ECE408 / CS483/CSE408
Spring 2018

Applied Parallel Programming

Lecture 1: Introduction

Course Goals

- Learn to program massively parallel processors and achieve
  - High performance
  - Functionality and maintainability
  - Scalability across future generations

- Technical subjects
  - Parallel programming basics
  - Principles and patterns of parallel algorithms
  - Programming API, tools and techniques
  - Processor architecture features and constraints
  - Killer apps

People

- Instructors:

  Prof. Wen-mei Hwu

  215 CSL, w-hwu@illinois.edu, 244-8270

  use ECE408 to start your e-mail subject line

  Office hours: 1:30-2:30pm Tuesdays; or by appointment

- Teaching Assistants:

  Carl Pearson (pearson@illinois.edu), Keven Feng (klfeng2@illinois.edu), Lan, Rui (ruilan2@illinois.edu) Wei Ren (weiren2@illinois.edu)

  Office hours: Check the course website

Brief Bio of Wen-mei Hwu

- Ph.D. from University of California, Berkeley, 1987
- Research: Computer Architecture and Parallel Computing
- Professor of ECE Illinois since 1987
- Chief Scientist Parallel Computing Institute, Co-PI Blue Waters
- Director of several research centers, most recently IBM C3SR
- Co-Founder, Advisor and Board Member to 8 startups to date
- Researcher, teacher, author, technologist, and entrepreneur

How we will run the course

- Previously recorded lecture video for each major topical segment is on-line.
  - In Class Schedule page of course wiki
  - Your responsibility is to absorb BEFORE corresponding lecture
- Lectures by me when I am in town.
- View previously recorded lecture videos when I am traveling – One of the TAs will come to class to discuss finer points, answer questions (flipped class).

Web Resources

- Website: https://wiki.illinois.edu/wiki/display/ECE408/
  - Use your Illinois NetID and Password to login
  - Handouts and lecture slides/recordings
  - Textbook chapters, documentation, software resources
  - Note: While we'll make an effort to post announcements on the web, we can't guarantee it, and won't make any allowances for people who miss things in class.
- Web board discussions in Piazza
  - Channel for electronic announcements
  - Forum for Q&A - the TAs and Professors read the board, and your classmates often have answers
- Compass - grades

Grading

- Exams: 40%
  - Exam 1: 20%
  - Exam 2: 20%
- Labs (Machine Problems): 35%
  - Passing Datasets 90%
  - Reasonable-looking answers to questions
- Project: 25%
  - Automatic Demo/Functionality/Coding Style: 50%
  - Automatic Performance Ranking with full functionality: 50%
  - Detailed Rubric will be posted

Academic Honesty

- You are allowed and encouraged to discuss assignments with other students in the class. Getting verbal advice/help from people who've already taken the course is also fine.
- Any reference to assignments from previous terms or web postings is unacceptable
- Any copying of non-trivial code is unacceptable
  - Non-trivial = more than a line or so
  - Copying includes reading someone else's code and then going off to write your own.

Academic Honesty (cont.)

- Giving/receiving help on an exam is unacceptable

- Penalties for academic dishonesty:
  - Zero on the assignment/exam for the first occasion
  - Automatic failure of the course for repeat offenses

Text/Notes

1. D. Kirk and W. Hwu, “Programming Massively Parallel Processors – A Hands-on Approach,” Morgan Kaufman Publisher, 3rd edition, 2016, ISBN 978-0123814722
   - all chapters used in this course are in Wiki

2. NVIDIA, *NVidia CUDA C Programming Guide*, version 7.5 or later (reference book)

Tentative Schedule

(Class Schedule Tab in Wiki)

- Week 1:
  - Tu: Lecture 1: Introduction
  - Th: Lecture 2: CUDA Intro
- Week 2:
  - Tu: Lecture 3: Data Parallelism Model
  - Th: Lecture 4: CUDA Memory Model
  - Due: MP-0, installation, test account, MP-1, vector addition
- Week 3:
  - Tu: Lecture 5: CUDA Memory Model
  - Th: Lecture 6: Performance Considerations
  - Due: MP-2, simple matrix multiplication

A major paradigm shift

- **In the 20th Century, we were able to understand, design, and manufacture what we can measure**
  - Physical instruments and computing systems allowed us to see farther, capture more, communicate better, understand natural processes, control artificial processes…

A major paradigm shift

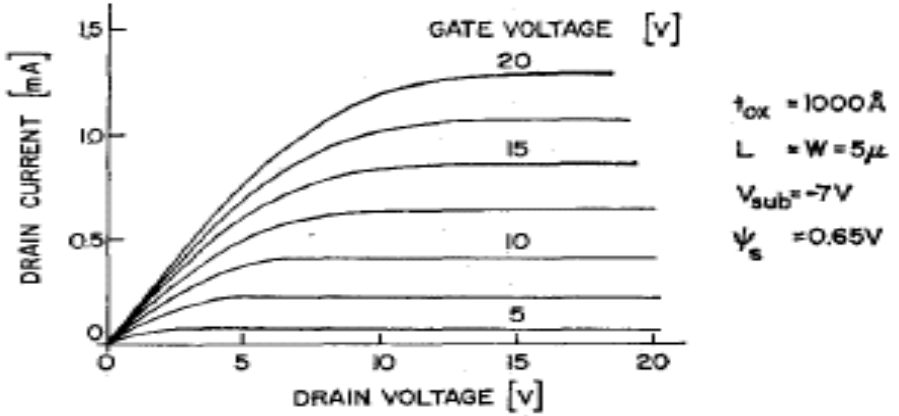
- **In the 21st Century, we are able to understand, design, and create what we can compute**
  - Computational models are allowing us to see even farther, going back and forth in time, learn better, test hypothesis that cannot be verified any other way, create safe artificial processes…

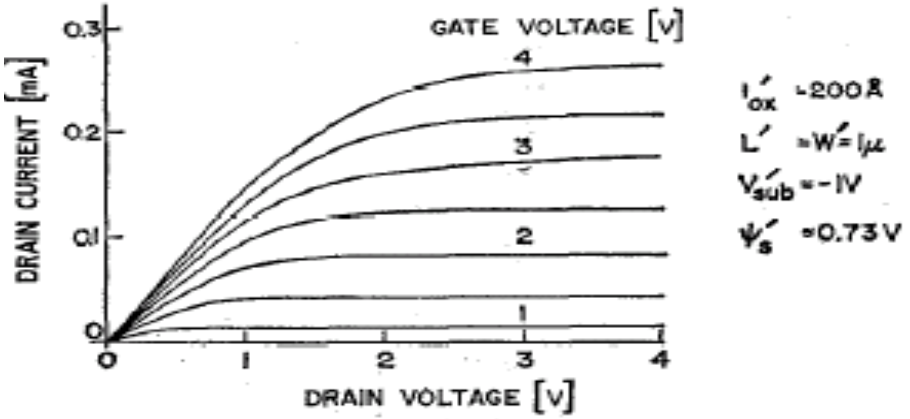
Examples of Paradigm Shift

20th Century	21st Century
▪ Small mask patterns	▪ Optical proximity correction
▪ Electronic microscope and Crystallography with computational image processing	▪ Computational microscope with initial conditions from Crystallography
▪ Anatomic imaging with computational image processing	▪ Metabolic imaging sees disease before visible anatomic change
▪ Teleconference	▪ Tele-emersion
▪ GPS	▪ Self-driving cars

POST-DENNARD TECHNOLOGY PIVOT – PARALLELISM AND HETEROGENEITY

Dennard Scaling of MOS Devices

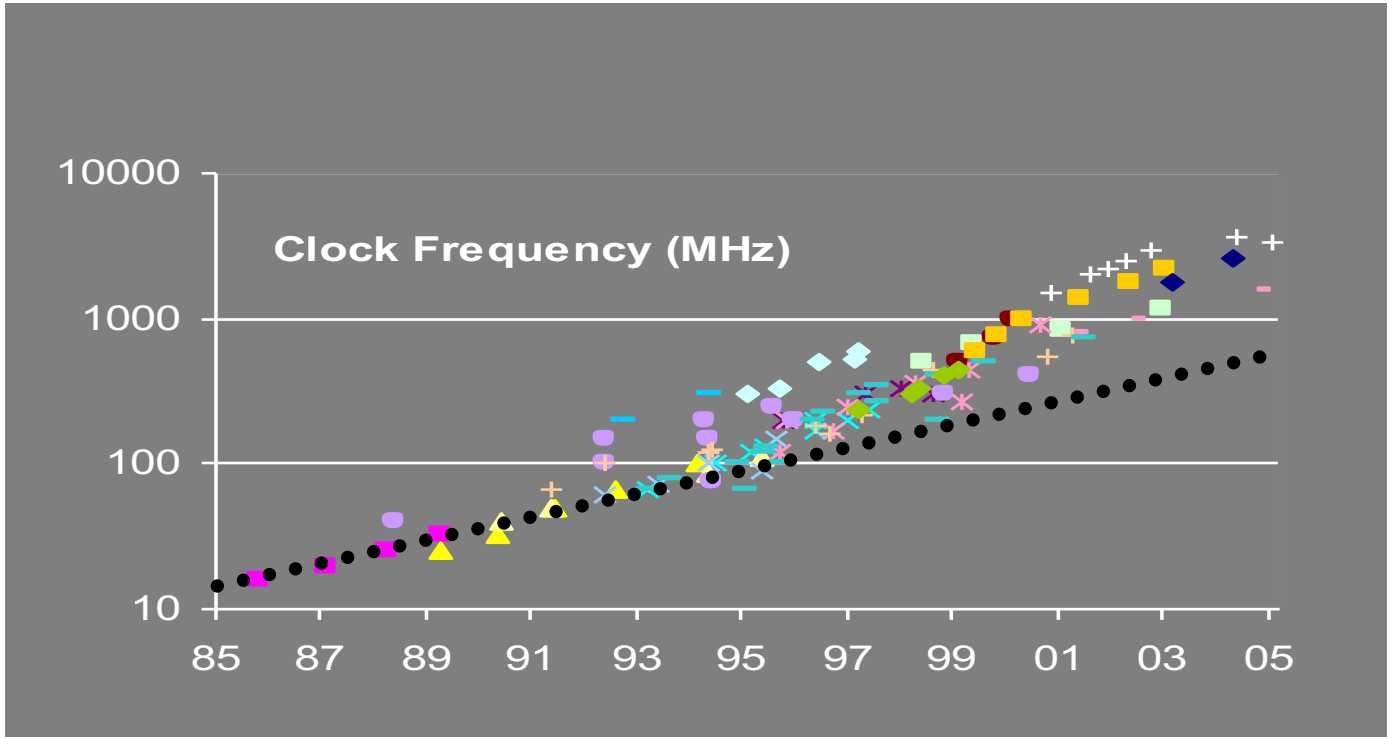




JSSC Oct **1974**, page 256

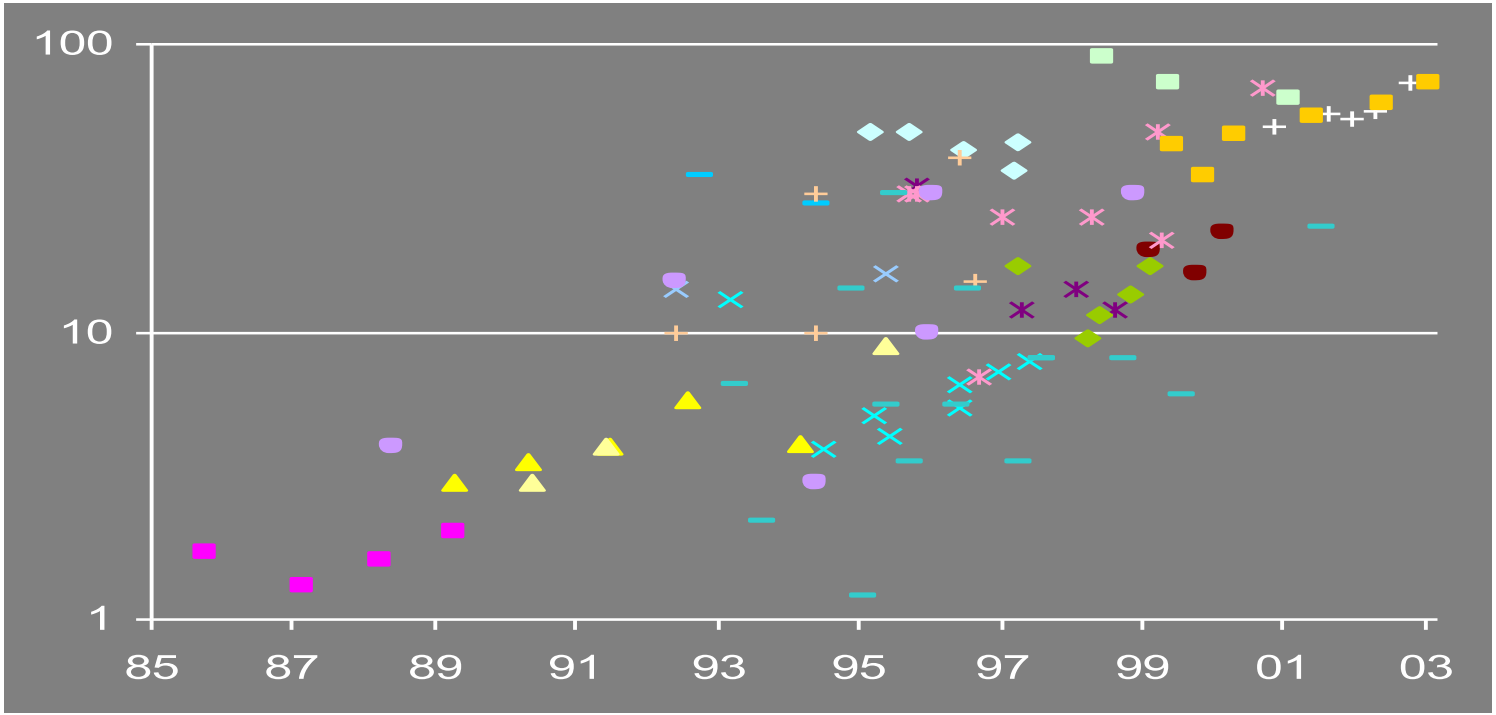
- In this ideal scaling, as L → α*L
  - V_{DD} → α*V_{DD}, C → α*C, i → α*i
  - Delay = CV_{DD}/I scales by α, so f → 1/α
  - Power for each transistor is CV^2*f and scales by α^2
    - keeping total power constant for same chip area

Frequency Scaled Too Fast 1993-2003



Total Processor Power Increased

(super-scaling of frequency and chip size)



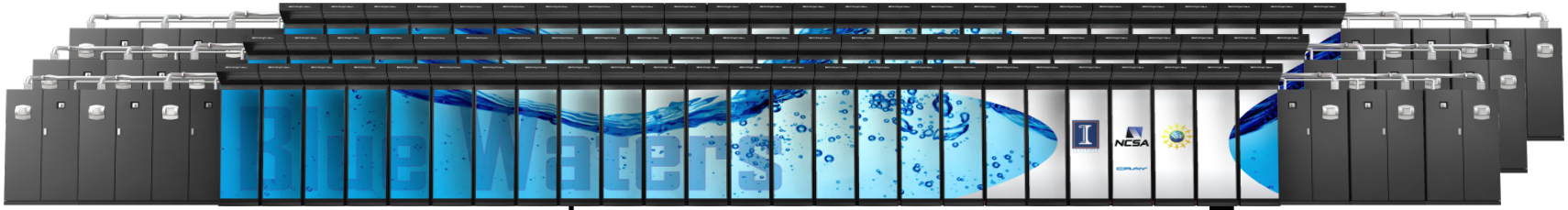
Post-Dennard Pivoting

- Multiple cores with more moderate clock frequencies
- Heavy use of vector execution
- Employ both latency-oriented and throughput-oriented cores
- 3D packaging for more memory bandwidth

Blue Waters Computing System

Operational at Illinois since 3/2013

49,504 CPUs -- 4,224 GPUs



12.5 PF
1.6 PB DRAM
$250M

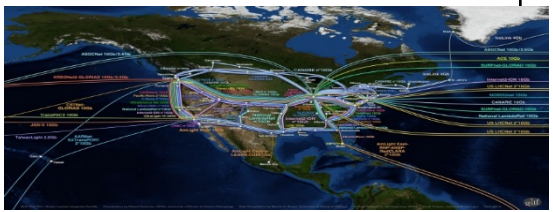
10/40/100 Gb Ethernet Switch

IB Switch

>1 TB/sec

120+ Gb/sec

100 GB/sec



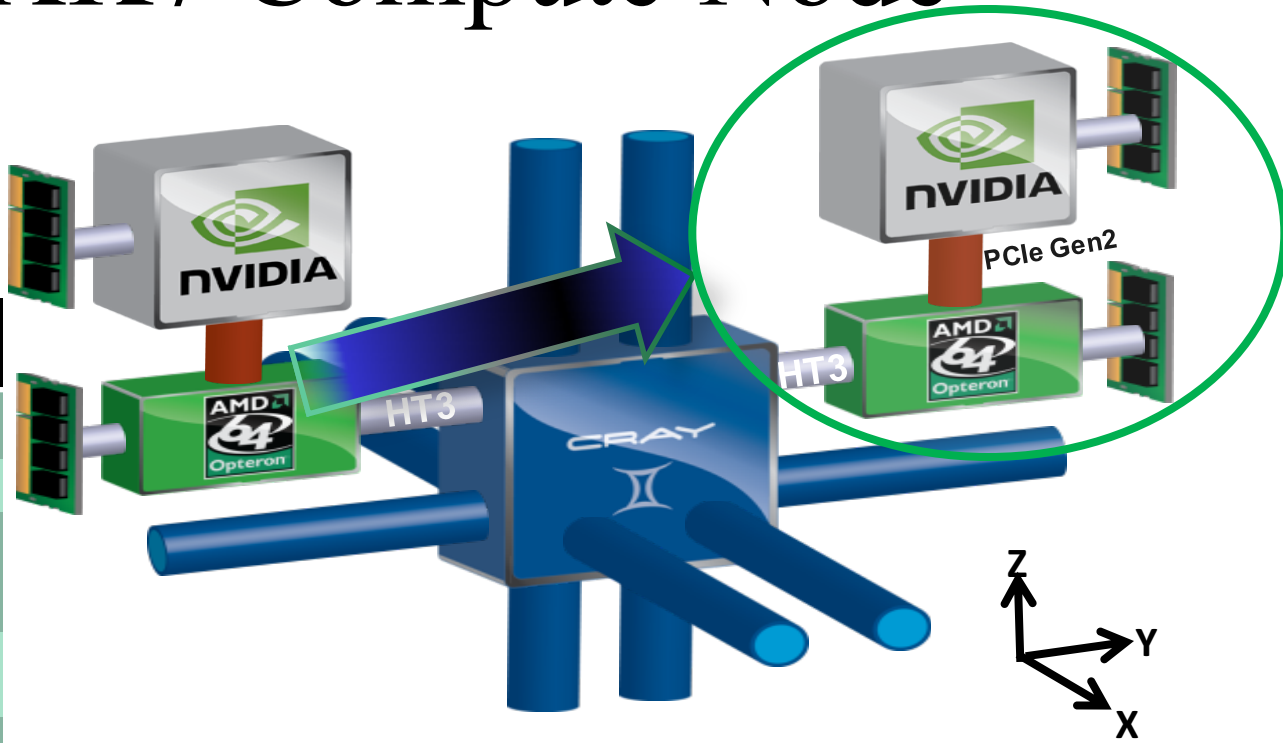
WAN

Spectra Logic: 300 PBs

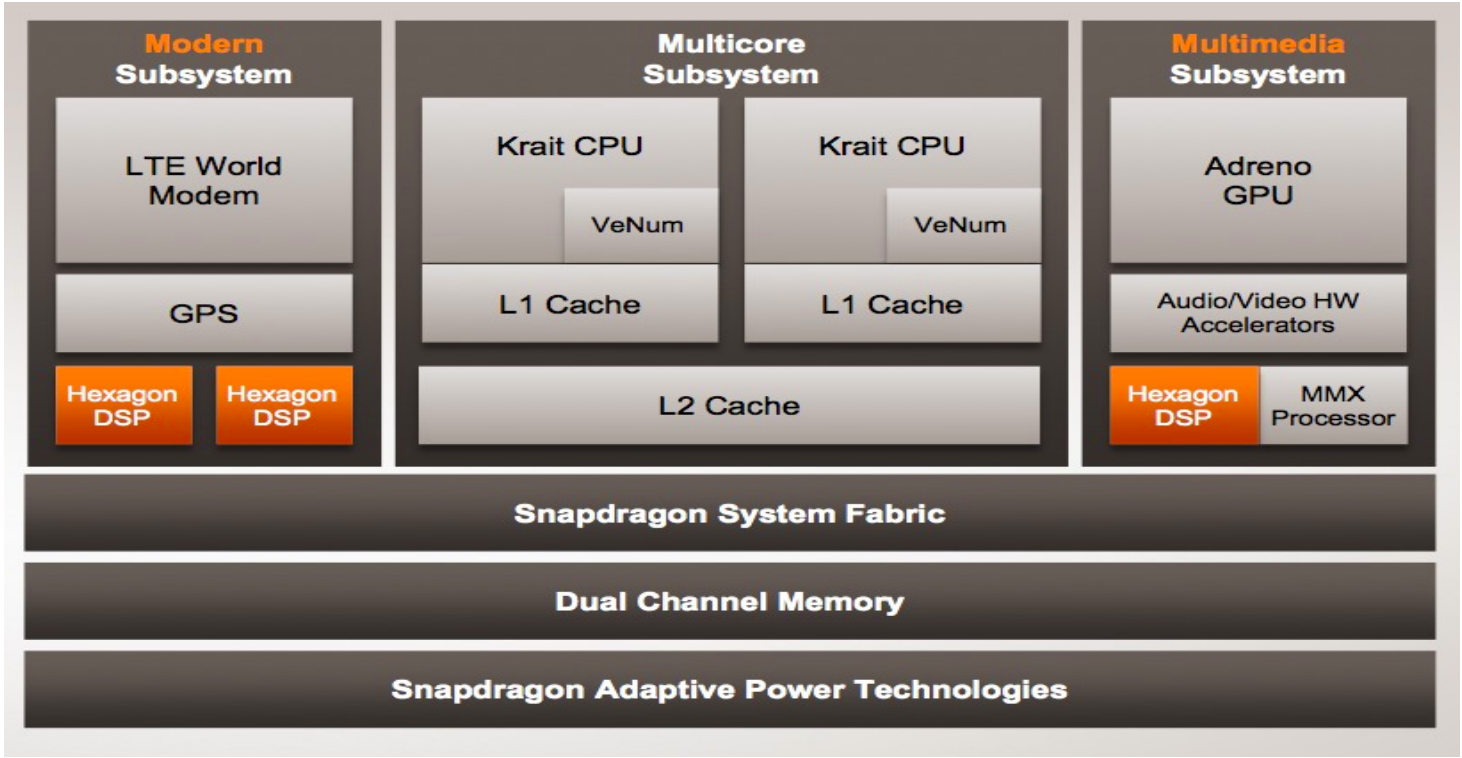
Sonexion: 26 PBs

Cray XK7 Compute Node

XK7 Compute Node Characteristics
AMD Series 6200 (Interlagos)
NVIDIA Kepler
Host Memory 32GB 1600 MT/s DDR3
NVIDIA Tesla X2090 Memory 6GB GDDR5 capacity
Gemini High Speed Interconnect
Keplers in final installation

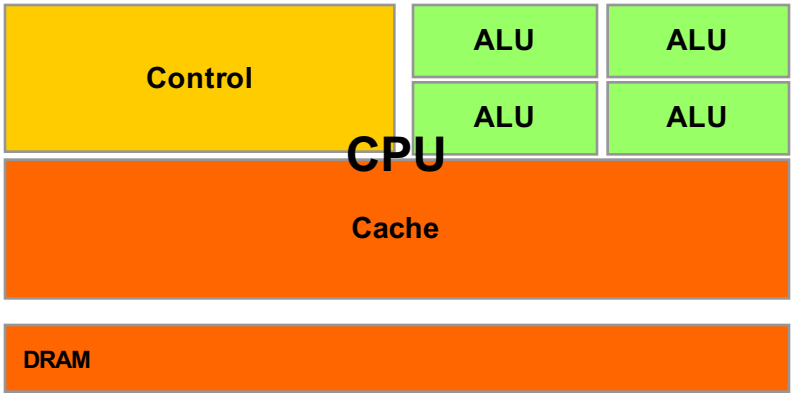


Qualcomm SoC for Mobile



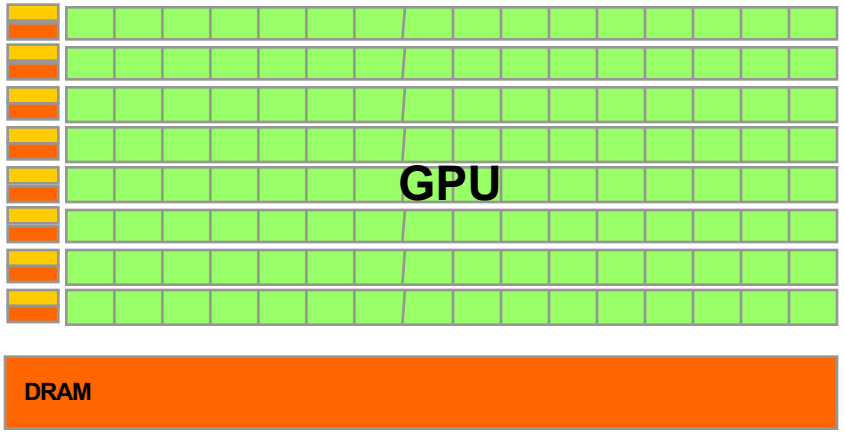
CPUs: Latency Oriented Design

- High clock frequency
- Large caches
  - Convert long latency memory accesses to short latency cache accesses
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency
- Powerful ALU
  - Reduced operation latency



GPUs: Throughput Oriented Design

- Moderate clock frequency
- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many, long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies



Applications Benefit from Both CPU and GPU

- CPUs for sequential parts where latency matters
  - CPUs can be 10+X faster than GPUs for sequential code
- GPUs for parallel parts where throughput wins
  - GPUs can be 10+X faster than CPUs for parallel code

Winning Strategies Use Both CPU and GPU

- CPUs for sequential parts where latency hurts
  - CPUs can be 10+X faster than GPUs for sequential code
- GPUs for parallel parts where throughput wins
  - GPUs can be 10+X faster than CPUs for parallel code

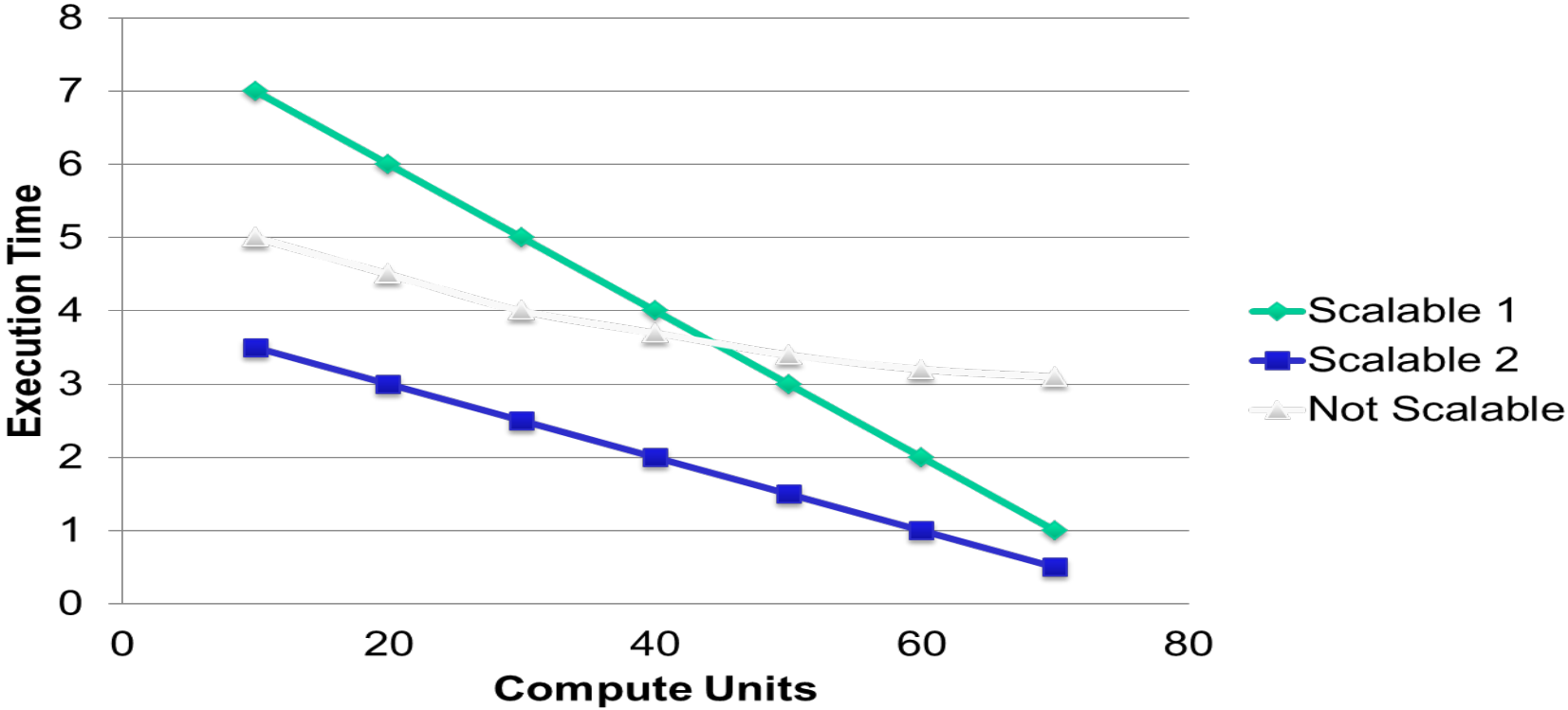
Heterogeneous Parallel Computing Applications

Financial Analysis	Scientific Simulation	Engineering Simulation	Data Intensive Analytics	Medical Imaging
Digital Audio Processing	Digital Video Processing	Computer Vision	Machine Learning	Electronic Design Automation
Biomedical Informatics	Statistical Modeling	Ray Tracing Rendering	Interactive Physics	Numerical Methods

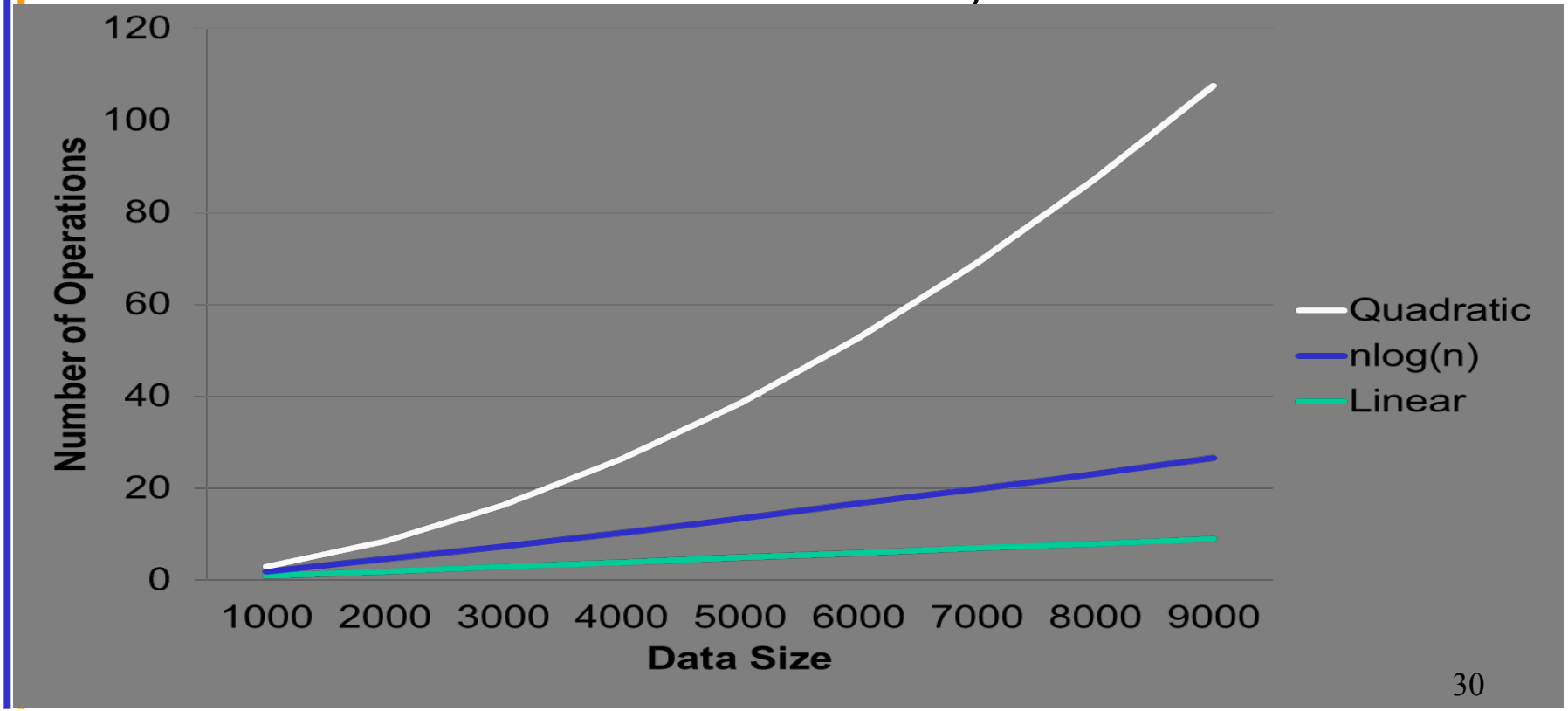
Parallel Programming Work Flow

- Identify compute intensive parts of an application
- Adopt/create scalable algorithms
- Optimize data arrangements to maximize locality
- Performance Tuning
- Pay attention to code **portability**, **scalability**, and **maintainability**

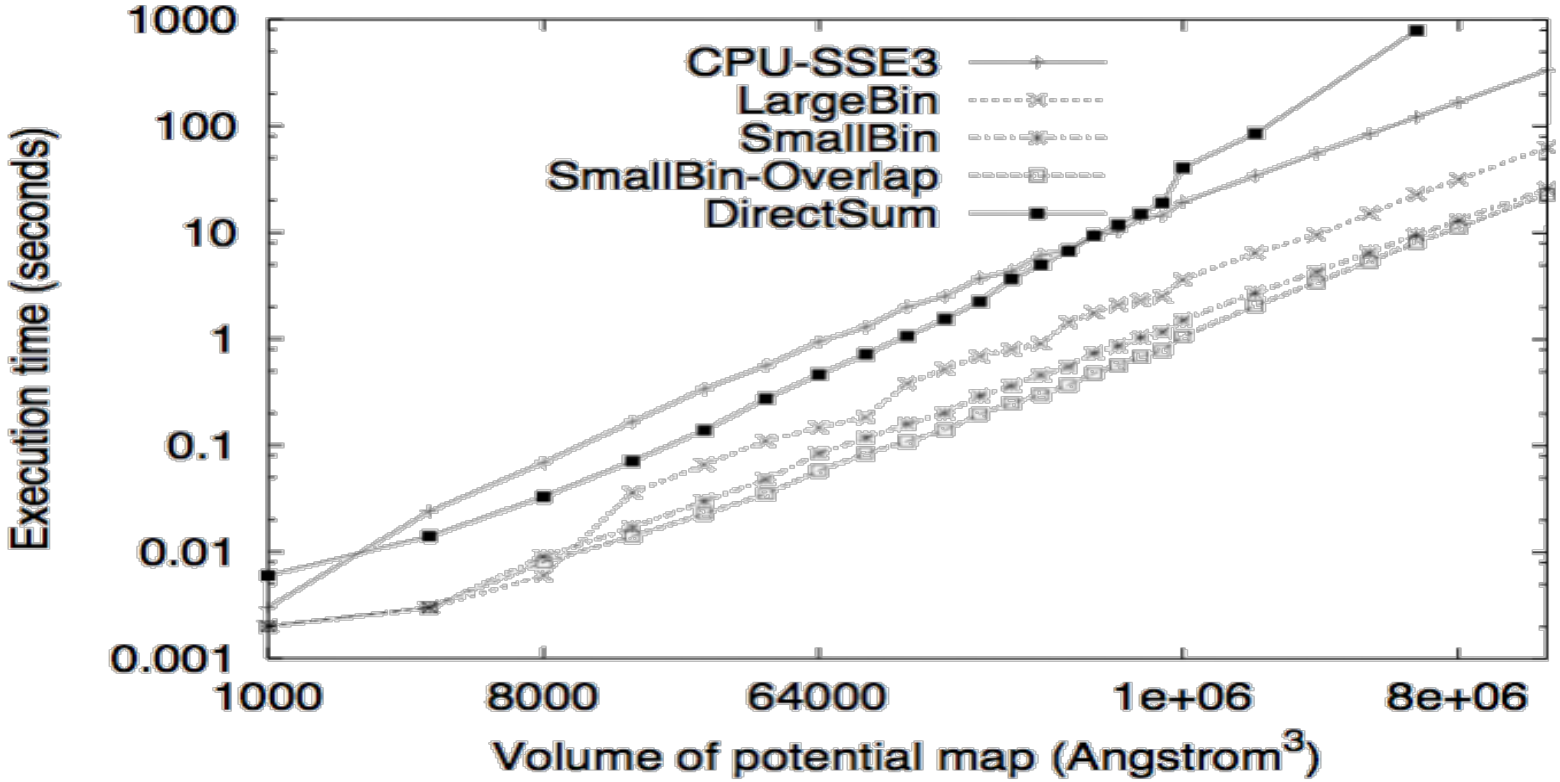
Parallelism Scalability



Algorithm Complexity and Data Scalability



A Real Example of Data Scalability Particle-Mesh Algorithms

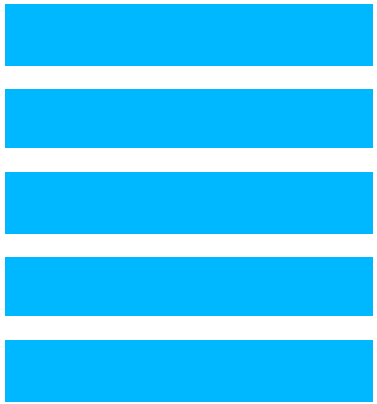




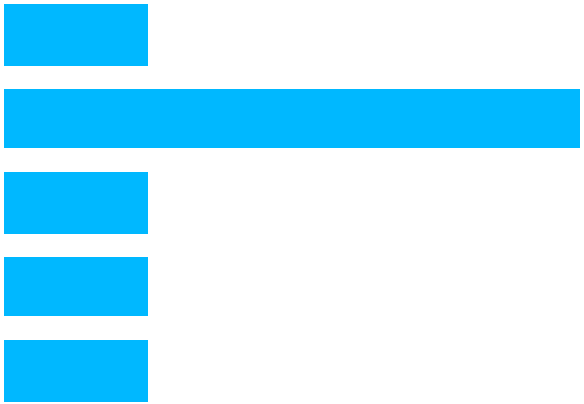

Massive Parallelism - Regularity


Load Balance

- The total amount of time to complete a parallel job is limited by the thread that takes the longest to finish



good



bad!

Global Memory Bandwidth

Ideal

Reality

Conflicting Data Accesses Cause Serialization and Delays

- Massively parallel execution cannot afford serialization

- Contentions in accessing critical data causes serialization

What is the stake?

- Scalable and portable software lasts through many hardware generations

Scalable algorithms and libraries can be the best legacy we can leave behind from this era

ANY MORE QUESTIONS?