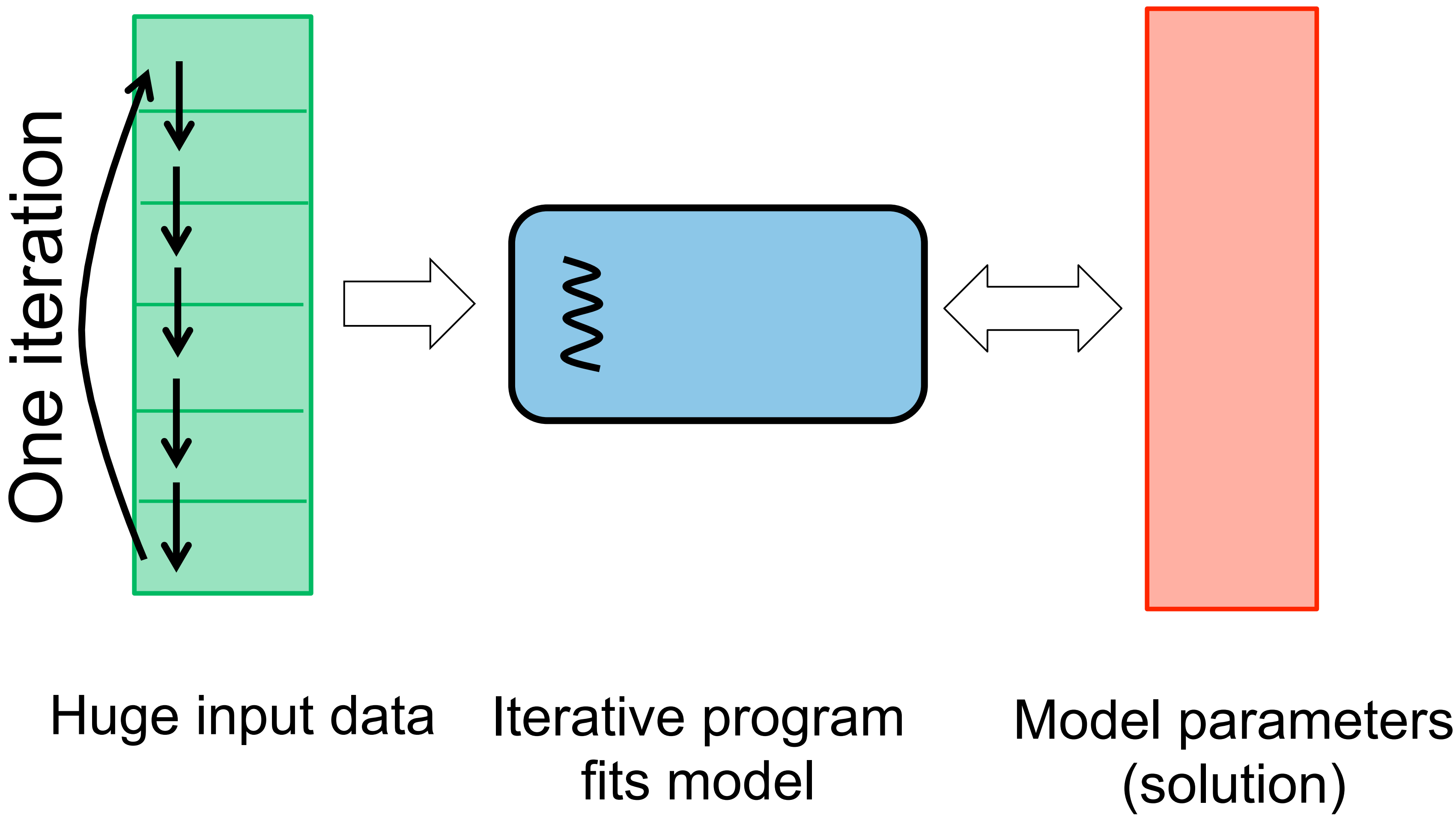
Exploiting Bounded Staleness to Speed up Big Data Analytics

Henggang Cui

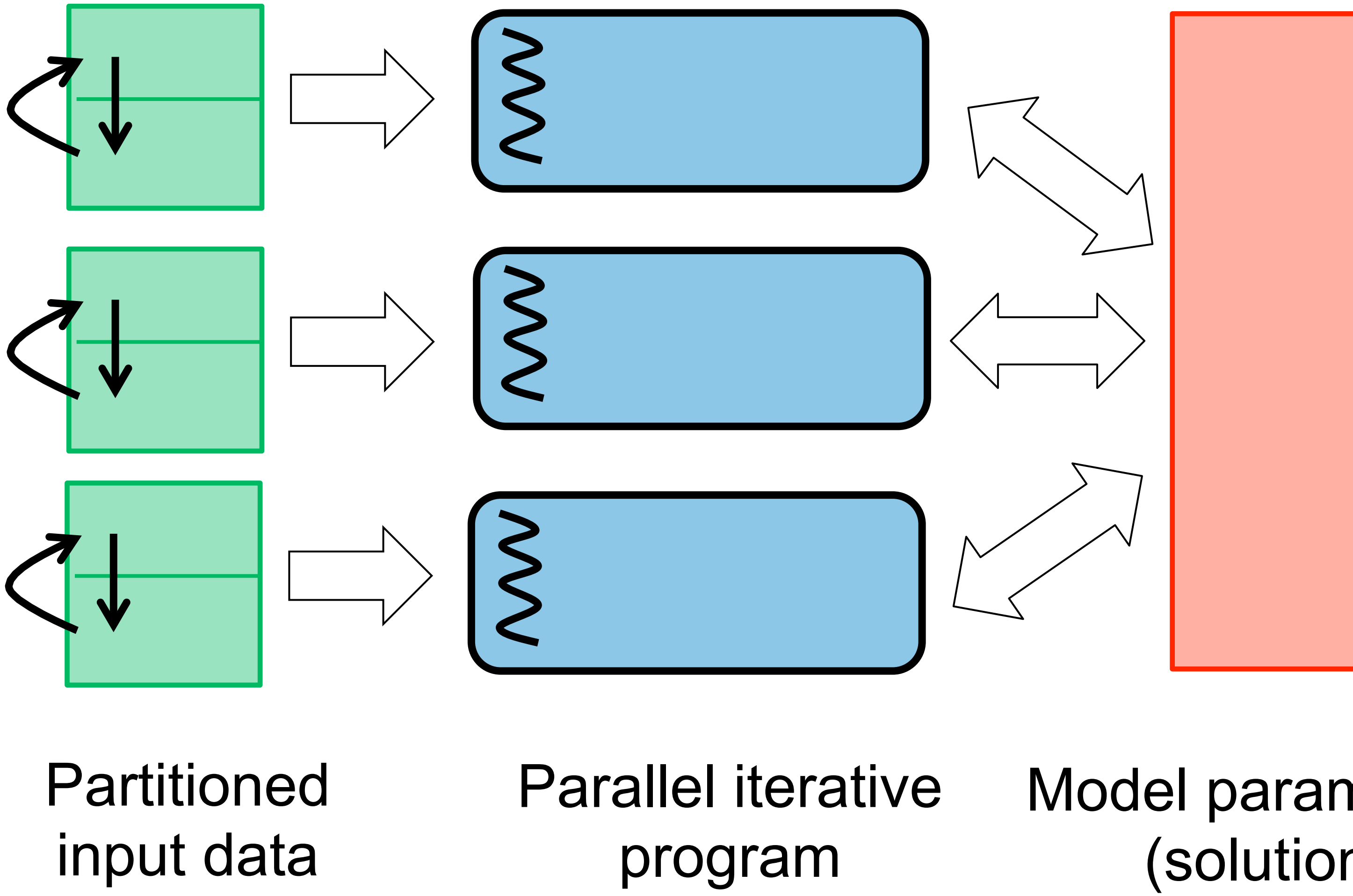
James Cipar, Qirong Ho, Jin Kyu Kim,
Seunghak Lee, Abhimanu Kumar, Jinliang Wei, Wei Dai,
Greg Ganger, Phil Gibbons (Intel), Garth Gibson, and Eric Xing

Carnegie Mellon University

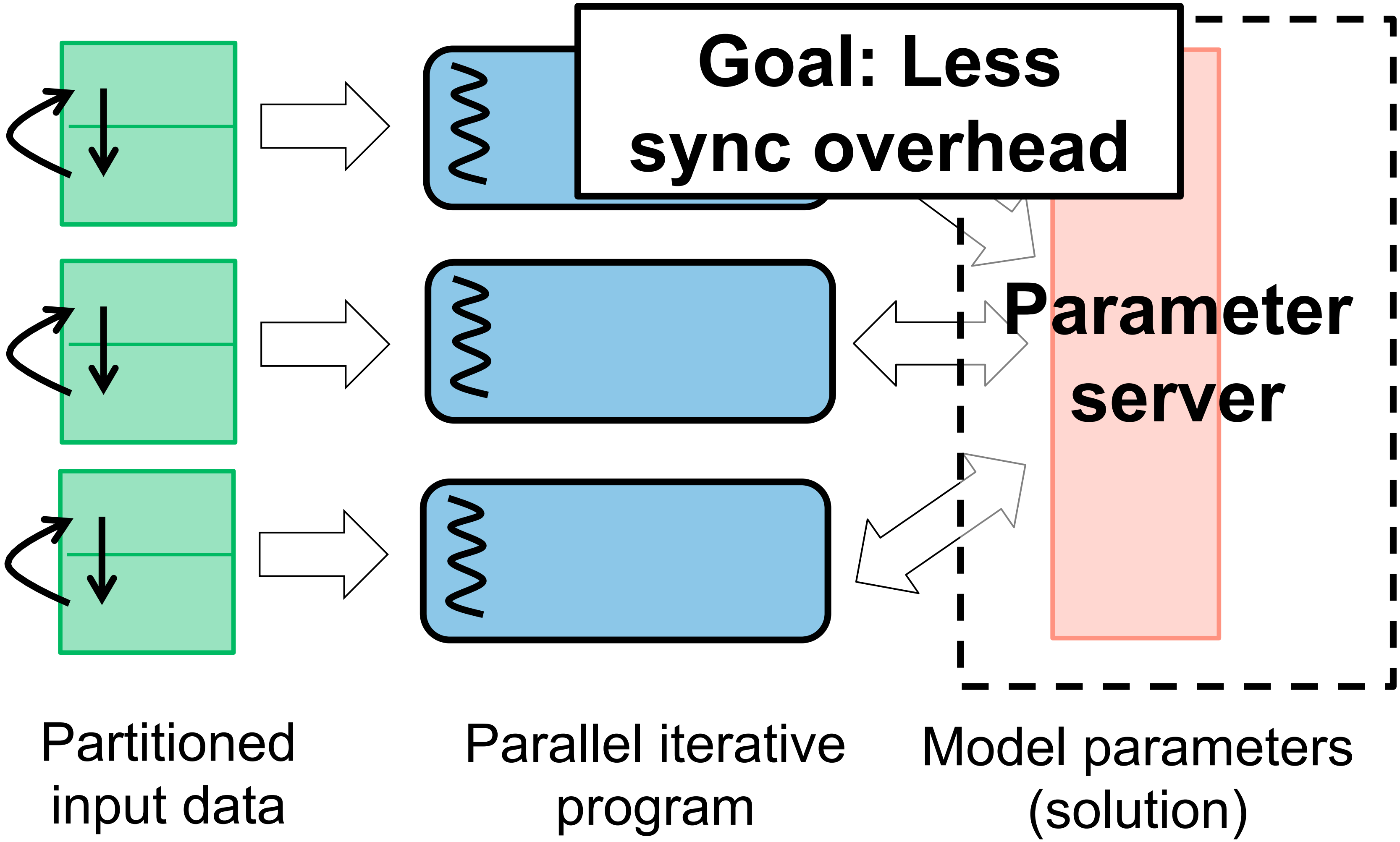
Big Data Analytics Overview



Big Data Analytics Overview



Big Data Analytics Overview



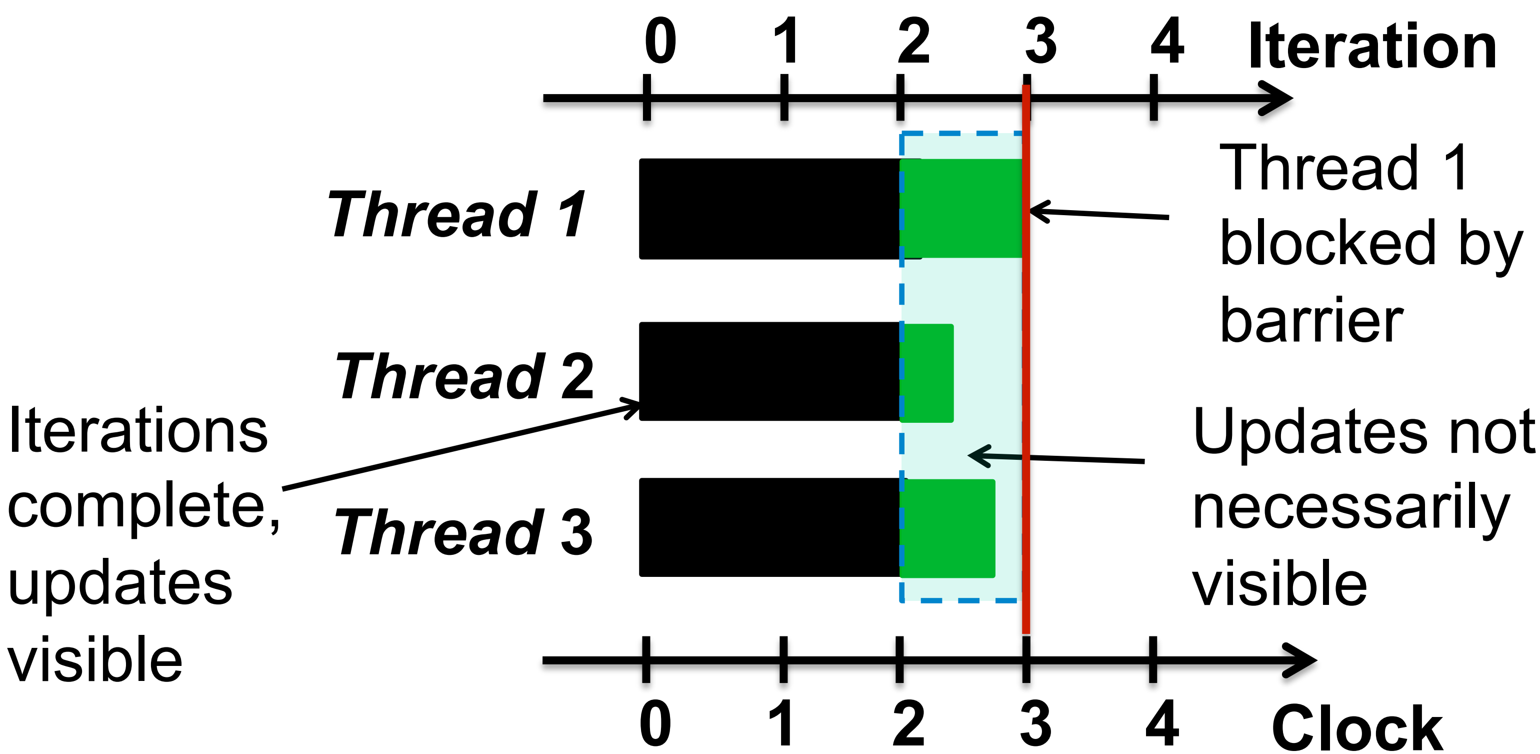
Outline

- 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

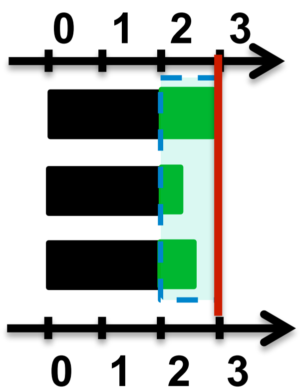
- A barrier every **clock** (a.k.a. epoch)
  - In ML apps, often one iteration over input data

Thread progress illustration:



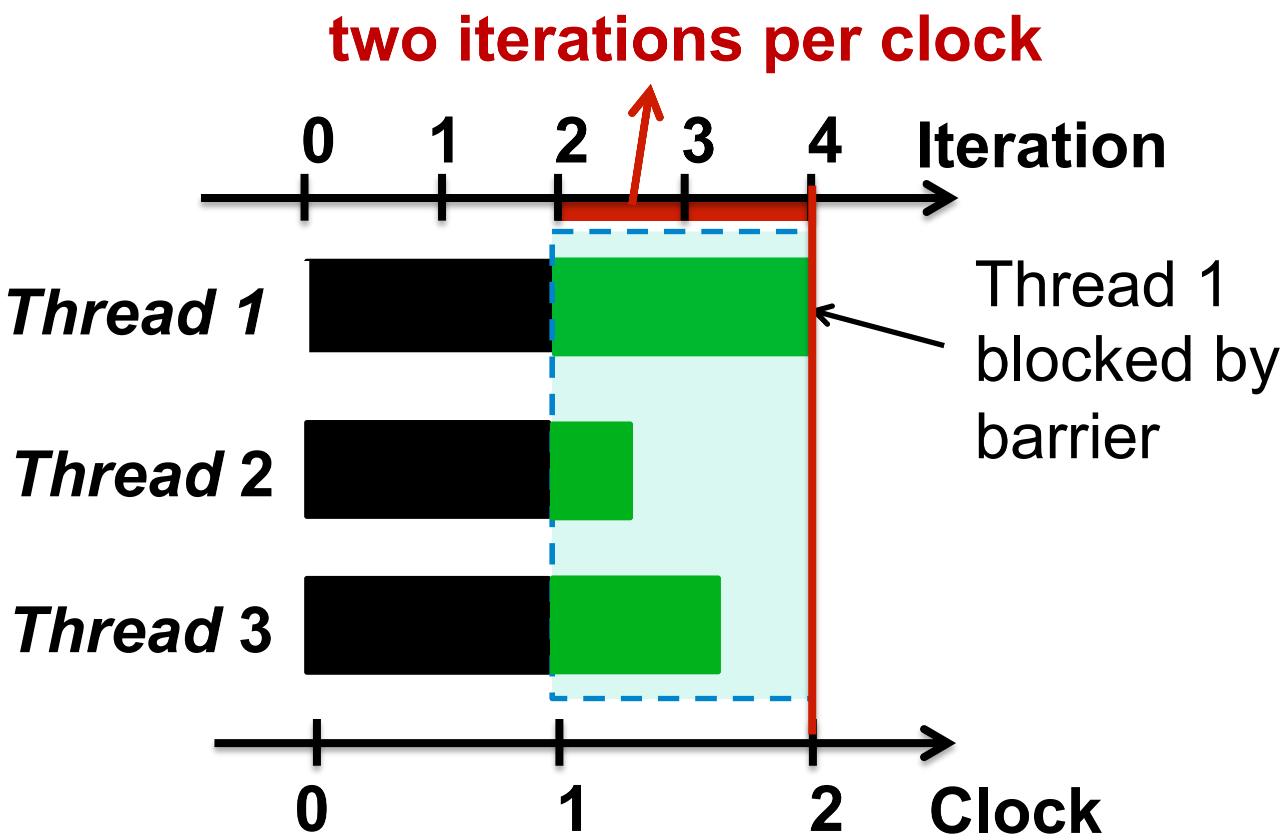
Data Staleness

- In BSP, threads can see "out-of-date" values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**



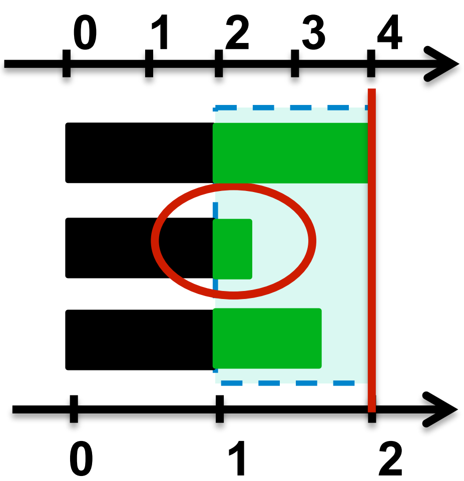
Arbitrarily-sized BSP (A-BSP)

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



Problem of (A-)BSP: Stragglers

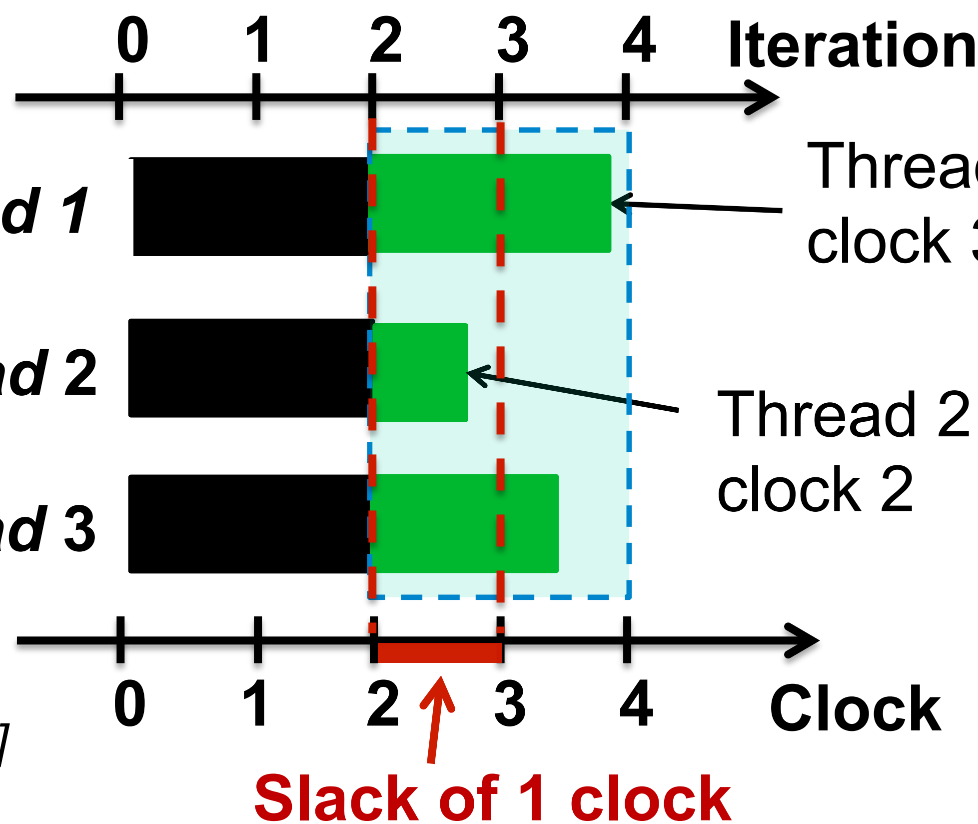
- A-BSP still has the straggler problem
  - A slow thread will slow down all
  - Stragglers are common in large systems



- 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

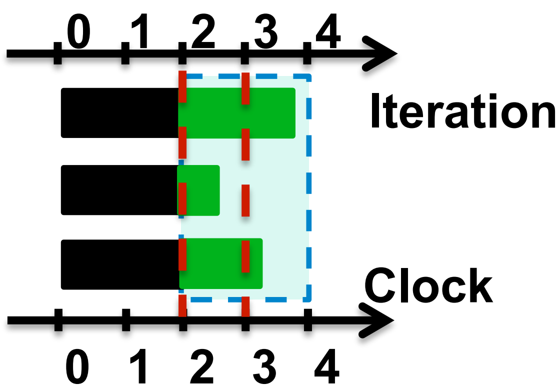


[HotOS'13, NIPS'13]

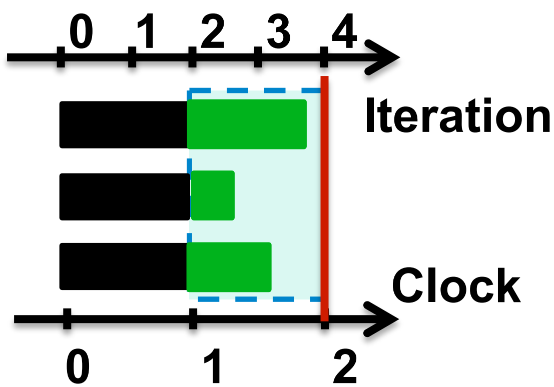
Two Dimensional Config. Space

- 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

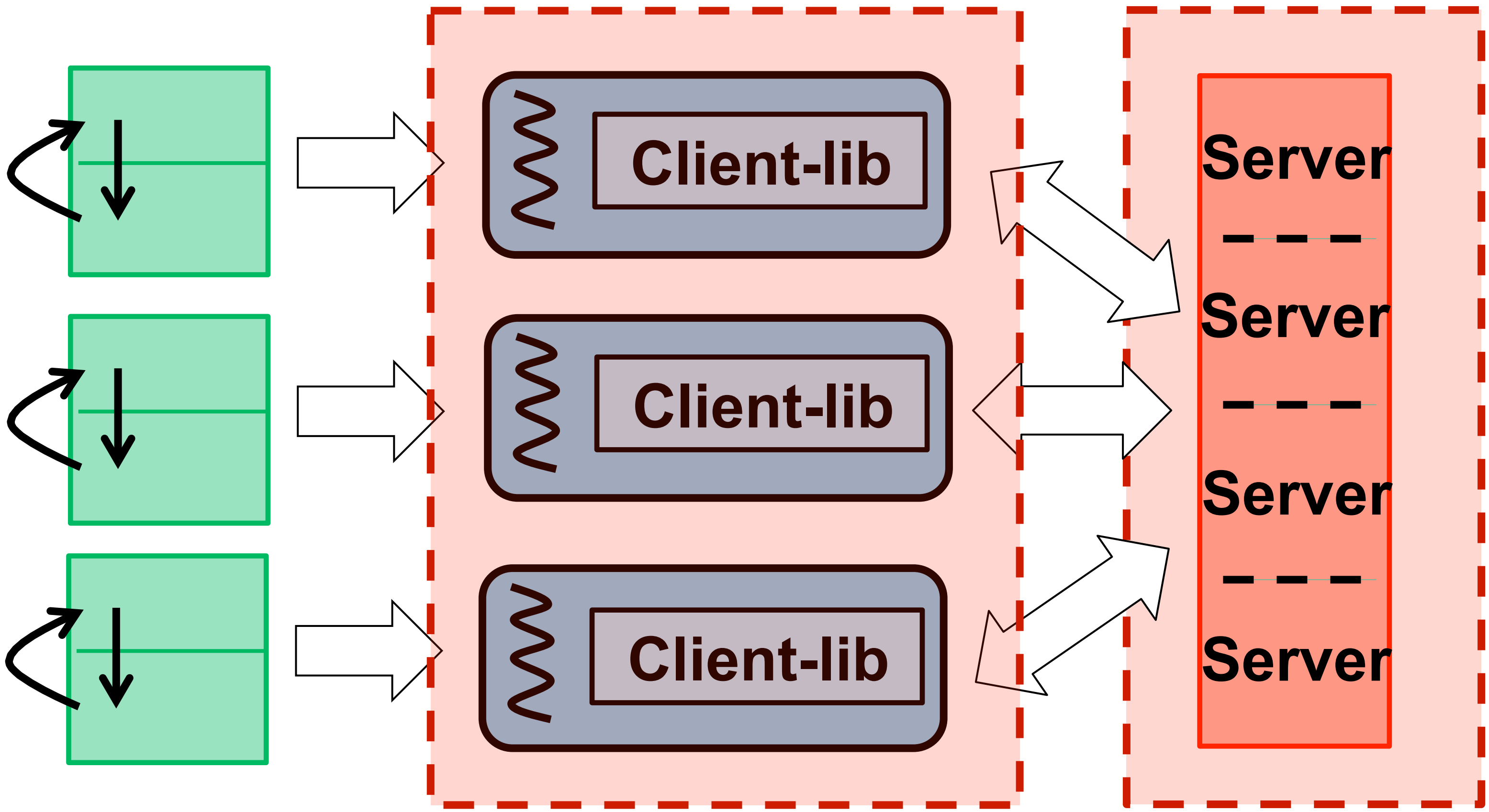
SSP (iters-per-clock=1, slack=1):



A-BSP (iters-per-clock=2, slack=0):



LazyTable Architecture

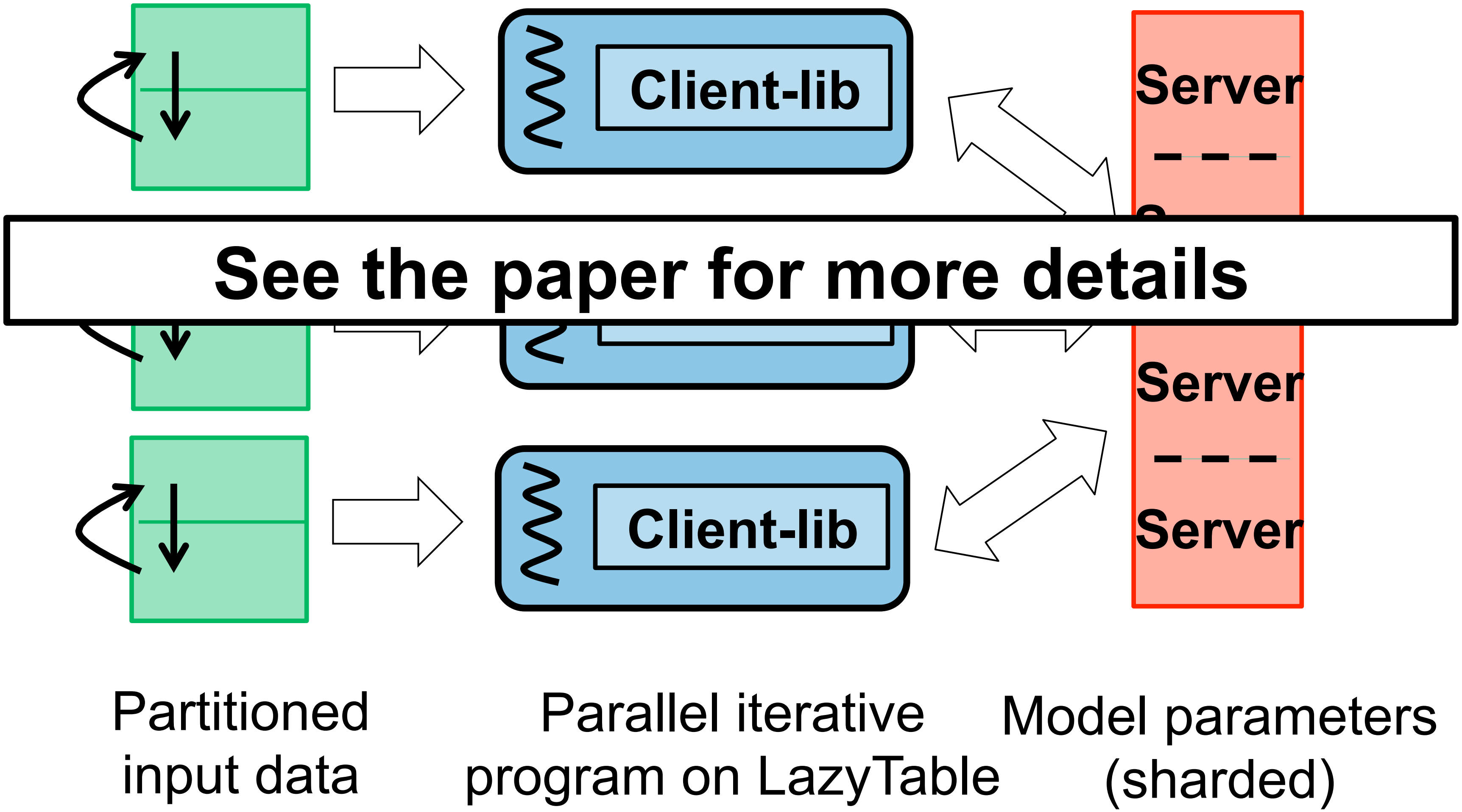


Partitioned input data

Parallel iterative program on LazyTable

Model parameters (sharded)

LazyTable Architecture



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

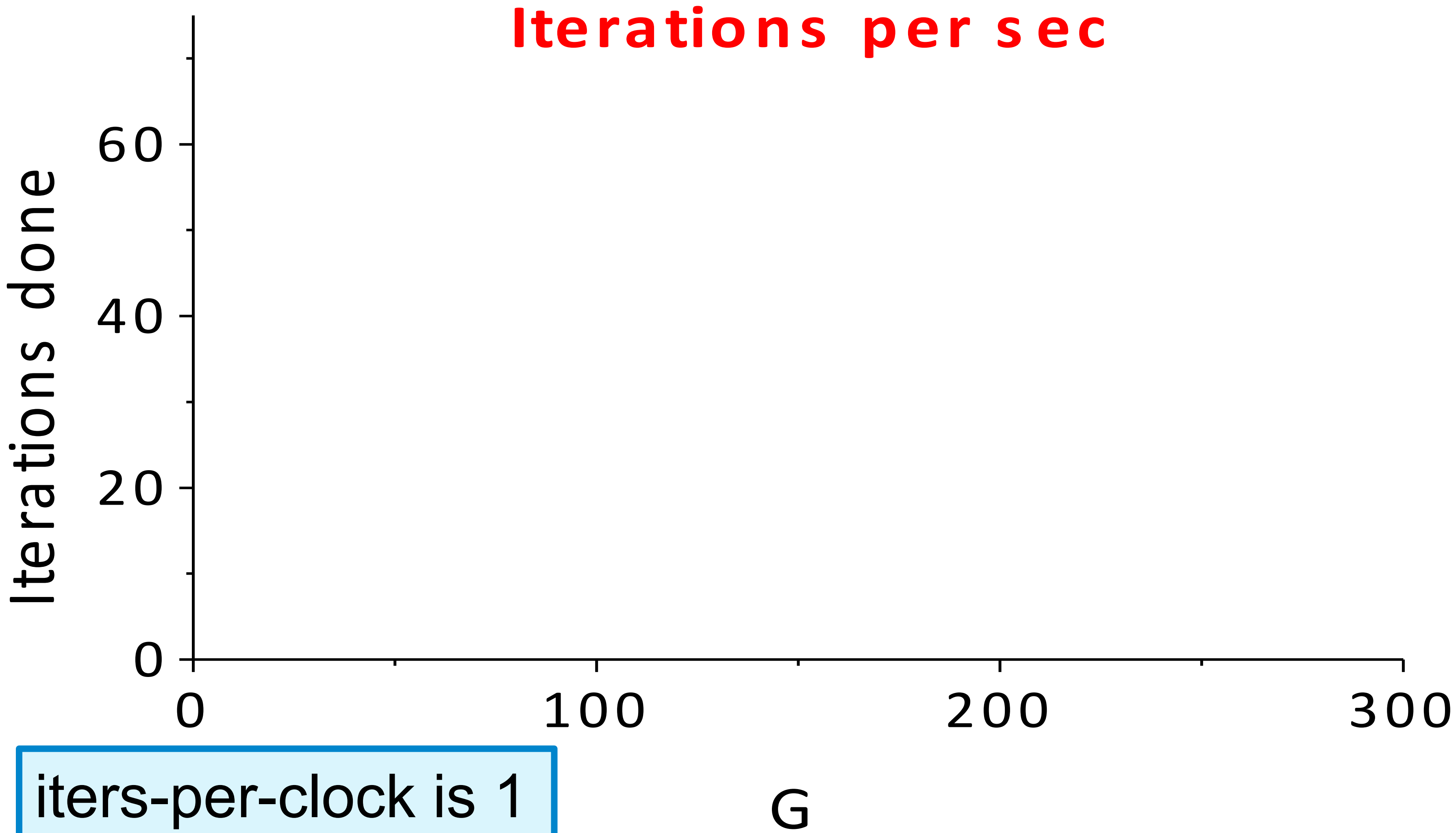
Controlling Data Staleness

- SSP
  - Larger slack -> more staleness

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

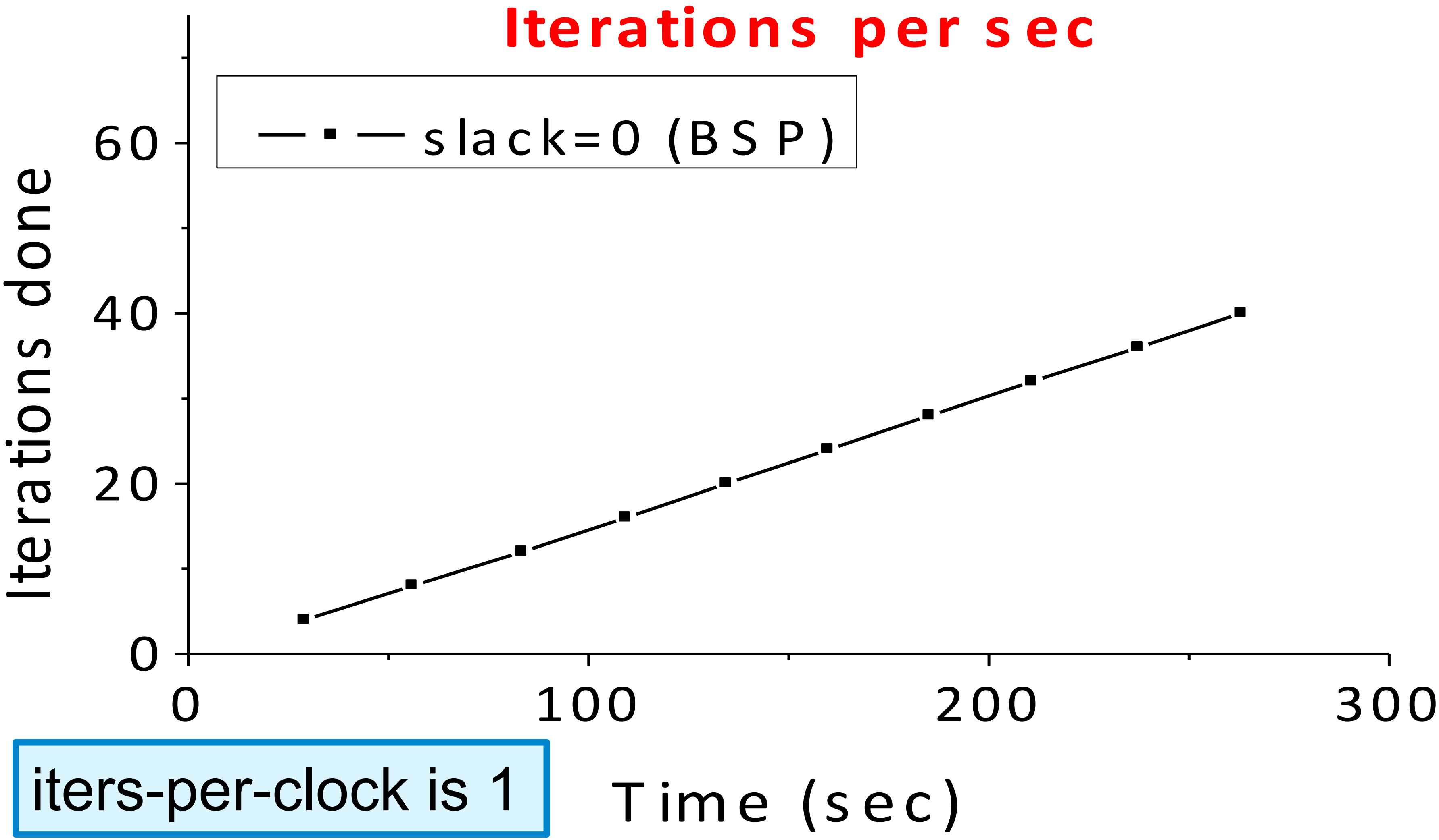
- The tradeoffs with increased staleness

Staleness Increases Iters/sec



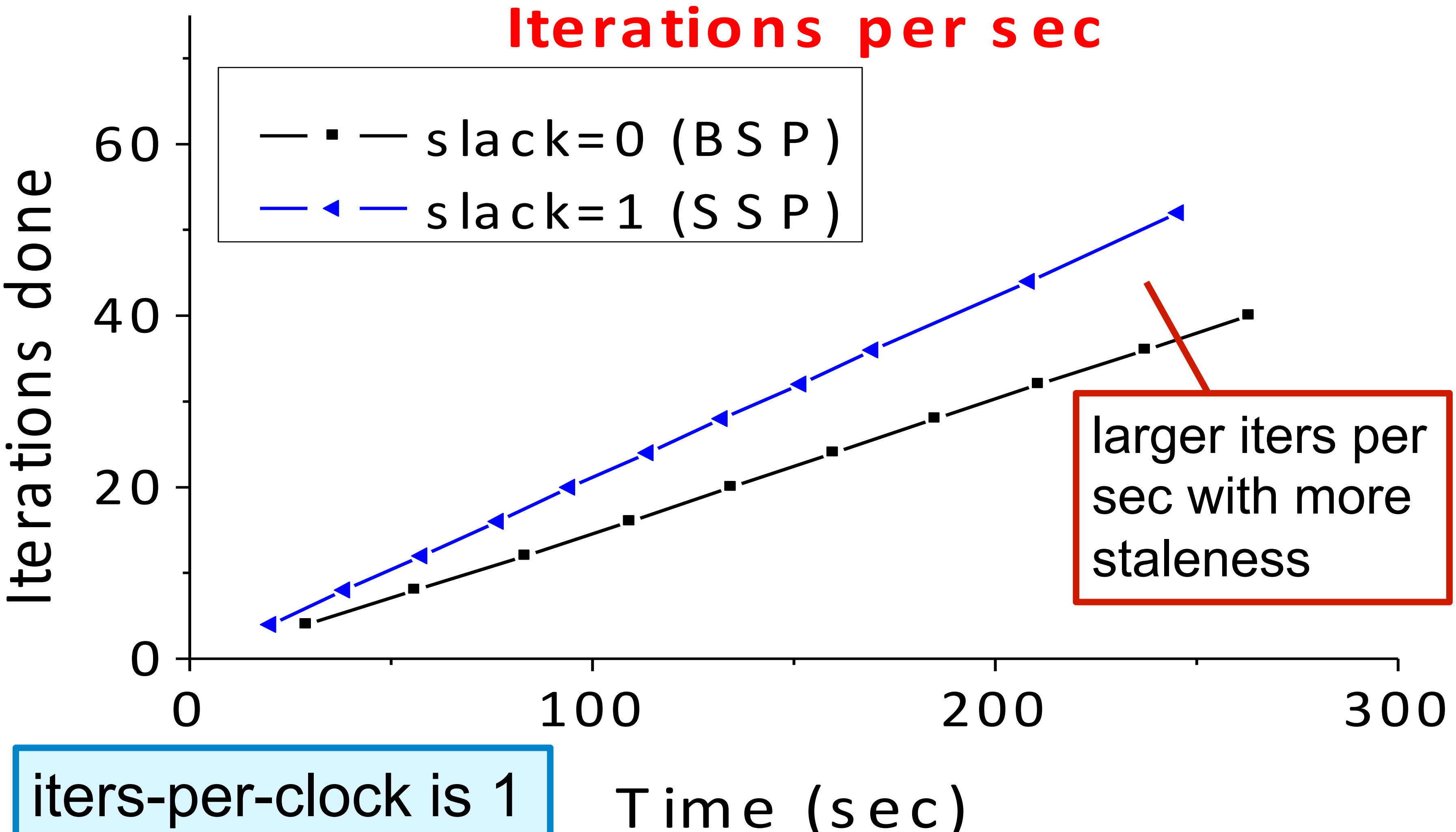
iters-per-clock is 1

Staleness Increases Iters/sec



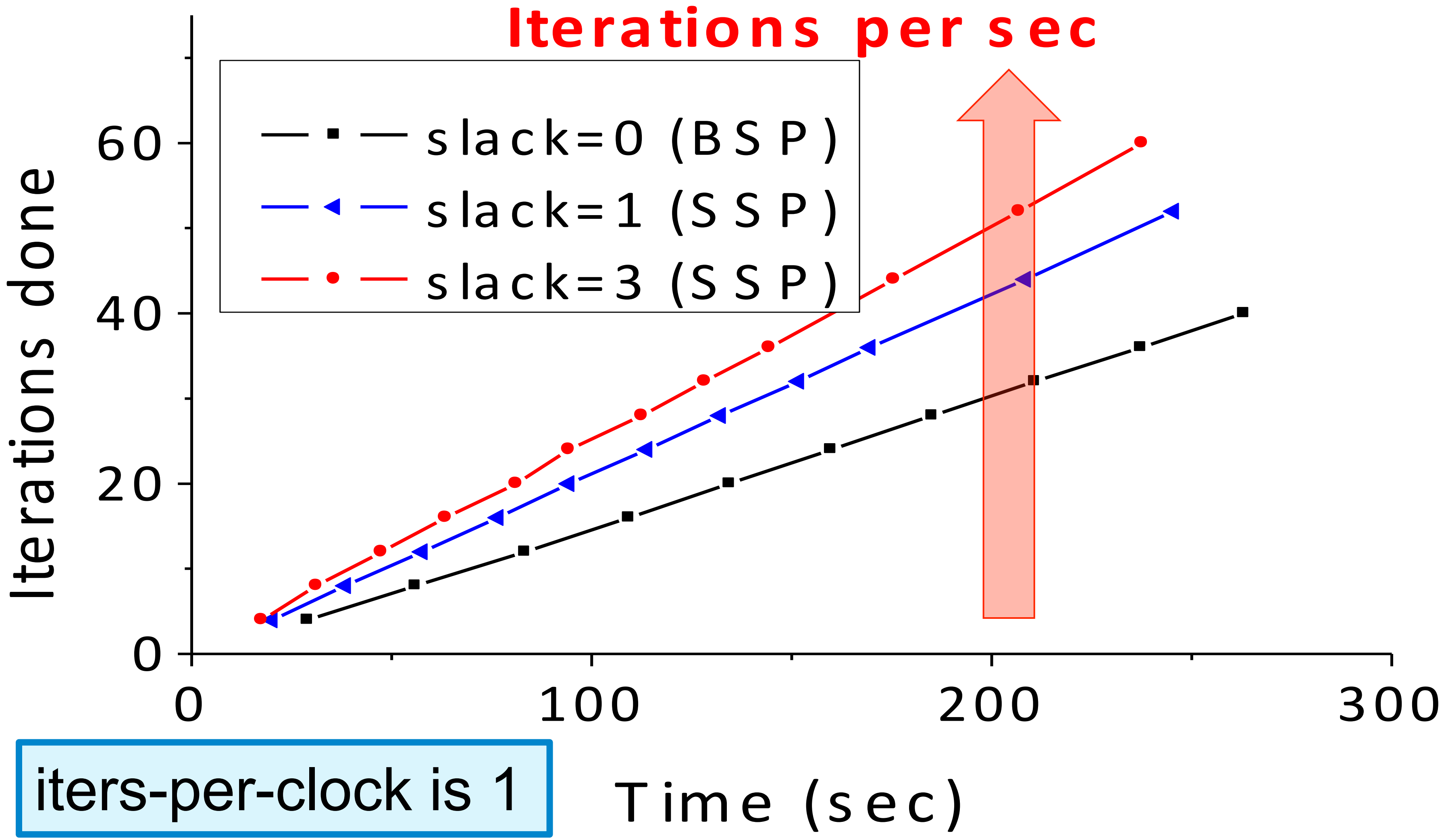
iters-per-clock is 1

Staleness Increases Iters/sec



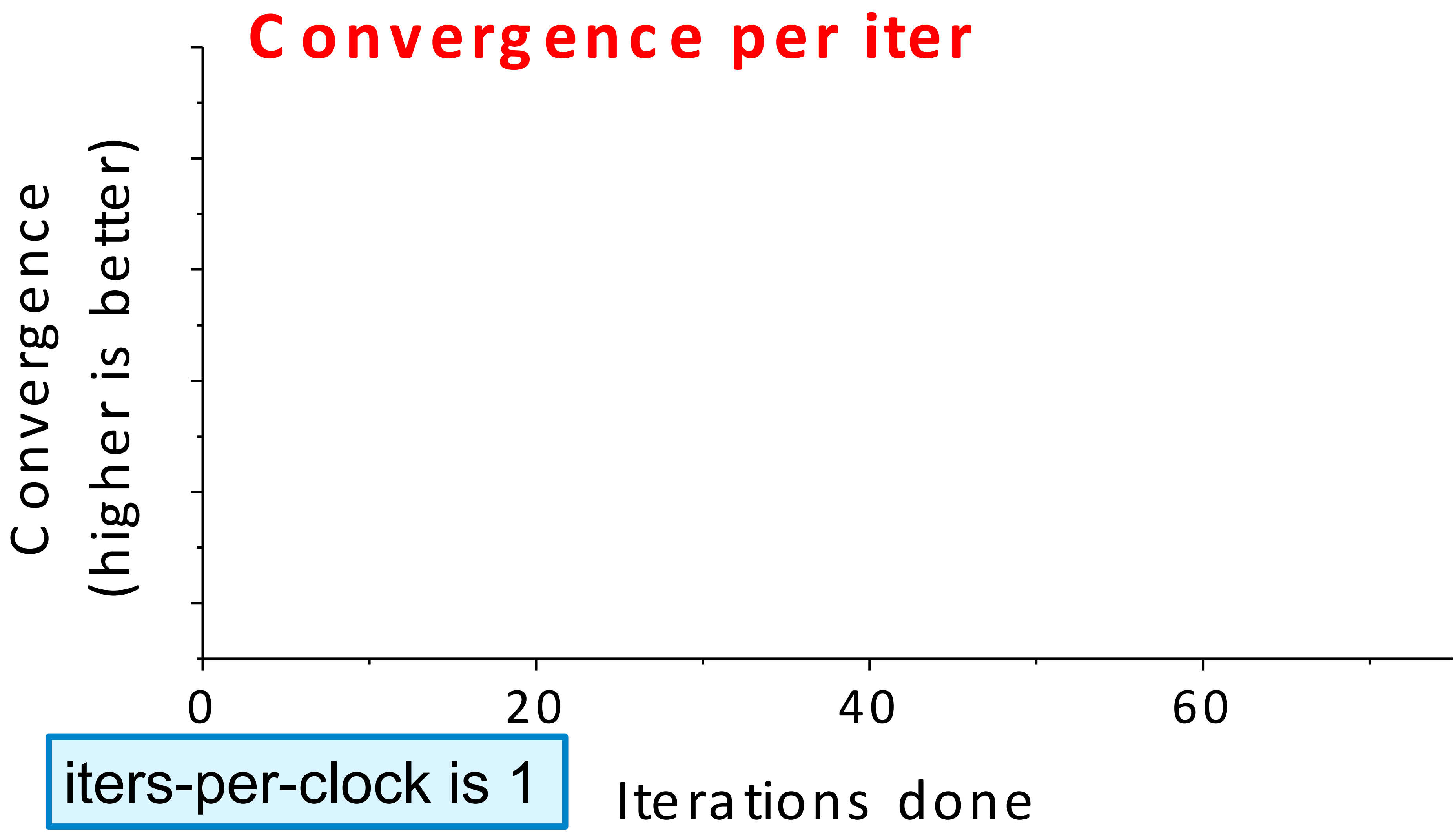
iters-per-clock is 1

Staleness Increases Iters/sec



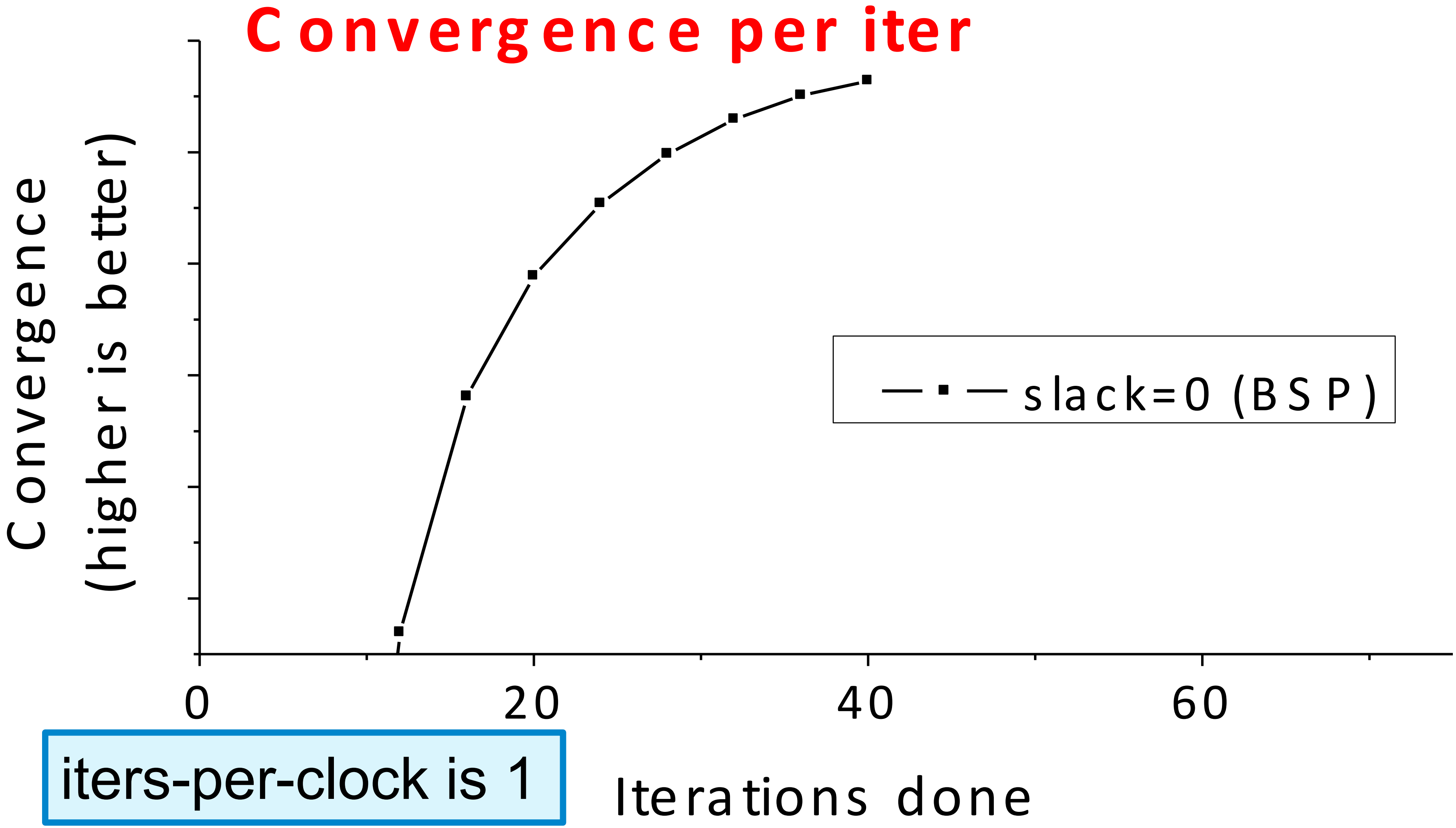
iters-per-clock is 1

Staleness Reduces Converge/iter



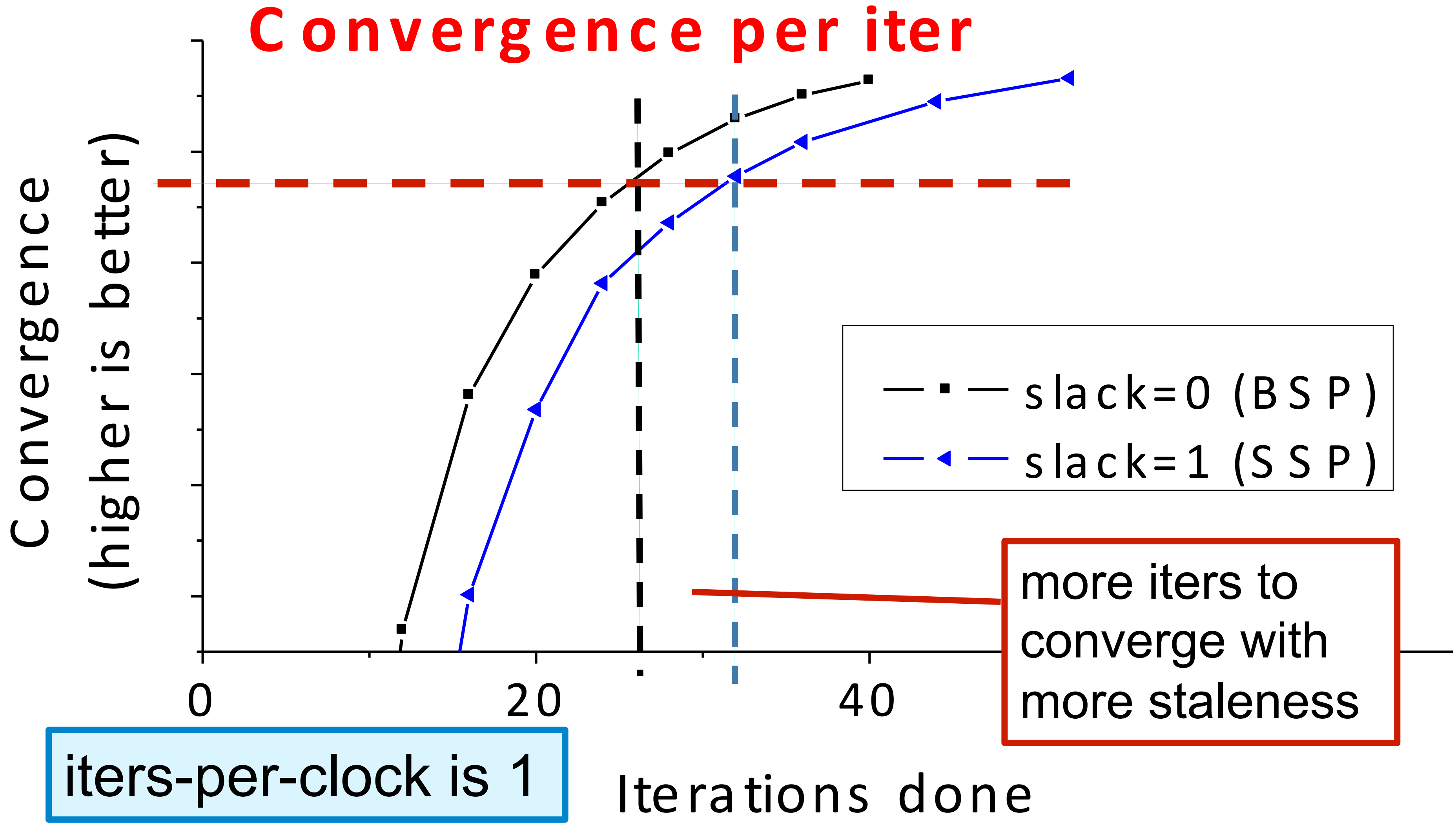
iters-per-clock is 1

Staleness Reduces Converge/iter



iters-per-clock is 1

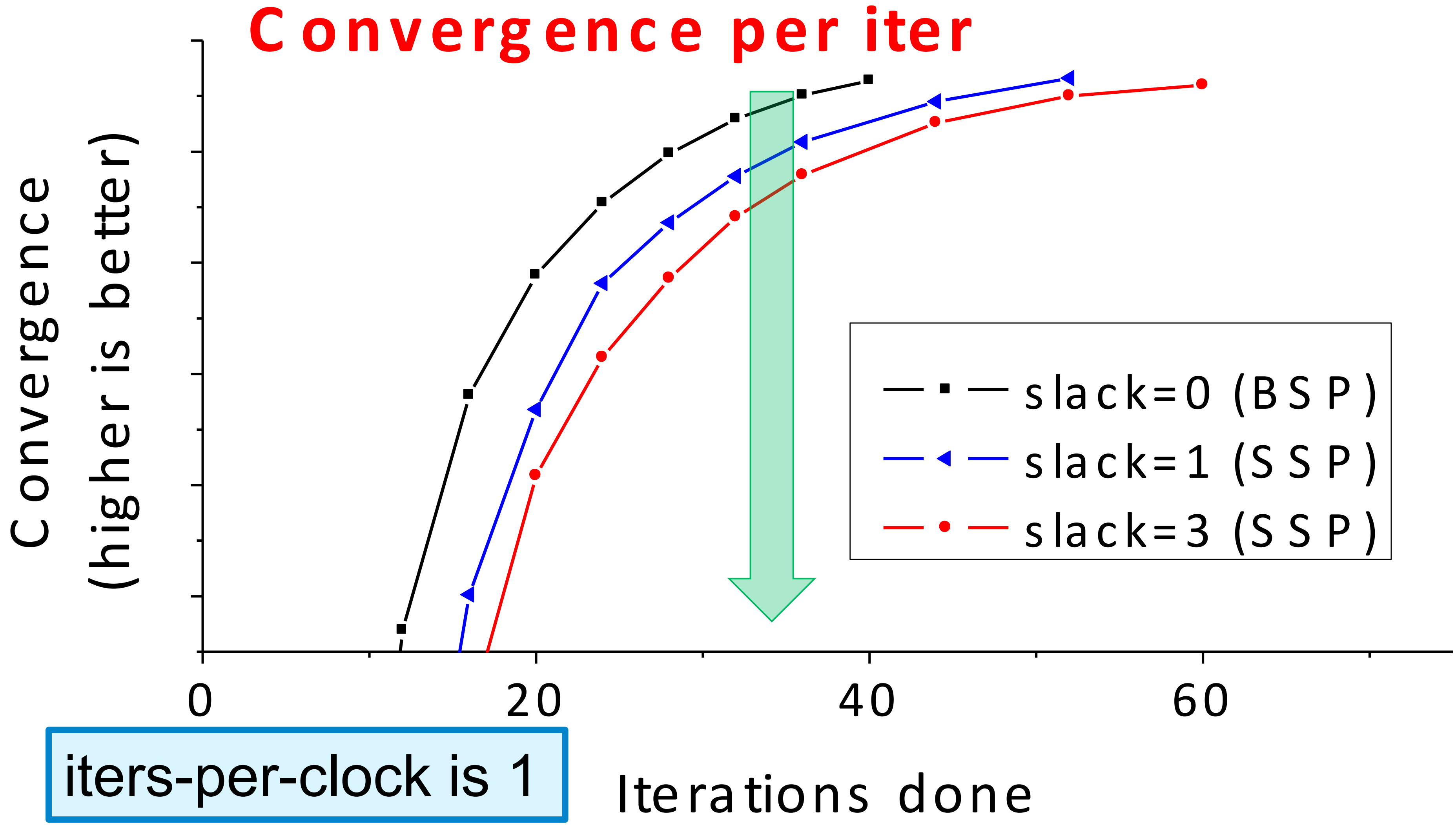
Staleness Reduces Converge/iter



more iters to converge with more staleness

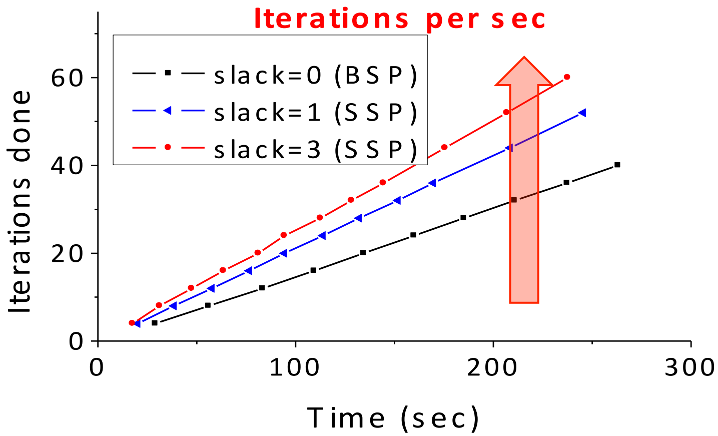
iters-per-clock is 1

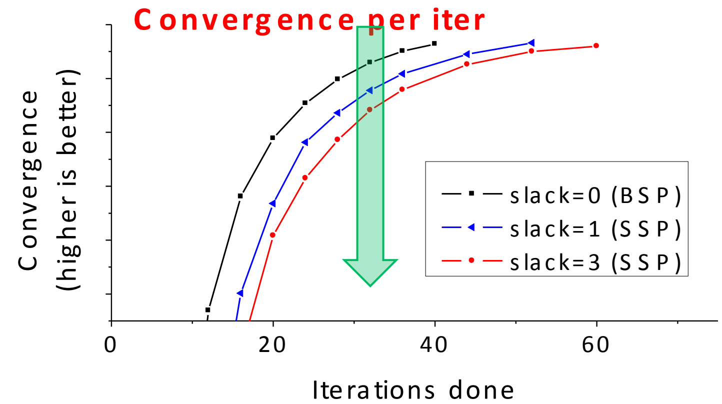
Staleness Reduces Converge/iter

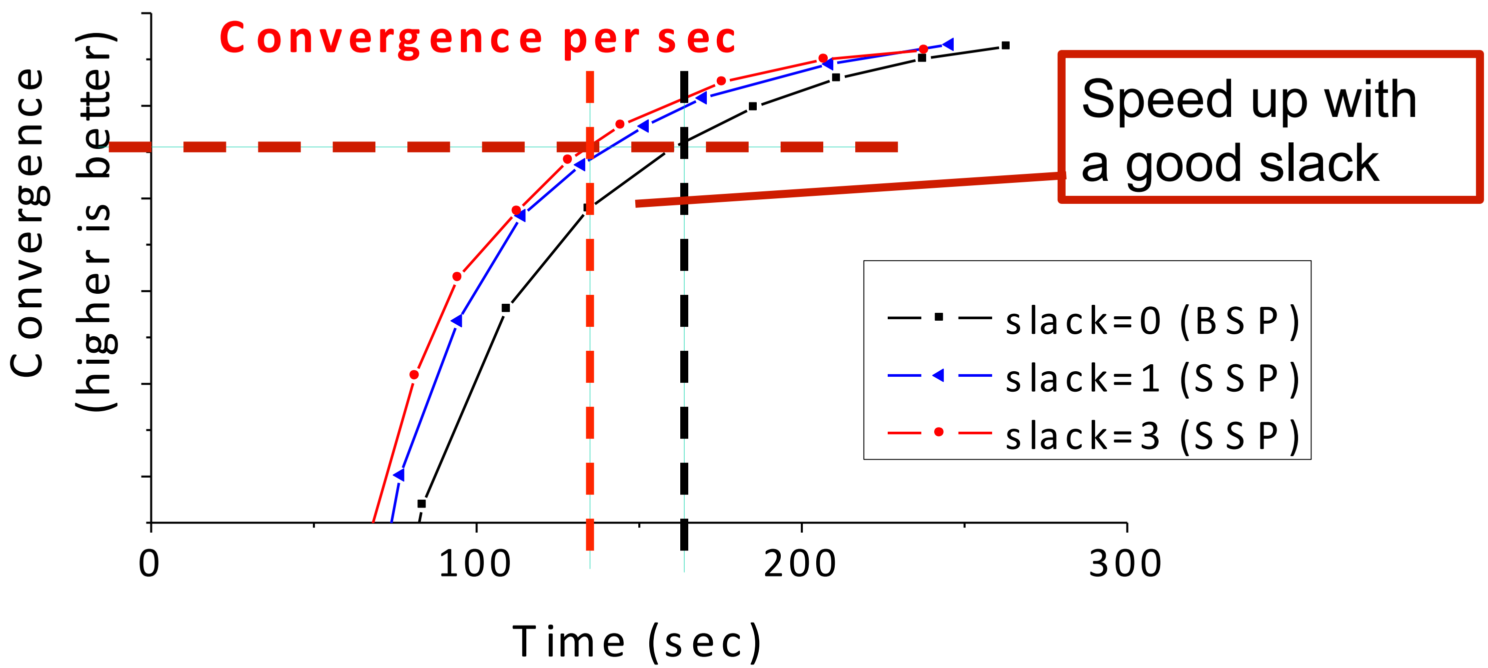


iters-per-clock is 1

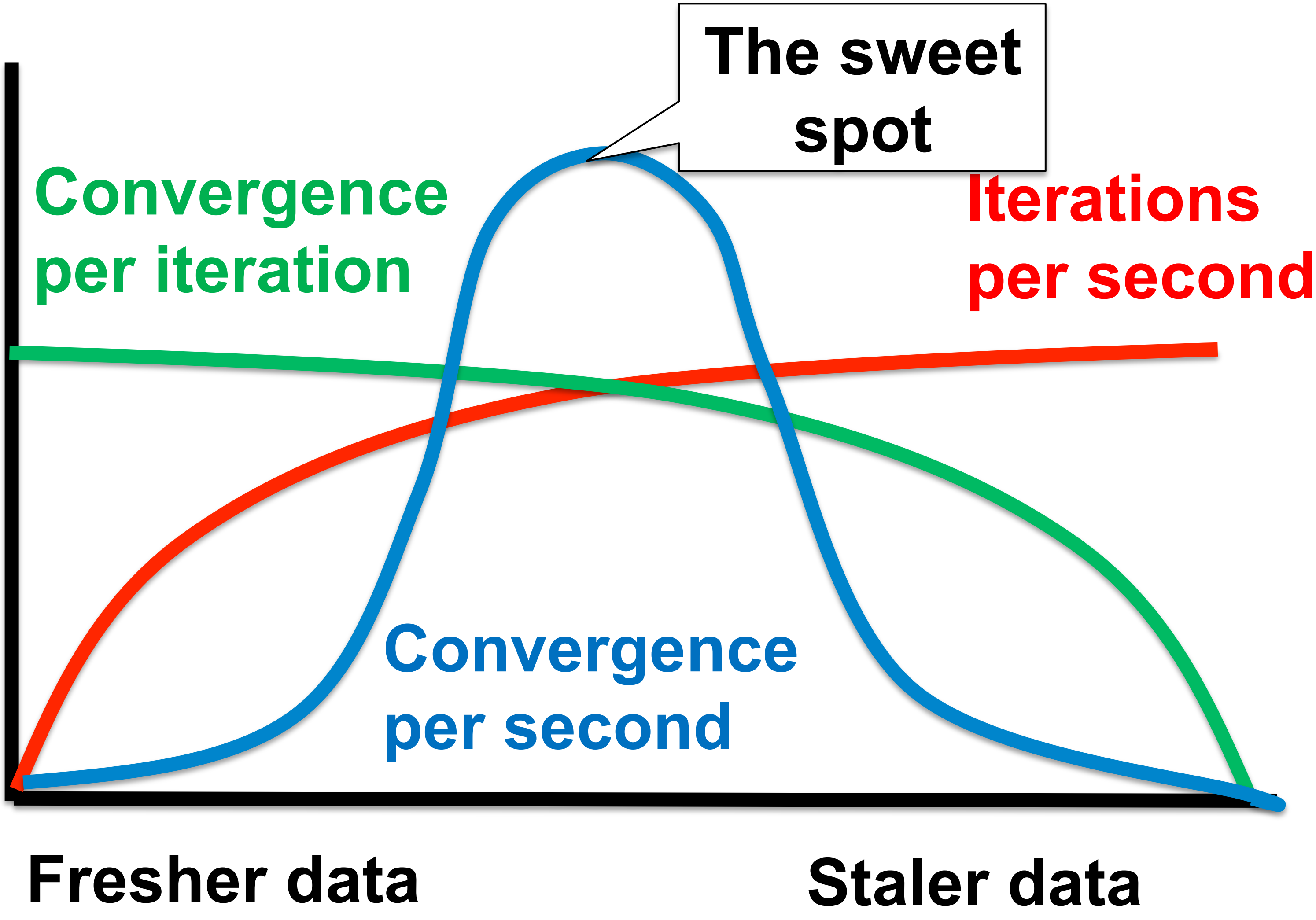
Sweet Spot Balances the Two







Key Takeaway Insight #1



SSP vs A-BSP

- Similar performance
  - In the absence of stragglers

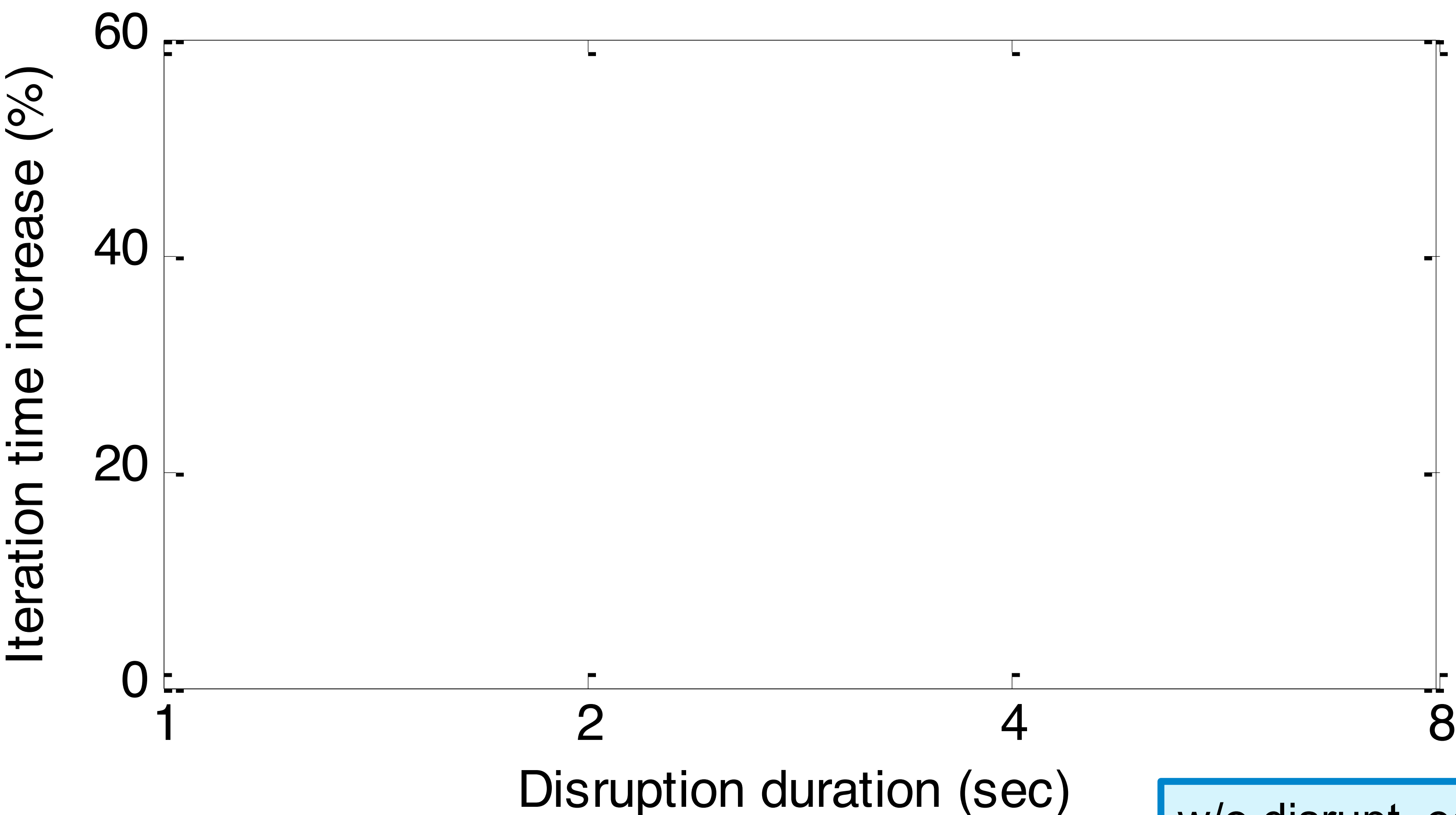
What about environment with stragglers?

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

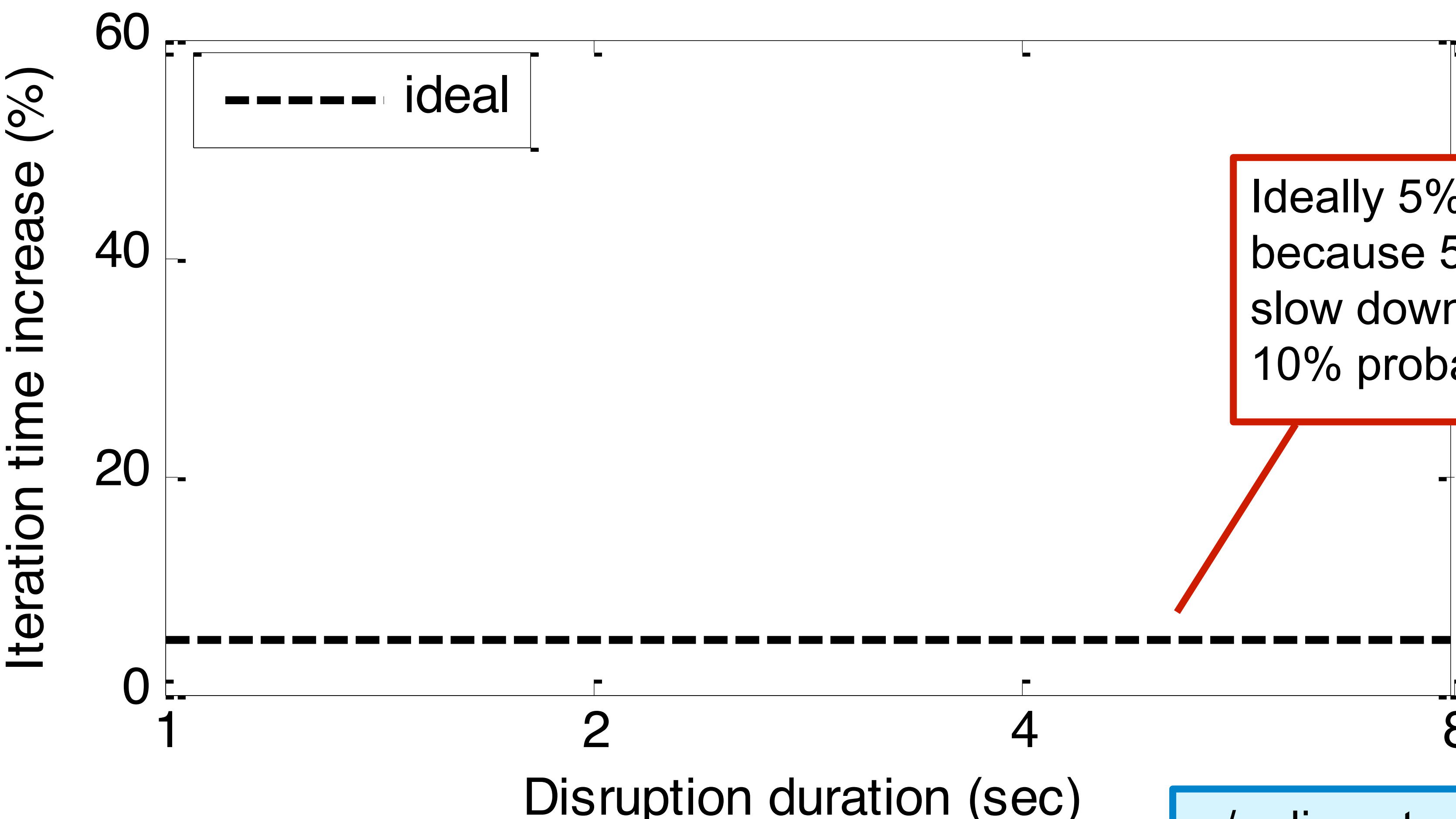
More straggler experiments in the paper

Straggler Results #1



w/o disrupt, each iter takes 4.2 sec

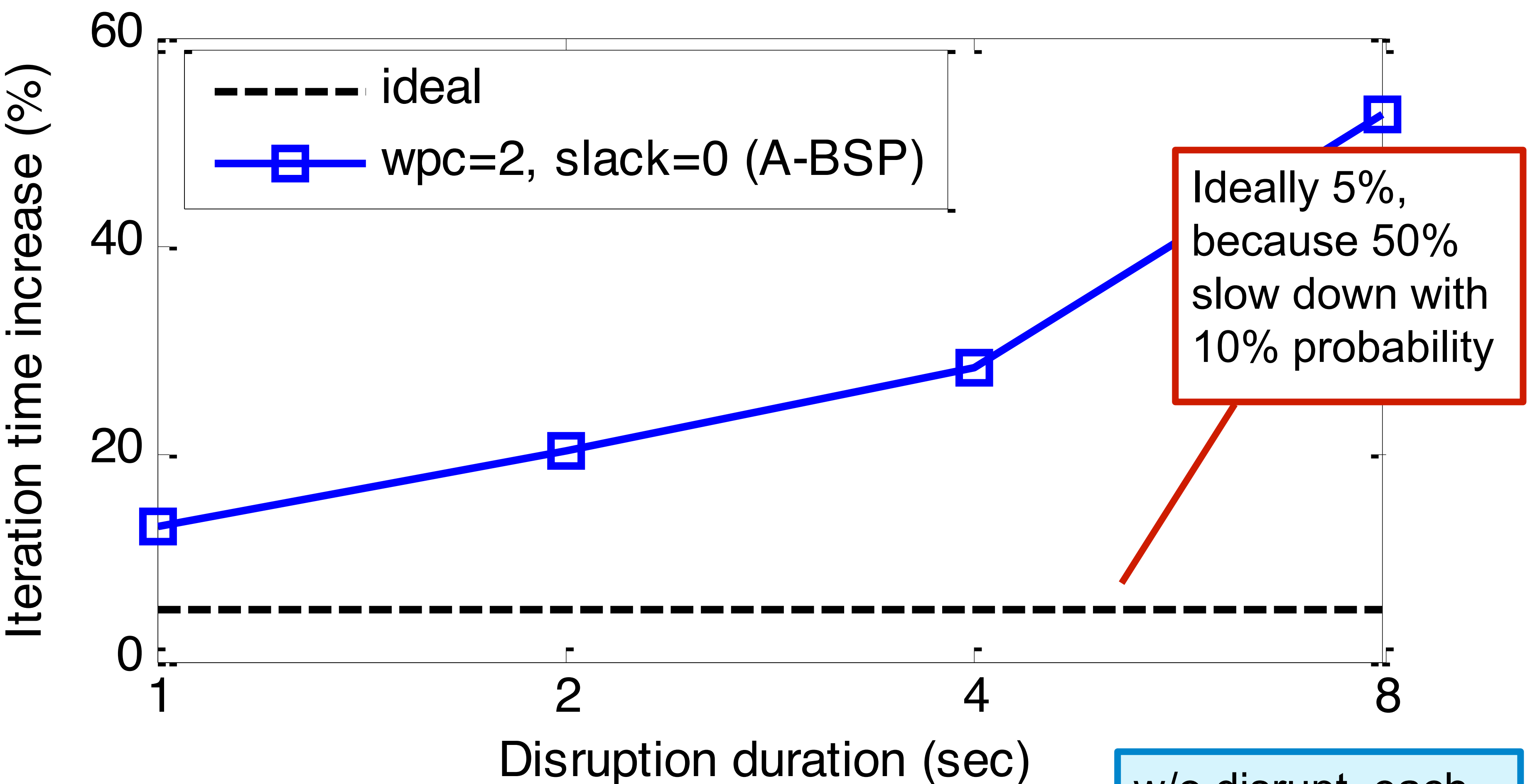
Straggler Results #1



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

w/o disrupt, each iter takes 4.2 sec

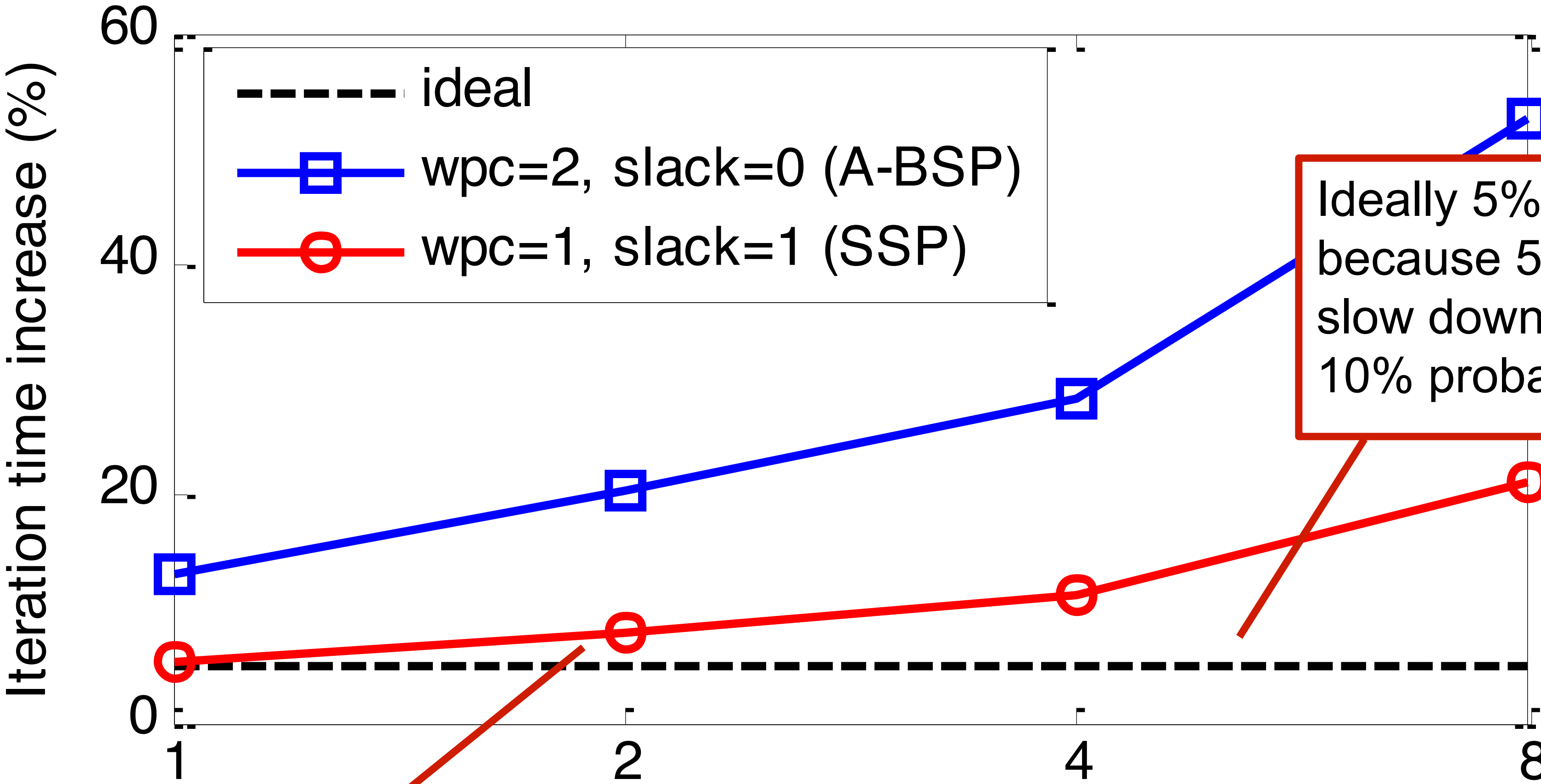
Straggler Results #1



w/o disrupt, each iter takes 4.2 sec

Carnegie Mellon
Parallel Data Laboratory

Straggler Results #1



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

SSP tolerates transient stragglers

w/o disrupt, each iter takes 4.2 sec

Conclusion

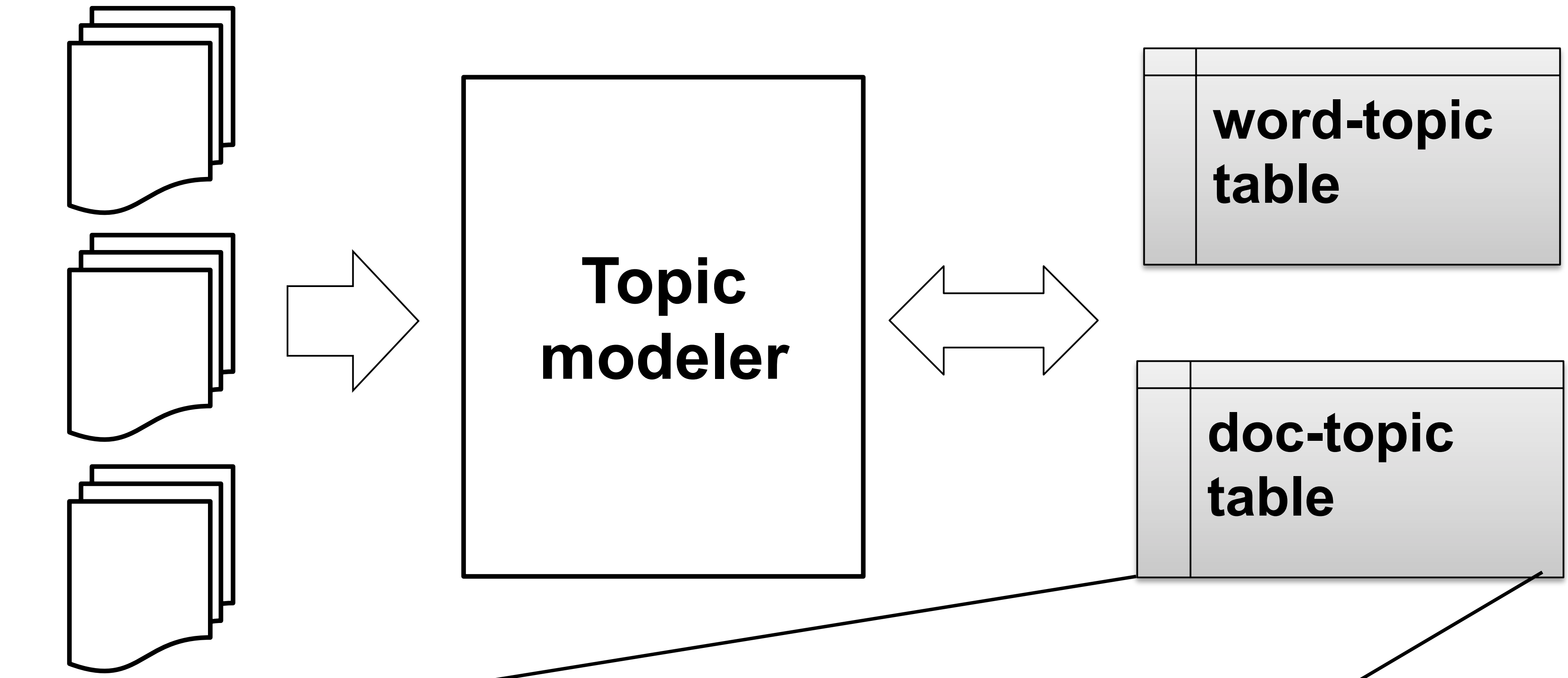
- Staleness should be tuned
  - By iters-per-clock and/or slack
- LazyTable implements SSP and A-BSP
  - See paper for details
- Key results from experiments
  - Both SSP and A-BSP are able to exploit the staleness sweet-spot for faster convergence
  - SSP is tolerant of small transient stragglers
 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

References

- J. Cipar, G. Ganger, K. Keeton, C. B. Morrey, III, C. A. Soules, and A. Veitch. LazyBase: trading freshness for performance in a scalable database. Eurosys'12.
- J. Cipar, Q. Ho, J. K. Kim, S. Lee, G. R. Ganger, G. Gibson, K. Keeton, and E. Xing. Solving the straggler problem with bounded staleness. HotOS'13.
- NYTimes: http://archive.ics.uci.edu/ml/datasets/Bag+of+Words
- Q. Ho, J. Cipar, H. Cui, S. Lee, J. Kim, P. Gibbons, G. Gibson, G. Ganger, and E. Xing. More effective distributed ML via a stale synchronous parallel parameter server. NIPS'13.

BACK-UP

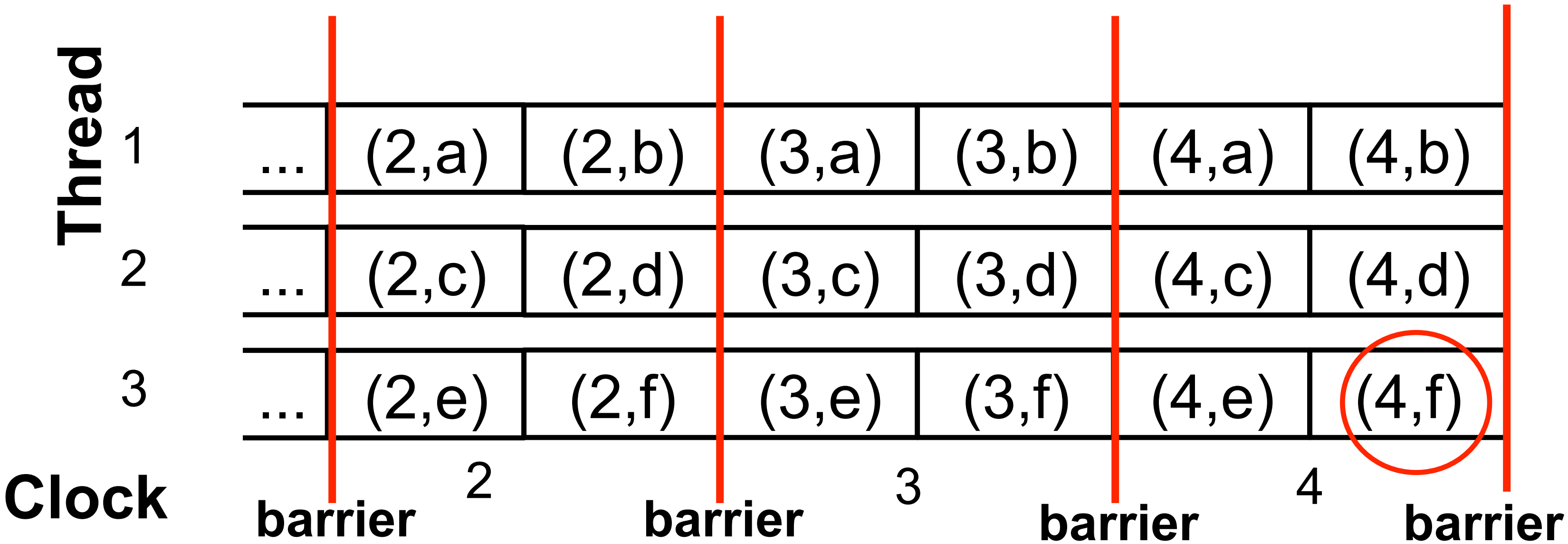
Example: Topic Modeling



...	...	...	...
Doc i	Topic 1: 0.8	Topic 2: 0.1	...
...	...	...	...

BSP Progress and Staleness

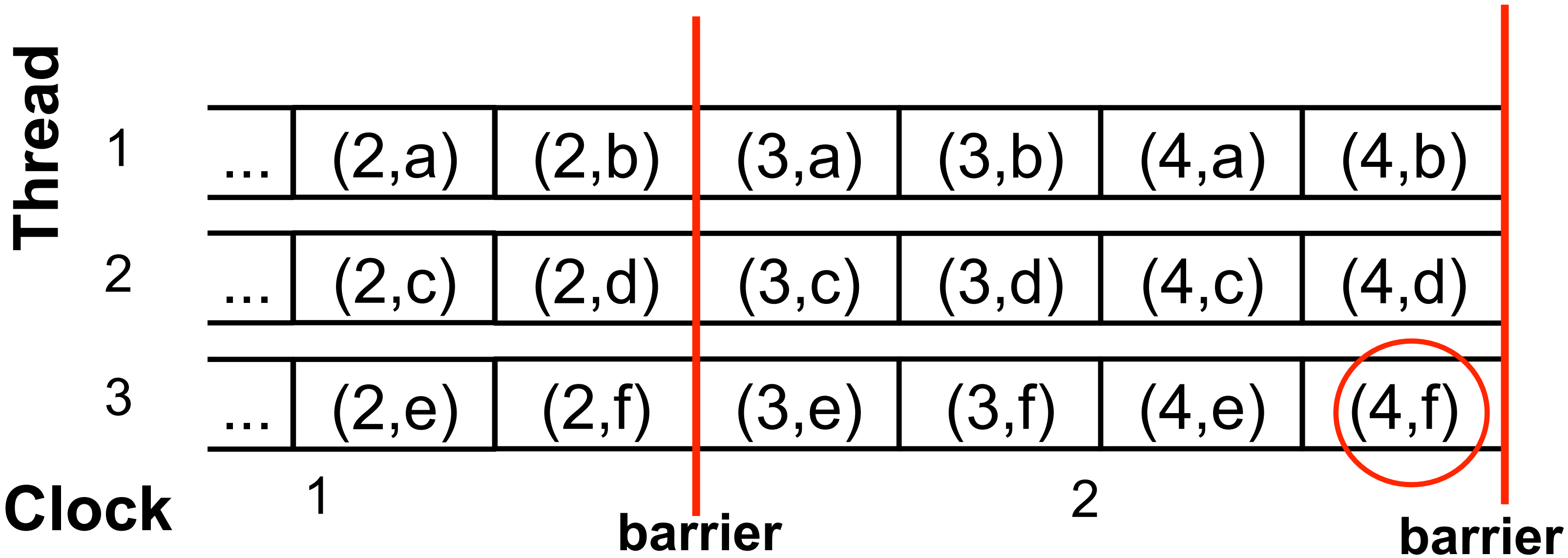
- `(i, j)` represents `iteration i`, `work uint j`



Carnegie Mellon
Parallel Data Laboratory

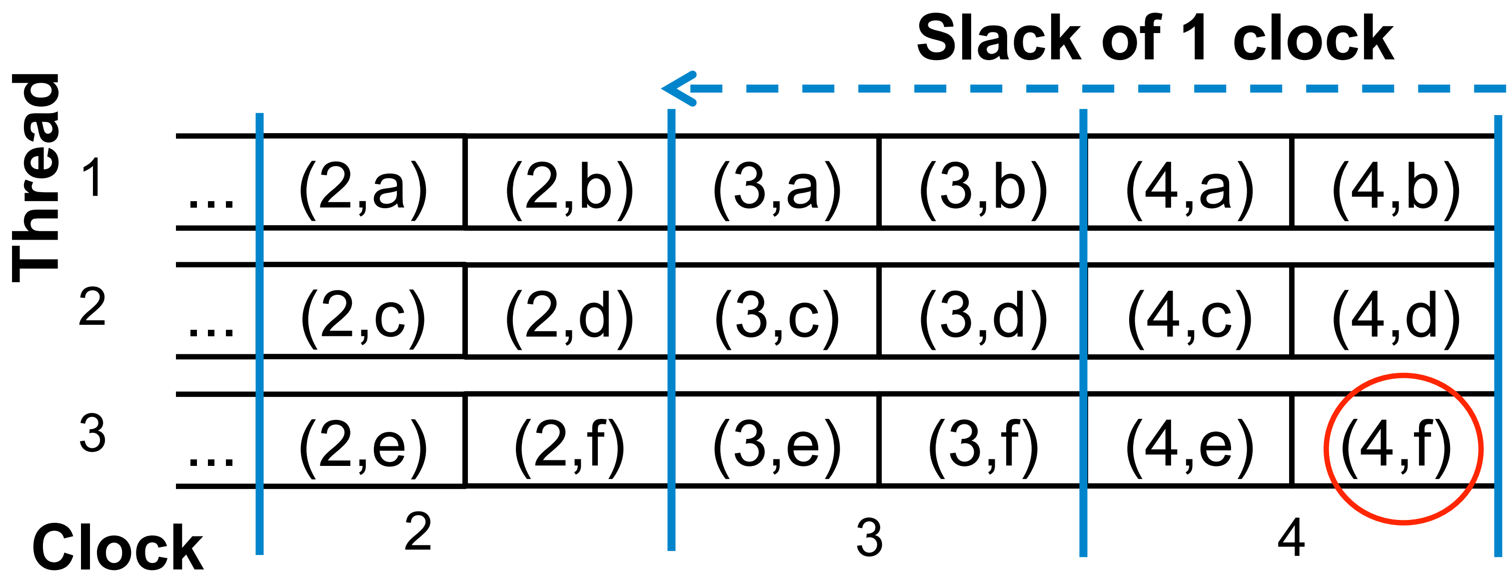
A-BSP Progress and Staleness

- A-BSP, wpc = 2 iterations



SSP Progress and Staleness

- SSP, wpc = 1 iteration, slack = 1 clock



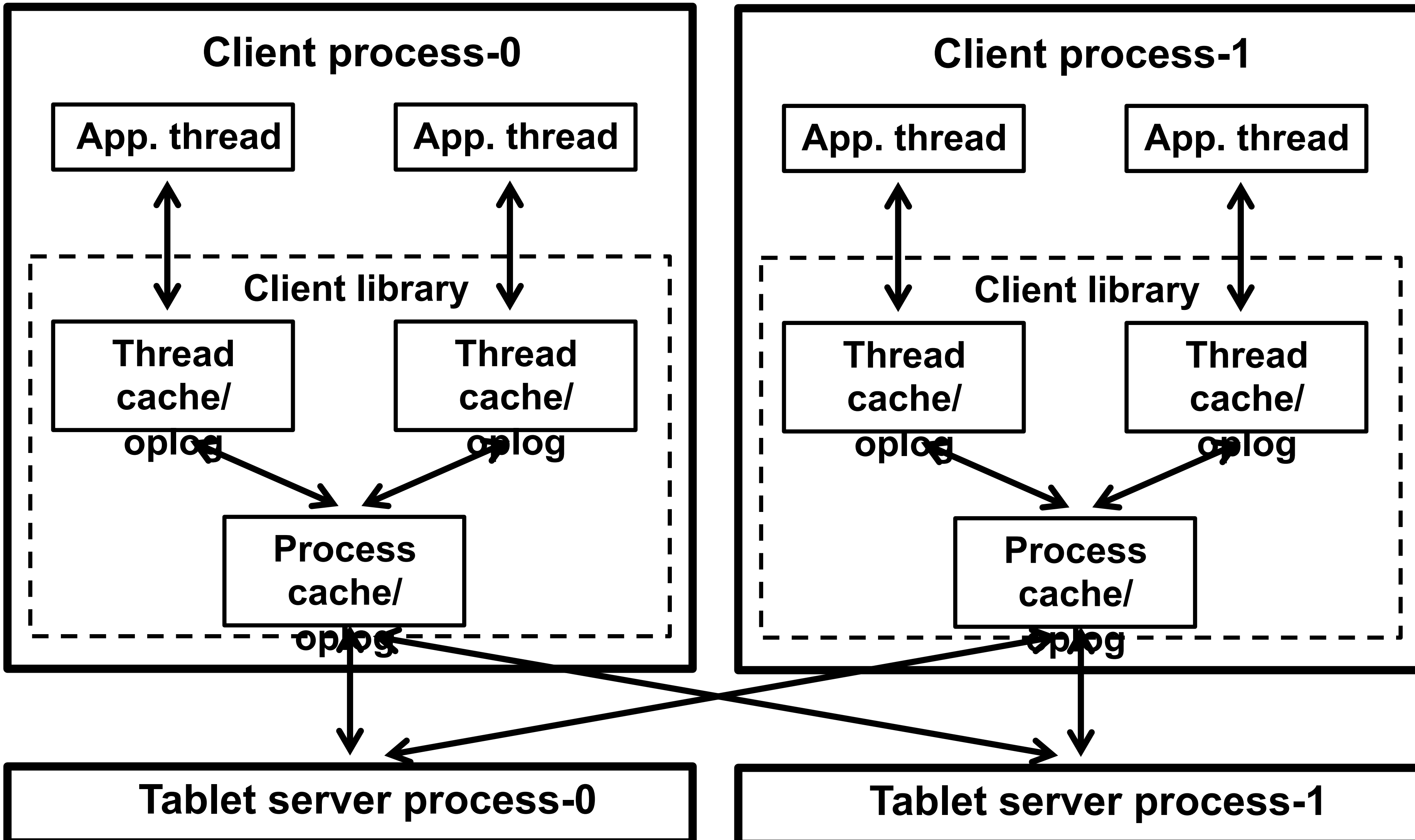
- Same staleness bound as the A-BSP one
  - But more flexible
- Data staleness for SSP with wpc and slack:

wpc x (slack + 1) - 1

SSP VS A-BSP

- A-BSP is SSP with a slack of zero
- Data staleness bound
  - SSP {wpc, slack} == A-BSP {wpc x (slack + 1), 0}
- SSP is a "pipelined" version of A-BSP
  - Tolerates transient stragglers

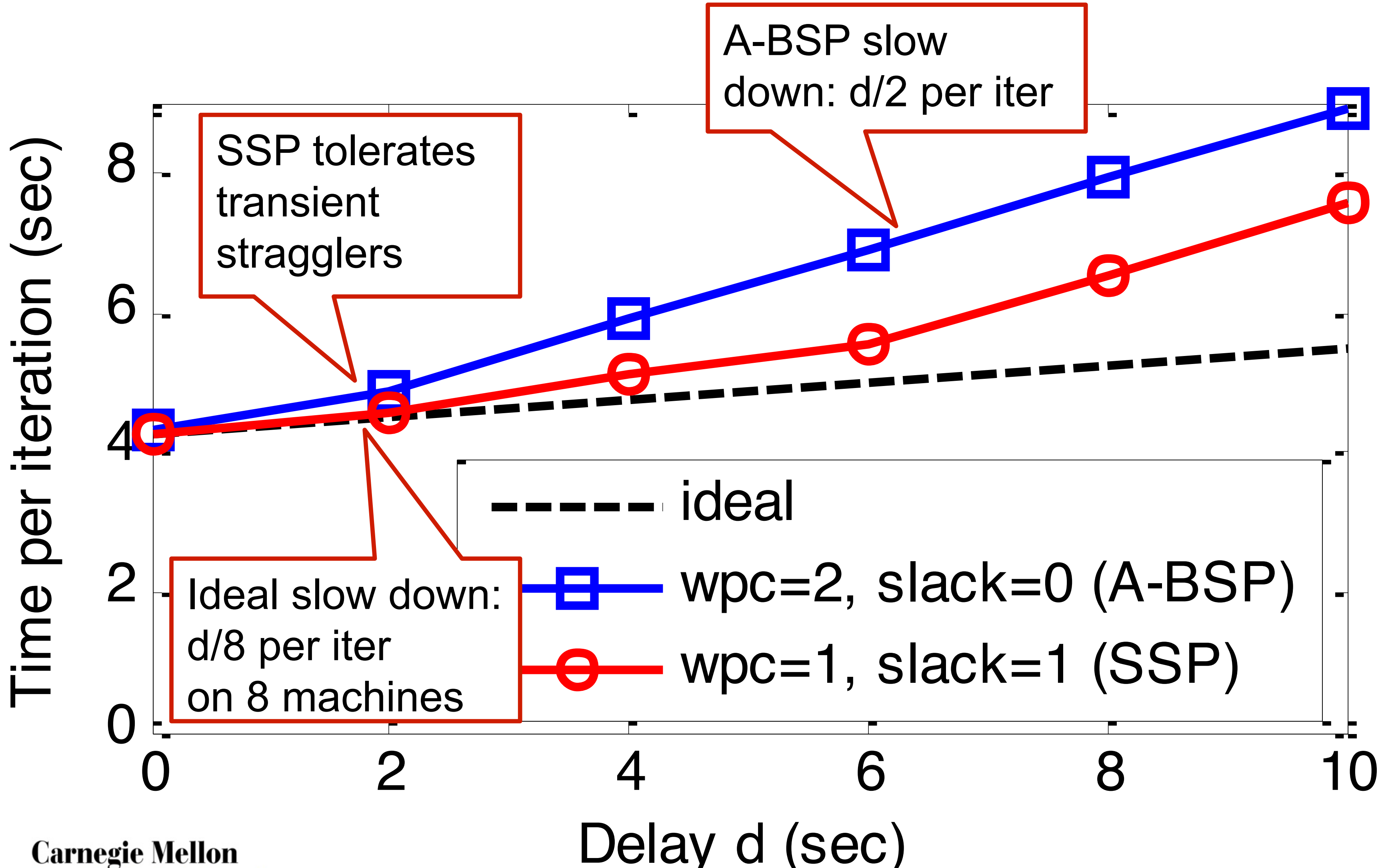
LazyTable Architecture



Stragglers: Delay

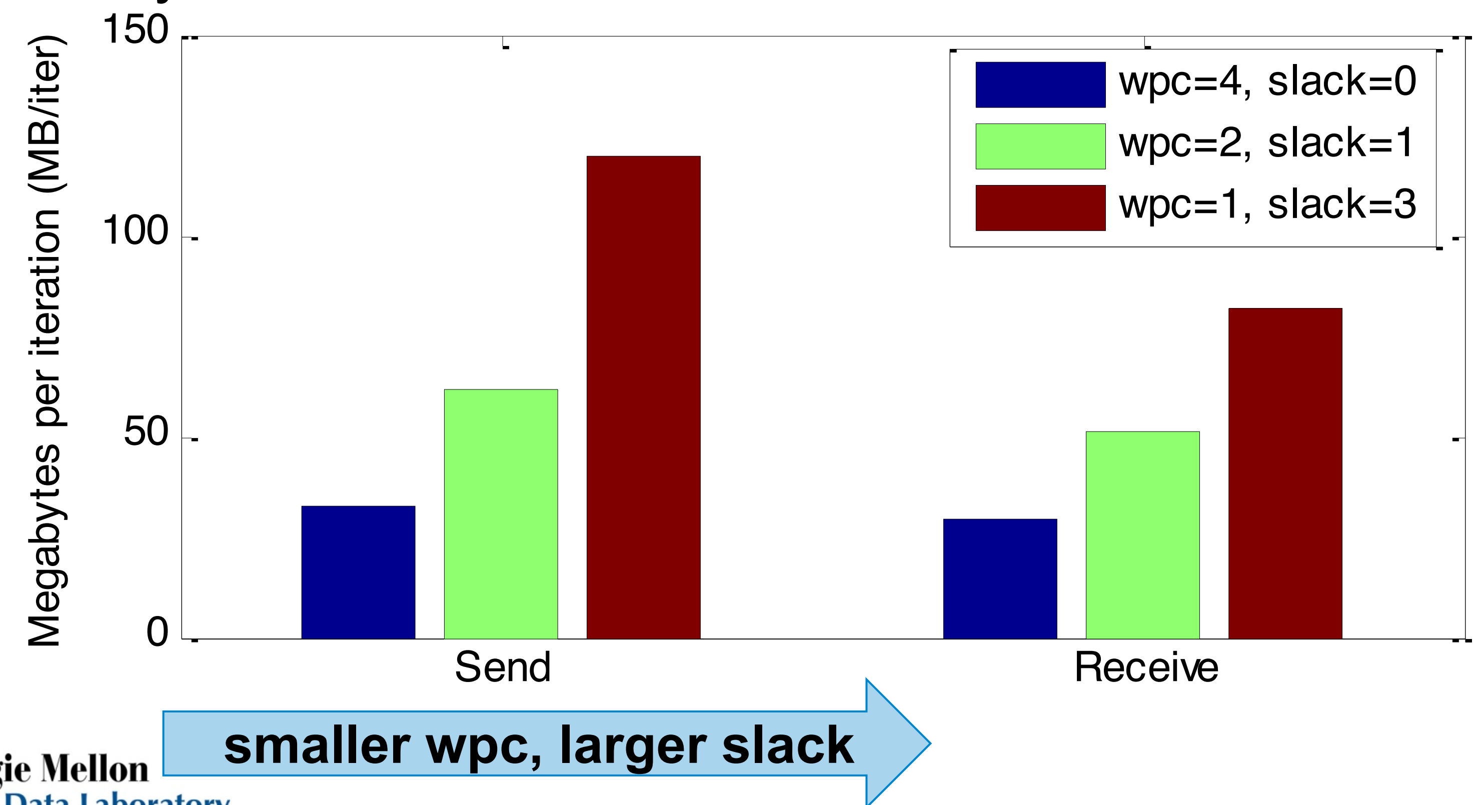
- Delaying some threads
  - Artificially introduce stragglers to the system
  - Have some threads sleep() for a time
- Experiment setup
  - Threads sleep "d" seconds in turn
    - Threads of machine "i" sleep at iteration "i"
  - Compare influence of different "d"

Stragglers: Delay (Results)



The Cost of Increased Flexibility

- Comparing {wpc=X, ...} with {wpc=2X, ...}
  - Bytes sent doubled (send update twice as often)
  - Bytes received almost doubled



Key Takeaway Insight #3

- SSP incurs more traffic
  - Finer grained division of clocks
  - Avoiding barriers is still a win,
    if communication is not the bottleneck

Prefetching

- Conservative prefetch
  - refresh row only when it's too stale
- Aggressive prefetch
  - refresh row every clock

Prefetching (results)

- wpc = 1 iter, slack = 7 clocks