“To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”

— Edsger Dijkstra, The Humble Programmer, CACM
Course Information

- **Lecture Time:** TuTh 5:20 pm - 6:40 pm
- **Location:** Earth & Space 069, West Campus
- **Instructor:** Rezaul A. Chowdhury
- **Office Hours:** TuTh 12:00 pm - 1:30 pm, 1421 Computer Science
- **Email:** rezaul@cs.stonybrook.edu
- **TA:** No idea!
- **TA Office Hours:** Same as above
- **TA Email:** Same as above
- **Class Webpage:**
**Prerequisites**

- **Required:** Background in algorithms analysis
  ( e.g., CSE 373 or CSE 548 )

- **Required:** Background in programming languages ( C / C++ )

- **Helpful but Not Required:** Background in computer architecture

- **Please Note:** This is not a course on
  - Programming languages
  - Computer architecture

- **Main Emphasis:** Parallel algorithms ( for supercomputing )
Course Organization

- **First Part:** 11 Lectures
  - Introduction (2)
  - Shared-memory parallelism & Cilk (2)
  - Distributed-memory parallelism & MPI (2)
  - GPGPU computation & CUDA (2)
  - MapReduce & Hadoop (2)
  - Cloud computing (1)

- **Second Part:**
  - Paper presentations
  - Group projects
Grading Policy

- Programming assignments (best 3 of 4): 15%
- Paper presentation (one): 25%
- Report on a paper presented by another student (one): 10%
- Group project (one): 40%
  - Proposal (in-class): Feb 28
  - Progress report (in-class): April 10
  - Final presentation (in-class): May 8 - 15
- Class participation & attendance: 10%
Programming Environment

This course is supported by educational grants from

- Extreme Science and Engineering Discovery Environment (XSEDE): https://www.xsede.org
- Amazon Web Services (AWS): http://aws.amazon.com

We will use XSEDE for homeworks/projects involving
  - Shared-memory parallelism
  - Distributed-memory parallelism

And AWS for those involving
  - GPGPUs
  - MapReduce
Programming Environment

On XSEDE we have access to

- Ranger: ≈ 4,000 compute nodes with 16 cores/node
- Lonestar 4: ≈ 2,000 compute nodes with 12 cores/node

World's Most Powerful Supercomputers in June, 2008
(www.top500.org)
Recommended Texts

No required textbook.
Some useful ones are as follows

Supercomputing & Parallel Computing
## Top 10 Supercomputing Sites in Nov. 2011

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Computer/Years Vendor</th>
<th>Cores</th>
<th>R&lt;sub&gt;max&lt;/sub&gt;</th>
<th>R&lt;sub&gt;peak&lt;/sub&gt;</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RIKEN Advanced Institute for Computational Science (AICS) Japan</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu</td>
<td>705024</td>
<td>10510.00</td>
<td>11280.38</td>
<td>12659.9</td>
</tr>
<tr>
<td>2</td>
<td>National Supercomputing Center in Tianjin China</td>
<td>NUDT X44 HPR, Xeon X5670 6C 2.93 GHz NVIDIA 2050 / 2010 NUDT</td>
<td>186368</td>
<td>2566.00</td>
<td>4701.00</td>
<td>4040.0</td>
</tr>
<tr>
<td>3</td>
<td>DOE/SC/Oak Ridge National Laboratory United States</td>
<td>Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.</td>
<td>224162</td>
<td>1759.00</td>
<td>2331.00</td>
<td>6950.0</td>
</tr>
<tr>
<td>4</td>
<td>National Supercomputing Centre in Shenzhen (NSCS) China</td>
<td>Dawning TC3800 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 / 2010 Dawning</td>
<td>120640</td>
<td>1271.00</td>
<td>2984.30</td>
<td>2580.0</td>
</tr>
<tr>
<td>5</td>
<td>CSIC Center, Tokyo Institute of Technology Japan</td>
<td>HP ProLiant SL 390s G7 Xeon 6C X5670 Nvidia GPU, Linux/Windows / 2010 NEC/HP</td>
<td>73278</td>
<td>1192.00</td>
<td>2287.63</td>
<td>1398.6</td>
</tr>
<tr>
<td>6</td>
<td>DOE/NNSA/LANL/SNL United States</td>
<td>Cray XE6, Opteron 6136 8C 2.40GHz, Custom / 2011 Cray Inc.</td>
<td>142272</td>
<td>1110.00</td>
<td>1365.81</td>
<td>3980.0</td>
</tr>
<tr>
<td>7</td>
<td>NASA/Ames Research Center/NAS United States</td>
<td>SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 GHz, Infiniband / 2011 SGI</td>
<td>111104</td>
<td>1088.00</td>
<td>1315.33</td>
<td>4102.0</td>
</tr>
<tr>
<td>8</td>
<td>DOE/SC/BNL/NERSC United States</td>
<td>Cray XE6, Opteron 6172 12C 2.10GHz, Custom / 2010 Cray Inc.</td>
<td>153408</td>
<td>1054.00</td>
<td>1288.63</td>
<td>2910.0</td>
</tr>
<tr>
<td>9</td>
<td>Commissariat a l’Energie Atomique (CEA) France</td>
<td>Bull bulx super-node S6010/S6030 / 2010 Bull</td>
<td>138368</td>
<td>1050.00</td>
<td>1254.55</td>
<td>4590.0</td>
</tr>
<tr>
<td>10</td>
<td>DOE/NNSA/LANL United States</td>
<td>BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 GHz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM</td>
<td>122400</td>
<td>1042.00</td>
<td>1375.78</td>
<td>2345.0</td>
</tr>
</tbody>
</table>

Source: www.top500.org
Top 500 Supercomputing Sites
(Cores / System)

Source: www.top500.org
Why Parallelism?
Moore’s Law

curve shows transistor count doubling every two years

Unicore Performance

Source: Chung-Ta King, Department of Computer Science, National Tsing Hua University
Unicore Performance Has Hit a Wall!

Some Reasons

- Lack of additional ILP
  (Instruction Level Hidden Parallelism)
- High power density
- Manufacturing issues
- Physical limits
- Memory speed
Unicore Performance: No Additional ILP

Exhausted all ideas to exploit hidden parallelism?

- Multiple simultaneous instructions
- Dynamic instruction scheduling
- Branch prediction
- Out-of-order instructions
- Speculative execution
- Pipelining
- Non-blocking caches, etc.
**Unicore Performance: High Power Density**

- Dynamic power, \( P_d \propto V^2 f C \)
  
  - \( V \) = supply voltage
  - \( f \) = clock frequency
  - \( C \) = capacitance

- But \( V \propto f \)
- Thus \( P_d \propto f^3 \)

*Source: Patrick Gelsinger, Intel Developer Forum, Spring 2004 (Simon Floyd)*
Unicore Performance: High Power Density

- Changing $f$ by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?

Source: Andrew A. Chien, Vice President of Research, Intel Corporation
Unicore Performance: High Power Density

- Changing $f$ by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?

Source: Andrew A. Chien, Vice President of Research, Intel Corporation
Unicore Performance: High Power Density

- Changing $f$ by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?

Source: Andrew A. Chien, Vice President of Research, Intel Corporation
Unicore Performance: Manufacturing Issues

– Frequency, \( f \propto 1 / s \)
  – \( s = \) feature size (transistor dimension)
– Transistors / unit area \( \propto 1 / s^2 \)
– Typically, die size \( \propto 1 / s \)
– So, what happens if feature size goes down by a factor of \( x \)?
  – Raw computing power goes up by a factor of \( x^4 \)!
  – Typically most programs run faster by a factor of \( x^3 \) without any change!

Source: Kathy Yelick and Jim Demmel, UC Berkeley
Unicore Performance: Manufacturing Issues

As feature size decreases

- Manufacturing cost goes up
  - Cost of a semiconductor fabrication plant doubles every 4 years (Rock’s Law)
- Yield ( % of usable chips produced ) drops

Source: Kathy Yelick and Jim Demmel, UC Berkeley
Unicore Performance: Physical Limits

Execute the following loop on a serial machine in 1 second:

```
for ( i = 0; i < 10^{12}; ++i )
    z[ i ] = x[ i ] + y[ i ];
```

- We will have to access $3 \times 10^{12}$ data items in one second
- Speed of light is, $c \approx 3 \times 10^8$ m/s
- So each data item must be within $c / 3 \times 10^{12} \approx 0.1$ mm from the CPU on the average
- All data must be put inside a $0.2$ mm $\times$ $0.2$ mm square
- Each data item ($\geq 8$ bytes) can occupy only $1$ Å$^2$ space! (size of a small atom!)

Source: Kathy Yelick and Jim Demmel, UC Berkeley
Unicore Performance: Memory Wall

Source: Rick Hetherington, Chief Technology Officer, Microelectronics, Sun Microsystems
Moore’s Law Reinterpreted

Source: Report of the 2011 Workshop on Exascale Programming Challenges
Cores / Processor (General Purpose)

Source: Andrew A. Chien, Vice President of Research, Intel Corporation
No Free Lunch for Traditional Software

Source: Simon Floyd, Workstation Performance: Tomorrow's Possibilities (Viewpoint Column)
Insatiable Demand for Performance

Numerical Weather Prediction

Problem: \((\text{temperature, pressure, ..., humidity, wind velocity})\)
\[
\leftarrow f(\text{longitude, latitude, height, time})
\]

Approach (very coarse resolution):
- Consider only modeling fluid flow in the atmosphere
- Divide the entire global atmosphere into cubic cells of size 1 mile \(\times\) 1 mile \(\times\) 1 mile each to a height of 10 miles
  \(\approx 2 \times 10^9\) cells
- Simulate 7 days in 1 minute intervals
  \(\approx 10^4\) time-steps to simulate
- 200 floating point operations (flop) / cell / time-step
  \(\approx 4 \times 10^{15}\) floating point operations in total
- To predict in 1 hour \(\approx 1\) Tflop/s (Tera flop / sec)
Insatiable Demand for Performance

Some Useful Classifications of Parallel Computers
Parallel Computer Memory Architecture (Shared Memory)

- All processors access all memory as global address space
- Changes in memory by one processor are visible to all others
- Tow types:
  - Uniform Memory Access (UMA)
  - Non-Uniform Memory Access (NUMA)

Source: Blaise Barney, LLNL
Parallel Computer Memory Architecture (Shared Memory)

Advantages

- User-friendly programming perspective to memory
- Fast data sharing

Disadvantages

- Difficult and expensive to scale
- Correct data access is user responsibility

Source: Blaise Barney, LLNL
Parallel Computer Memory Architecture
(Distributed Memory)

- Each processor has its own local memory — no global address space
- Changes in local memory by one processor have no effect on memory of other processors
- Communication network to connect inter-processor memory

Source: Blaise Barney, LLNL
Parallel Computer Memory Architecture
( Distributed Memory )

Advantages

- Easily scalable
- No cache-coherency needed among processors
- Cost-effective

Disadvantages

- Communication is user responsibility
- Non-uniform memory access
- May be difficult to map shared-memory data structures to this type of memory organization

Source: Blaise Barney, LLNL
Parallel Computer Memory Architecture
( Hybrid Distributed-Shared Memory )

- The share-memory component can be a cache-coherent SMP or a Graphics Processing Unit (GPU)
- The distributed-memory component is the networking of multiple SMP/GPU machines
- Most common architecture for the largest and fastest computers in the world today

Source: Blaise Barney, LLNL
**Flynn’s Taxonomy of Parallel Computers**

**Flynn’s classical taxonomy (1966):**
Classification of multi-processor computer architectures along two independent dimensions of *instruction* and *data*.

<table>
<thead>
<tr>
<th></th>
<th>Single Data (SD)</th>
<th>Multiple Data (MD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Instruction (SI)</td>
<td>SISD</td>
<td>SIMD</td>
</tr>
<tr>
<td>Multiple Instruction (MI)</td>
<td>MISD</td>
<td>MIMD</td>
</tr>
</tbody>
</table>
Flynn’s Taxonomy of Parallel Computers

SISD

- A serial (non-parallel) computer
- The oldest and the most common type of computers
- Example: Uniprocessor uncore machines

Source: Blaise Barney, LLNL
Flynn’s Taxonomy of Parallel Computers

SIMD

- A type of parallel computer
- All PU’s run the same instruction at any given clock cycle
- Each PU can act on a different data item
- Synchronous (lockstep) execution
- Two types: processor arrays and vector pipelines
- Example: GPUs (Graphics Processing Units)
Flynn’s Taxonomy of Parallel Computers

**MISD**
- A type of parallel computer
- Very few ever existed

**MIMD**
- A type of parallel computer
- Synchronous /asynchronous execution
- Examples: most modern supercomputers, parallel computing clusters, multicore PCs

Source: Blaise Barney, LLNL
Parallel Algorithms
Warm-up

“The way the processor industry is going, is to add more and more cores, but nobody knows how to program those things. I mean, two, yeah; four, not really; eight, forget it.”

— Steve Jobs, NY Times interview, June 10 2008
Parallel Algorithms Warm-up (1)

Consider the following loop:

\[
\text{for } i = 1 \text{ to } n \text{ do}
\]

\[
C[i] \leftarrow A[i] \times B[i]
\]

- Suppose you have an infinite number of processors/cores
- Ignore all overheads due to scheduling, memory accesses, communication, etc.
- Suppose each operation takes a constant amount of time
- How long will this loop take to complete execution?
  - \(O(1)\) time
Parallel Algorithms Warm-up (2)

Now consider the following loop:

\[ c \leftarrow 0 \]

\[ \text{for } i = 1 \text{ to } n \text{ do} \]
\[ c \leftarrow c + A[i] \times B[i] \]

― How long will this loop take to complete execution?

― \(O(\log n)\) time
Now consider quicksort:

\[
QSort( A )
\]

\[
if |A| \leq 1 \text{ return } A
\]

\[
else \quad p \leftarrow A[ \text{rand}( |A| ) ]
\]

\[
\text{return } QSort( \{ x \in A : x < p \} )
\]

\[
\quad \# \{ p \} \#
\]

\[
QSort( \{ x \in A : x > p \} )
\]

— Assuming that \( A \) is split in the middle everytime, and the two recursive calls can be made in parallel, how long will this algorithm take?

— \( O( \log^2 n ) \) time