COL380

Introduction to
Parallel & Distributed Programming

2-0-2

Subodh Kumar

Agenda

- Course structure and policies
- Review of basic concepts
  - ➡ OS, Compilers, Architecture

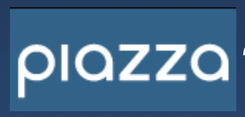
First thing First

- COL331 is a ~~pre~~ co-requisite
- Those without COL331 will be de-registered at the end of the add-drop period

Resources

- http://www.cse.iitd.ac.in/~subodh/courses/COL380
  - ➡ Persistent Info: Policies, Resources, Links, Slides, Textbook

- Assignments, Test, Discussion



- ? Course discussion

- Urgent announcements subodh@cse.iitd.ac.in
  Subject: [COL380] …

- Grading

Lectures

- Lectures
  - ➡ Tuesday and Friday
- Mondays for doubts, Q&A
  - ➡ And announced quizzes

Grading

- Assignments: 29.
  - A0: 4, A1-A3: 5 each, A4: 10 Two deadlines, resp. correct and performant
  - Marks for performance (only if correct); >100% possible
  - Checked for similarity; some will give **viva**
- Programming Test: 6.
- Quiz(zes): 10.
  - Announced / Unannounced in class
- Minor: 25. Major 30.
  - Objective + subjective

* Do not share code
* Do not discuss, except with me or TA

Similarity 1 ⇒

0 mark + 1 letter-grade penalty

Similarity 2 ⇒

F grade + Disciplinary com.

Other Policies

- Late Assignment Submission:
  - ➡ 0.25 marks/6H of delay or part thereof; up to 3 days of delay
- 100% attendance is required
  - ➡ <75% in any month ⇒ take a **viva** on the material covered that month
    - ▶ Range of marks in the viva: -2.5 to 0
  - ➡ 75% pre-minor required for reminor, 75% in semester for remajor
- Audit Policy:
  - ➡ No audits

Other Policies

- Late Assignment Submission:
  - ➡ 0.25 marks/6H of delay or part thereof; up to 3 days of delay
- 100% attendance is required
  - ➡ <75% in any month ⇒ take a viva on the material covered that month
    - Range of marks in the viva: -2.5 to 0
  - ➡ 75% pre-minor required for minor, 75% in semester for pass
- Audit Policy:
  - ➡ No audits

VIVA

Usually on Monday 12-1 (Keep slot free.)

Viva call by email before that Monday at 8am

If viva not on Monday, email will be ~24 hours in advance

If you fail to show up, you will get the minimum marks

If you are sick, send a request to reschedule BEFORE the viva

(In case of such request, it is YOUR responsibility to find an alternate time within 1 week of the original time.)

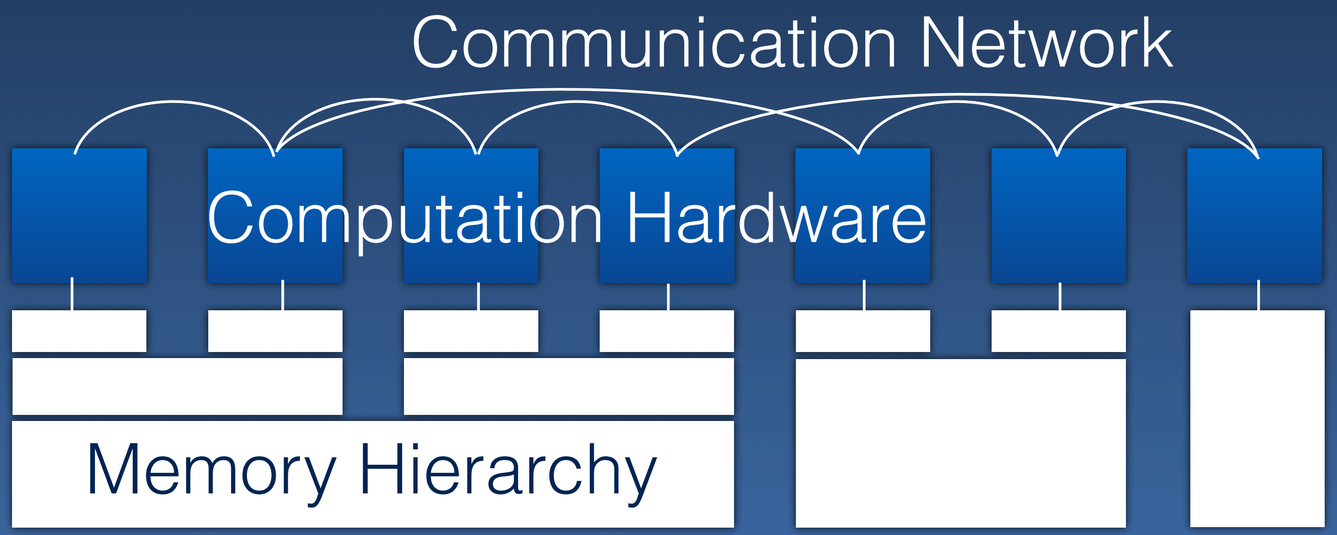
About

User Program
Programming Model
Compilers and Runtime
Communication Abstraction
Hardware Abstraction

- Decide what to do
- Where
- When
- How to ask the system to do it
- Estimate and Measure Efficiency

- Algorithms
- Programs
- System S/W
- Hardware

Synchronization
Communication
Load distribution
Latency
Bandwidth
Scaling



Course Content

Introduction to concurrency, Race conditions, Atomicity, Semantics of concurrent programs, Examples of distributed algorithms, Client-Server paradigm
Parallel architecture, Flynn's classification
Shared-memory programming with reference to memory consistency, cache in-coherence, false sharing and mutual exclusion
Message passing, High level and collective constructs, Point-to-point communication, multicast and broadcast, Blocking versus non-blocking styles for communication, Message buffering
Theoretical models of parallel computation and algorithm analysis, Examples of reduction, prefix-sum
Performance metrics: Time, work, Scalability.
Task/Communication Dependence graphs, Task decomposition, Data-parallel decomposition, Pipelining
Synchronization, barriers, Progress, Livelock/Deadlock

Learning Goals

- Write scalable and efficient parallel programs
  - ➡ OpenMP, MPI, Cuda, Development tools
  - ➡ Understand the issues in tool design and implementation
  - ➡ Profile and Debug
- Understand, measure, predict and analyze parallel performance
- Examples of parallel and distributed algorithms and data structures
- Understand parallelism in I/O and memory
- Understand nomenclature, literature, documentation

Learning Goals

- Write scalable and efficient parallel programs
  - ➡ OpenMP, MPI, Cuda, Development tools
  - ➡ Understand the issues in tool design and impl
  - ➡ Profile and Debug
- Understand, measure, predict and analyze parallel performance
- Examples of parallel and distributed algorithms and data structures
- Understand parallelism in I/O and memory
- Understand nomenclature, literature, documentation

Need
- strong C++ skills
- OS and Architecture concepts
- Background in Algorithms

Keys to success

- Be regular (Slides only have topics)

- Read

- Program all assignments yourself

  - ➡ Be curious. Try out ‘what-if’ variations and see what happens

- Talk to the instructor

First Parallel Program

Instruction level Parallelism

```
float dot(float *a, float *b, int n)
{
    int i;
    float s0=0, s1=0, s2=0, s3=0;
    for(i=0; i<n/4*4; i+=4)
        s0 += a[i]*b[i];
        s1 += a[i+1]*b[i+1];
        s2 += a[i+2]*b[i+2];
        s3 += a[i+3]*b[i+3];
    for(; i<n; i++)
        s0 += a[i]*b[i];
    return s0+s1+s2+s3;
}
```

unrolled loop

Independent statements

Architecture can issue multiple instructions

Architecture can prefetch data

Compilers also re-order instructions

- Process & thread
  - ➡ Scheduling, Context-switching and concurrency
  - ➡ User space, Kernel space
  - ➡ System calls
- Address space & Name space
- Virtual Memory
  - ➡ Caches
- Synchronization

- ✦ Shell (bash)
- ✦ time
- ✦ PMU (perf)
- ✦ Context switch
  - ✦ and scheduling
- ✦ System calls
- ✦ Interrupts