Lecture 20: Parallel Algorithms
Single processor vs. many processors
Does it always help to have more processors?

Is the problem “hole” or a “ditch”?
The questions of the day:

• How much does it help to have extra processors?
  • What kind of speedups?

• When doesn’t it help?
Example:

Adding $n$ integers $x_1, \ldots, x_n$

see board...
Computation DAG

• Parallel time:
  • Each node of DAG represents O(1) computation
  • Each level of DAG can be computed in parallel

• Fastest parallel algorithm:
  • Length of longest path in DAG corresponds to min parallel time
  • Max number of nodes in a level corresponds to number of processors needed to get best time
But first... what is a good model?

- Memory
  - Shared?
    - multicore
  - Distributed?
    - Connection machine
- How many processors?
  - Does the program know?
  - Can it change during execution?
  - Who handles the “scheduling”?
Programming models

Some buzzwords:

• **static threading** -- software abstraction of virtual processors sharing common memory

• **dynamic threading** – programmers specify parallelism without worrying about communication protocols, load balancing,...
Memory model (for today)

• Shared memory –
  • Many processors can read from same location at same time
  • Careful: what if more than one processor writes to same memory location at same time?
  • Today – make sure to avoid
    • Concurrent Read, Exclusive Write (CREW) PRAM
  • Other possibilities –
    • Ok as long as write the same thing?
      • Concurrent Read, Concurrent Write (CRCW) PRAM
    • Break the computation?
Processors (for today)

• Assume there are as many as we want
  • But, evaluate algorithm based on processor usage...
  • Should at most be poly in input size
Scheduling the processors

- Do we know computation DAG in advance?
- Type of control
  - Distributed control –
    - better for practice/more complicated
  - Centralized control
    - today
Performance measures

See board for defs of work, span, running time, speedup, parallelism

See board for “work law” and “span law”
Greedy scheduler

Assign as many “strands” as possible at each time

Thm. For greedy schedule, $T_p \leq \frac{T_1}{p} + T_\infty$

Proof:

“Complete step” – all processors compute

Claim 1: $\leq \lfloor \frac{T_1}{p} \rfloor$ complete steps

Claim 2: incomplete steps reduce span by 1
More examples

• Matrix Multiply
• Merging sorted lists
What problems are “holes”?

- As hard to parallelize as any problem in P:
  - Circuit evaluation: Given a circuit and inputs, what is the output?
  - Linear programming
- Unknown:
  - GCD
  - Factoring
Have a great Thanksgiving!!!!