LazyTable: Distributed Machine Learning with the Stale Synchronous Parallel Model
Qirong Ho, Henggang Cui, James Cipar, Jin Kyu Kim, Abhimanu Kumar, Seunghak Lee, Wei Dai, Jinliang Wei, Greg Ganger, Phil Gibbons*, Garth Gibson, Eric Xing (CMU, *Intel)

PARALLEL MACHINE LEARNING
- Learn model parameters from a big dataset
  › Work is partitioned among multiple threads
  › Each thread processes a partition of input data
  › Threads iteratively update the shared parameter state based on their input data
- Parameter server
  › Maintains shared values for worker threads
  › Tradeoff between fresh views and synchronization
- Most ML algorithms tolerate bounded staleness
  › Common model: Bulk Synchronous Parallel
    • Barrier and data update at end of each clock
    • Worker guaranteed to see updates up to previous clock
  › New model: Stale Synchronous Parallel

WHY DOES SSP CONVERGE?
- Theorem 1: SSP approximates sequential execution
  › Error at each update is strictly bounded
- Theorem 2: For iterative-convergent ML problems, SSP guarantees algorithm convergence
  › Hence, ML algorithms converge under SSP
    › Albeit via a noisy trajectory

STATELY SYNCHRONOUS PARALLEL MODEL
- Tunable data staleness ("slack")
  Any thread can be up to slack clocks ahead of slowest thread
  - Slack bound of 3 clocks
  - Results visible, because old enough
  - Results visible, because read-my-writes
  - Results not necessarily visible

LAZYTABLE SYSTEM OVERVIEW
- Parameter server based on SSP
  › A client library with a cluster of tablet servers
  › Multiple layers of caches and operation logs
  › Closer caches tend to be more stale, but faster
  › Slack bound specified in each read operation
  › Data allowed to be "slack" clocks stale
  › Cache data returned, if fresh enough

RESULTS & DIRECTIONS
- Many results found on companion poster
- Key takeaways: converge faster with SSP
  › More staleness → more iters/sec, less effective/iter
    › Sweet spot balances the two
  › Works well for range of ML approaches
    › Topic Modeling (LDA with Gibbs sampling)
    › Sparse Matrix Factorization (stochastic gradient descent)
    › Shotgun (coordinate gradient descent)
  › Continuing to explore iterative nature
    › Better data assignment to tablet servers
    › Memory/thread scheduling on multi-core machines
    › Try for near-ideal straggler tolerance

Theorem 1: SSP approximates sequential execution
   Maximum extent of error for this update:
Next state = previous state + noisy gradient

Next state = previous state + noisy gradient