## Your teaching staff!

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vivek Sarkar</strong> (Instructor)</td>
<td><strong>Shams Imam</strong> (Co-instructor)</td>
<td><strong>Max Grossman</strong> (Head TA)</td>
<td><strong>Pranath Chatarasi</strong> (Grad TA)</td>
<td><strong>Arghya Chatterjee</strong> (Grad TA)</td>
<td><strong>Yuhan Peng</strong> (Grad TA)</td>
<td><strong>Jonathan Sharman</strong> (Grad TA)</td>
</tr>
<tr>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
</tr>
<tr>
<td><strong>Peter Elmers</strong> (UG TA)</td>
<td><strong>Nicholas Hanson-Holtry</strong> (UG TA)</td>
<td><strong>Ayush Narayan</strong> (UG TA)</td>
<td><strong>Alitha Partono</strong> (UG TA)</td>
<td><strong>Tom Roush</strong> (UG TA)</td>
<td><strong>Hunter Tidwell</strong> (UG TA)</td>
<td><strong>Bing Xue</strong> (UG TA)</td>
</tr>
<tr>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
<td><a href="#">Image</a></td>
</tr>
</tbody>
</table>
What is Parallel Computing?

- **Parallel computing**: using multiple processors in parallel to solve problems more quickly than with a single processor and/or with less energy

- **Example of a parallel computer**
  - An 8-core Symmetric Multi-Processor (SMP) consisting of four dual-core chip microprocessors (CMPs)

Source: Figure 1.5 of Lin & Snyder book, Addison-Wesley, 2009
All Computers are Parallel Computers --- Why?
Moore’s Law and Dennard Scaling

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 1-2 years (Moore’s Law)
⇒ area of transistor halves every 1-2 years
⇒ feature size reduces by √2 every 1-2 years

Dennard Scaling states that power for a fixed chip area remains constant as transistors grow smaller.
Recent Technology Trends

- **Chip density (transistors)** is increasing ~2x every 2 years
- ⇒ number of processors doubles every 2 years as well
- **Clock speed** is plateauing below 10 GHz so that **chip power** stays below 100W
- **Instruction-level parallelism (ILP)** in hardware has also plateaued below 10 instructions/cycle
- ⇒ **Parallelism must be managed by software!**

Source: Intel, Microsoft (Sutter) and Stanford (Olukotun, Hammond)
Parallelism Saves Power
(Simplified Analysis)

Nowadays (post Dennard Scaling), Power \sim (\text{Capacitance}) \times (\text{Voltage})^2 \times (\text{Frequency})
and maximum Frequency is capped by Voltage

\[ \text{\implies Power is proportional to } (\text{Frequency})^3 \]

Baseline example: single 1GHz core with power P

Option A: Increase clock frequency to 2GHz \text{ \implies Power} = 8P

Option B: Use 2 cores at 1 GHz each \text{ \implies Power} = 2P

• Option B delivers same performance as Option A with 4x less power ... provided software can be decomposed to run in parallel!
## A Real World Example

- Fermi vs. Kepler GPU chips from NVIDIA’s GeForce 600 Series
  
  —Source: [http://www.theregister.co.uk/2012/05/15/nvidia_kepler_tesla_gpu_revealed/](http://www.theregister.co.uk/2012/05/15/nvidia_kepler_tesla_gpu_revealed/)

<table>
<thead>
<tr>
<th></th>
<th>Fermi chip (released in 2010)</th>
<th>Kepler chip (released in 2012)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of cores</strong></td>
<td>512</td>
<td>1,536</td>
</tr>
<tr>
<td><strong>Clock frequency</strong></td>
<td>1.3 GHz</td>
<td>1.0 GHz</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>250 Watts</td>
<td>195 Watts</td>
</tr>
<tr>
<td><strong>Peak double precision floating point performance</strong></td>
<td>665 Gigaflops</td>
<td>1310 Gigaflops (1.31 Teraflops)</td>
</tr>
</tbody>
</table>
What is Parallel Programming?

- Specification of operations that can be executed in parallel
- A parallel program is decomposed into sequential subcomputations called *tasks*
- Parallel programming constructs define task creation, termination, and interaction
Example of a Sequential Program: Computing the sum of array elements

Algorithm 1: Sequential ArraySum

Input: Array of numbers, X.
Output: \( \text{sum} = \text{sum of elements in array } X \).

\[
\text{sum} \leftarrow 0; \\
\text{for } i \leftarrow 0 \text{ to } X\.\text{length} - 1 \text{ do} \\
\quad \text{sum} \leftarrow \text{sum} + X[i]; \\
\text{return } \text{sum};
\]

Observations:

- The decision to sum up the elements from left to right was arbitrary
- The computation graph shows that all operations must be executed sequentially
Parallelization Strategy for two cores (Two-way Parallel Array Sum)

Task 0: Compute sum of lower half of array

Task 1: Compute sum of upper half of array

Compute total sum

Basic idea:

• Decompose problem into two tasks for partial sums
• Combine results to obtain final answer
• Parallel divide-and-conquer pattern
Async and Finish Statements for Task Creation and Termination (Pseudocode)

async S
- Creates a new child task that executes statement S

finish S
- Execute S, but wait until all asyncs in S’s scope have terminated.

// T₀ (Parent task)
STMT₀;
finish {  //Begin finish
    async {
        STMT₁;  //T₁ (Child task)
    }
    STMT₂;  //Continue in T₀
        //Wait for T₁
}  //End finish
STMT₃;  //Continue in T₀
Two-way Parallel Array Sum using async & finish constructs

Algorithm 2: Two-way Parallel ArraySum

Input: Array of numbers, X.
Output: sum = sum of elements in array X.

// Start of Task T1 (main program)
sum1 ← 0; sum2 ← 0;
// Compute sum1 (lower half) and sum2 (upper half) in parallel.
finish{
    async{
        // Task T2
        for i ← 0 to X.length/2 − 1 do
            sum1 ← sum1 + X[i];
    };
    async{
        // Task T3
        for i ← X.length/2 to X.length − 1 do
            sum2 ← sum2 + X[i];
    };
}
// Task T1 waits for Tasks T2 and T3 to complete
// Continuation of Task T1
sum ← sum1 + sum2;
return sum;
Course Syllabus

- Fundamentals of Parallel Programming taught in three modules
  1. Parallelism
  2. Concurrency
  3. Locality & Distribution
- Each module is subdivided into units, and each unit into topics
- Lecture and lecture handouts will introduce concepts using pseudocode notations
- Labs and programming assignments will be in Java 8
  - Initially, we will use the Habanero-Java (HJ) library developed at Rice as a pedagogic parallel programming model
    - HJ-lib is a Java 8 library (no special compiler support needed)
    - HJ-lib contains many features that are easier to use than standard Java threads/tasks, and are also being added to future parallel programming models
  - Later, we will learn parallel programming using standard Java libraries, and combinations of Java libs + HJ-lib
Grade Policies

Course Rubric

- Homeworks (5) 40% (written + programming components)
  - Weightage proportional to # weeks for homework
- Exams (2) 40% (scheduled midterm + scheduled final)
- Quizzes & Labs 10% (quizzes on edX, labs graded as in COMP 215)
- Class Participation 10% (classroom Q&A, Piazza discussions, in-class worksheets)

Grading curve (we reserve the right to give higher grades than indicated below!)

>= 90% => A or A+

>= 80% => B, B+, or A-

>= 70% => C+ or B-

others => C or below
Next Steps

• IMPORTANT:
  — Send email to comp322-staff@rice.edu if you did NOT receive a welcome email from us
  — Bring your laptop to this week’s lab at 7pm on Wednesday (Section A01: DH 1064, Section A02: DH 1070)
  — Watch videos for topics 1.2 & 1.3 for next lecture on Wednesday

• Complete each week’s assigned quizzes on edX by 11:59pm that Friday. This week, you should submit quizzes for lecture & demonstration videos for topics 1.1, 1.2, 1.3, 1.4

• HW1 will be assigned on Jan 15th and be due on Jan 28th

• See course web site for syllabus, work assignments, due dates, ...
  • http://comp322.rice.edu
OFFICE HOURS

• Regular office hour schedule will be posted for Jan 19th onwards

• This week’s office hours are as follows
  — TODAY (Jan 11), 2pm - 3pm, Duncan Hall 3092
  — FRIDAY (Jan 15), 2pm - 3pm, Duncan Hall 3092

• Send email to instructors (vsarkar@rice.edu, shams@rice.edu) if you need to meet some other time this week

• And remember to post questions on Piazza!