CS/EE 217

GPU Programming and Architecture

Lecture 1: Introduction

Course Goals

- Learn how to program GPGPU processors and achieve
  - high performance
  - functionality and maintainability
  - scalability across future generations
- Technical subjects
  - principles and patterns of parallel algorithms
  - processor architecture features and constraints
  - programming API, tools and techniques

Course Staff

- Professor:

  Nael Abu-Ghazaleh

  WCH-441, (951) 827-2347

  Use 217 to start your e-mail subject line

  Office hours: TBD soon; or by appointment

- Teaching Assistants:
  - We will have one!
    - If we can find one
  - Office hours: TBA
- Class may be moving in time and space
  - Sorry, I will let you know soon

Web Resources

- Course website: http://www.cs.ucr.edu/~nael/217-f15
  - Handouts and lecture slides
  - Resources, announcements, projects, …
  - 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.
- Piazza for discussions
  - Channel for electronic announcements
  - Forum for Q&A – course staff read the board, and your classmates often have answers
- iLearn for submissions and grades

Grading

- Exam+Final: 35%
- Labs (Programming assignments): 35%
- Project: 30%
  - Design Document: 25%
  - Project Presentation: 25%
  - Demo/Functionality/Performance/Report: 50%

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
  - 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 for the first occasion
  - Automatic failure of the course for repeat offenses
  - UCR academic honesty policy trumps any instructor policies

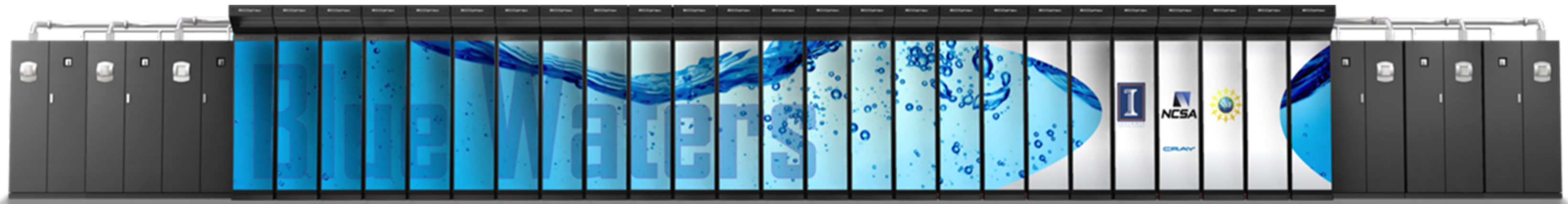
Team Projects

- Work can be divided up between team members in any way that works for you
- However, each team member will demo the final checkpoint of each project individually, and will get a separate demo grade
  - This will include questions on the entire design
  - Rationale: if you don't know enough about the whole design to answer questions on it, you aren't involved enough in the project

Text/Notes

1. D. Kirk and W. Hwu, "Programming Massively Parallel Processors – A Hands-on Approach, Second Edition"
2. CUDA by example, *Sanders and Kandrot*
3. *Nvidia CUDA C Programming Guide*
   - https://docs.nvidia.com/cuda/cuda-c-programming-guide/
4. Occasional research papers
5. Lecture notes on class website
   - Tentative schedule on class website
   - Will try to assign reading ahead of time

Blue Waters Hardware



Cray System & Storage cabinets:	• >300
Compute nodes:	• >25,000
Usable Storage Bandwidth:	• >1 TB/s
System Memory:	• >1.5 Petabytes
Memory per core module:	• 4 GB
Gemin Interconnect Topology:	• 3D Torus
Usable Storage:	• >25 Petabytes
Peak performance:	• >11.5 Petaflops
Number of AMD Interlogos processors:	• >49,000
Number of AMD x86 core modules:	• >380,000
Number of NVIDIA Kepler GPUs:	• >3,000

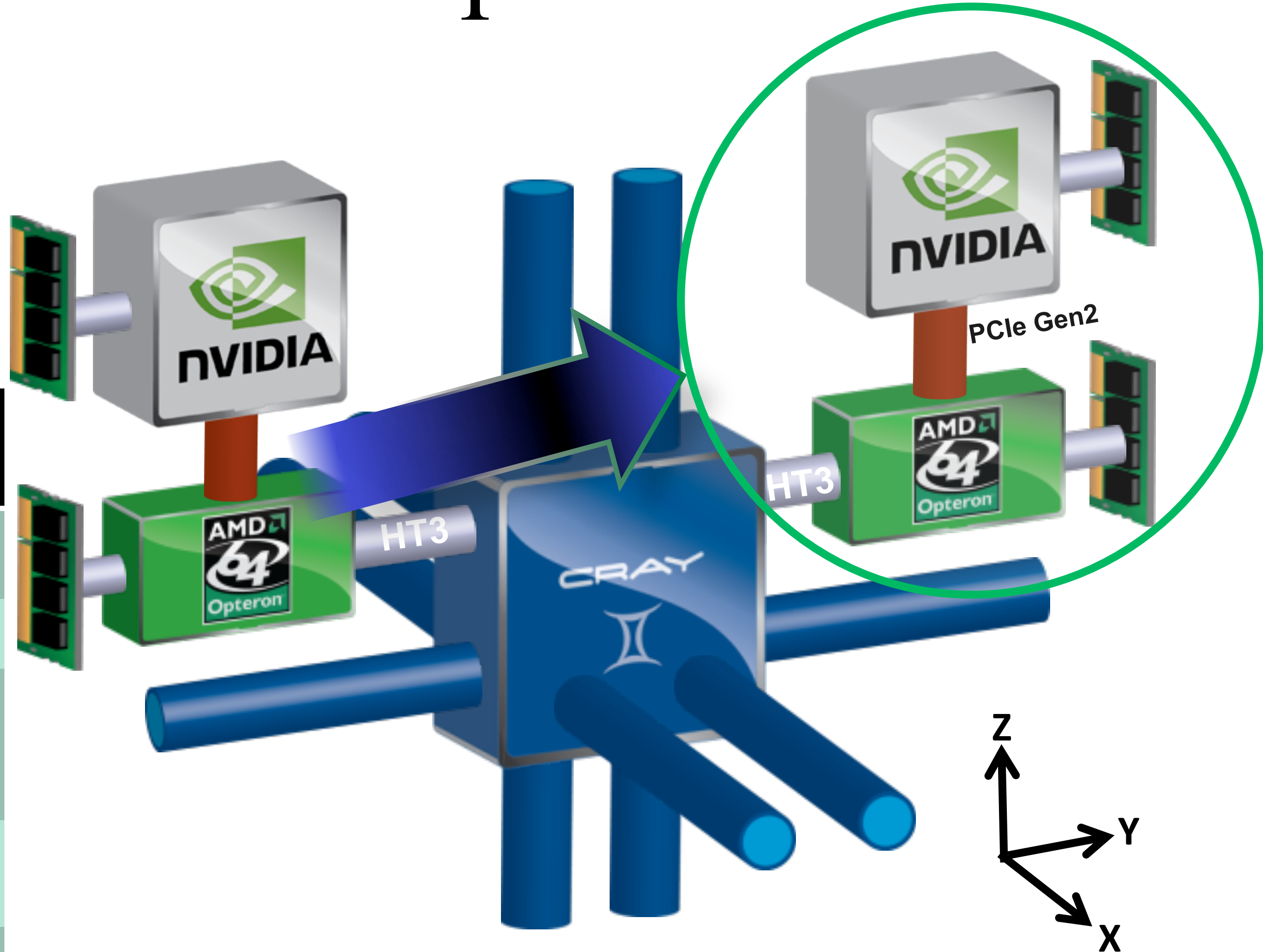
NSF

NCSA

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

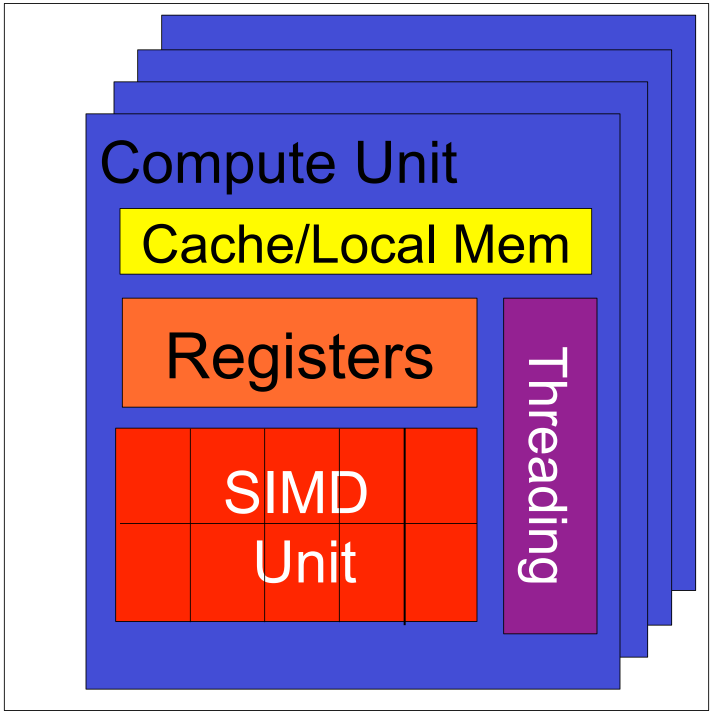
Cray XK7 Compute Node

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

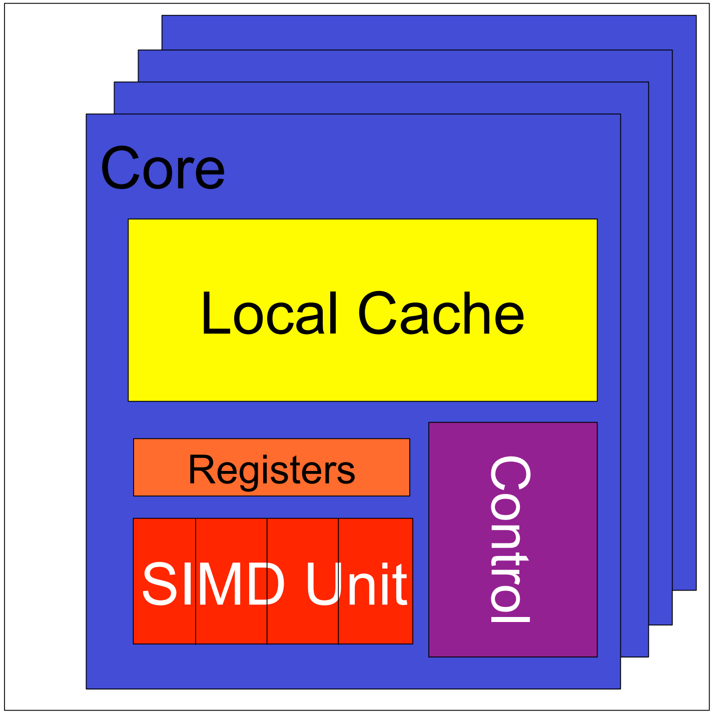


CPU and GPU have very different design philosophy

GPU
Throughput Oriented Cores

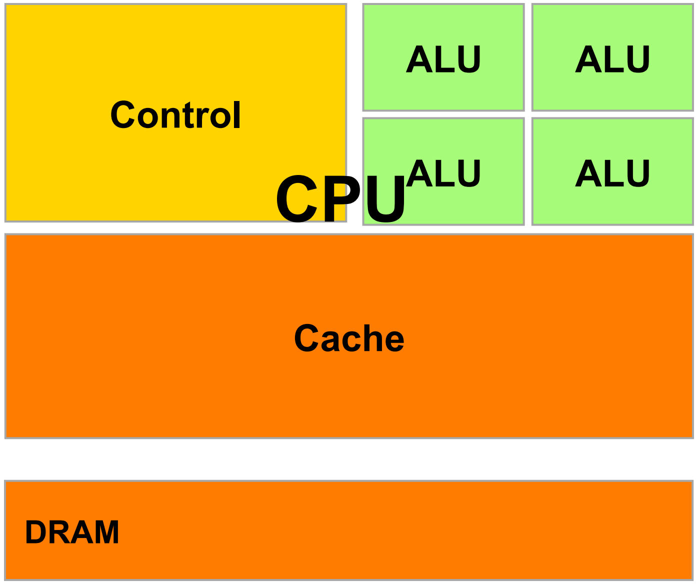


CPU
Latency Oriented Cores



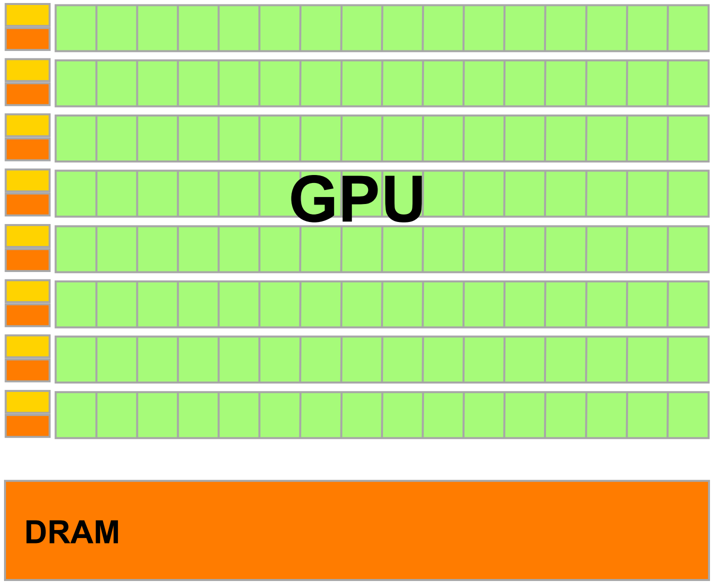
CPUs: Latency Oriented Design

- 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

- 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



Heterogeneous Computing: Use 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

Heterogeneous parallel computing is catching on.

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

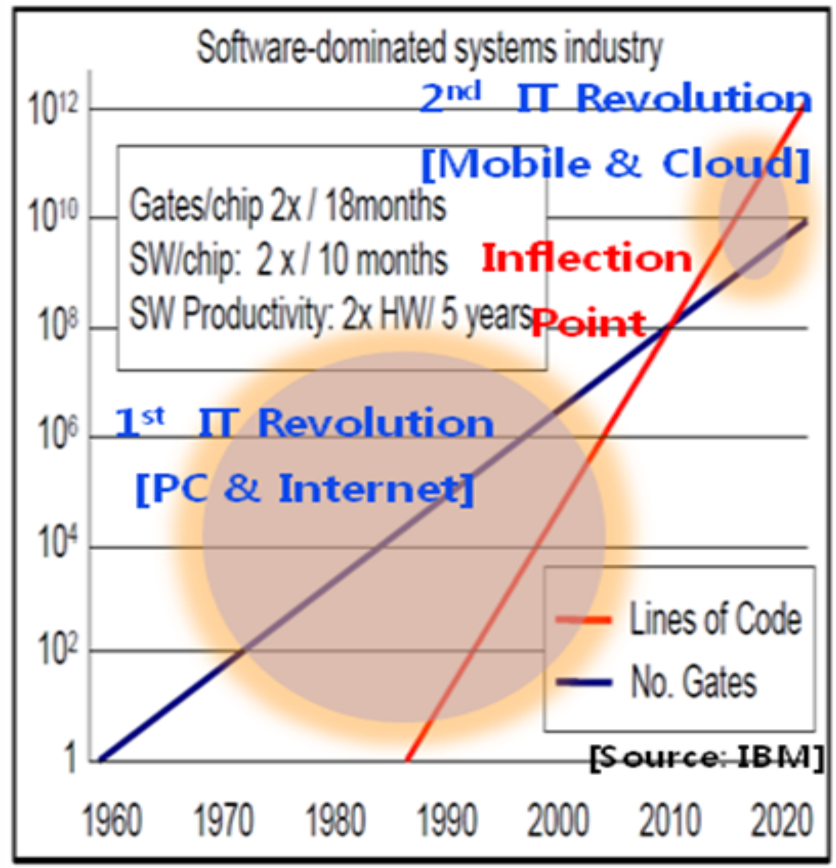
- 280 submissions to GPU Computing Gems
  - 110 articles included in two volumes

Parallel Programming Work Flow

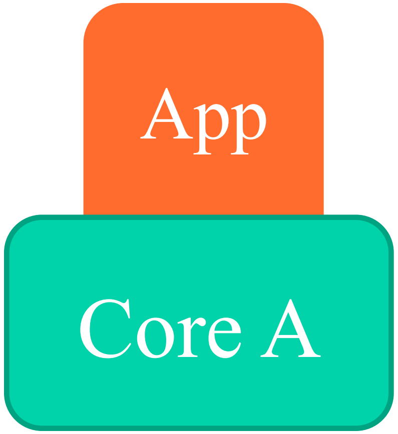
- Identify compute intensive parts of an application
- Adopt scalable algorithms
- Optimize data arrangements to maximize locality
- Performance Tuning
- Pay attention to code portability and maintainability

Software Dominates System Cost

- SW lines per chip increases at 2x/10 months
- HW gates per chip increases at 2x/18 months
- Future system must <u>minimize software redevelopment</u>

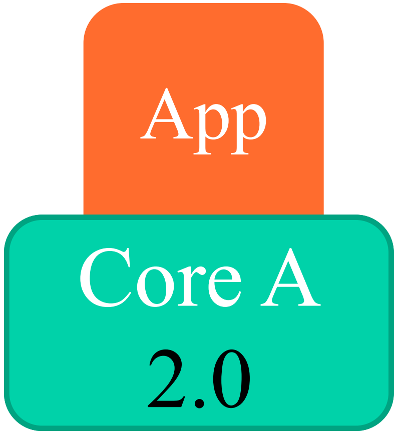


Keys to Software Cost Control



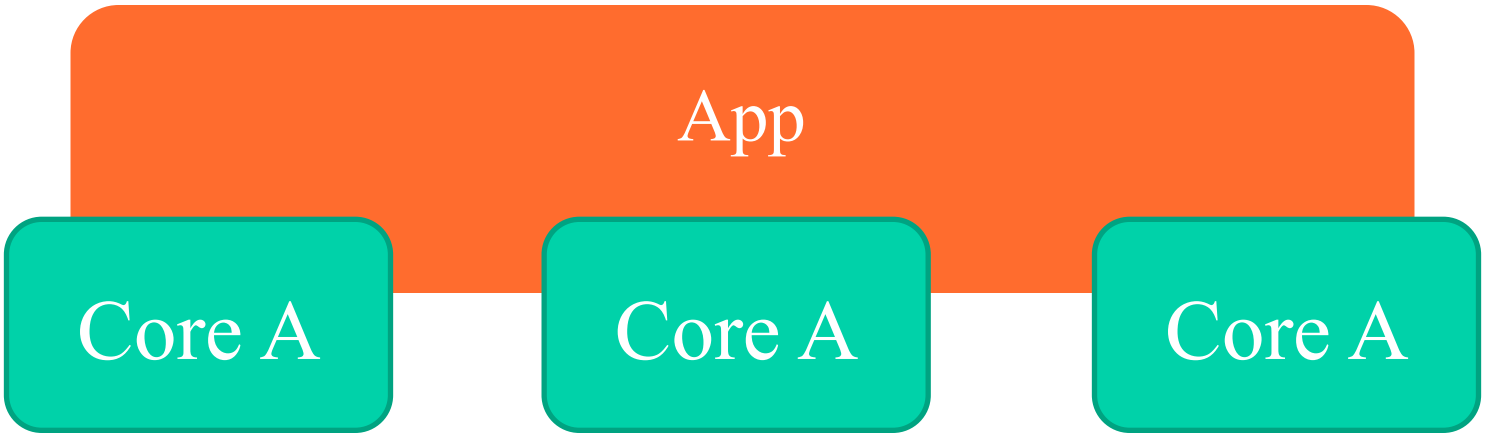
- Scalability

Keys to Software Cost Control



- Scalability
  - The same application runs efficiently on new generations of cores

Keys to Software Cost Control

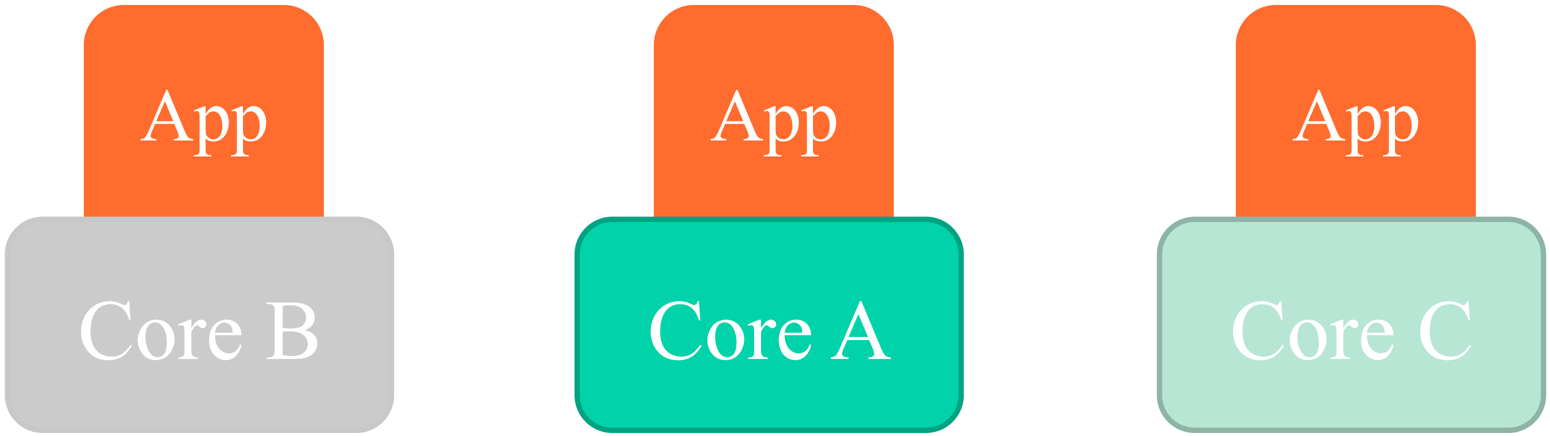


- Scalability
  - The same application runs efficiently on new generations of cores
  - The same application runs efficiently on more of the same cores

Scalability and Portability

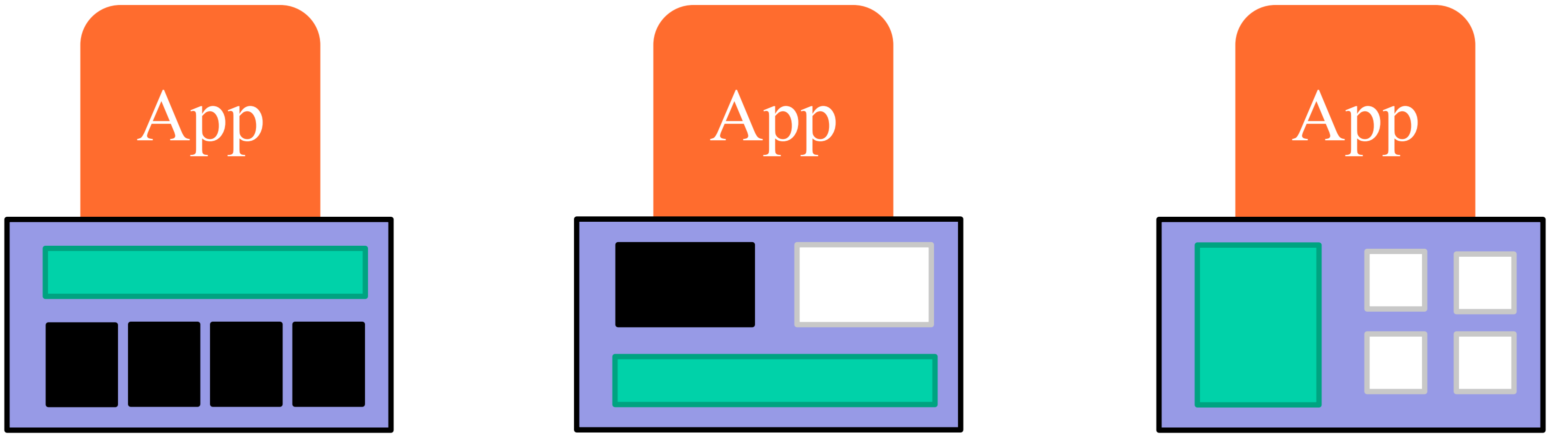
- Performance growth with HW generations
  - Increasing number of compute units
  - Increasing number of threads
  - Increasing vector length
  - Increasing pipeline depth
  - Increasing DRAM burst size
  - Increasing number of DRAM channels
  - Increasing data movement latency
- Portability across many different HW types
  - Multi-core CPUs vs. many-core GPUs
  - VLIW vs. SIMD vs. threading
  - Shared memory vs. distributed memory

Keys to Software Cost Control



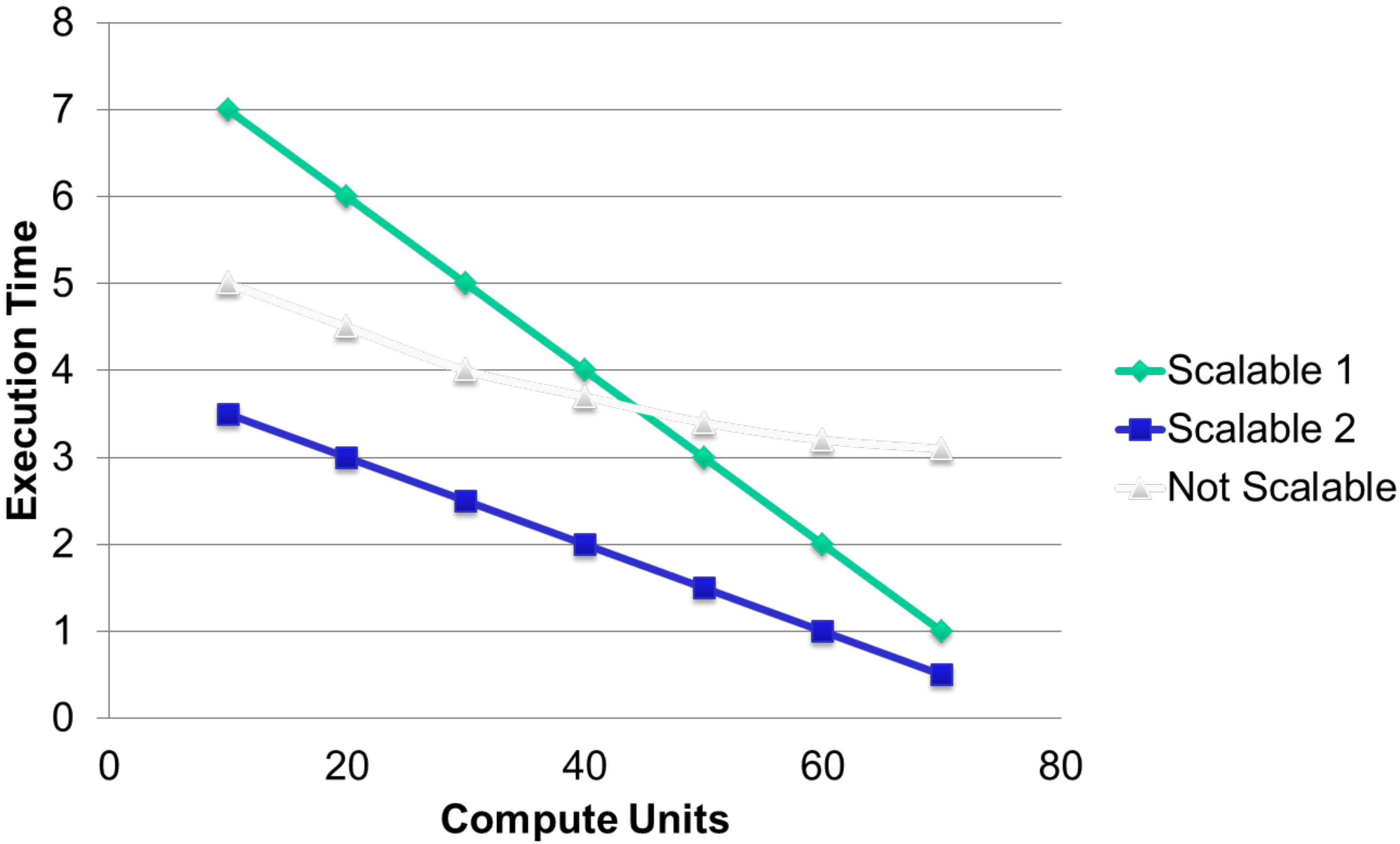
- Scalability
- Portability
  - The same application runs efficiently on different types of cores

Keys to Software Cost Control

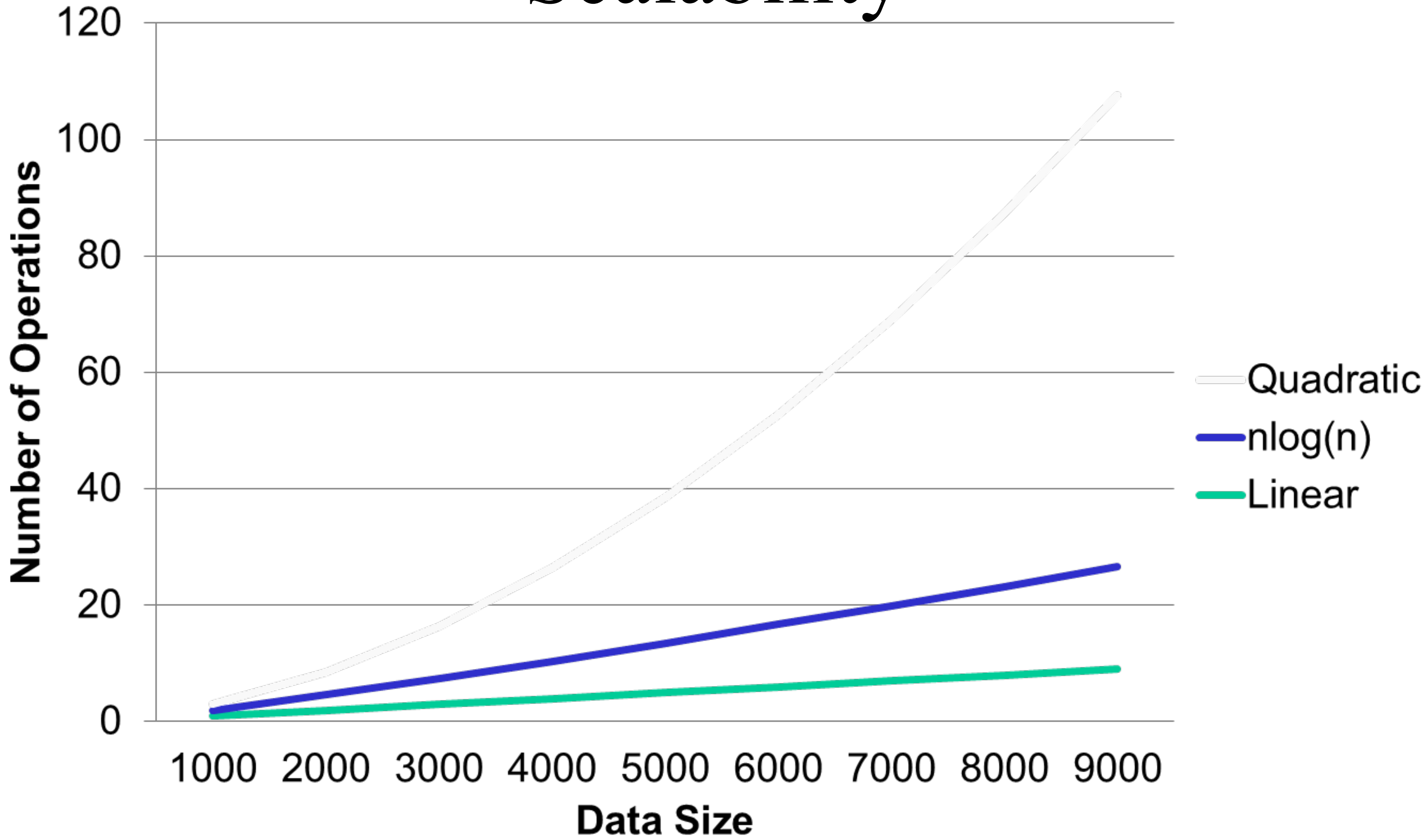


- Scalability
- Portability
  - The same application runs efficiently on different types of cores
  - The same application runs efficiently on systems with different organizations and interfaces

Parallelism Scalability



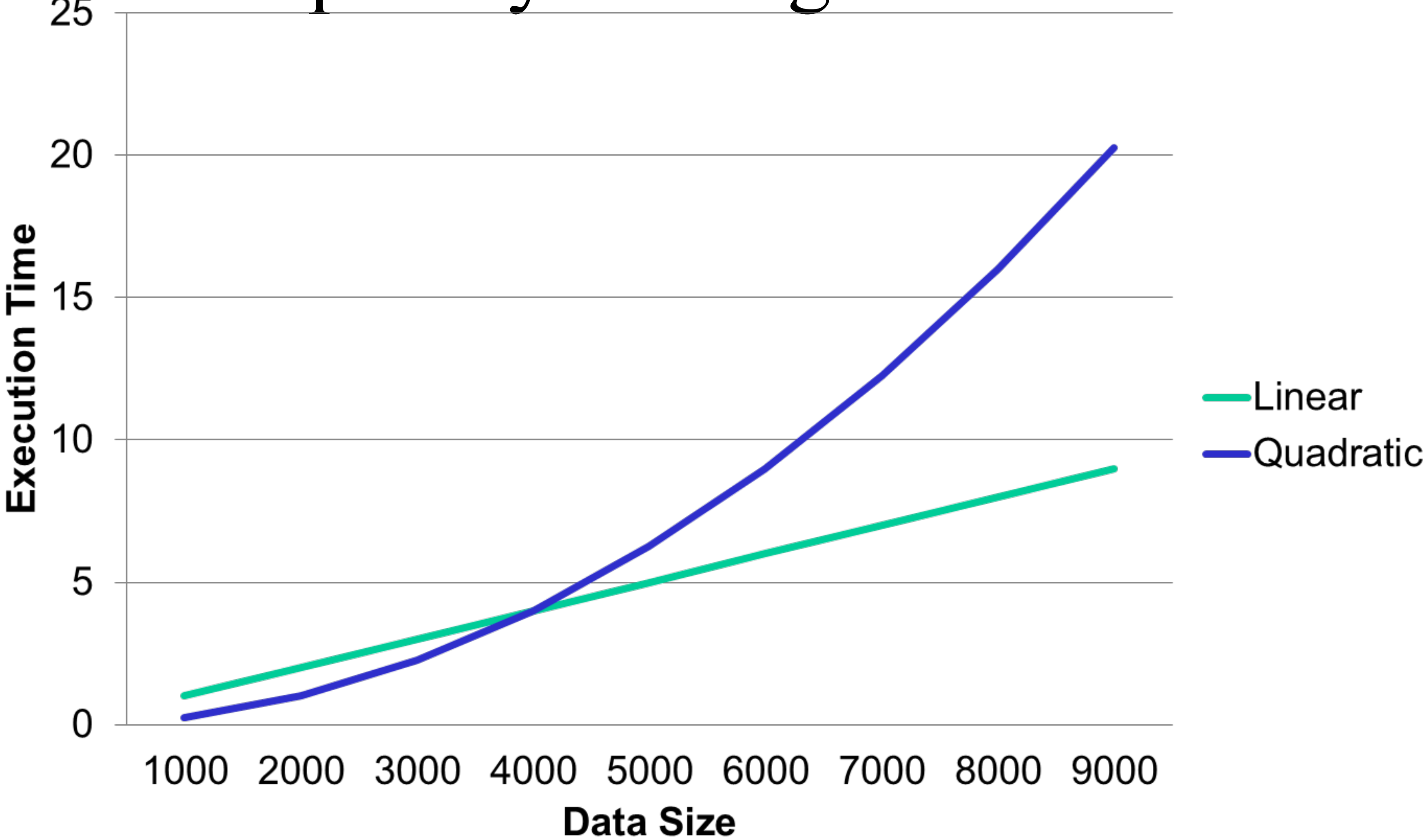
Algorithm Complexity and Data Scalability



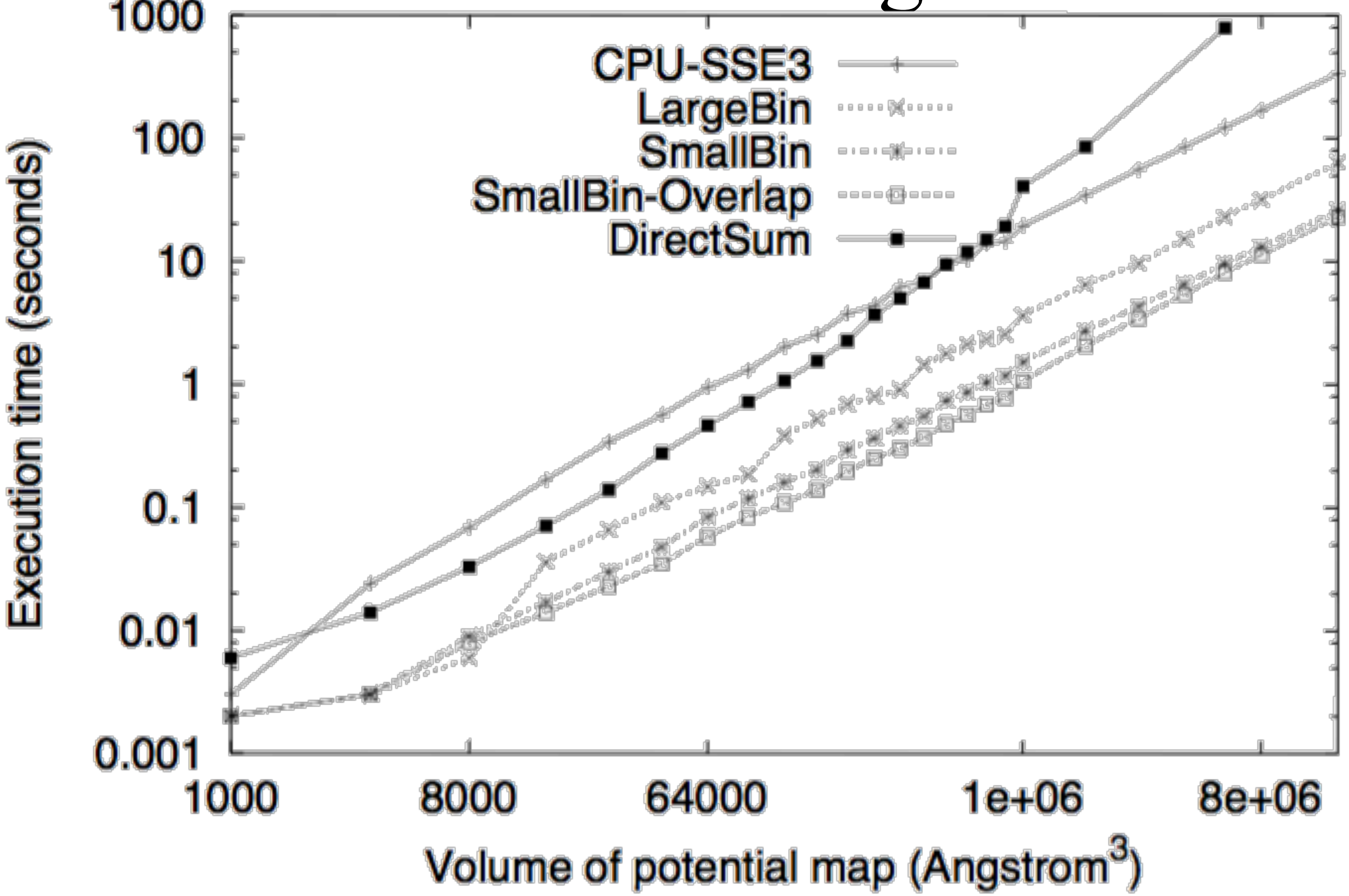
Why is data scalability important?

- Any algorithm complexity higher than linear is not data scalable
  - Execution time explodes as data size grows even for an n*log(n) algorithm
- Processing large data sets is a major motivation for parallel computing
- A sequential algorithm with linear data scalability can outperform a parallel algorithm with n*log(n) complexity
  - log(n) grows to be greater than degree of HW parallelism and makes parallel algorithm run slower than sequential algorithm

Parallelism cannot overcome complexity for large data sets



A Real Example of Data Scalability Particle-Mesh Algorithms



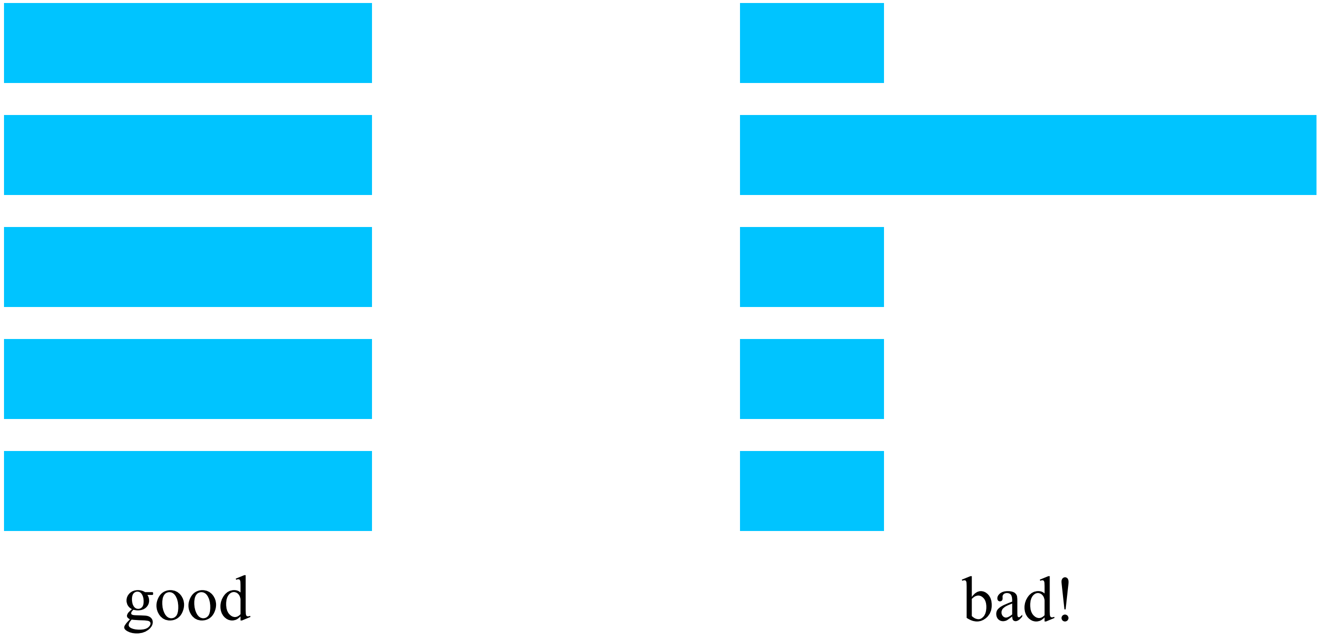


Massive Parallelism - Regularity



Load Balance

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



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

QUESTIONS?