Exploting Bounded Staleness To Speed Up Big Data Analytics
Henggang Cui, James Cipar, Qirong Ho, Jin Kyu Kim, Abhimanyu Kumar, Seunghak Lee, Wei Dai, Jinliang Wei, Greg Ganger, Phil Gibbons*, Garth Gibson, Eric Xing  (CMU, *Intel)

BOUNDSTALENESSINPARALLELML
- Can tune staleness of shared data
- Arbitrarily-sized Bulk Synchronous Parallel (A-BSP)
  > A barrier every some amount of work (a clock)
  > BSP with tunable “work per clock” (WPC)
- Stale Synchronous Parallel (SSP)
  > Fastest worker ≤ “slack” clocks ahead of slowest
  > Tunable “slack” (see LazyTables poster)
- Data staleness bound for “(wpc, slack)”
  > wpc x (slack + 1)
  > SSP (wpc, slack) == A-BSP (wpc x (slack + 1), 0)

EXPERIMENTAL SETUP
- App: Topic Modeling (LDA with Gibbs sampling)
  > Nytimes dataset (300k documents, 100m words)
  > Similar results for other ML apps
- Hardware (2 clusters)
  > Default: 8 64-core machines with 128GB RAM
  > vCloud: 8 64-core machines with 128GB RAM

SPEED-EFFECTIVENESS TRADEOFF
- Controlled by staleness bound
- SSP: fixed wpc, change slack
  > More staleness → more iters/sec, less convergence/iter
  > A sweet spot in the middle
- A-BSP: slack is always zero, change wpc
  > Similar tradeoffs

TOLENTABILITYOFSTRAGGLERS
- SSP’s slack mitigates small transient delays
- Ex: Delayed threads
  > Threads on machine i sleep() “d” seconds at iteration i
- Ex: Background work (on vCloud)
  > Disrupter process on each machine consumes 50% CPU in
each time slot (“t” seconds) with probability 10%

COMMUNICATIONOVERHEAD
- Total traffic drops as WPC grows
- Updates sent every clock, reads on many
- SSP uses smaller WPC for same staleness bound