Exploiting Bounded Staleness to Speed up Big Data Analytics

Henggang Cui

James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Abhimanu Kumar, Jinliang Wei, Wei Dai, Greg Ganger, Phil Gibbons (Intel), Garth Gibson, and Eric Xing

Carnegie Mellon University
Big Data Analytics Overview

Iterative program fits model

One iteration

Huge input data

Iterative program fits model

Model parameters (solution)
Big Data Analytics Overview

Partitioned input data → Parallel iterative program → Model parameters (solution)
Big Data Analytics Overview

Partitioned input data

Parallel iterative program

Model parameters (solution)

Goal: Less sync overhead

Parameter server
Outline

• Two novel synchronization approaches
  • Arbitrarily-sized Bulk Synchronous Parallel (A-BSP)
  • Stale Synchronous Parallel (SSP)
• LazyTable architecture overview
• Taste of experimental results
Bulk Synchronous Parallel

- A barrier every **clock** (a.k.a. epoch)
  - In ML apps, often one iteration over input data

Thread progress illustration:

- **Thread 1**
  - blocked by barrier
- **Thread 2**
  - Updates not necessarily visible
- **Thread 3**
  - Iterations complete, updates visible
Data Staleness

- In BSP, threads can see "out-of-date" values
  - May not see others' updates right away
  - Convergent apps usually tolerate that
- Allowing more staleness for speed
  - Less synchronizing among threads
  - More using cached values
  - More delaying and batching of updates
- But, too much staleness hurts convergence
  - Important to have staleness bound
  - Staleness should be tunable
Arbitrarily-sized BSP (A-BSP)

- Work in each clock can be more than one iteration
  - Less synchronization overhead

![Diagram showing two iterations per clock with threads blocked by barrier](image)

Thread 1

Thread 2

Thread 3

Iteration

Clock

Henggang Cui © October 14
Problem of (A-)BSP: Stragglers

• A-BSP still has the straggler problem
  • A slow thread will slow down all
  • Stragglers are common in large systems

• Many reasons for stragglers
  • Hardware: lost packets, SSD cleaning, disk resets
  • Software: garbage collection, virtualization
  • Algorithmic: calculating objectives and stopping conditions
Stale Synchronous Parallel (SSP)

- Threads are allowed to be slack clocks ahead of the slowest thread

[HotOS’13, NIPS’13]
Two Dimensional Config. Space

- Iters-per-clock and slack are both tunable
  - A-BSP is SSP with a slack of zero
  - Every SSP config. has an A-BSP counterpart with the same data staleness bound

SSP (iters-per-clock=1, slack=1):  
A-BSP (iters-per-clock=2, slack=0):
LazyTable Architecture

Partitioned input data

Parallel iterative program on LazyTable

Model parameters (sharded)
LazyTable Architecture

Partitioned input data

Parallel iterative program on LazyTable

Model parameters (sharded)

See the paper for more details

Henggang Cui © October 14
Primary Experimental Setup

- **Hardware information**
  - 8 machines, each with 64 cores & 128GB RAM

- **Basic configuration**
  - One client & tablet server per machine
  - One computation thread per core
Application Benchmark #1

• Topic Modeling
  • Algorithm: Gibbs Sampling on LDA
  • Input: NY Times dataset
    – 300k docs, 100m words, 100k vocabulary
  • Solution quality criterion: Loglikelihood
    – How likely the model generates observed data
    – Becomes higher as the algorithm converges
    – A larger value indicates better quality

More apps described and used in paper
Controlling Data Staleness

• SSP
  • Larger slack -> more staleness

• A-BSP
  • Larger iterations-per-clock -> more staleness

• The tradeoffs with increased staleness
Iters-per-clock is 1
Staleness Increases Iters/sec

Iterations per sec

-- slack=0 (BSP)

Iters-per-clock is 1
Staleness Increases Iters/sec

Iters/sec

slack=0 (BSP)
slack=1 (SSP)

Iterations per sec

Iteraciones done

Time (sec)

larger iters per sec with more staleness

iters-per-clock is 1

http://www.pdl.cmu.edu/
Staleness Increases Iter/s/sec

Iter/s vs. Time (sec)
-
- slack=0 (BSP)
- slack=1 (SSP)
- slack=3 (SSP)

Iters-per-clock is 1
Staleness Reduces Converge/iter

Convergence per iter

Iters-per-clock is 1

Iterations done

Convergence (higher is better)
Staleness Reduces Converge/iter

Convergence per iter

Convergence (higher is better)

Iterations done
slack=0 (BSP)

iters-per-clock is 1

Iterations done

http://www.pdl.cmu.edu/

Henggang Cui © October 14
Staleness Reduces Converge/iter

Convergence per iter

- slack=0 (BSP)
- slack=1 (SSP)

Iters-per-clock is 1

More iters to converge with more staleness

Convergence (higher is better)
Staleness Reduces Convergence/iter

Convergence per iter

Convergence (higher is better)

Iterations done

slack=0 (BSP)

slack=1 (SSP)

slack=3 (SSP)

Staleness Reduces Converge/iter

iters-per-clock is 1

Iterations done

http://www.pdl.cmu.edu/
Sweet Spot Balances the Two

Convergence (higher is better)

Iterations done

slack=0 (BSP)
slack=1 (SSP)
slack=3 (SSP)

Convergence per iter

Convergence (higher is better)

Iterations done

slack=0 (BSP)
slack=1 (SSP)
slack=3 (SSP)

Convergence per sec

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Convergence per sec

Convergence (higher is better)

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)

0 100 200 300

0 20 40 60

Speed up with a good slack

Iterations per sec

Time (sec)
Key Takeaway Insight #1

The sweet spot

Convergence per iteration

Iterations per second

Convergence per second

Fresher data

Staler data
SSP vs A-BSP

- Similar performance
  - In the absence of stragglers

What about environment with stragglers?
Straggler Experiment #1

- Stragglers caused by background disruption
  - Fairly common in large, shared clusters
- Experiment setup
  - One disrupter process per machine
    - Uses 50% of CPU cycles
  - Work (disrupt) or sleep randomly for $t$ seconds
    - 10% work, 90% sleep

More straggler experiments in the paper
Straggler Results #1

w/o disrupt, each iter takes 4.2 sec
Straggler Results #1

Ideally 5%, because 50% slow down with 10% probability

w/o disrupt, each iter takes 4.2 sec
Straggler Results #1

- Disruption duration (sec)
- Iteration time increase (%)

**ideal**
- wpc=2, slack=0 (A-BSP)

- Ideally 5%, because 50% slow down with 10% probability

- w/o disrupt, each iter takes 4.2 sec
Straggler Results #1

SSP tolerates transient stragglers

Ideally 5%, because 50% slow down with 10% probability

w/o disrupt, each iter takes 4.2 sec

http://www.pdl.cmu.edu/
Conclusion

• Staleness should be tuned
  • By iters-per-clock and/or slack
• LazyTable implements SSP and A-BSP
  • See paper for details
• Key results from experiments
  • Both SSP and A-BSP are able to exploit the staleness sweet-spot for faster convergence
  • SSP is tolerant of small transient stragglers
  • But SSP incurs more communication traffic
References

• NYTomes: http://archive.ics.uci.edu/ml/datasets/Bag+of+Words