Exploiting Bounded Staleness to Speed Up Big Data Analytics

Garth Gibson & the BigLearning team
Carnegie Mellon University

http://www.istc-cc.cmu.edu/
Parallel ML Systems Architecture

Partitioned input data

Parallel iterative code

Model parameters
• Compare Bounded Async Bulk Synch Parallel (A-BSP) vs Stale Synch Parallel (SSP)
• Repetition-exploiting optimizations (to BSP)
• Managed (extra) Bandwidth SSP (MBSSP)
• Convergence-guided Scheduling (STRADS)
Bulk Synchronous Parallel

- A barrier every (logical) **clock**
- chunk of work, often 1 iteration on all input data

Thread progress illustration:

- **Thread 1**
  - Iterations complete, updates visible
  - Thread 1 blocked by barrier

- **Thread 2**
  - Updates not necessarily visible

- **Thread 3**
• Threads allowed to be slack clocks ahead of slowest thread, possibly reading stale data

[HotOS’13, NIPS’13]
Arbitrarily-sized BSP (A-BSP)

- Work in each clock can be more than one iteration
  - Less synchronization overhead (bounded asynch)

- A-BSP is SSP with a slack of zero

Two iterations per clock

```
Thread 1
Thread 2
Thread 3
```

```
0 1 2 3 4  
```

```
Iteration
```

```
Thread 1 blocked by barrier
```

- A-BSP is SSP with a slack of zero
• **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

• **Hardware information**
  - 8 machines, each with 64 cores & 128GB RAM
• **Basic configuration**
  - One client & tablet server per machine
  - One computation thread per core
Staleness Increases Iters/sec

Iterations per sec

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

Iters-per-clock is 1
Convergence per iter

Staleness Reduces Converge/iteration

Convergence (higher is better)

Iters-per-clock is 1

Iterations done

slack=0 (BSP)
slack=1 (SSP)
slack=3 (SSP)
Key Takeaway Insight

The sweet spot

Convergence per iteration

Convergence per second

Fresher data

Staler data

[ATC’14]
• Iterative code often very repetitive – exploit!
  • Virtual iteration
• Affinity allocation, static & precomputed policies, multiple levels of cache, update prefetching
  • Optimization effectiveness break-down:

[under submission]

Apply Systems Experience to BSP

Lead: Henggang Cui
Managed Bandwidth SSP (MBSSP)

• In SSP, communication and computation are overlapped, but every update is treated equally
• But not every update is equally important to convergence (e.g. small vs. large deltas)
• MBSSP exploits network bandwidth not fully utilized to transmit pending updates sooner
• Early transmissions may speed convergence
  ▫ And may allow greater staleness (latency hiding)
• What to send early? Random vs delta ordered

• Leads: Jinliang Wei, Wei Dai

[under submission]
Absolute Convergence Improved 40%

Early transmission reduces time needed to converge

Delta-importance-ordered achieves as much benefit as random early send with half the extra bandwidth

LDA (Gibbs Sampling)
NYT Dataset
8x64 core nodes
1GE network
Fits in memory
• It is beneficial to send out early model refinements even with bounded bandwidth.
• Early communication improves convergence enabling much larger staleness (latency hiding).
• Application-specific policies for preferring model refinements can make a big difference.
STRADS: Up Stack to ML Scheduling

- Uniform parameter update is not optimal
  - Use deeper knowledge of ML algorithms to update parameters at different rates for best convergence speed (like MBSSP)

- Random parameter selection for parallel update risks divergence (e.g. Shotgun Lasso)
  - Control errors when selecting parameters to update in parallel

- Leads: Jin Kyu Kim, Seunghak Lee
STRADS: Two Scheduling Policies

Weight of model parameters | Delta of x |
\[\begin{array}{c}
\Delta x_1 & 0.11 \\
\Delta x_2 & 0.01 \\
\Delta x_3 & 0.003 \\
\Delta x_4 & 0.15 \\
\Delta x_5 & 0.0001 \\
\Delta x_6 & 0.001 \\
\Delta x_7 & 0.07 \\
\Delta x_8 & 0.0003 \\
\Delta x_9 & 0
\end{array}\]

One update set in ready queue

Sampling based on delta distribution

Density

Delta x

New delta info

[arXiv(1406.4580)'14]

Dependency Checker & Filter

Workers in remote machines
Benefits of Two Scheduling Policies

Application: Lasso

Synthetic data: 450 by 100K
- Parameters are highly correlated.

w/o scheduling: Limit the degree of parallelism to 70 cores to avoid divergence

STRADS still converge with 120 cores

w/o scheduling diverges beyond 80 cores
ML Iterative Solver Execution Model

Scheduling/Fetch/Execution/Aggregation model

- Scheduling selects a chord to minimize aggregate errors of parallel update
- Parameters of a chord are selected to be approximately independent
System Issue: Pipeline Scheduling

Serial execution of chords is a performance bottleneck

Approach: Make scheduling decisions with latest data only for the scheduler’s partition of the (big) model parameters
One pipeline is not enough

1: schedule w/ local freshness; execute with global freshness

2: relax freshness of least important updates (relative to next Chord)
Application: Lasso
Data: Synthetic data
50K samples, 1M dimension

Depth refers to second pipeline

Time to objective value 0.061
STRADS Vision

- STRADS’ scheduling policies show order of magnitude faster convergence speed compared to parallel ML apps w/o scheduling.
- ML Apps (esp. with divergence risks) benefit from significant scheduling and bounded staleness to fully utilize parallelism.
- Concept of “iteration” is lost when importance guides update frequency (don’t just delay communication, delay computation too).
  - Staleness can still bound minimal update frequency.
- Fully utilizing hardware when scheduling is non-trivial adds additional reasons for exploiting staleness induced error tolerance.

- Three canonical ML applications (Lasso, Logistic Regression, SVM) implemented of STRADS framework so far.
• Compare Bounded Async Bulk Synch Parallel (A-BSP) vs Stale Synch Parallel (SSP)
  ▫ Similar best case speedups
  ▫ SSP tolerates (transient) stragglers (see paper)
• Repetition-exploiting optimizations (to BSP)
• Managed extra Bandwidth SSP (MBSSP)
  ▫ Smart early-notify speeds convergence
• Convergence-guided Scheduling (STRADS)
  ▫ Up the ML stack to control update order too
  ▫ Escape straightjacket of “the iteration”
  ▫ Tackle divergence head on & use staleness for latency hiding to better utilize hardware
• Carnegie Mellon Univ students on this project
  ▫ Henggang Cui
  ▫ Qirong Ho
  ▫ Jinliang Wei
  ▫ Wei Dai
  ▫ Jin Kyu Kim
  ▫ Seunghak Lee
  ▫ Abhimanu Kumar
  ▫ James Cipar
  ▫ Alexey Tumanov
  ▫ Lianghong Xu
  ▫ Jesse Haber-Kucharsky