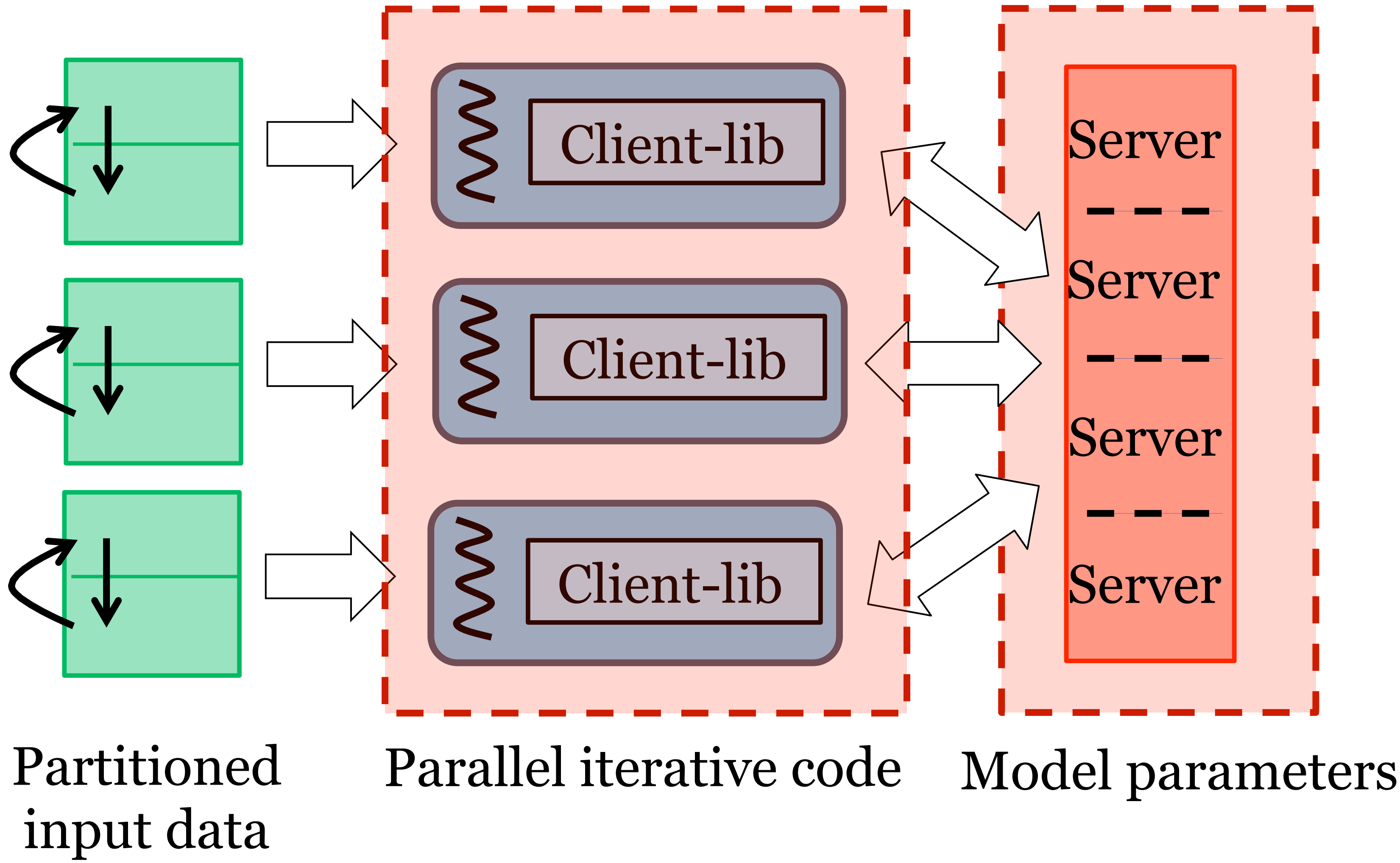
Exploiting Bounded Staleness to Speed Up Big Data Analytics

Garth Gibson & the BigLearning team
Carnegie Mellon University

http://www.istc-cc.cmu.edu/

Intel Science & Technology
Center for Cloud Computing

Parallel ML Systems Architecture



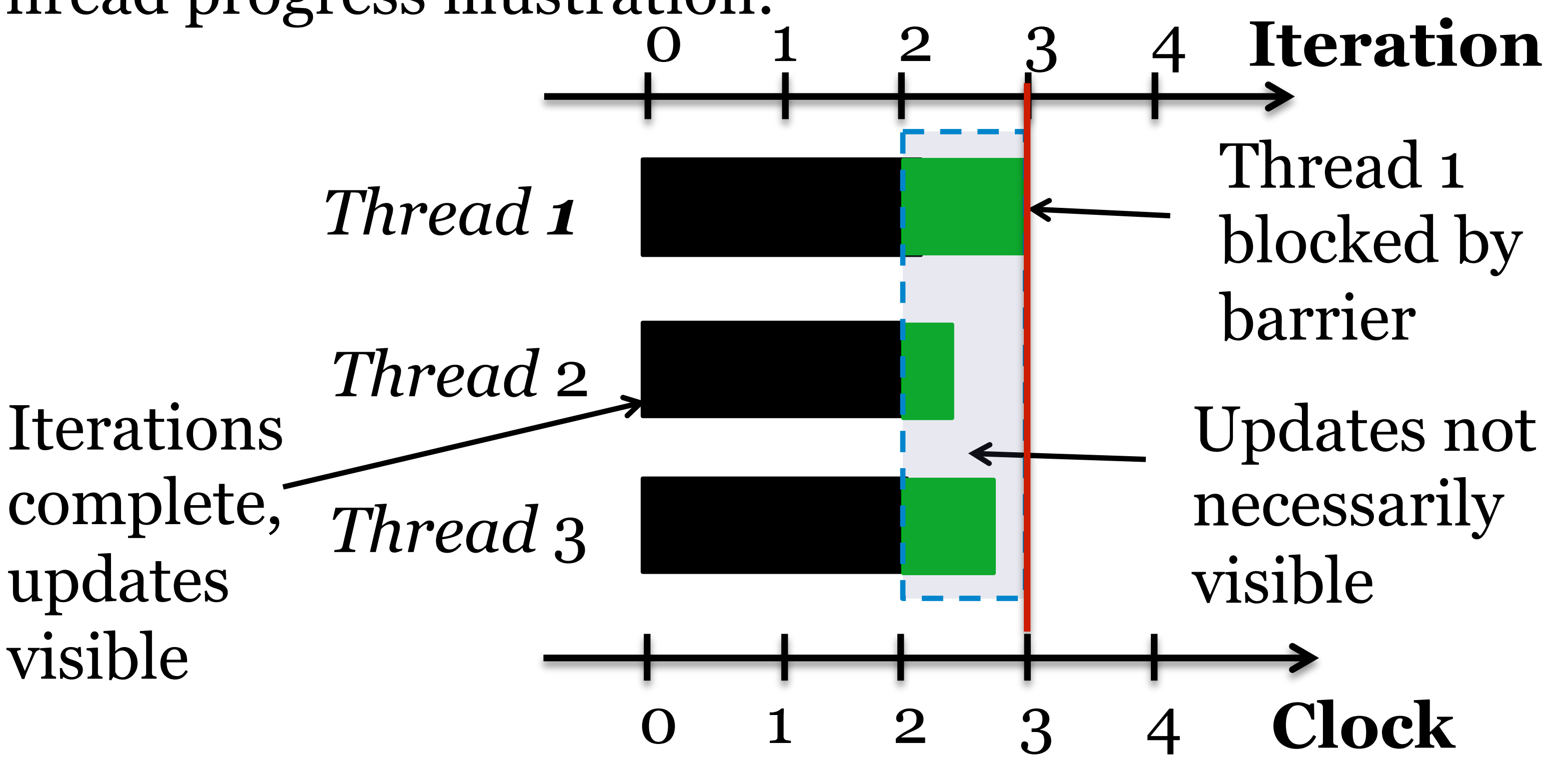
Agenda: Bound Staleness Project Suite

- 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

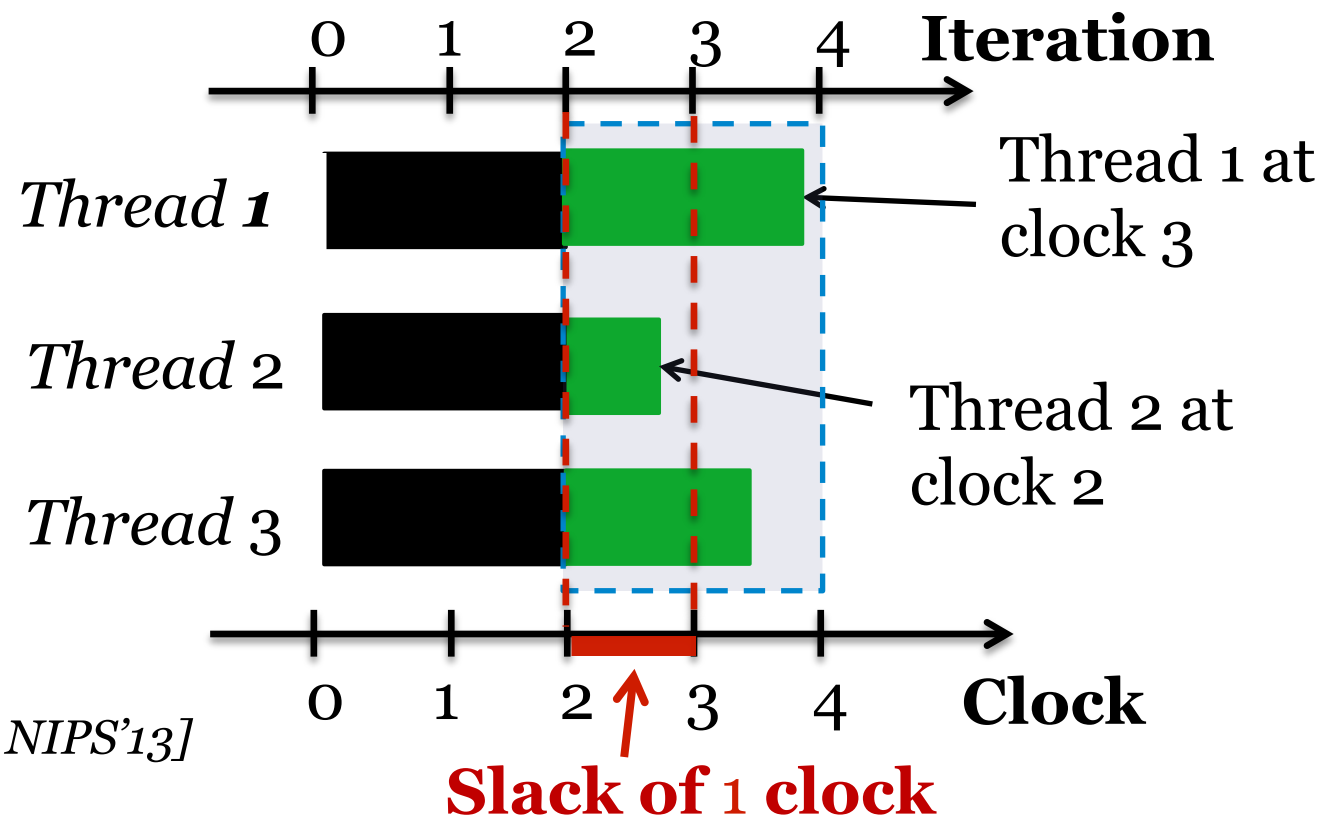
- A barrier every (logical) clock
  - chunk of work, often 1 iteration on all input data

Thread progress illustration:



Stale Synchronous Parallel (SSP)

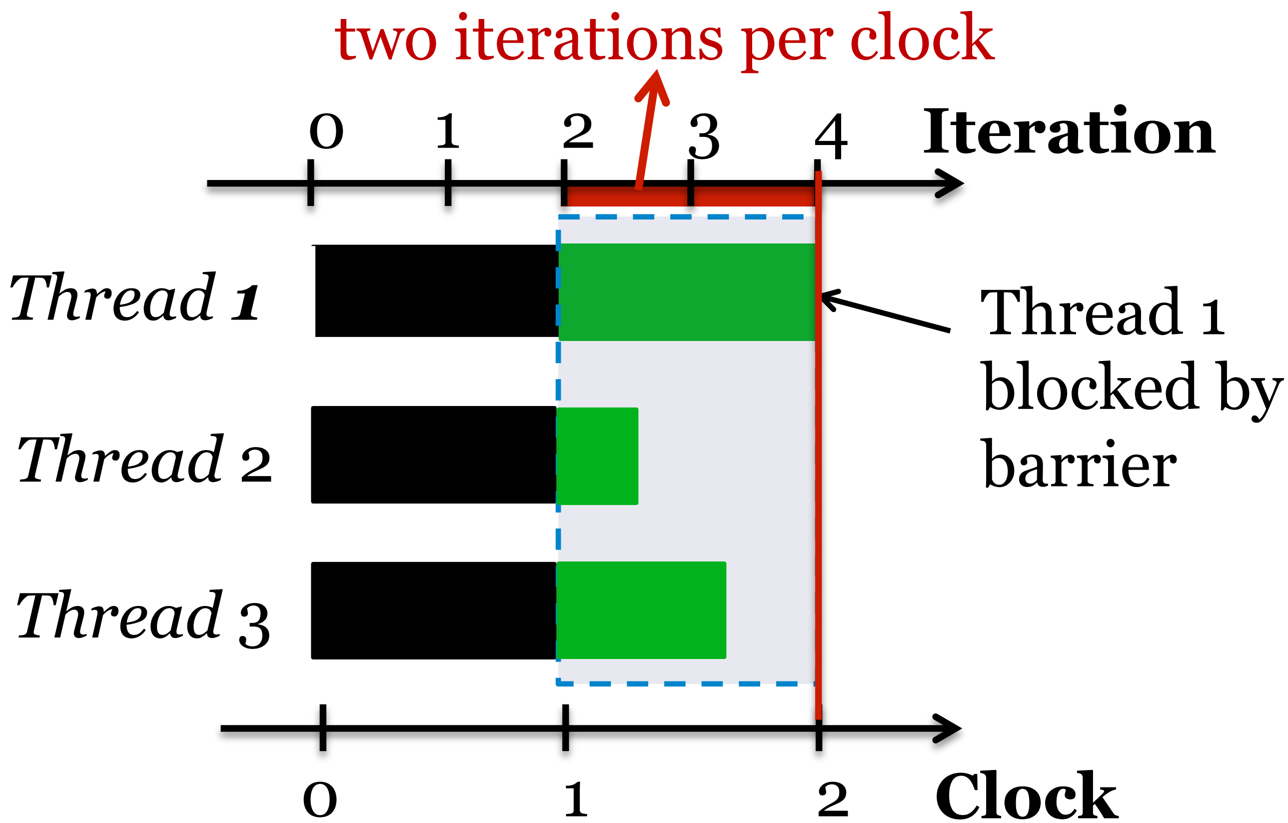
- Threads allowed to be slack clocks ahead of slowest thread, possibly reading stale data



[HotOS'13, NIPS'13]

Arbitrarily-sized BSP (A-BSP)

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



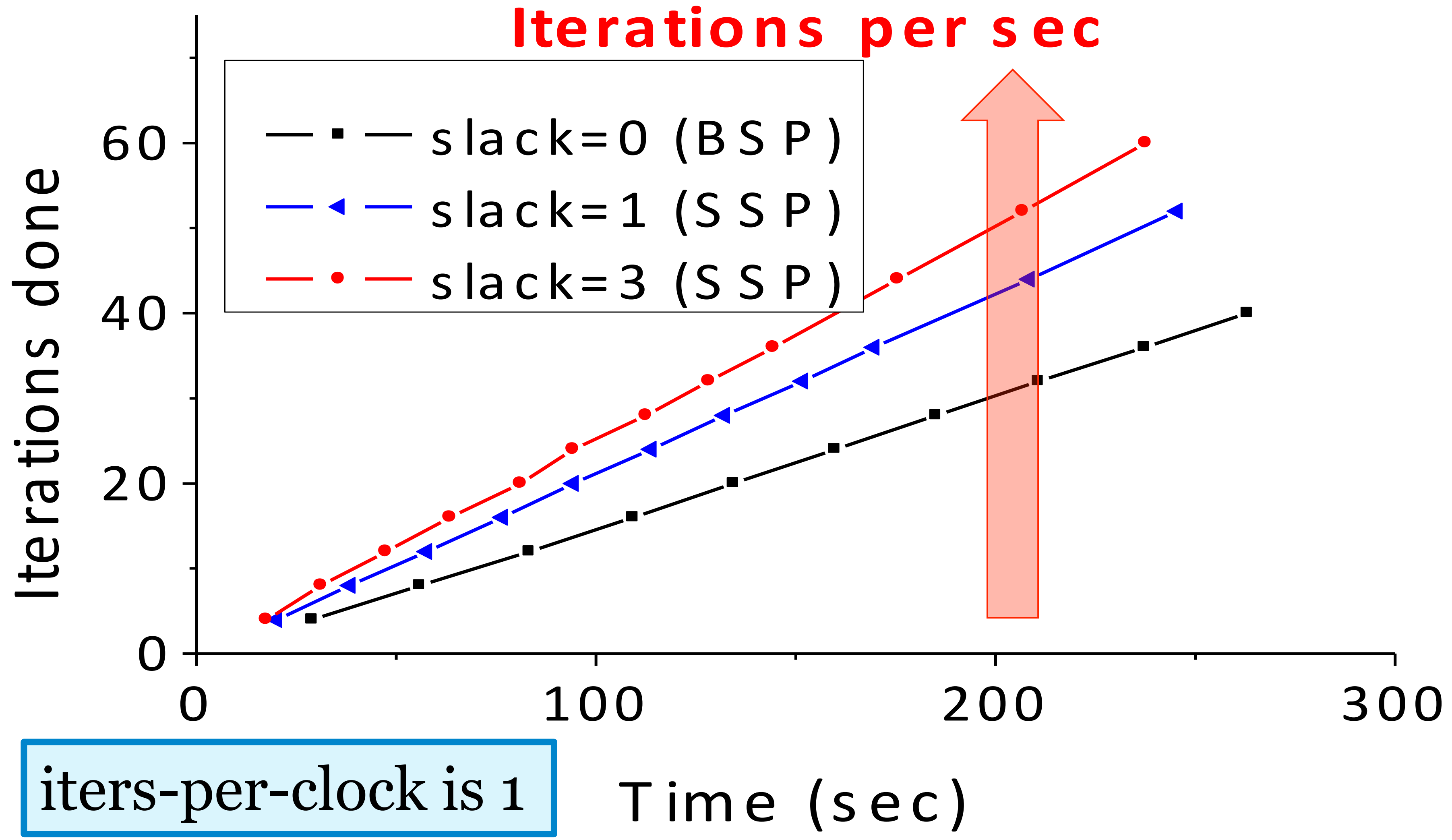
  - A-BSP is SSP with a slack of zero

Application Benchmark Example

- 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

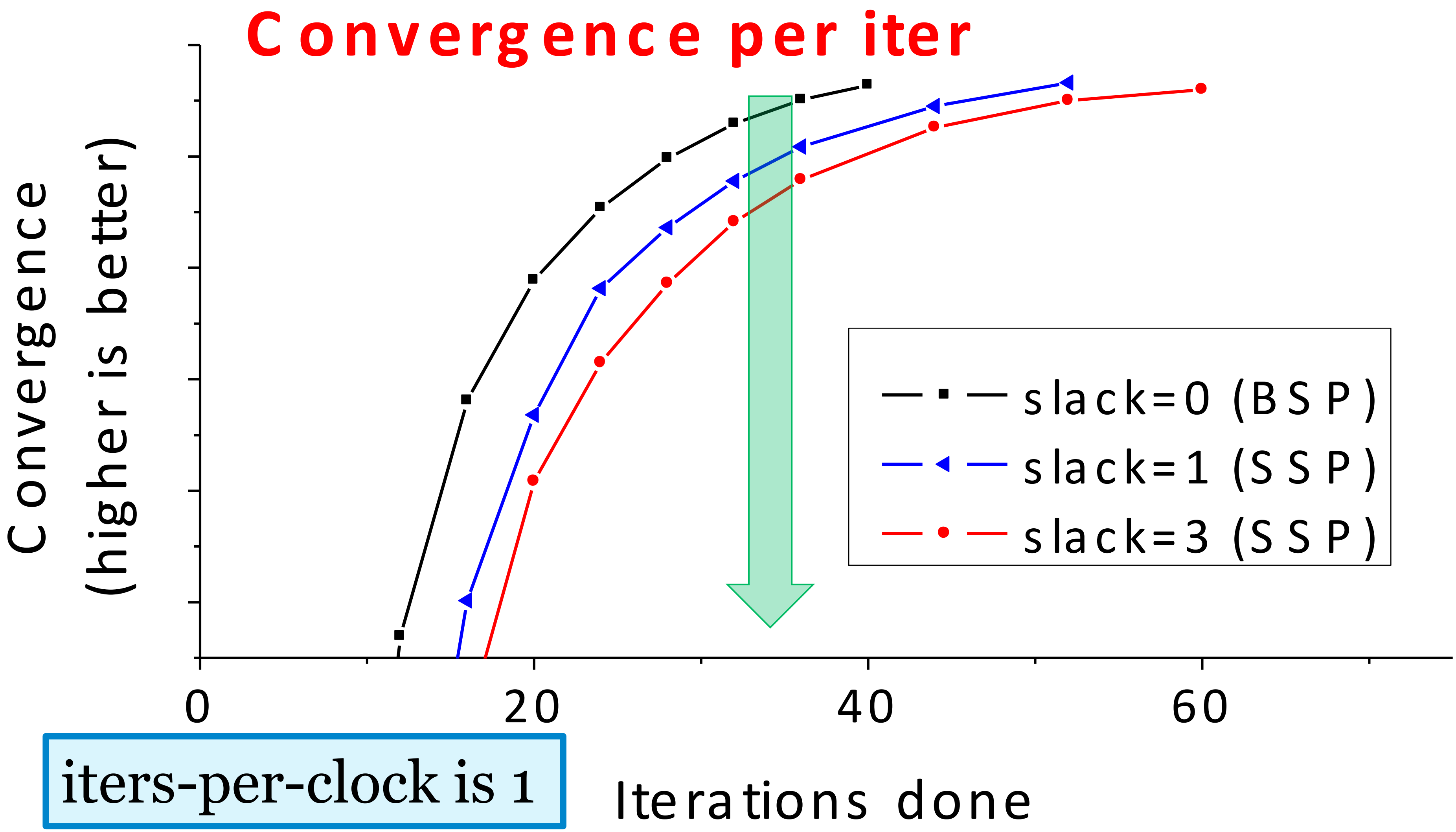
- Hardware information
  - 8 machines, each with 64 cores & 128GB RAM
- Basic configuration
  - One client & tablet server per machine
  - One computation thread per core

Staleness Increases Iters/sec



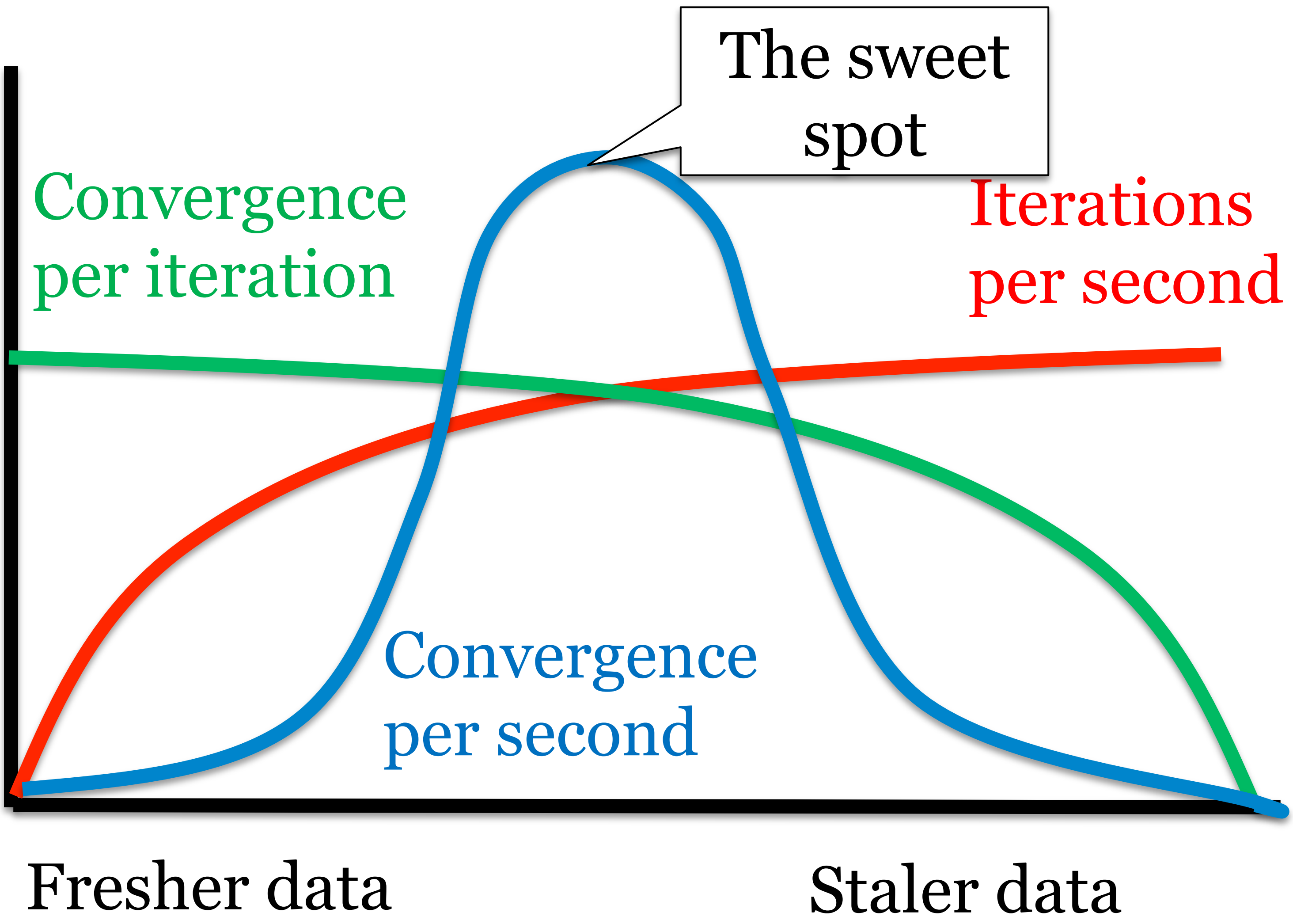
iters-per-clock is 1

Staleness Reduces Converge/iteration



iters-per-clock is 1

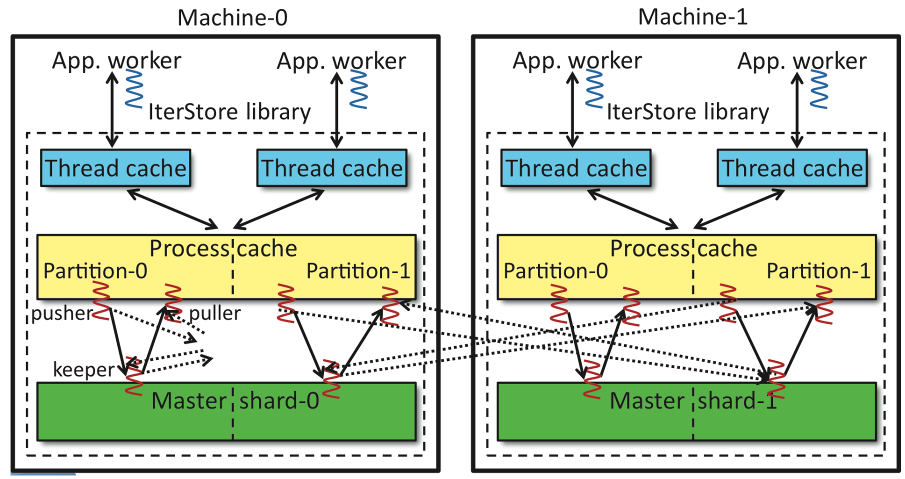
Key Takeaway Insight



[ATC'14]

Apply Systems Experience to BSP

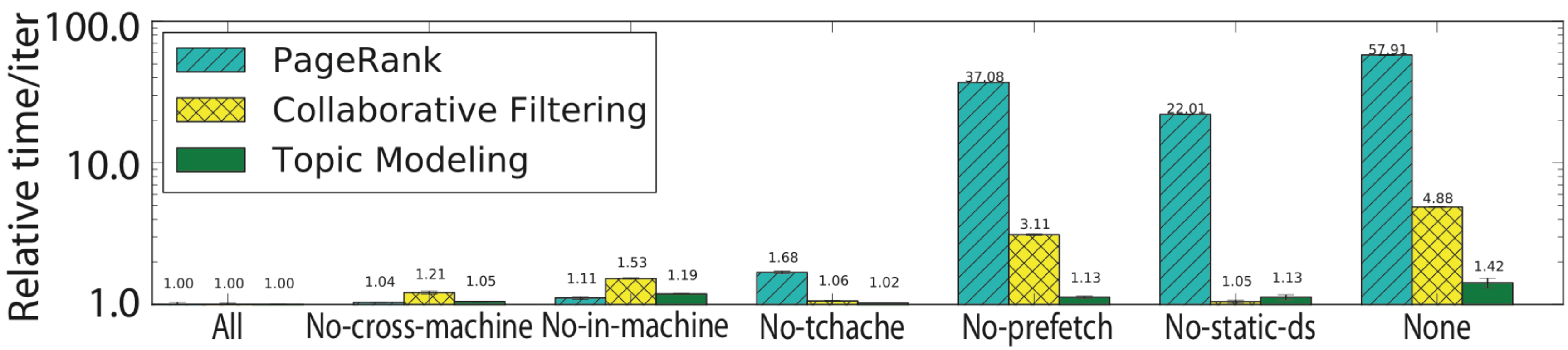
- Iterative code often very repetitive – exploit!
  - Virtual iteration
- Affinity allocation, static & precomputed policies, multiple levels of cache, update prefetching
-



[under submission]

Lead: Henggang Cui

- **Optimization effectiveness break-down:**



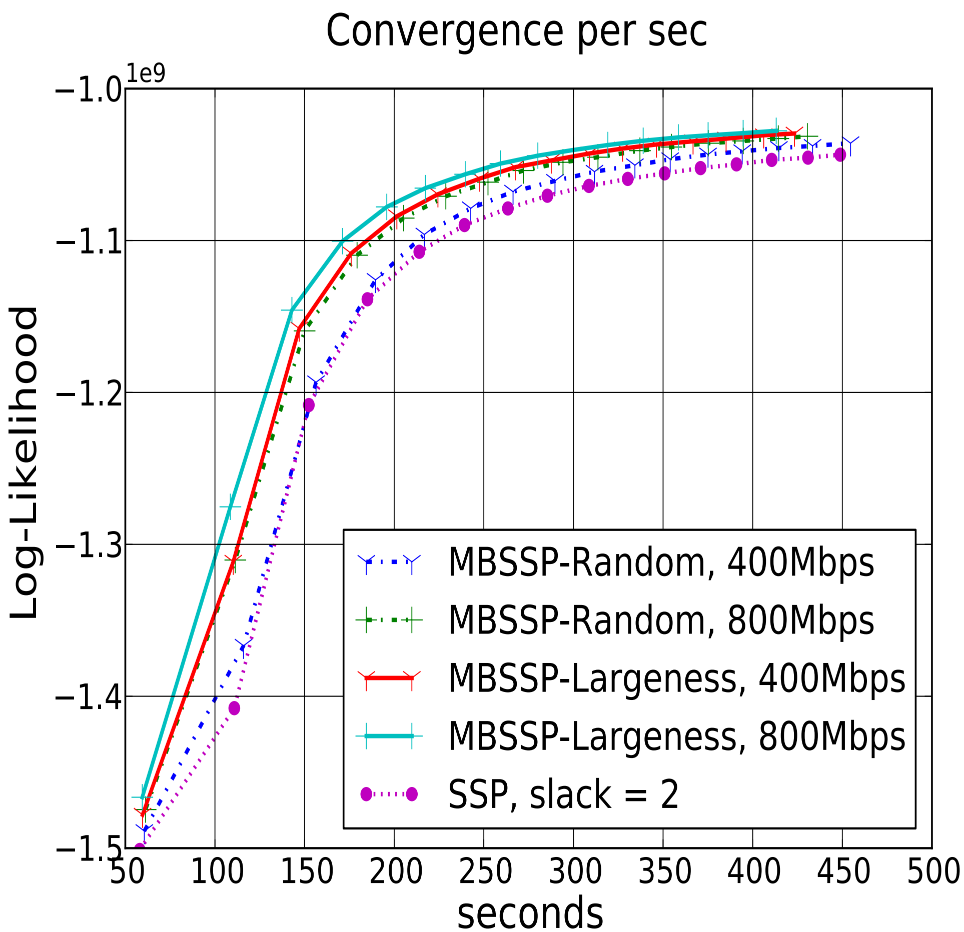
Managed Bandwidth SSP (MBSSP)

- 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

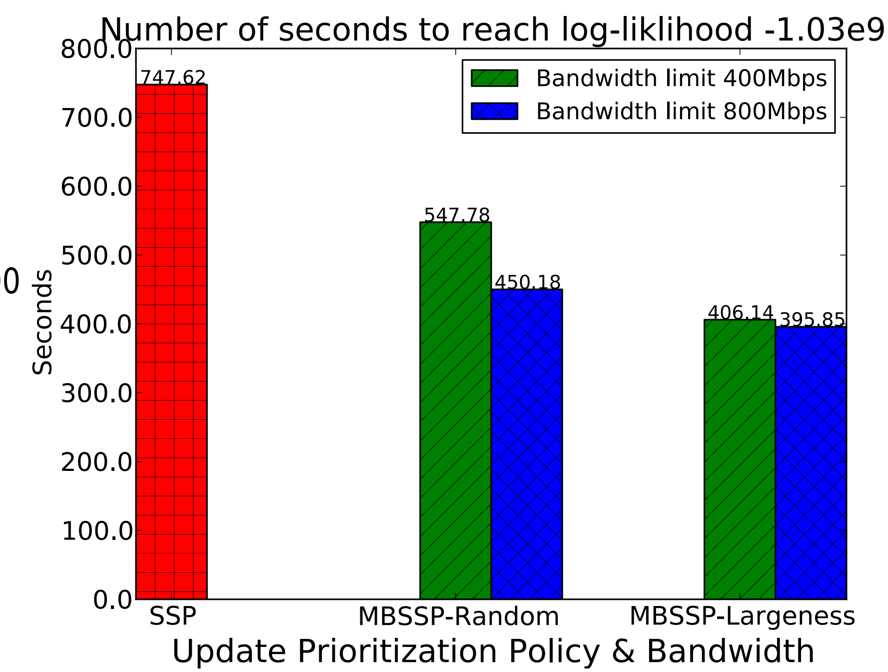
[under submission]

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

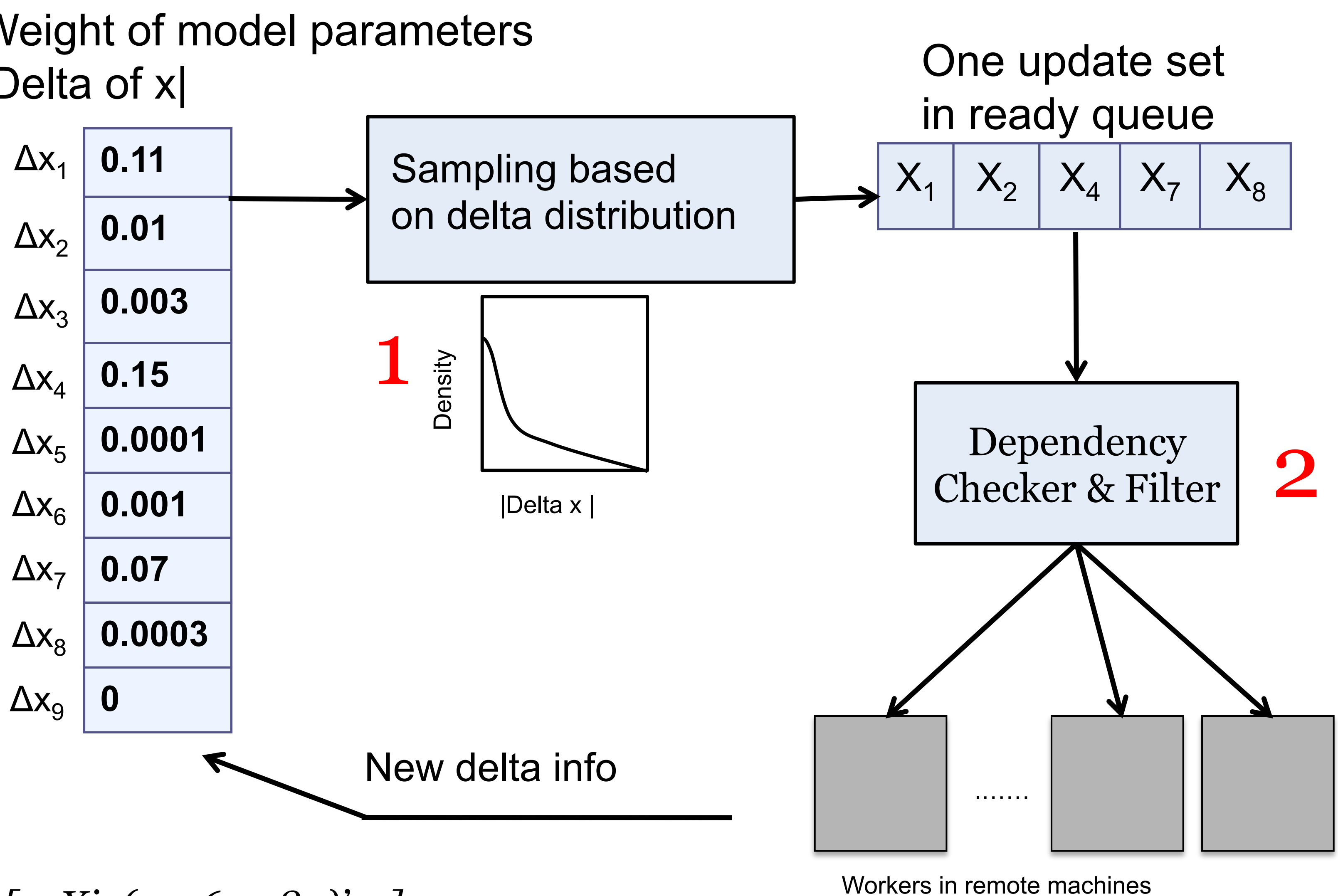
MBSSP Vision

- 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

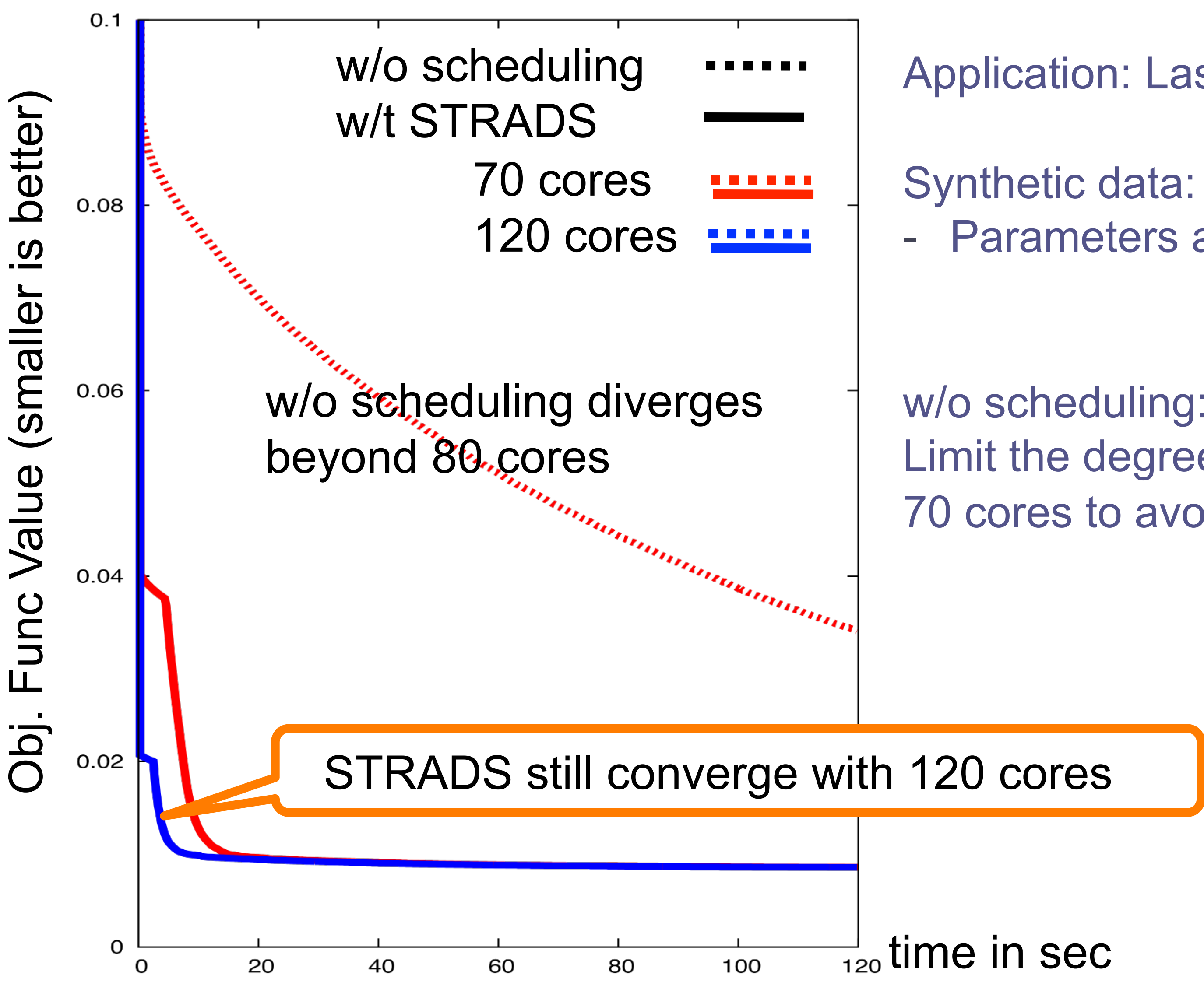
- Uniform parameter update is not optimal
  - Use deeper knowledge of ML algorithms to update parameters at different rates for best convergence speed (like MBSSP)
- Random parameter selection for parallel update risks divergence (e.g. Shotgun Lasso)
  - Control errors when selecting parameters to update in parallel
- Leads: Jin Kyu Kim, Seunghak Lee

STRADS: Two Scheduling Policies



[arXiv(1406.4580)'14]

Benefits of Two Scheduling Policies



Application: Lasso

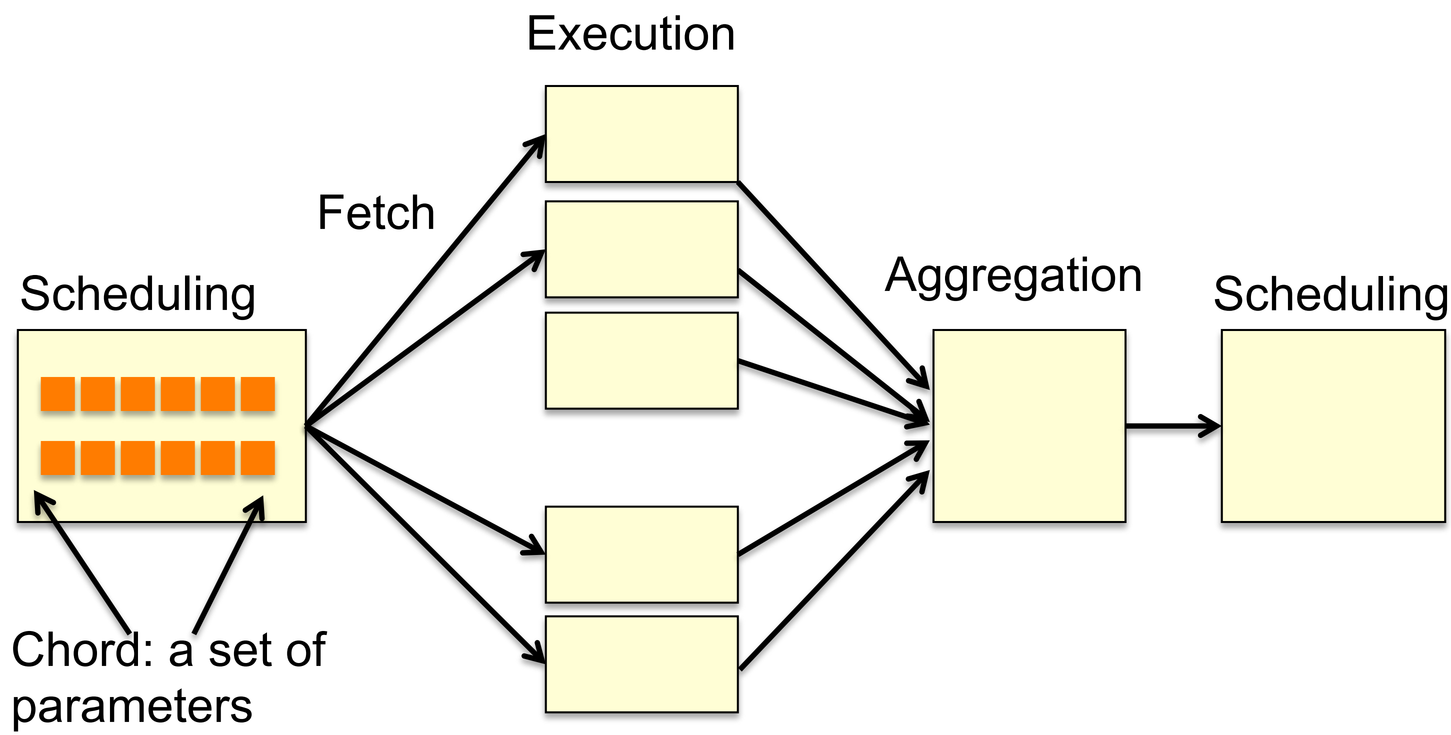
Synthetic data: 450 by 100K

- Parameters are highly correlated.

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

ML Iterative Solver Execution Model

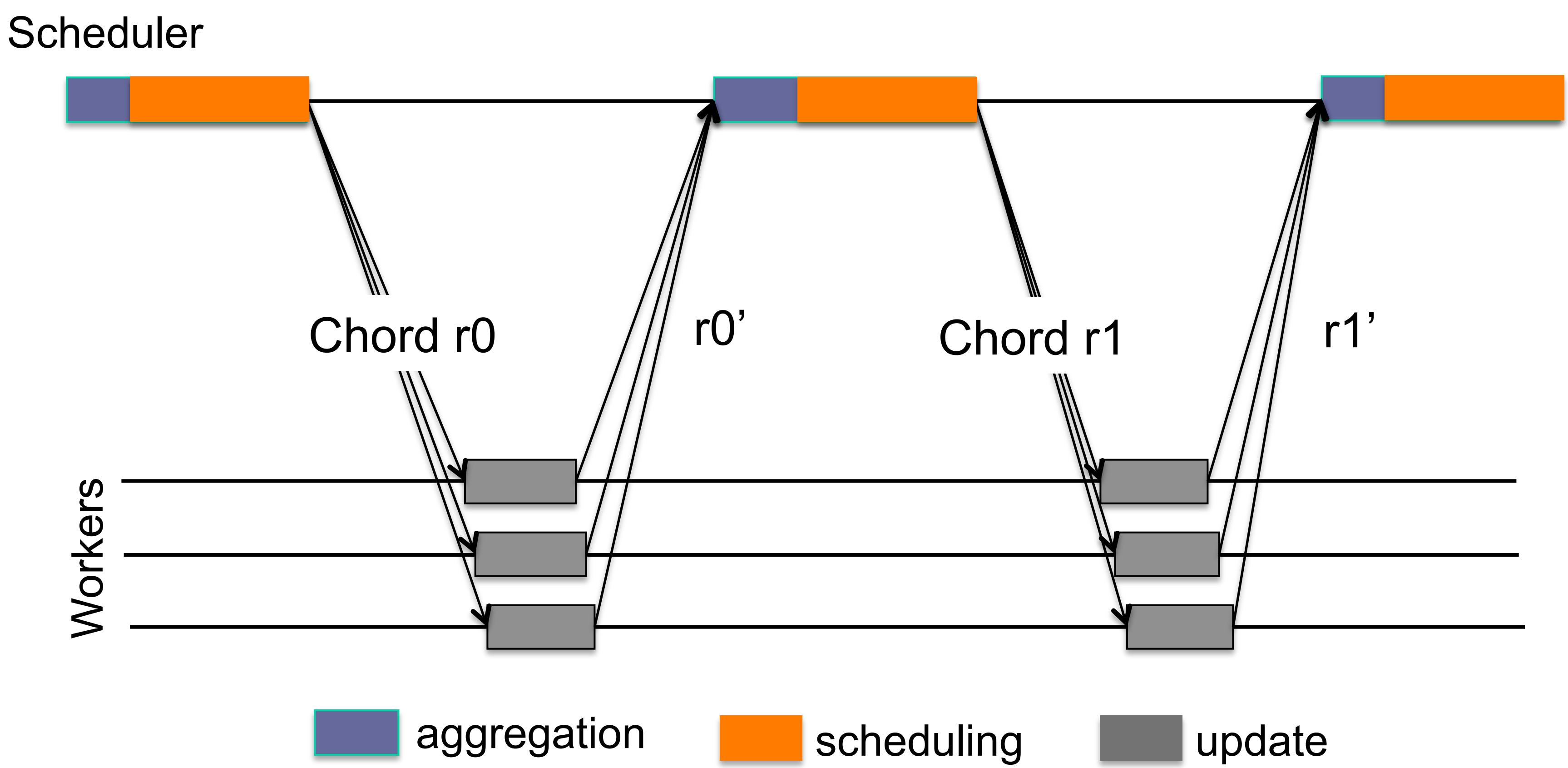
Scheduling/Fetch/Execution/Aggregation model



- 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

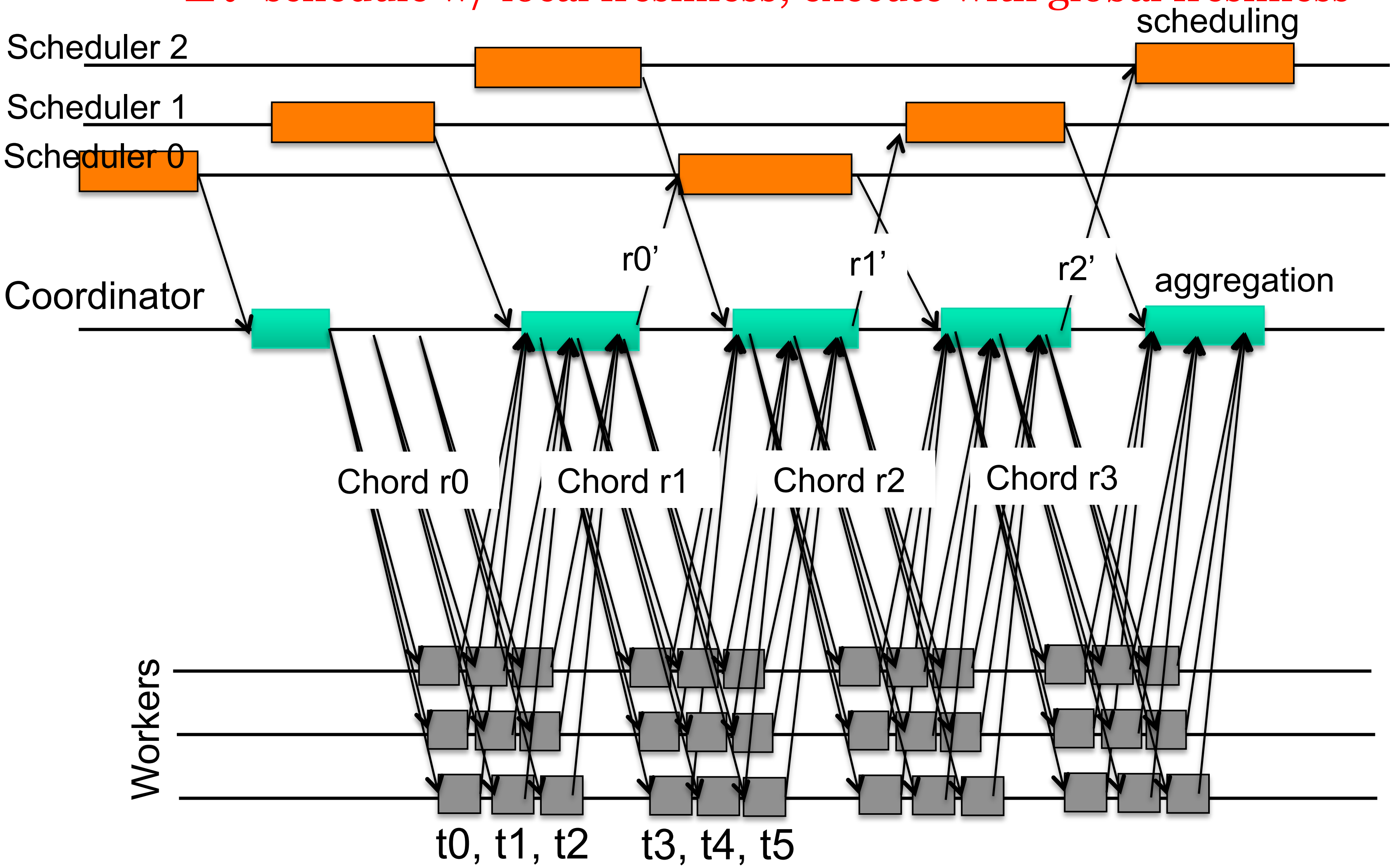
Serial execution of chords is a performance bottleneck



Approach: Make scheduling decisions with latest data only for the scheduler's partition of the (big) model parameters

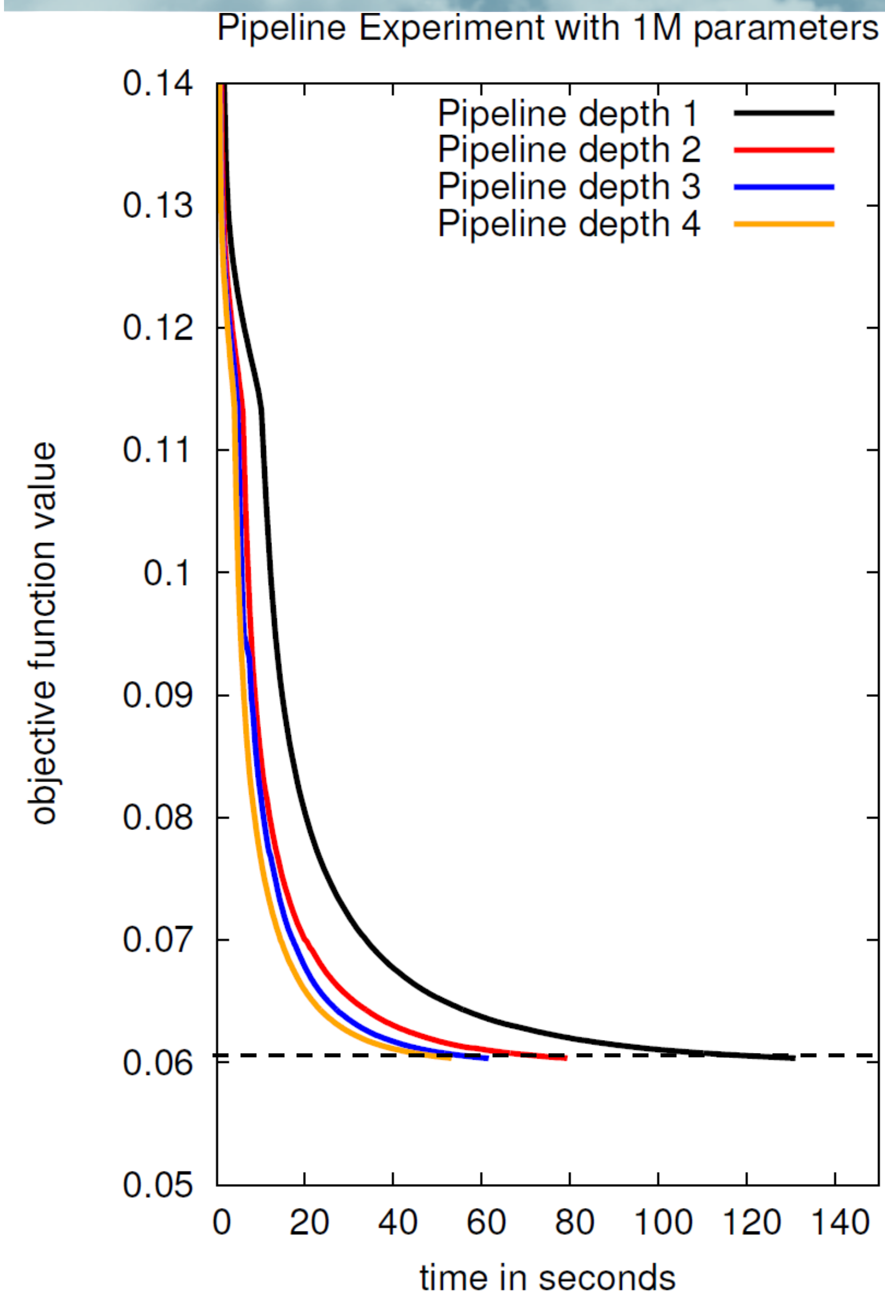
One pipeline is not enough

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



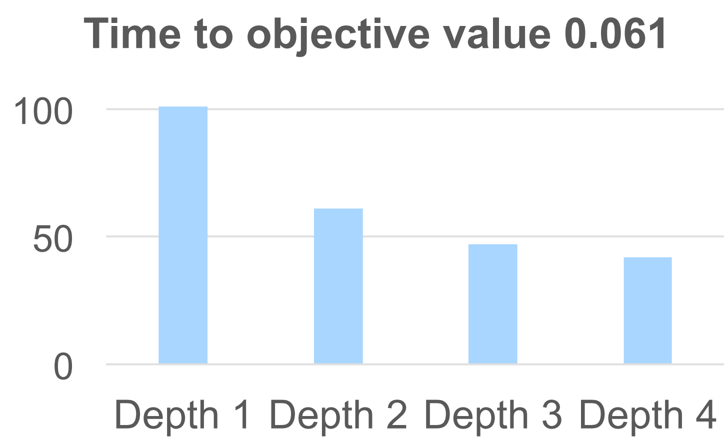
2: relax freshness of least important updates (relative to next Chord)

STRADS Dual Pipeline Convergence



Application: Lasso
Data: Synthetic data
50K samples, 1M dimension

Depth refers to second pipeline



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.

Closing: Bound Staleness Project Suite

- 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

Contributing Students

- 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