Compsci 590.3: Introduction to Parallel Computing

Alvin R. Lebeck

Some slides based on those from the University of Oregon
General Information

• Instructor: Alvin R. Lebeck
  Office: D308 LSRC
  email: alvy@duke.edu
  Office Hours: Monday 2:00-3:00, Tuesday 3-4pm (except tomorrow 😔)
    » If you have classes/work/practice at these times, email me to schedule a meeting.

• No TAs … affects how this course works

• Introductions…
  ▪ Name: grad/ugrad, dept & field of interest, why you are here…
What this course is and is not

• Introduction to parallel programming
• Introduction to parallel hardware (high level)
• Introduction to several parallel programming environments
• Introduction to performance in parallel systems

• Not a parallel algorithms course
• Not a parallel architecture course
• Not a parallel systems course
• Not an advanced methods for HPC course
Goals of the Course

• Understand parallelism in modern computers
• How to develop parallel programs using several parallel programming paradigms
• Understand a common set of “patterns” for parallelism
• Understand performance issues for parallel programs
• Complete a sizable parallel program in a group…

• If, at any point, it’s not clear why I’m talking about some topic, please ask!
Access to Information and Communication

• Course web page
  ▪ [https://www.cs.duke.edu/courses/fall15/compsci590.3/](https://www.cs.duke.edu/courses/fall15/compsci590.3/)
  ▪ Syllabus, schedule, homework, documentation/resources, etc.
  ▪ Slides available shortly before or after class…

• Git repository for submission (gitlab.oit.duke.edu)

• Sakai
  ▪ For posting grades
  ▪ Some resources
  ▪ Bulk email announcements occasionally

• Piazza for announcements, Q&A
  ▪ Use this instead of sending email to me, best way to get a quick answer
  ▪ Many eyes on questions & answers, minimizes confusion and/or misinformation

• You are expected to monitor these modes of communication
General Info

• I AM NOT PERFECT!
  ▪ Ask questions, if you are confused or need clarification others probably need help too.

• Required Textbook
  ▪ Additional online material
Grading

• Grade Breakdown
  ▪ 10% Class Participation
  ▪ 30% Homework/Labs (groups)
  ▪ 20% Midterm exam (in class)
  ▪ 40% Class Project (Presentation and Document)

• Late Policy
  ▪ No late submissions.
  ▪ In fairness to all, only exceptions are Dean’s excuses
• No extra credit in this course
• Plan your time accordingly…
Homework and Project

• Homework
  ▪ Groups of 3 (one or two exceptions due to number of students)
  ▪ Programming exercises, (approximately one per week)
    » given serial code
    » make it parallel
    » analyze performance
  ▪ Graded on 0-3 scale

• Project
  ▪ Groups of 3 or more
  ▪ Goal: Scalable parallel application
  ▪ Three parts:
    1. Proposal (written and short presentation, October 7)
    2. Final Presentation (Nov 23)
    3. Final Report (Dec 4)
Academic Conduct

• Do Not Cheat!
• Academic Conduct
  ▪ Duke Community Standard
  ▪ Studying high level concepts together in groups is encouraged
  ▪ Common examples of cheating: running out of time on an assignment and then use someone else's solution, person asks to borrow solution “just to take a look”, using previous semester solutions, using solutions from the internet or any archive (paper or electronic), copying an exam question, line by line help from another student …
• Zero tolerance for academic misconduct
  ▪ We do use software to compare your code/solutions, and it works really well!
• Homework and projects are in groups, but work must be within the group and include appropriate attribution to each group member.
Course Problems

• Can’t make midterm, other conflicts
  ▪ Dean’s excuses are only acceptable reasons to reschedule
  ▪ Tell me early and I will schedule an alternate time
• If you are having problems
  ▪ See me
  ▪ See DUS/DGS
  ▪ See Academic Dean (very good resource)
Our Environment

• We will use the following machines for this course.
  ▪ <phe, phi, pho, phum>.cs.duke.edu
  ▪ Access is via ssh using a terminal (login with Duke NetID)
  ▪ No shared file system (see git below)
• Compilers are from Intel (icc and icpc)
• Unix make utility
• Some shell scripts
• Git
  ▪ Source code version control / project management
  ▪ Retrieve source I provide
  ▪ Work together on homework & project
  ▪ Submit your solutions
• SSH Keys
  ▪ Simplifies access to git and remote login to machines
Todo

• Remote Access (docs page of web site)
  ▪ If you registered for class Friday or later or not yet, I need NetID to grant access to machines.

• Unix tutorial (docs page of web site)

• C & C++tutorial (docs page of web site)

• Signup for piazza

• Homework 0 (Getting Started)
  ▪ Sign in to gitlab.oit.duke.edu
  ▪ Set up SSH keys (see http://doc.gitlab.com/ce/ssh/README.html)
  ▪ Work with repository
  ▪ Part 1. Compile and run serial hello world

• Now for the real stuff…first any questions?
Questions

1. What is parallel programming? High Level
2. Why do we need parallel programming?
3. What are some applications?
4. What are the forms of parallelism?
5. What things do you need to worry about when writing parallel code?
1. What is Parallel Programming?

- Writing programs that exploit multiple hardware resources simultaneously to solve a given problem.
- This is not just concurrency!
  - There is concurrency on computers that can do only one operation at a time---Operating System support for context switching.
- It is not just having parallel hardware!
  - You can write programs that are not parallel and run many of them on computers with many processors.
- Concurrency + parallel hardware = parallel processing
- Writing programs for parallel processing = parallel programming
2. Why do we need parallel programming?

- We want more performance
  - Solve same problem size faster
  - Solve a bigger problem in same time
  - Solve a bigger problem with constant error

- The hardware is parallel....so we should use it

- Let’s go a bit deeper on this…
System Organization

- Processor
- Cache
- Memory Bus
- I/O Bridge
- Core Chip Set
- Main Memory
- I/O Bus
  - Disk Ctrl
  - Flash Ctrl
  - Graphics Controller
  - Network Interface
- Disk
- Flash
- Graphics
- Network
The Sequential (Von Neumann) Model

- Implicit model of all modern ISAs
  - Often called Von Neumann, but in ENIAC before
- Basic feature: the **program counter (PC)**
  - Defines **total order** of dynamic instructions
    - Next PC = PC++ unless instruction says otherwise
  - Order and **named storage** define computation
    - Value flows from instruction X to Y via storage A iff…
    - X names A as output, Y names A as input…
    - And Y after X in total order
- Processor implicitly executes loop at left
  - Instruction execution assumed atomic
  - Instruction X finishes before instruction X+1 starts
- This is one hardware thread.
- All of this implemented using transistors
  - ~10nm
Microprocessor Transistor Trends (Intel CPUs)
Single Processor Performance

Move to multi-processor

Image Copyright, Elsevier Inc.

© Alvin R. Lebeck

Compsci 590.3 Parallel Computing
More Recent Processor: Intel Core i7

- 12 “processors” (6 cores, multiple threads per core)
- Includes special instructions for parallel operations
- Other versions of this have 16 “processors” (8 cores, two threads/core)
- Xeon Phi: 60 cores, 4 threads/core
- NVIDIA GPU: Thousands of threads…
Questions

1. What is parallel programming? High Level
2. Why do we need parallel programming?
3. What are some applications?
4. What are the forms of parallelism?
5. What things do you need to worry about when writing parallel code?
3. What Applications

- Machine Learning
- Drug development
- Health and disease modeling
- Weather / Climate prediction
- Evolution of galaxies
- Oil reservoir simulation
- Automobile crash tests
- VLSI CAD

- Typically model physical systems or phenomena
- Problems are 2D or 3D
- Usually require “number crunching”
- Involve “true” parallelism
4. Forms of Parallelism

- Parallel Hardware, programming model, programming language
- Parallel Hardware != Programming model != Programming language
- Hardware parallelism across scales
  - Bit
  - Operand (e.g., int, double)
  - Instruction (add, mul)
  - Threads
  - Cores
  - Machines
  - Data Centers
5. What are some issues?

- Specify form of parallelism to exploit?
- Specify the amount of parallelism?
- Name a datum across processors?
- Communicate values?
- Coordinate and synchronize (wait for a value to be produced)?
  
  - The programming model helps with this…
  - Von Neumann -> sequential programming model
Programming Model

• How does the programmer view the system?
  ▪ This is NOT necessarily the same as how the system actually behaves!!

• Shared memory: processors execute instructions and communicate by reading/writing a globally shared memory

• Message passing: processors execute instructions and communicate by explicitly sending messages

• Data parallel: processors do the same instructions at the same time, but on different data (Single Instruction Multiple Data)
Historical View

• Architecture $\rightarrow$ Programming Model
  ▪ Join at network $\rightarrow$ program with message passing model
  ▪ Join at memory $\rightarrow$ program with shared memory model
  ▪ Join at processor $\rightarrow$ program with SIMD or data parallel

• Programming Model $\rightarrow$ Architecture
  ▪ Message-passing programs on message-passing arch
  ▪ Shared-memory programs on shared-memory arch
  ▪ SIMD/data-parallel programs on SIMD/data-parallel arch
  ▪ Slippery slope that led to LISP machines … returning a bit now….

• But
  ▪ Isn’t hardware basically the same? Processors, memory, & I/O?
  ▪ Realization that most programming models could be supported on any hardware, and now hardware has mixed types of parallelism…

• But, But: One model may be more efficient than another…
1990’s Parallel Computer Architecture

- Extension of traditional computer architecture to support communication and cooperation
  - Communications architecture

![Diagram showing layers of parallel computing architecture]

- Multiprogramming
- Shared Memory
- Message Passing
- Data Parallel

User Level

System Level

Libraries and Compilers

Operating System Support

Communication Hardware

Physical Communication Medium

Programming Model

Communication Abstraction

Hardware/Software Boundary
Today’s Parallel Computer Architecture

- Multicore: Processors on one chip + SIMD
- Graphics Cards/ GPUs (AMD & NVIDIA)
  - Massive Data Parallel
  - Highly threaded
- Big Clusters (of multicore processors + GPUs)
- Special purpose (D.E. Shaw, IBM Blue Gene, Micron Automata, etc.)
- We are going to start with Shared Memory
- Then GPUs
- Then Message Passing
Simple Problem

for i = 1 to N
    A[i] = (A[i] + B[i]) * C[i]
    sum = sum + A[i]

• How do I make this parallel?
Simple Problem

for i = 1 to N
    A[i] = (A[i] + B[i]) * C[i]
    sum = sum + A[i]

• Split the loops
  » Independent iterations

for i = 1 to N
    A[i] = (A[i] + B[i]) * C[i]
for i = 1 to N
    sum = sum + A[i]

• Data flow graph?
Data Flow Graph

2 + N-1 cycles to execute on N processors
But with what assumptions?
Partitioning of Data Flow Graph


global synch
Shared-Memory Programming Model

• Provide a shared-memory abstraction
  ▪ Familiar and efficient for programmers

Memory System
**Shared Memory Architectures**

- Communication, sharing, and synchronization with loads/stores (reads/writes) on shared variables

![Shared Memory Architectures Diagram](image)

© Alvin R. Lebeck

Compsci 590.3 Parallel Computing 34
Return of the Simple Problem (Shared Memory)

```java
private int i, my_start, my_end, mynode;
shared float A[N], B[N], C[N], sum;
for i = my_start to my_end
    A[i] = (A[i] + B[i]) * C[i]
GLOBAL_SYNCH;
if (mynode == 0)
    for i = 1 to N
        sum = sum + A[i]
```

- Can run this pseudo code on any machine that supports shared memory
- Next Time programming environments to do this!
  - language and systems
Summary

• Parallel Programming has gotten easier, but it is still not automatic
• Remote Access to machines
• Unix
• C/C++
• Homework 0 (Getting Started)
  ▪ Git (gitlab.oit.duke.edu)
  ▪ SSH Keys
  ▪ Part 1. Compile and run hello world on course machine