Scheduling and Threads

10/5/15
Recall: Batch scheduling

- Jobs arriving in system over time, waiting for access to processor
- Each job has release time and processing time
- Common algorithms:
  - First-Come First-Served (FCFS)
  - Shortest Processing Time (SPT)
  - Shortest Remaining Processing Time (SRPT)
- Metric: Total response time
  - Response for job $j = \text{completion}_j - \text{arrival}_j$
  - Sum this over all jobs
What is the total response time of the jobs below if scheduled by SRPT (allowing preemptions)?

A. 15
B. 17
C. 20
D. 21
E. 24

<table>
<thead>
<tr>
<th>Job</th>
<th>arrival</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
What is the total response time of the jobs below if scheduled by SRPT (allowing preemptions)?

A. 15
B. 17
C. 20
D. 21
E. 24

<table>
<thead>
<tr>
<th>Job</th>
<th>arrival</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
How does this differ from actual CPU scheduling?

(Is anything the same?)
Threads
Thread

• Unit of program execution
  – has own program counter, stack, etc

• All threads of a process share address space
  – same global variables, state of open files etc
Moore’s law

Figure: Herb Sutter “The free lunch is over: A fundamental turn toward concurrency in software” Dr. Dobb’s Journal, 30(3), March 2005. 
http://www.gotw.ca/publications/concurrency-ddj.htm
How hot is your CPU?

• https://www.youtube.com/watch?v=7uBNCN6v_gk#t=30
Parallel computing in the small

http://i561.photobucket.com/albums/ss59/gamenews86/Die_Map.jpg
Why have multiple threads?

• Performance
  – run like this
  instead of this
Why have multiple threads?

• Performance
  – run like this instead of this

• Responsiveness
  – one thread runs user interface while others compute in background (ex: mobile platforms, web servers)
Two relevant concepts

• Parallelism
  – Using more resources to complete job faster
  – Ex: multiple cooks splitting food prep

• Concurrency
  – Managing access to shared resources
  – Ex: two cooks both trying to get dish into oven
Situation: You call a professor to see if they are free to meet. They say “yes”, but another student is in their office by the time you get there.

Is this situation an example of an issue with parallelism or concurrency?

A. Primarily parallelism
B. Primarily concurrency
C. Equally both
D. Neither
E. What are we talking about again?
Situation: You call a professor to see if they are free to meet. They say “yes”, but another student is in their office by the time you get there.
Is this situation an example of an issue with parallelism or concurrency?
A. Primarily parallelism
B. Primarily concurrency
C. Equally both
D. Neither
E. What are we talking about again?
Situation: For a group project, you and your teammates divvy up the tasks to be completed and set up a meeting to put it all together. Is this situation an example of parallelism or concurrency?

A. Primarily parallelism
B. Primarily concurrency
C. Equally both
D. Neither
E. What are we talking about again?
Situation: For a group project, you and your teammates divvy up the tasks to be completed and set up a meeting to put it all together. Is this situation an example of parallelism or concurrency?

A. Primarily parallelism
B. Primarily concurrency
C. Equally both
D. Neither
E. What are we talking about again?
Fork-join pattern

Thread Team

Initial Thread

Fork

Join

http://cnx.org/contents/66607f05-723f-a3ef-0c1f17ee00ee@7.1:6/An_Introduction_to_High-Performance_Computing
Speedup

\[
\text{Speedup} = \frac{\text{Serial (non-parallel) running time}}{\text{Parallel running time}}
\]

- Linear speedup: speedup equal to the number of processing elements
- Sublinear speedup: less than this
- Superlinear speedup: more than this
Why not linear speedup? (1)

• Some parts of the code can’t run in parallel
  – Initialization
  – I/O
  – critical sections: areas where we ensure at most one thread is running
Why not linear speedup? (1)

If $B =$ fraction of program that must run serially

$T_1 =$ total time on 1 processing element

What is best possible time on $p$ elements?

A. $T_1/p + B$

B. $T_1B/p$

C. $T_1(1-B)/p + B$

D. $T_1(1-B)/p + T_1B$

E. None of the above
Why not linear speedup? (1)

If $B =$ fraction of program that must run serially
$T_1 =$ total time on 1 processing element

What is best possible time on $p$ elements?

A. $T_1/p + B$
B. $T_1B/p$
C. $T_1(1-B)/p + B$
D. $T_1(1-B)/p + T_1B$ (called Amdahl’s Law)
E. None of the above