Introduction

CPS343

Parallel and High Performance Computing

Spring 2020

Outline

Outline

Course Details

- Meeting Time and Place: 2:10–3:10 p.m. Monday, Wednesday, and Friday
  - Roughly two days per week we will meet in KOSC 125
  - Approximately one day per week we will meet in KOSC 244
- Course Website:
  `http://www.math-cs.gordon.edu/courses/cps343`
- Office Hours:
  - Monday and Wednesday: 3:20–4:20 p.m.,
  - Tuesday and Thursday: 1:00–3:00 p.m.,
  - and by appointment.
- Prerequisites: Ability to program in C/C++ and Python. Knowledge of computer organization, parallel programming constructs such as Java threads, and experience with linear algebra/matrices will all be helpful.

Textbooks

Our main textbook is **Multicore and GPU Programming: An Integrated Approach** by Gerassimos Barlas, Morgan Kaufman/Elsevier, 2015.

The following recommended additional texts are available free on-line:

1. **Introduction to High-Performance Scientific Computing** by Victor Eijkhout, 2016.
2. **Parallel Computing for Science and Engineering** by Victor Eijkhout, 2017.
3. **Designing and Building Parallel Programs** by Ian Foster, 1995.
4. **MPI: The Complete Reference** by Marc Snir, et.al., 1996.

Links for these can be found on the course web site.

Course Content

This course will cover the following topic areas:

- Overview and history of parallel and high performance computing
- Parallel and high performance computing (HPC) architectures
- Tools for parallel programming (OpenMP, MPI, CUDA, OpenACC, etc.)
- High performance issues (memory hierarchy, caching, bandwidth, etc.)
- Parallel computation issues (partition, synchronization, load balancing, etc.)
- Survey of parallel application problems

Course Objectives

After completing this course you should be able to:

- analyze a programming task and identify what portions admit a parallel implementation
- Use various numerical libraries and work with common formats for large data files
- use OpenMP to develop applications for multi-core computers
- use the MPI standard to develop applications for clusters
- use CUDA, OpenACC and/or Thrust to develop applications for GPU hardware
- understand and address issues arising in installation and management of HPC systems

Outline

Mobile Device Policy

- Laptops, tablets, and other mobile devices may be used only when appropriate for the current classroom activity.
- You may not use a mobile phone or other device for texting or otherwise communicating with others during class. (This activity prevents you from fully concentrating on our topic and is distracting to those around you and to the professor.)

Attendance and Participation

- Regular, consistent attendance is expected
- Assigned readings must be completed before class so you are able to discuss the covered material
- You are expected to be engaged in all class discussions and activities
- Class will be a mixture of presentation and discussion: you are encouraged to ask questions as the arise!

Homework Assignments

Homework problems will be assigned throughout the semester.

- some will be pencil (or pen) and paper problems
- some will involved programming
- some may involve writing short reports

You are permitted to discuss problems with one another but all written work or code you turn in should reflect your own understanding.

Projects

1. I anticipate assigning four programming projects:
   - performance measuring and tuning and working with large data files
   - using OpenMP
   - using MPI
   - using CUDA

2. There will also be a final project at end of the semester.
   - Topics will be chosen in consultation with me
   - Each student will make a presentation to the class on their work at the end of the semester
   - Collaboration on a single project is not allowed. . .
     . . . but two students may choose related or complementary projects and share information
   - You are encouraged to talk about your projects with each other and help each other as necessary

Quizzes and Exams

Quizzes

- approximately seven quizzes during the semester
- may be announced or unannounced
- short — 10 minutes or less
- cover assigned reading, class presentation, homework, hands-on exercises, or projects

Exams

- two hour-long exams
- each exam will cover about half of the material

Grading weights

Component	Percentage
Class preparedness and participation	10%
Written assignments	15%
Quizzes	10%
Programming projects	30%
Final project and presentation	15%
Midterm exam	10%
Final exam	10%

Outline

Definitions

Parallel Computing

By *parallel computing* we mean any computing that consists of multiple tasks being executed simultaneously as part of a single job.

This is different than *multiprocessing*, which refers to a computer having multiple on-going jobs at any one time.

Definitions

Parallel Computing

By *parallel computing* we mean any computing that consists of multiple tasks being executed simultaneously as part of a single job.

This is different than *multiprocessing*, which refers to a computer having multiple on-going jobs at any one time.

High Performance Computing

Synonymous with *supercomputing*, HPC is focused on running jobs that

- take a long time to run but must be run as fast as possible (different than *real-time* programming where speed is also very important but at a different scale), and/or
- use more data than can be handled by "normal" computers.

Definitions

Parallel Computing

By *parallel computing* we mean any computing that consists of multiple tasks being executed simultaneously as part of a single job.

This is different than *multiprocessing*, which refers to a computer having multiple on-going jobs at any one time.

High Performance Computing

Synonymous with *supercomputing*, HPC is focused on running jobs that

- take a long time to run but must be run as fast as possible (different than *real-time* programming where speed is also very important but at a different scale), and/or
- use more data than can be handled by "normal" computers.

Thus, HPC is an umbrella which covers parallel computing in the cases where the goal of the parallel program is to tackle large jobs as quickly as possible.

Definitions

Parallel Computing

the use of two or more processors (computers), *usually in the same system*, working simultaneously to solve a single problem.

Definitions

Parallel Computing

the use of two or more processors (computers), *usually in the same system*, working simultaneously to solve a single problem.

Distributed Computing

any computing that involves multiple computers *remote from each other* that each have a role in a computation problem or information processing.

Definitions

Parallel Computing

the use of two or more processors (computers), *usually in the same system*, working simultaneously to solve a single problem.

Distributed Computing

any computing that involves multiple computers *remote from each other* that each have a role in a computation problem or information processing.

Characteristic	**Parallel**	**Distributed**
Overall goal	speed	convenience
Interactions	frequent	infrequent
Granularity	fine	course
Reliability	assumed	not assumed

Why study parallel computing?

Question: Why study parallel computing?

Answer:

Why study parallel computing?

Question: Why study parallel computing?

Answer: Because virtually all computers are now parallel!

Why study parallel computing?

Question: Why study parallel computing?

Answer: Because virtually all computers are now parallel!

- single-core computers stopped getting faster around 2005.

Why study parallel computing?

Question: Why study parallel computing?

Answer: Because virtually all computers are now parallel!

- single-core computers stopped getting faster around 2005.
- multi-core processors have become ubiquitous—from computers, to tablets and phones

Why study parallel computing?

Question: Why study parallel computing?

Answer: Because virtually all computers are now parallel!

- single-core computers stopped getting faster around 2005.
- multi-core processors have become ubiquitous—from computers, to tablets and phones
- effective parallel algorithms are often different than effective serial algorithms

Why study parallel computing?

Question: Why study parallel computing?

Answer: Because virtually all computers are now parallel!

- single-core computers stopped getting faster around 2005.
- multi-core processors have become ubiquitous—from computers, to tablets and phones
- effective parallel algorithms are often different than effective serial algorithms
- parallel algorithms are necessary to take advantage of computer hardware now and (at least) in the near future

Why study high performance computing?

Question: Why study high performance computing?

Answer:

Why study high performance computing?

Question: Why study high performance computing?

Answer: Because performance is always important!

Why study high performance computing?

Question: Why study high performance computing?

Answer: Because performance is always important!

- Supercomputers occupy an important niche; not a lot of consumer appeal but indispensable in the modern world

Why study high performance computing?

Question: Why study high performance computing?

Answer: Because performance is always important!

- Supercomputers occupy an important niche; not a lot of consumer appeal but indispensable in the modern world
- getting the best performance out of a program is important for all levels of coding; from embedded controllers to mobile devices to supercomputers

Scientific Computing

Much of this class will be focused on problems that are categorized as *scientific computing* or *numerical computing* problems.

Such problems include:

- Simulation (weather forecasting, galaxy formation, fluid flow)
- Data mining (gene sequencing, signal processing)
- Machine Learning
- Visualization to present *data* as *pictures* to provide understanding and insight

While HPC is often associated with scientific and numerical computing, it has applications in many other areas as well — consider Google's ability to quickly resolve search queries.

FLOPS

- A *FLOP* is a **F**loating **P**oint **OP**eration.
- The term *FLOPS* means **F**loating **P**oint **O**peration **P**er **S**econd. Also written as *FLOP/S*. Note that FLOPs is the plural of FLOP.
- *MFLOPS* means 1 million FLOPS
- Sometimes this means 1000^2 and sometimes it means 1024^2.
- Todays laptop computers are capable of computations in the *GFLOPS* (gigaflops) range; 1 GFLOP = 1000 MFLOP.
- Most current supercomputers work in the *TFLOPS* (teraflops) or *PFLOPS* (petaflops) range
  - 1 TFLOP = 1000 GFLOP
  - 1 PFLOP = 1000 TFLOP.

Exascale Challenge

Prior to 2016 the goal was to have computers working in the *EFLOPS* (exaflops) range by 2018. This is usually referred to as "exascale" computing to emphasize the fact that it's not just computation rate but also data set size that will be HUGE!

Exascale Challenge

Prior to 2016 the goal was to have computers working in the *EFLOPS* (exaflops) range by 2018. This is usually referred to as "exascale" computing to emphasize the fact that it's not just computation rate but also data set size that will be HUGE!

How did we do?

Exascale Challenge

Prior to 2016 the goal was to have computers working in the *EFLOPS* (exaflops) range by 2018. This is usually referred to as "exascale" computing to emphasize the fact that it's not just computation rate but also data set size that will be HUGE!

How did we do? We didn't make it.

In November 2017 the fastest computer in the world (according to the most commonly used benchmark) achieved 93 PFLOPs, It required over 15,370 kW of electricity every hour. Using the current residential rate for electricity in Massachusetts, it cost over $1,840 per hour to run.

Exascale Challenge

Prior to 2016 the goal was to have computers working in the *EFLOPS* (exaflops) range by 2018. This is usually referred to as "exascale" computing to emphasize the fact that it's not just computation rate but also data set size that will be HUGE!

How did we do? We didn't make it.

In November 2017 the fastest computer in the world (according to the most commonly used benchmark) achieved 93 PFLOPs, It required over 15,370 kW of electricity every hour. Using the current residential rate for electricity in Massachusetts, it cost over $1,840 per hour to run.

Two years later, in November 2019, the fastest computer achieved 148.6 PFLOPs. However, it only requires 10,000 kW of electricity per hour, or about $1,000 per hour.

Exascale Challenge

To develop a practical exascale computer we need to overcome multiple challenges:

- Processor architecture is still unknown.
- System power is the primary constraint for the exascale system: simply scaling up from today's requirements for a petaflop computer, the exaflop computer in 2020 would require 200 MW, which is untenable. The target is 20–40 MW in 2020 for 1 exaflop.
- Memory bandwidth and capacity are not keeping pace with the increase in flops: technology trends against a constant or increasing memory per core. Although the memory per flop may be acceptable to applications, memory per processor will fall dramatically, thus rendering some of the current scaling approaches useless
- Clock frequencies are expected to decrease to conserve power; as a result, the number of processing units on a single chip will have to increase – this means the exascale architecture will likely be high-concurrency – billion-way concurrency is expected.

Exascale Challenge

- Cost of data movement, both in energy consumed and in performance, is not expected to improve as much as that of floating point operations, thus algorithms need to minimize data movement, not flops
- A new programming model will be necessary: even heroic compilers will not be able to hide the level of concurrency from applications
- The I/O system at all levels – chip to memory, memory to I/O node, I/O node to disk – will be much harder to manage, as I/O bandwidth is unlikely to keep pace with machine speed
- Reliability and resiliency will be critical at the scale of billion-way concurrency: "silent errors," caused by the failure of components and manufacturing variability, will more drastically affect the results of computations on exascale computers than today's petascale computers

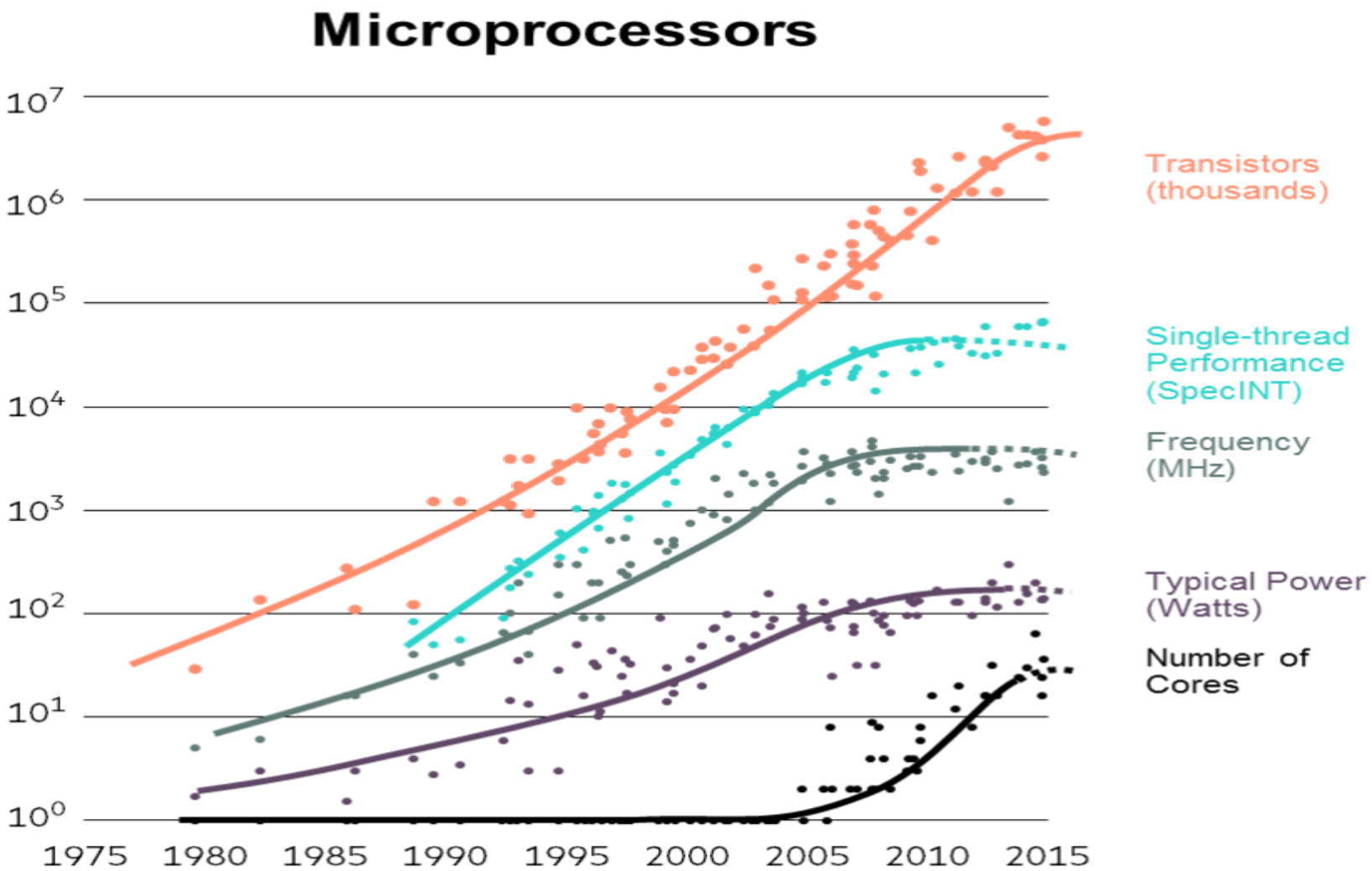
Outline

Moore's Law

What we now call **Moore's Law** was described by Gordon Moore in 1965 as the tendency for the number of transistors in an integrated circuit to double approximately every two years.

- sometimes quoted as *eighteen months* rather than two years
- transistor density has been (and still can) continue to increase
- clock speeds have leveled off

Moore's Law



https://www.weforum.org/agenda/2018/09/end-of-an-era-what-computing-will-look-like-after-moores-law/

Power and Heat

Notes on the graph:

- The "number of transistors" curve continues to follow Moore's Law
- The curves we really care about (clock speed, power) have leveled off
- In fact clock speed has actually started to decline...
- The problem is really one of *power density*; how much power is being consumed (and therefore heat generated) per unit area

Power and Heat

Notes on the graph:

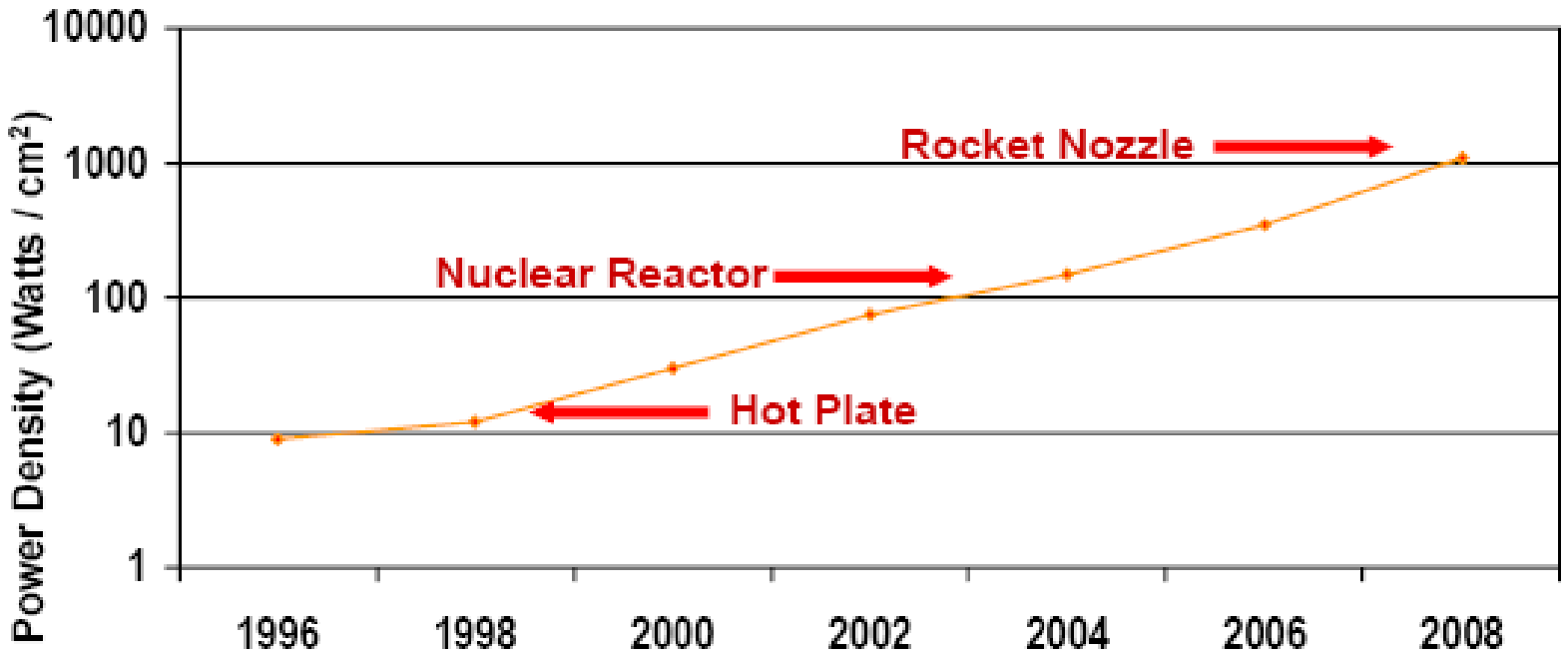
- The "number of transistors" curve continues to follow Moore's Law
- The curves we really care about (clock speed, power) have leveled off
- In fact clock speed has actually started to decline...
- The problem is really one of *power density*; how much power is being consumed (and therefore heat generated) per unit area

You may have noticed that the number of cores seems to be leveling off as well – more about this later.

Power and Heat

Moore's Law Extrapolation:

Power Density for Leading Edge Microprocessors



Power Density Becomes Too High to Cool Chips Inexpensively

Source: Shekhar Borkar, Intel Corp

Source: `http://www.cs.kent.edu/~jbaker/ParallelProg-Sp11/slides/Baker-Chpt%201.ppt`

Shift to Multi-core

- Drive to increasingly complex processor design and faster clock speeds replaced with increased number of processor cores
- current CPUs typically have between 2 and 16 cores
- GPU devices typically have between 16 and several thousand cores (Nvidia Tesla P100 GPU has 3,584 CUDA cores, 16GB of CoWoS HBM2 memory, and is capable of 4.7 Teraflops (double precision))
- MIC devices have many more cores than the typical CPUs (Intel Xeon Phi Coprocessor 7295 has 72 cores running at 1.5GHz (1.6GHz max))

Shift to Multi-core

- Drive to increasingly complex processor design and faster clock speeds replaced with increased number of processor cores
- current CPUs typically have between 2 and 16 cores
- GPU devices typically have between 16 and several thousand cores (Nvidia Tesla P100 GPU has 3,584 CUDA cores, 16GB of CoWoS HBM2 memory, and is capable of 4.7 Teraflops (double precision))
- MIC devices have many more cores than the typical CPUs (Intel Xeon Phi Coprocessor 7295 has 72 cores running at 1.5GHz (1.6GHz max))

Moral: *Today all computers are parallel computers!*