18-742 Spring 2011
Parallel Computer Architecture
Lecture 1: Introduction

Prof. Onur Mutlu
Carnegie Mellon University
Agenda

- Course logistics, info, requirements
  - Who are we?
  - Who should take this course?
  - What will you learn?
  - What will I expect?
  - Policies

- Homework and reading for next time

- Some fundamental concepts
Quiz 0 (Student Info Sheet)

- Due at the end of class

- Our way of getting to know about you fast

- All grading predicated on passing Quiz 0
  - But, you are not in this room for grades anyway
Course Info: Who Are We?

- Instructor: Prof. Onur Mutlu
  - onur@cmu.edu
  - Office: HH-A305, Hours: W 2:30-3:30pm (or by appointment)
  - http://www.ece.cmu.edu/~omutlu
  - Research: Computer architecture, many-core systems, interconnection networks, hardware/software interaction and co-design, fault tolerance, hardware security

- Teaching Assistant
  - Chris Craik (ccraik@ece.cmu.edu), Office: HH-A5
  - Office hours TBD

- Course Administrative Assistant
  - Lydia Corrado, Office: HH D-level Course Hub
What This Course is About

- **Goal 1:**
  - Build a **strong understanding of the fundamentals of the architecture of parallel computers** and the tradeoffs made in their design.
  - Examine how architectures are designed to exploit and extract different types of parallelism. The focus will be on fundamentals, tradeoffs in parallel architecture design, and cutting-edge research.

- **Goal 2:**
  - **Do research in parallel computer architecture.** You will conduct 1) a literature survey of very recent papers on a topic and 2) an open-ended research project to advance the state of the art.
  - Get familiar with research papers. Deliver technical talks of both your survey and project findings to the entire class.
Who Should Take This Course?

- This course is entirely optional: advanced graduate course
- You should be self-motivated and enthusiastic to do research in computer architecture
- Must have done well in Graduate Comp Arch (740 or 18-741)
  - B or above
  - If not, you have to convince me you know the required basics
- Must be enthusiastic enough to
  - Work hard
  - Read a lot of research papers
  - Pace yourself without deadlines; be self motivated
  - Discover on your own (research project)
  - Make a difference (advance the state of the art)
Where to Get Up-to-date Course Info?

- **Website:** http://www.ece.cmu.edu/~ece742
  - Lecture notes
  - Readings
  - Project info
  - Discussion boards – share information

- **Blackboard:** Linked from website
  - For you to upload your work

- **Your email**

- **Me and the TAs**
Lectures and Course Schedule

- **Reserved Lecture Times:**
  - MWF 4:30-6:20pm
  - Scaife Hall 208
  - I intend to lecture 2/3 days on average. Days and load will be determined dynamically.

- **Tentative schedule in your syllabus**
  - But don’t believe all of it
  - Systems that perform best are usually dynamically scheduled.
    - Static vs. Dynamic Scheduling
    - Why do you *really* need dynamic scheduling?
Static versus Dynamic Scheduling

- **Static**: Done at compile time or parallel task creation time
  - Schedule does not change based on runtime information

- **Dynamic**: Done at run time (e.g., after tasks are created)
  - Schedule changes based on runtime information

- Example: Instruction scheduling
  - Why would you like to do dynamic scheduling?
  - What pieces of information are not available to the static scheduler?
Parallel Task Assignment: Tradeoffs

- Problem: $N$ tasks, $P$ processors, $N>P$. Do we assign tasks to processors statically (fixed) or dynamically (adaptive)?

- Static assignment
  + Simpler: No movement of tasks.
  - Inefficient: Underutilizes resources when load is not balanced

- Dynamic assignment
  + Efficient: Better utilizes processors when load is not balanced
  - More complex: Need to move tasks to balance processor load
  - Higher overhead: Task movement takes time, can disrupt locality
Parallel Task Assignment: Example

- Compute histogram of a large set of values
- Parallelization:
  - Divide the values across $T$ tasks
  - Each task computes a local histogram for its value set
  - Local histograms merged with global histograms in the end

```c
GetPageHistogram(Page *P)
For each thread: {
    /* Parallel part of the function */
    UpdateLocalHistogram(Fraction of Page)

    /* Serial part of the function */
    Critical Section:
    Add local histogram to global histogram

    Barrier
}
Return global histogram
```
Parallel Task Assignment: Example (II)

- How to schedule tasks updating local histograms?
  - Static: Assign equal number of tasks to each processor
  - Dynamic: Assign tasks to a processor that is available
  - When does static work as well as dynamic?

- Implementation of Dynamic Assignment with Task Queues

(a) Distributed Task Stealing
(b) Hierarchical Task Queuing
Software Task Queues

- What are the advantages and disadvantages of each?
  - Centralized
  - Distributed
  - Hierarchical
Task Stealing

- **Idea:** When a processor’s task queue is empty it steals a task from another processor’s task queue
  - Whom to steal from? (Randomized stealing works well)
  - How many tasks to steal?

+ Dynamic balancing of computation load

- Additional communication/synchronization overhead between processors
- Need to stop stealing if no tasks to steal
Parallel Task Assignment: Tradeoffs

- Who does the assignment? Hardware versus software?

- Software
  + Better scope
  - More time overhead
  - Slow to adapt to dynamic events (e.g., a processor becoming idle)

- Hardware
  + Low time overhead
  + Can adjust to dynamic events faster
  - Requires hardware changes (area and possibly energy overhead)
How Can the Hardware Help?

- Managing task queues in software has overhead
  - Especially high when task sizes are small

- An idea: Hardware Task Queues
  - Each processor has a dedicated task queue
  - Software fills the task queues (on demand)
  - Hardware manages movement of tasks from queue to queue
  - There can be a global task queue as well → hierarchical tasking in hardware

  - Optional reading
Dynamic Task Generation

- Does static task assignment work in this case?

- Problem: Searching the exit of a maze

```java
while (problem not solved)
    SubProblem = PriorityQ.remove()
    Solve(SubProblem)
    if(solved)
        break
    NewSubProblems = Partition(SubProblem)
    PriorityQ.insert(NewSubProblems)
```
What Will You Learn?

- Parallel computer designs
  - State-of-the-art as well as research proposals
  - Tradeoffs and how to make them
  - Emphasis on cutting-edge research

- Hands-on research in a parallel computer architecture topic
  - Semester-long project
  - How to design better architectures (not an intro course)

- How to dig out information
  - No textbook really required
  - But, see the syllabus anyway
What Do I Expect From You?

- Learn the material
- And, research it → find the original source of ideas
- Do the work & work hard
- **Ask questions, take notes, participate in discussion**
- Come to class on time

- Start early and focus on the research project
- If you want feedback, come to office hours

- This class will definitely be tough
How Will You Be Evaluated?

- Research Project + Presentation + Poster: 40%
- Literature Survey + Presentation: 20%
- Exam: 20%
- Reviews, Class Participation, Quizzes, HWs: 15%
- Our evaluation of your performance: 5%

- Grading will be back-end heavy. Most of your grade will be determined after late March (Exam is on March 21)
  - How you prepare and manage your time will determine it
Policies

- No late assignments accepted
- Everything must be your own work (unless otherwise specified)

- Projects and Literature Survey in groups of 2 or 3

- Cheating → Failing grade
  - No exceptions
Readings for This Week

- **Required – Enter reviews in the online system**
  - Levin and Redell, “How (and how not) to write a good systems paper,” OSR 1983.

- **Recommended**
  - Culler & Singh, Chapter 1