Solving the straggler problem with bounded staleness

Jim Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Gregory R. Ganger, Garth Gibson, Kimberly Keeton*, Eric Xing

PARALLEL DATA LABORATORY
Carnegie Mellon University

* HP Labs
Overview

It’s time for all applications (and systems) to worry about data freshness

• Current focus: parallel machine learning

• Often limited by synchronization overhead

• What if we explicitly allow stale data?
A typical ML algorithm

Input data
A typical ML algorithm

Input data

(1) initialization

Intermediate Program state
A typical ML algorithm

Input data

(1) initialization

Intermediate Program state

(2) Iterate, many small updates
A typical ML algorithm

Input data → (1) initialization → Intermediate Program state

(2) Iterate, many small updates
(3) Output results after many iterations

Output results
A typical ML algorithm

1. Initialization

Input data

Intermediate Program state

2. Iterate, many small updates

3. Output results after many iterations

Output results
Parallel ML

• Generally follows **bulk synchronous parallel** model

• Many iterations of
  1. **Computation**: compute new values
  2. **Synchronization**: wait for all other threads
  3. **Communication**: send new values to other threads
  4. **Synchronization**: wait for all other threads... again
All threads must be on the same iteration to continue.
Stragglers in BSP

Slow thread(s) will hold up entire application

• Predictable stragglers
  – Slow/old machine
  – Bad network card
  – More data assigned to some threads
Stragglers in BSP

Slow thread(s) will hold up entire application

• Predictable stragglers ➔ Easy case
  – Slow/old machine
  – Bad network card
  – More data assigned to some threads
Stragglers in BSP

Slow thread(s) will hold up entire application

• Predictable stragglers ➔ Easy case
• Unpredictable stragglers ➔ ???
Stragglers in BSP

Slow thread(s) will hold up entire application

• Predictable stragglers ➔ Easy case
• Unpredictable stragglers ➔ ???
  – Hardware: disk seeks, network, CPU interrupts
  – Software: garbage collection, virtualization
  – Algorithmic: Calculating objectives and stopping conditions
Stragglers in BSP

Slow thread(s) will hold up entire application

- Predictable stragglers ➔ Easy case
- Unpredictable stragglers
  - Hardware: disk seeks, network latency, CPU interrupts
  - Software: garbage collection, virtualization
  - Algorithmic: calculating objectives, stopping conditions

Don’t synchronize
Don’t synchronize?

• Well, don’t synchronize much
  – Read old (stale) results from other threads
  – Application controls how stale the data can be

• Machine learning can get away with that

• Algorithms are **convergent**
  – Given (almost) any state, will find correct solution
  – Errors introduced by staleness are usually ok
Points are initialized randomly,
Always settle to correct locations
Freshness and convergence: the sweet spot

Fresh reads/writes

Stale reads/writes
Freshness and convergence: the sweet spot

Fresh reads/writes vs. Stale reads/writes

Iterations per second
Freshness and convergence: the sweet spot

- Improvement per iteration
- Iterations per second

Fresh reads/writes vs. Stale reads/writes
Freshness and convergence: the sweet spot

- Improvement per iteration
- Iterations per second

Fresh reads/writes

Stale reads/writes
Freshness and convergence: the sweet spot

![Graph showing the relationship between freshness and convergence](image)

- Fresh reads/writes vs. Stale reads/writes
- Improvement per second
- The sweet spot
Stale synchronous parallel

• Allow threads to continue ahead of others
  – Avoids temporary straggler effects

• Application can limit allowed staleness
  – Ensure convergence
  – E.g. “threads may not be more than 3 iters ahead”
Threads proceed, possibly using stale data
Increased staleness can mask the effects of occasional delays.
Ongoing work

• Characterizing “staleness-tolerant” algorithms
  – Properties of algorithms, rules of thumb
  – Convergence proof

• Automatically tune freshness requirement

• Specify freshness by error bounds
  – “Read X with no more than 5% error”
Summary

Introducing staleness, but *not too much staleness*, can improve performance of machine learning algorithms.