수업개요

- dcslab.snu.ac.kr/courses/pp2011/

- 교재 1 : Designing and Building Parallel Programs
  by Ian Foster

- 교재 2 : Programming MPPs (draft)
  by David Kirk, Wen-mei Hwu

- 교재 3 : GPU programming guide
Programming Languages/Systems

- MPI
- OpenMP
- CUDA
- Cilk
Motivation and History
Tunnel Vision by Experts

❖ “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.
Tunnel Vision by Experts

❖ “640K [of memory] ought to be enough for anybody.”
   • Bill Gates, chairman of Microsoft, 1981.

❖ “On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it.”
   • Ken Kennedy, CRPC Directory, 1994
Why Faster Computers?

- Solve compute-intensive problems faster
  - Make infeasible problems feasible
  - Reduce design time

- Solve larger problems in same amount of time
  - Improve answer’s precision
  - Reduce design time

- Gain competitive advantage
Definitions

- **Parallel computing**
  - Using parallel computer to solve single problems faster

- **Parallel computer**
  - Multiple-processor system supporting parallel programming

- **Parallel programming**
  - Programming in a language that supports concurrency explicitly
Why MPI?

- MPI = “Message Passing Interface”
- Standard specification for message-passing libraries
- Libraries available on virtually all parallel computers
- Free libraries also available for networks of workstations or commodity clusters

- MPI, the complete reference
Why OpenMP?

- OpenMP is an application programming interface (API) for shared-memory systems.
- Supports higher performance parallel programming of symmetrical multiprocessors.
- OpenMP.org
Why Cilk?

- Cilk is a C/C++ extension which makes it easy to program multithreaded programs
- With the wide availability of multi-core CPUs, it’s the most natural way of programming

- www.cilk.com
Why CUDA?

- A quiet revolution!
  - 367 vs. 32 GFLOPS

- www.nvidia.com/object/cuda_home.htm
Why CUDA?

A quiet revolution!
Three Laws of Computing:

- **Moore's Law.**
  - Transistors on a single chip doubles ~ every 18–24 months.

- **Gilder's Law.**
  - Aggregate bandwidth triples ~ every year.

- **Metcalfe's Law.**
  - The value of a network may grow exponentially with the number of participants.

Source: Cambridge Energy Resource Associates
Still, it is true in terms of the “# of transistors”.

http://en.wikipedia.org/wiki/Moore's_law
But, Things have changed.

# of transistors ≠ clock speed

- **Moore’s Law**
  - For a long time, it meant that *clock speeds* would also *double* along the same timeline.
  - This trend has ground to a halt!

- **Intel made an announcement in 2003**
  - that they were moving away
    - from trying to increase the *clock speed*
    - to adding *more cores*
Is there speed limit?

Little progress have been made since the curve flattens out around 2007!
Moore’s Law Extrapolation:
Power Density for Leading Edge Microprocessors

Power Density (Watts / cm²)


Rocket Nozzle
Nuclear Reactor
Hot Plate

Power Density Becomes Too High to Cool Chips Inexpensively

Source: Shekhar Borkar, Intel Corp
Is the Speed of light fast enough?

- **299 792 458 m / s**
- **Modern computer clock speed**
  - 3 GHz
  - 3,000,000,000 cycles per second
- **How far can the light travel during 1 cycle?**
  - About 10 cm

- **Electronic circuits are small, but......**
Consider the 1 Tflop/s sequential machine:

- Data must travel some distance, \( r \), to get from memory to CPU.
- To get 1 data element per cycle, this means \( 10^{12} \) times per second at the speed of light, \( c = 3 \times 10^8 \) m/s. Thus \( r < \frac{c}{10^{12}} = 0.3 \) mm.

Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:

- Each bit occupies about 1 square Angstrom, or the size of a small atom.

No choice but parallelism
Revolution is Happening Now

- Chip density is continuing increase $\sim 2x$ every 2 years
  - Clock speed is not
  - Number of processor cores may double instead
- There is little or no hidden parallelism (ILP) to be found
- Parallelism must be exposed to and managed by software

Source: Intel, Microsoft (Sutter) and Stanford (Olukotun, Hammond)
Multicore

- **What is it?**
  - a processing system composed of two or more independent cores.

- **Modification to hardwares**
  - Integration of cores
  - Interconnect between components
    - AMD: Direct Connect Architecture
    - Intel: QuickPath
  - Shared cache between cores
    - AMD: shared L3
    - Intel: shared L2
Traditional Scaling Process

Speedup

User code

Traditional Uniprocessor

Time: Moore’s law

- 1.8x
- 3.6x
- 7x
Multicore Scaling Process

Speedup

1.8x

3.6x

7x

User code

Multicore

Unfortunately, not so simple...
Real-World Scaling Process

Speedup

1.8x

2x

2.9x

User code

Multicore

Parallelization and Synchronization require great care...
What is the problem here?

- Linear speedup is only possible when processes run totally independent of each other.
- Even then, OS interferes.
  - Shared data structures for
    - Scheduling
    - Interrupt handling

- If processes have to communicate (most cases), it gets worse......
No more free speedup, unless ...

- Redesign the Software to
  - exploit parallelism
  - provide scalability

Data Explosion comes to rescue!
- Plenty of PARALLELISM
Multi-Core Products

**AMD**

- **Turion X2**
  - 2.1~2.4 GHz
  - L1: 128KB
  - L2/core: 512KB

- **Athlon X2**
  - L1: 128KB
  - L2/core: 512KB

- **Phenom II X4**
  - 2.5~3.2 GHz
  - L1/core: 128KB
  - L2/core: 512KB
  - L3: 6MB

- **Phenom II X3**
  - 2.6~2.8 GHz
  - L1/core: 128KB
  - L2/core: 512KB
  - L3: 6MB

- **Opteron (Codename: Istanbul)**
  - L1/core: 128KB
  - L2/core: 512KB
  - L3: 6MB

  • 12, 16 cores will be ready on 2011.

**Intel**

- **Core2Duo**
  - 1.8~3.3 GHz
  - L1: 64KB
  - L2: 2~6MB
  - L3: 6MB

- **Core2Quad**
  - 2~3 GHz
  - L1/core: 128KB
  - L2: 4~12MB

- **Core i7**
  - 2.93 GHz
  - L1/core: 128KB
  - L2: 8MB

  Each core implements “HyperThreading” ➞ Total 8 processing threads
Berkeley NOW Project (1995)
Beowulf Concept

- NASA (Sterling and Becker)
- Commodity processors
- Commodity interconnect
- Linux operating system
- Message Passing Interface (MPI) library
- High performance/$ for certain applications
<table>
<thead>
<tr>
<th>NAME/MANUFACTURER/COMPUTER</th>
<th>LOCATION</th>
<th>COUNTRY</th>
<th>CORES</th>
<th>R_{TeraFlops}</th>
</tr>
</thead>
<tbody>
<tr>
<td>K Computer</td>
<td>RIKEN</td>
<td>Japan</td>
<td>548,352</td>
<td>8.16</td>
</tr>
<tr>
<td>Tianhe-1A</td>
<td>NUDT/NSCC/Tianjin</td>
<td>China</td>
<td>186,368</td>
<td>2.56</td>
</tr>
<tr>
<td>Jaguar</td>
<td>DOE/SC/ORNL</td>
<td>USA</td>
<td>224,162</td>
<td>1.76</td>
</tr>
<tr>
<td>Nebulae</td>
<td>NSCS</td>
<td>China</td>
<td>120,640</td>
<td>1.27</td>
</tr>
<tr>
<td>Tsubame 2.0</td>
<td>TiTech</td>
<td>Japan</td>
<td>73,278</td>
<td>1.19</td>
</tr>
</tbody>
</table>

**PERFORMANCE DEVELOPMENT**

<table>
<thead>
<tr>
<th>Year</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>1 GFlop/s</td>
</tr>
<tr>
<td>1994</td>
<td>10 GFlop/s</td>
</tr>
<tr>
<td>1995</td>
<td>100 GFlop/s</td>
</tr>
<tr>
<td>1996</td>
<td>1 TFlop/s</td>
</tr>
<tr>
<td>1997</td>
<td>10 TFlop/s</td>
</tr>
<tr>
<td>1998</td>
<td>100 TFlop/s</td>
</tr>
<tr>
<td>1999</td>
<td>N=500</td>
</tr>
<tr>
<td>2000</td>
<td>N=1</td>
</tr>
<tr>
<td>2001</td>
<td>1.17 TFlop/s</td>
</tr>
<tr>
<td>2002</td>
<td>8.7 GFlop/s</td>
</tr>
<tr>
<td>2003</td>
<td>58.8 PFlop/s</td>
</tr>
<tr>
<td>2004</td>
<td>518 PFlop/s</td>
</tr>
<tr>
<td>2005</td>
<td>40.2 TFlop/s</td>
</tr>
<tr>
<td>2006</td>
<td>60.4 GFlop/s</td>
</tr>
<tr>
<td>2007</td>
<td>99.7 GFlop/s</td>
</tr>
<tr>
<td>2008</td>
<td>0.4 Calor/s</td>
</tr>
<tr>
<td>2009</td>
<td>0.3 Calor/s</td>
</tr>
<tr>
<td>2010</td>
<td>0.2 Calor/s</td>
</tr>
<tr>
<td>2011</td>
<td>0.1 Calor/s</td>
</tr>
<tr>
<td>2012</td>
<td>0.0 Calor/s</td>
</tr>
<tr>
<td>2013</td>
<td>0.0 Calor/s</td>
</tr>
<tr>
<td>2014</td>
<td>0.0 Calor/s</td>
</tr>
<tr>
<td>2015</td>
<td>0.0 Calor/s</td>
</tr>
<tr>
<td>2016</td>
<td>0.0 Calor/s</td>
</tr>
<tr>
<td>2017</td>
<td>0.0 Calor/s</td>
</tr>
<tr>
<td>2018</td>
<td>0.0 Calor/s</td>
</tr>
</tbody>
</table>
Programming Parallel Computers

- Extend compilers: translate sequential programs into parallel programs
- Extend languages: add parallel operations
- Add parallel language layer on top of sequential language
- Define totally new parallel language and compiler system
Current Status

- **Low-level approach is most popular**
  - Augment existing language with low-level parallel constructs
  - MPI and OpenMP are examples

- **Advantages of low-level approach**
  - Efficiency
  - Portability

- **Disadvantage: More difficult to program and debug**
Thank You!

Q & A