Lecture 12. 10/19/2010

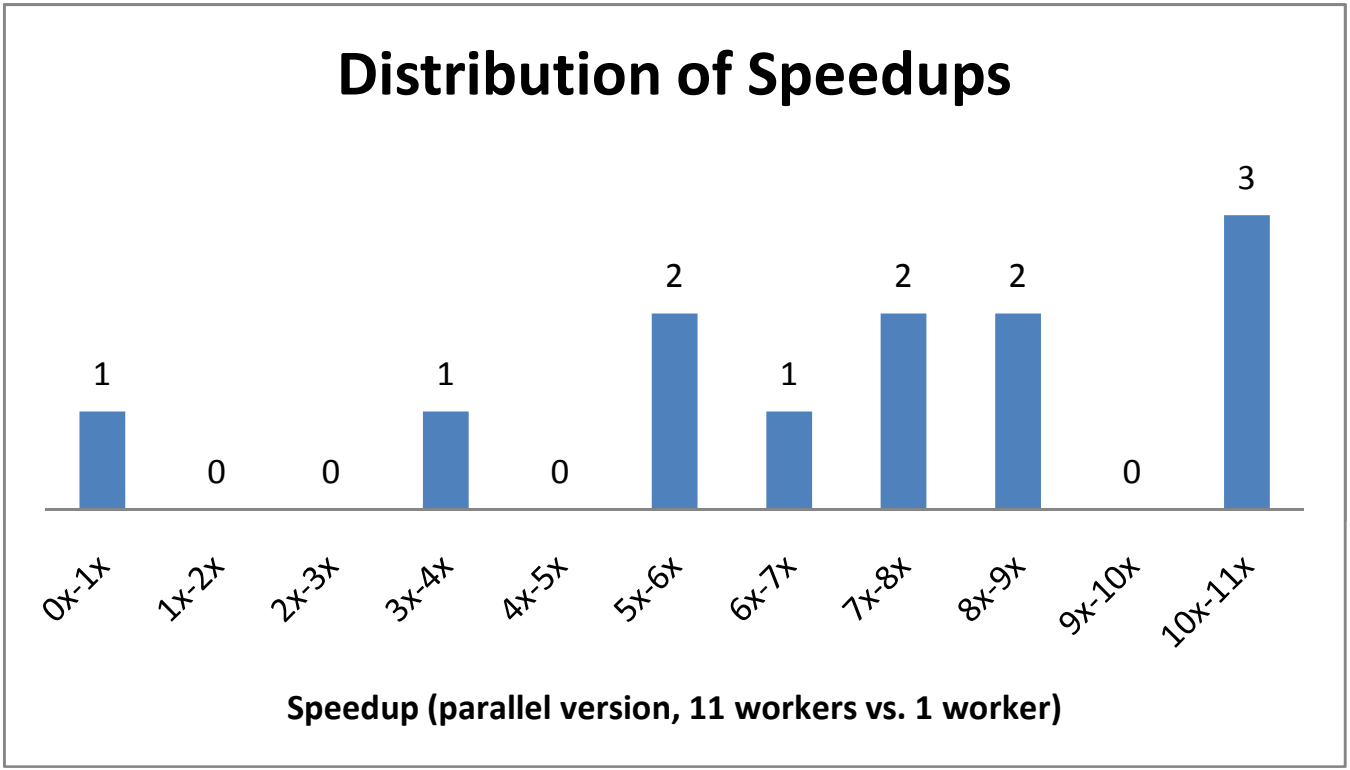
final projects
labs 3 and 4
an "indirect" quiz review

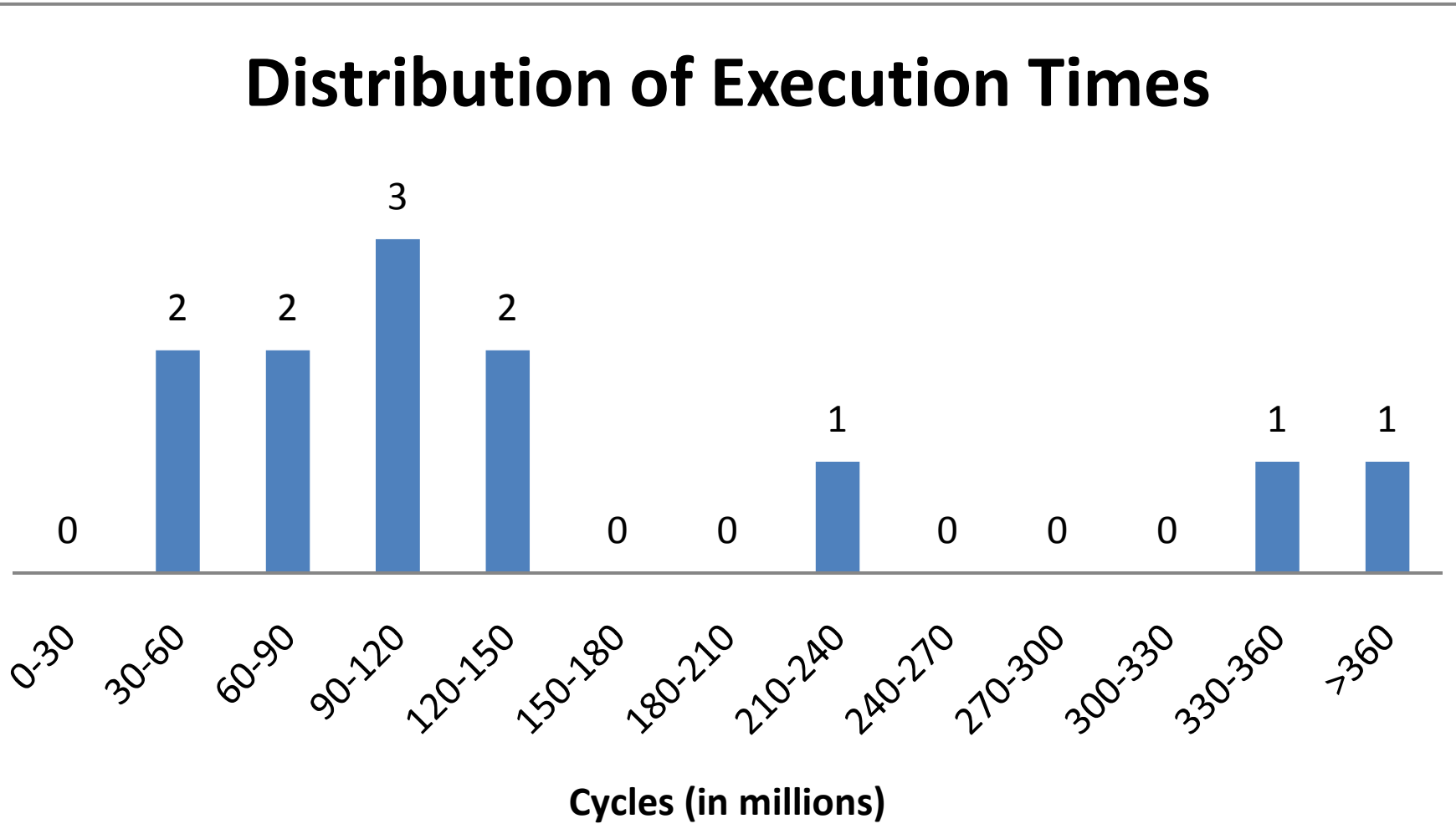
final projects:
  1 person or 2-person team
  abstract (11/5), proposal (11/12), report (12/3)
  in-class presentation (12/6)
  driven by apps (TSP, etc. maybe microbenchmark)
    TSP
    Jacobi
    Parallel sort
    Map-reduce
    ...
    Nice to have apps with much synchronization/coordination
  project ideas:
    cache-coherence
    read/write locks with a parallel app that benefits from them
    scalable locks
    monitors
    fetch-and-add
    accumulator functions in ring
    flow control for messages
    emulate memory hierarchy
    hardware-accelerated work-stealing library
    active messages
    hardware multicast
    improve ring protocol:
      remove train
      several modules
    prefetch module
    block transfer/swap
    transactional memory (v6)
  [Look at TMC C library C* CMMD, CM-5]

lab 4: barrier implementations: sw and hw
  algorithms sw
    hardware broadcast
      number of ring messages?
      why correct?
    using ptp messages: broadcast in sw.
      ordering a problem?
      number of ring messages?
    organize nodes in a ring, master passes token twice around the ring
      [number of ring messages? skip]
    what bugs?
  algorithms hw
    like hardware broadcast: count, when n reset to 0
      set done when count reaches n
      order locker, messenger, and barrier unit
    what bugs?

lab 3: tsp mp
  algorithmic
    divide up the work

```
    static partition
    dynamic with work queue
    master-less work queue
  reduce the amount of work
    pick bound using approximate TSP solver
    explore greedily
    sort input matrix
    eliminate path permutations
  management of the bound
    how to propogagate updates?
    how often to poll?
    how often is the bound updated?
  what bugs?
performance
  speedup vs. performance
    show jonathan charts
  how to measure?
    average over many graphs
  what hacks?
    how to encode work?
    how to implement present? (bit vector)
  how general are the hacks?
```


Distribution of Speedups
1
0
0
1
0
2
1
2
2
0
3
0x-1x
1x-2x
2x-3x
3x-4x
4x-5x
5x-6x
6x-7x
7x-8x
8x-9x
9x-10x
10x-11x
Speedup (parallel version, 11 workers vs. 1 worker)


Distribution of Execution Times
0
2
2
3
2
0
0
1
0
0
0
1
1
0-30
30-60
60-90
90-120
120-150
150-180
180-210
210-240
240-270
270-300
300-330
330-360
>360
Cycles (in millions)