Concurrent: Introduction

Arvind Krishnamurthy

Why parallel computing? (the traditional reasons)

- Traditional scientific and engineering paradigm:
  1. Do theory or paper design – too complex
  2. Build systems and perform experiments

- Limitations:
  - Too difficult -- complex equations, build large wind tunnels.
  - Too expensive -- build a throw-away passenger jet.
  - Too slow -- wait for climate or galactic evolution.
  - Too dangerous -- weapons, drug design, climate experimentation.

- Simulation: third pillar of science
  1. Use high performance computer systems to simulate the phenomenon
  2. Base on known physical laws and efficient numerical methods.

Global Climate Modeling Problem

- Problem is to compute:
  \( f(\text{latitude, longitude, elevation, time}) \rightarrow \text{temperature, pressure, humidity, wind velocity} \)

- Approach:
  1. Discretize the domain, e.g., a measurement point every 1km
  2. Devise an algorithm to predict weather at time t+1 given t

- Uses:
  - Predict major events
  - e.g., El Niño
  - Set air emissions standards

Parallel Computing in Web Search

- Functional parallelism: crawling, sorting, indexing
  1. Finding information amidst junk

- Parallelism between queries: multiple users

- Preprocessing of the web data set to help find information:
  1. Find most relevant data, classify data
  2. Page rank calculations for determining popularity

- General theme of sifting through large, unstructured data sets:
  1. when to put white socks on sale
  2. what advertisements should you receive
  3. finding medical problems in a community

Global Climate Modeling Computation

- One piece is modeling the fluid flow in the atmosphere
  1. Solve Navier-Stokes problem
  2. Roughly 100 Flops per grid point

- Computational requirements: (with 1 minute timestep)
  1. To match real-time, need 5x \(10^9\) Flops in 60 seconds = 8 GFlop/s
  2. Weather prediction (7 days in 24 hours) \(\rightarrow\) 56 GFlop/s
  3. Climate prediction (50 years in 30 days) \(\rightarrow\) 4.8 TFlop/s
  4. To use in policy negotiations (50 years in 12 hours) \(\rightarrow\) 288 TFlop/s

- To double the grid resolution, computation is at least 8x

- Current simulations are coarser than this
Document Retrieval Computation

- Approach:
  - Store the documents in a large (sparse) matrix
  - Use Latent Semantic Indexing (LSI), or related algorithms to "partition"
  - Needs large sparse matrix-vector multiply

- Ten million documents in typical matrix.
- Web storage increasing 2x every 5 months.
- Similar ideas may apply to image retrieval.

Parallel Computing: of limited interest?

- Argument #1: Maybe a few programs need large amounts of compute power, while the rest don't
  - "I think there is a world market for maybe five computers."
  - Thomas Watson, chairman of IBM, 1943.
  - "There is no reason for any individual to have a computer in their home"
  - Ken Olson, president and founder of digital equipment corporation, 1977.
  - "40K of memory ought to be enough for anybody."
  - Bill Gates, chairman of Microsoft, 1981.

- Argument #2: Just wait for next year's processor which will be about 2x faster
  - There are limitations as to how fast uniprocessors can perform
  - Hardware designers are putting multiple processors on the same chip

- Argument #3: Programming parallel machines is rather complex. Too hard to write correct and efficient parallel programs
  - Well, we all love a challenge, don't we?

Technology Trends: Microprocessor Capacity

- 2X transistors/Chip Every 1.5 years Called "Moore's Law"
- Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.
- Microprocessors have become smaller, denser, and more powerful.

Impact of Device Shrinkage

- What happens when the feature size (transistor size) shrinks by a factor of x?
  - Clock rate goes up by x because wires are shorter
  - Transistors per unit area goes up by \( x^2 \)
  - Die size also tends to increase
  - Typically another factor of \( \sim x \)
  - Raw computing power of the chip goes up by \( \sim x^3 \)
  - Of which \( x^3 \) is devoted either to parallelism or locality

Concurrency in its full glory

- Concurrency issues are not specific to a few niche applications
- Processor-level parallelism:
  - Bit-level parallelism: within floating point operations, etc.
  - Instruction level parallelism: multiple instructions execute per clock cycle.
  - Memory system parallelism: overlap of memory operations with computation
- Operating systems:
  - Multiple services executing concurrently
  - Different hardware devices being managed simultaneously
  - RAID (redundant array of inexpensive disks): multiple disks being accessed simultaneously

Concurrency in its full glory (contd.)

- Distributed systems:
  - Servers and clients need to coordinate their actions
  - Networked systems (such as the Internet) need to concurrently compute routes
  - Distributed transaction systems need concurrency control to provide strong correctness properties while achieving high performance
  - Peer-to-peer systems might comprise of a large number of nodes maintaining a large number of resources
- This course:
  - Expanded view of concurrent systems
  - Includes study of distributed systems and algorithms as well
  - Limited focus on traditional scientific applications
Concurrency Landscape

<table>
<thead>
<tr>
<th>Shared memory systems</th>
<th>Distributed memory systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors access shared memory</td>
<td>Processors exchange messages</td>
</tr>
<tr>
<td>Tightly coupled systems</td>
<td>Loosely coupled systems</td>
</tr>
<tr>
<td>Thread programming model</td>
<td>Message passing model</td>
</tr>
<tr>
<td>Small scale (tens of processors)</td>
<td>Large scale (up to thousands)</td>
</tr>
</tbody>
</table>

Parallel Computing | Distributed computing
--- | ---
Single shared goal | Complex autonomous agents
Simple failure model | Processors fail-stop/byzantine
Processor set is static | Dynamic node join/leave
Homogeneous systems | Heterogeneous systems
Mostly synchronous execution | Mostly asynchronous execution

Course Structure

- Parallel algorithms for shared memory machines (PRAM algorithms)
- Parallel algorithms for distributed memory machines (different interconnection networks)
- Distributed algorithms (elections, routing, etc.)
- Parallel architectures and languages
- Parallel linear algebra
- Distributed peer-to-peer systems, distributed transaction systems
- Canonical parallel applications

Course Work

- Individual home-works (about 20% of the grade)
- Programming assignments & semester-long project as teams (about 50% of the grade)
- Course material:
  - Handouts of book chapters
  - Technical papers
  - Most material will be on class webpage
- Two exams (about 30% of the grade)