600-Core GPU" on Wed., Jun. 15.

© W. Feng, 2011

Acknowledgements

- Worldwide Collaborators
- Staff: Heshan Lin and Mark Gardner
- Students
  - Mayank Daga (VT → AMD)
  - Kenneth Lee
  - Gabriel Martinez (VT → Intel)
  - Thomas Scogland
  - Balaji Subramaniam
  - Jing Zhang

© W. Feng, 2011

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

SyNeRG

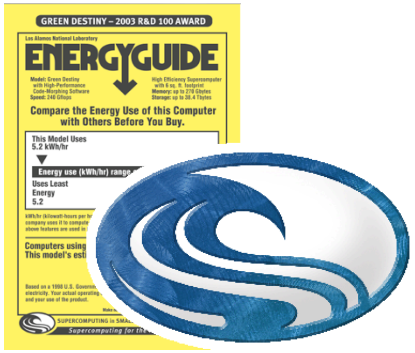
http://synergy.cs.vt.edu/

CHREC
NSF Center for High-Performance Reconfigurable Computing

http://www.chrec.org/

mpiBLAST

http://www.mpiblast.org/

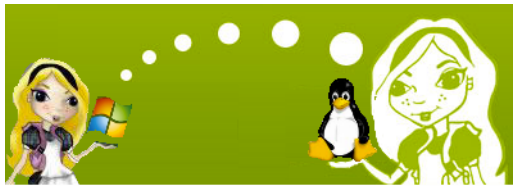


SUPERCOMPUTING in SMALL SPACES

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/

© W. Feng, 2011