Straggler Exploitation in Distributed Computing Systems with Task Grouping

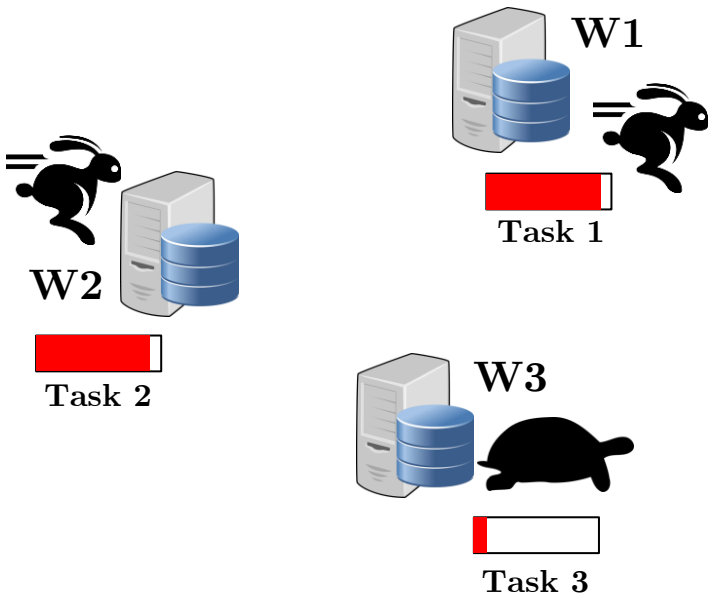
Tharindu Adikari
University of Toronto

Joint work with Haider Al-Lawati,
Jason Lam, Zhenhua Hu, and Stark Draper

Allerton Conference
Fall 2023

Stragglers **Lengthen** Running Time

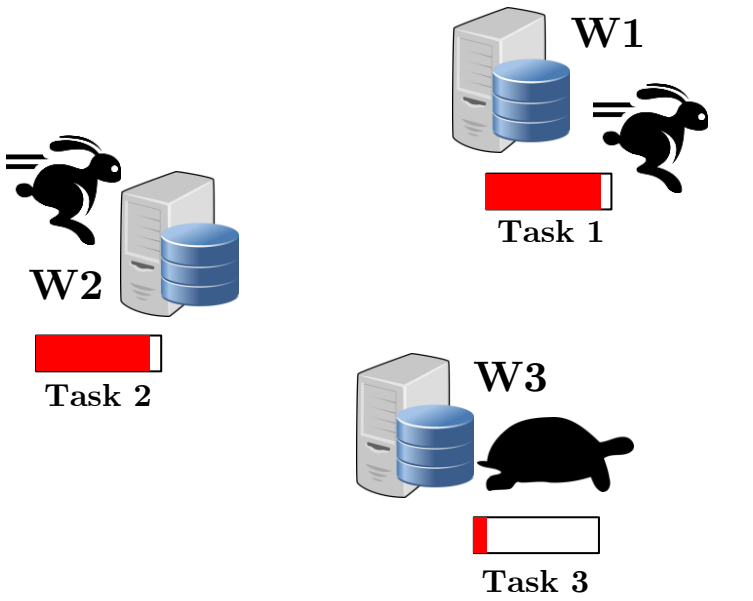
Setup: 3 tasks, 3 workers



- Some nodes are **SLOW** (Stragglers).
- Have to wait for the **slowest node**!

Stragglers **Lengthen** Running Time

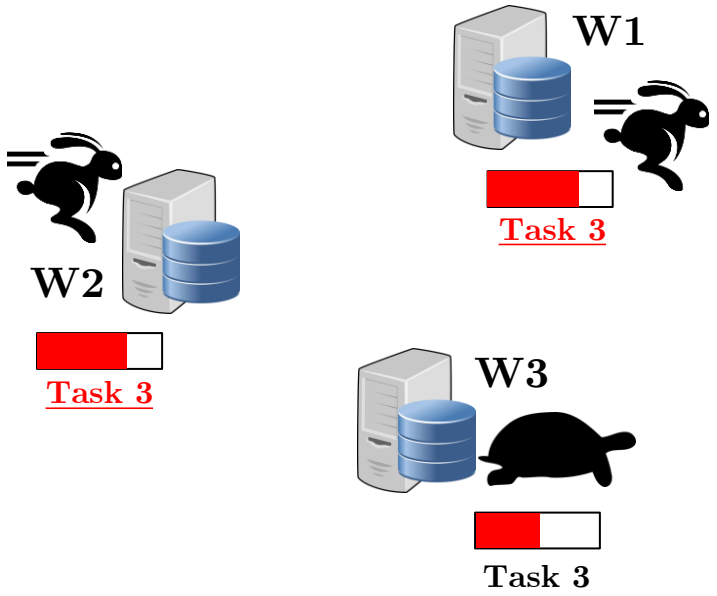
Setup: 3 tasks, 3 workers



- Some nodes are **SLOW** (Stragglers).
- Have to wait for the **slowest node**!

When tasks 1 & 2 finish

Standard replication



- **Replicate** Task 3 on W1 & W2.
- Store the result of the first to finish.

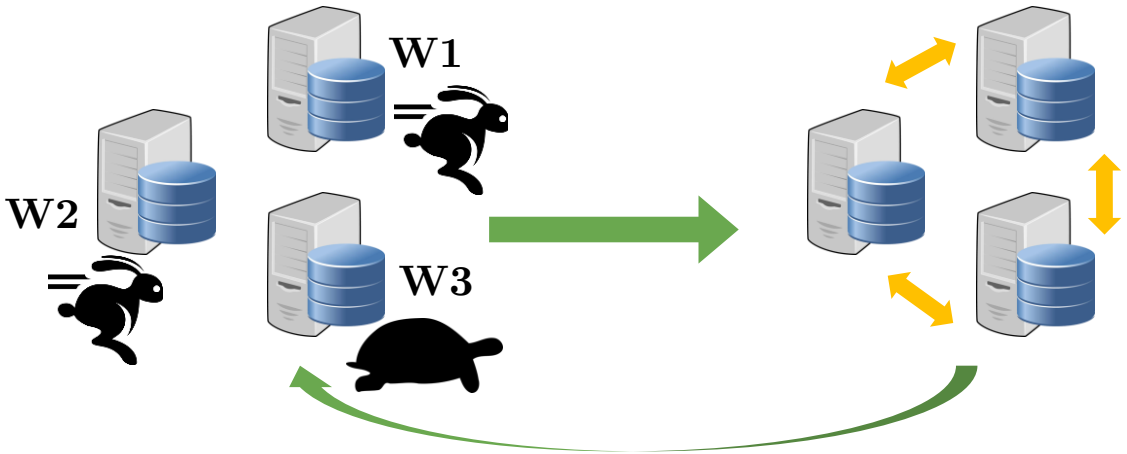
Algorithms exist for incorporating other information to prioritize replication: LATE, Mantri, MCP, Dolly

Zaharia (2008, 2013), Ananthanarayanan (2010), Chen (2013), Xu (2013, 2014, 2015)

Straggler **Exploitation** in SGD

Step 1: Compute
equal workloads

Step 2: Communicate
to synchronize

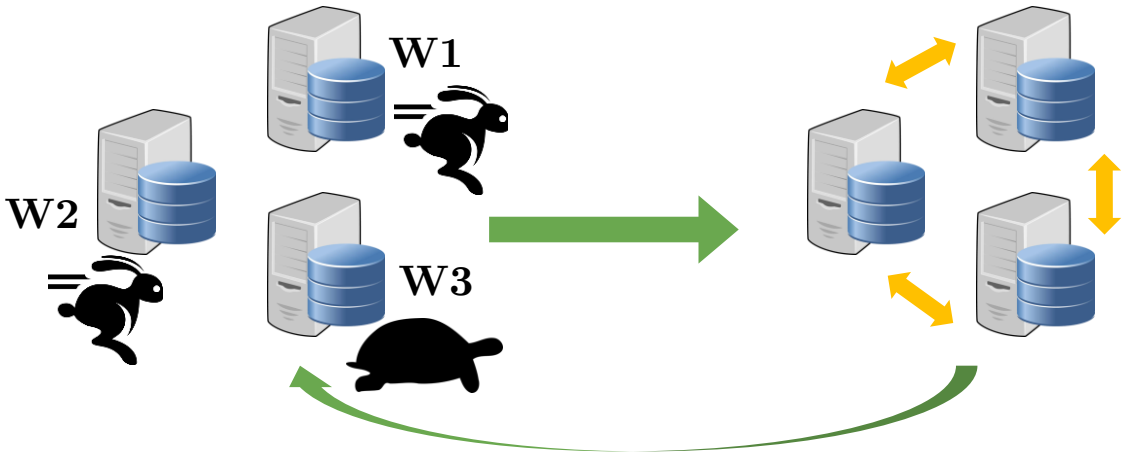


- Compute until a **time limit** in Step 1.
- Sync. with **partial results** in Step 2.

Straggler **Exploitation** in SGD

Step 1: Compute
equal workloads

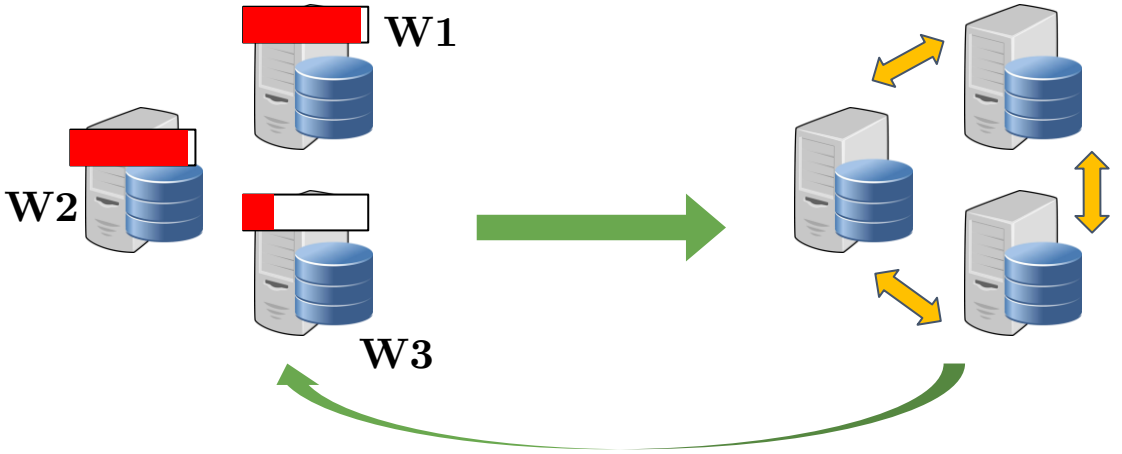
Step 2: Communicate
to synchronize



- Compute until a **time limit** in Step 1.
- Sync. with **partial results** in Step 2.

Step 1: Compute
For only 10 seconds

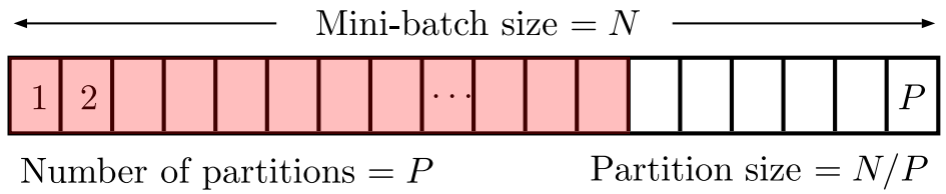
Step 2: Communicate
Partial computations



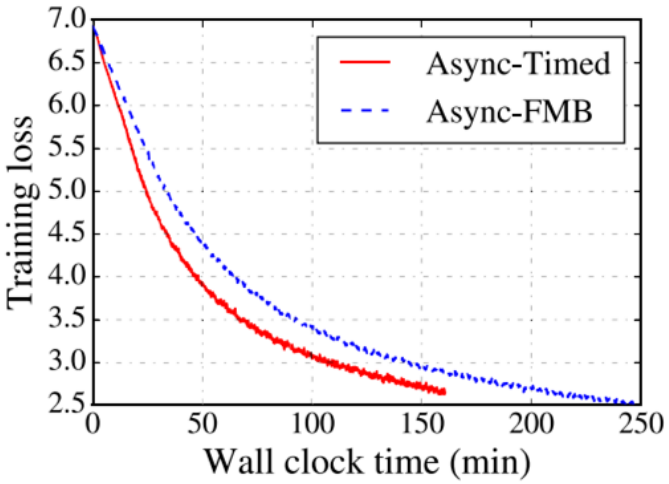
- Everybody's work is taken into account!
- Time for an iteration is LOWER!

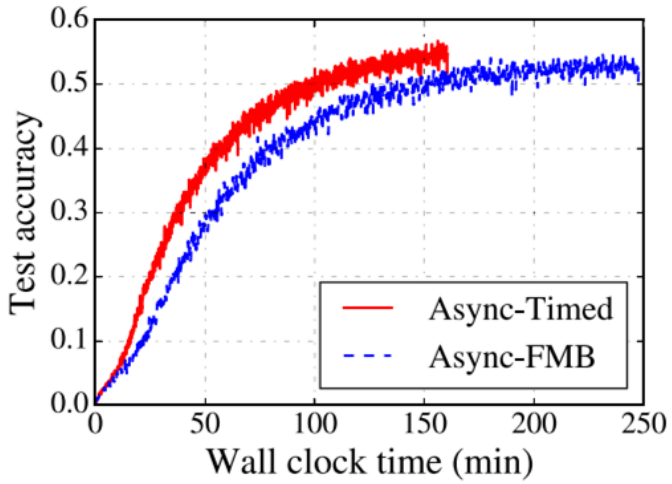
Ferdinand et al. (2017), Reisizadeh et al. (2019), Al-Lawati et al. (2021)

Straggler **Exploitation** in SGD: Results



The higher the granularity of splitting,
the higher the chance of stopping at the time limit.





AlexNet training on ImageNet dataset,
3 workers, natural stragglers.

Async-FMB : Equal workload approach
50% accuracy by **150 mins**

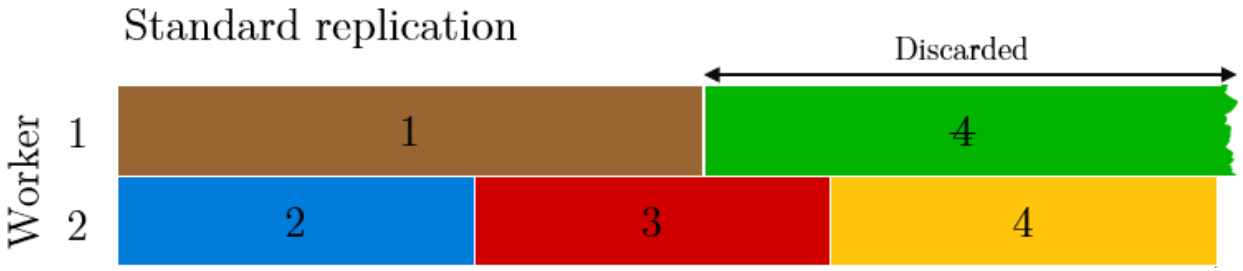
Async-Timed: Time-limited approach
50% accuracy by **100 mins**

Time limited approach is around **33% faster!**

Al-Lawati et al. (2021)

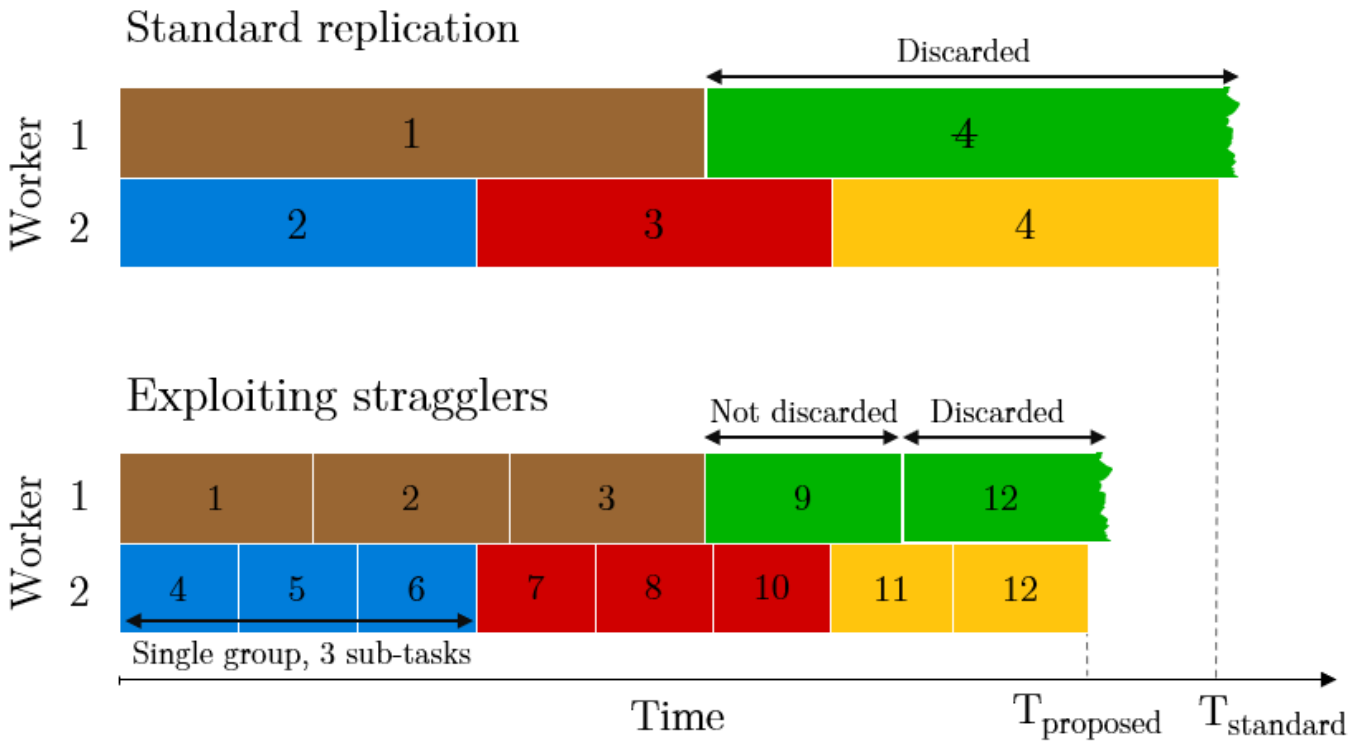
Straggler **Exploitation** with **General Computations**

- Computations must be exact (unlike 'stochastic' gradient descent).
- Hard to implement a time limit. >> Key Idea: **Increase task granularity**.



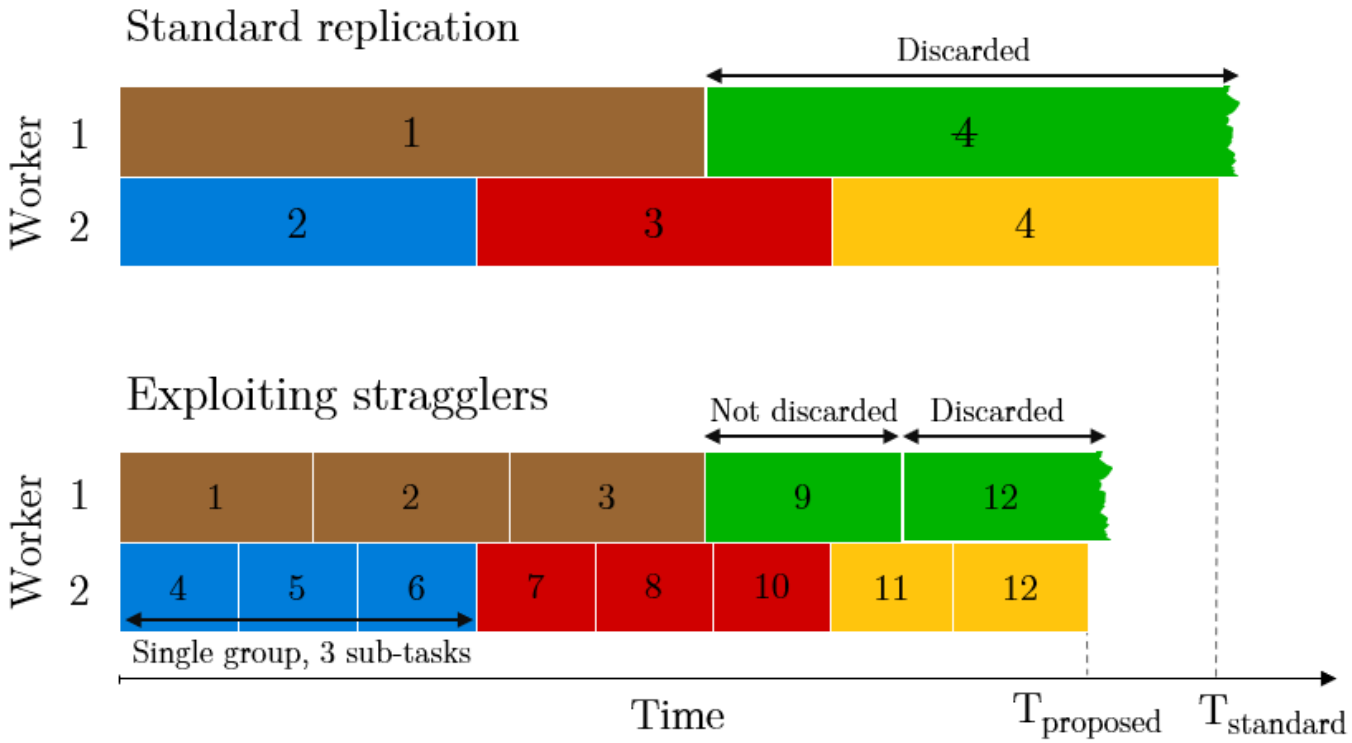
Straggler **Exploitation** with **General Computations**

- Computations must be exact (unlike 'stochastic' gradient descent).
- Hard to implement a time limit. >> Key Idea: **Increase task granularity**.



Straggler **Exploitation** with **General Computations**

- Computations must be exact (unlike 'stochastic' gradient descent).
- Hard to implement a time limit. >> Key Idea: **Increase task granularity**.



- Hard to split some tasks into **sub tasks**.
- Second method resembles **grouping** of tasks.

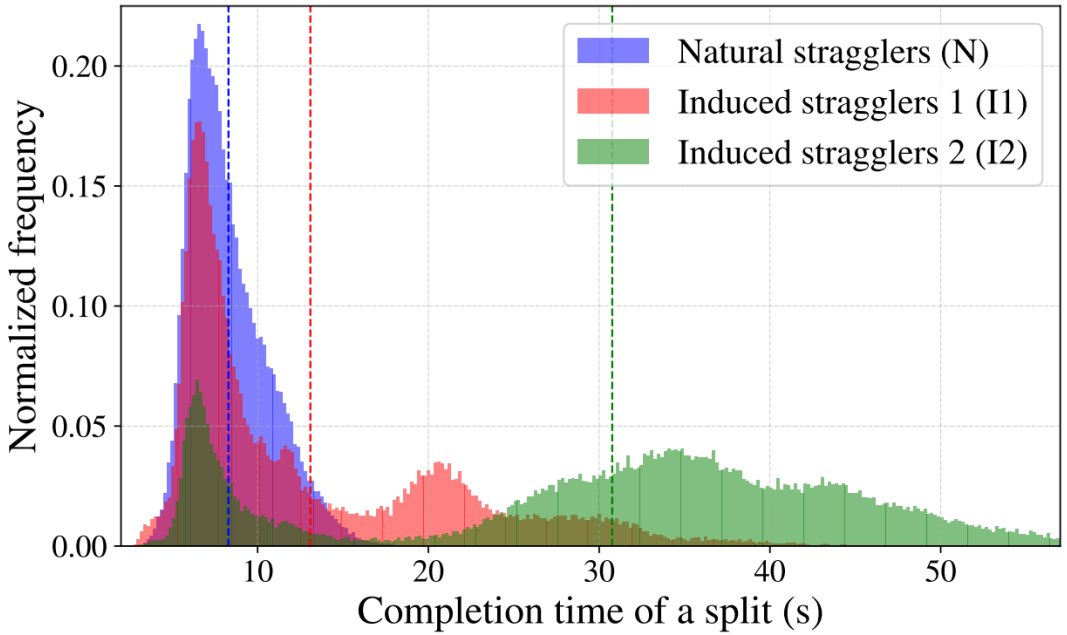
- Methodology summary:

1. **Assign groups of tasks to workers.**
2. **Increase worker updates.**
3. **Skip completed tasks.**

Amazon EC2 Tests

32 workers, 50 GB word count workload.
392 files (tasks) of 128 MB each.

Implementation with YARN and HDFS.



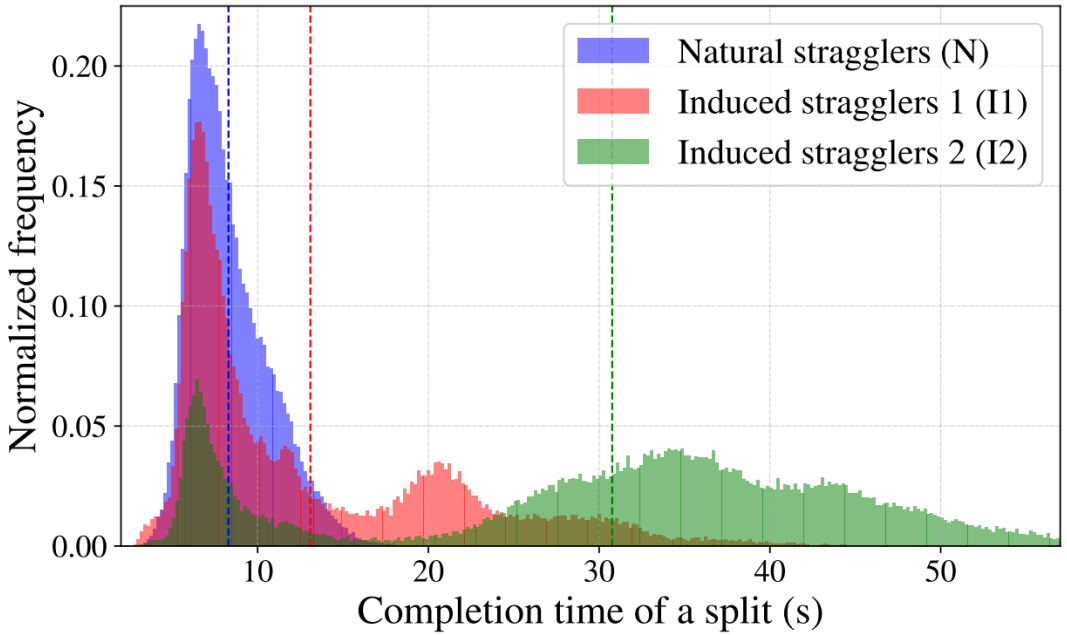
Amazon EC2 Tests

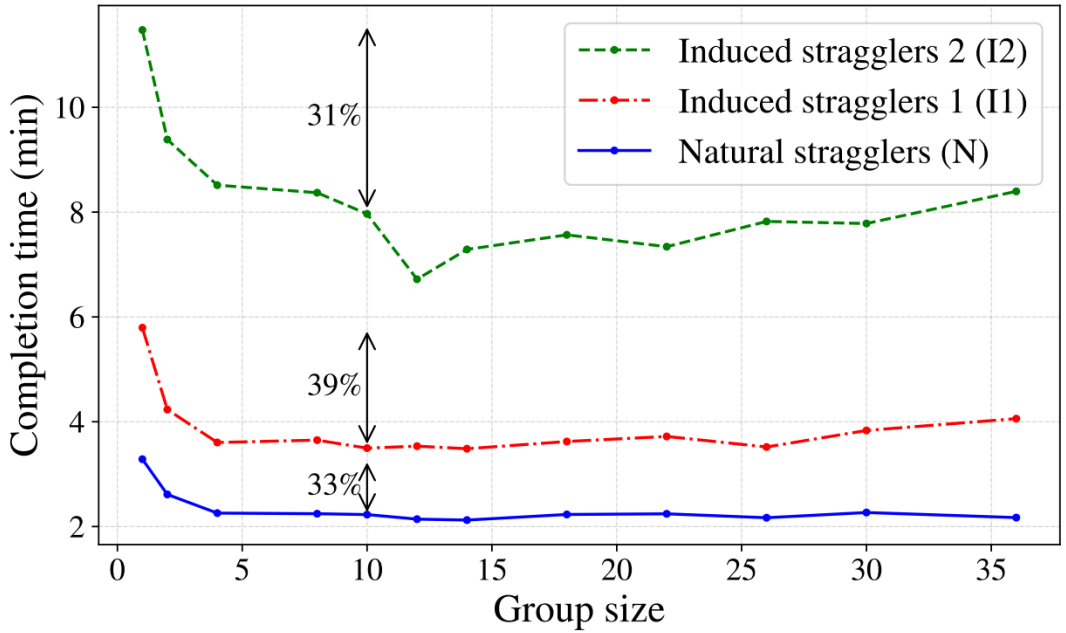
32 workers, 50 GB word count workload.
392 files (tasks) of 128 MB each.

Implementation with YARN and HDFS.

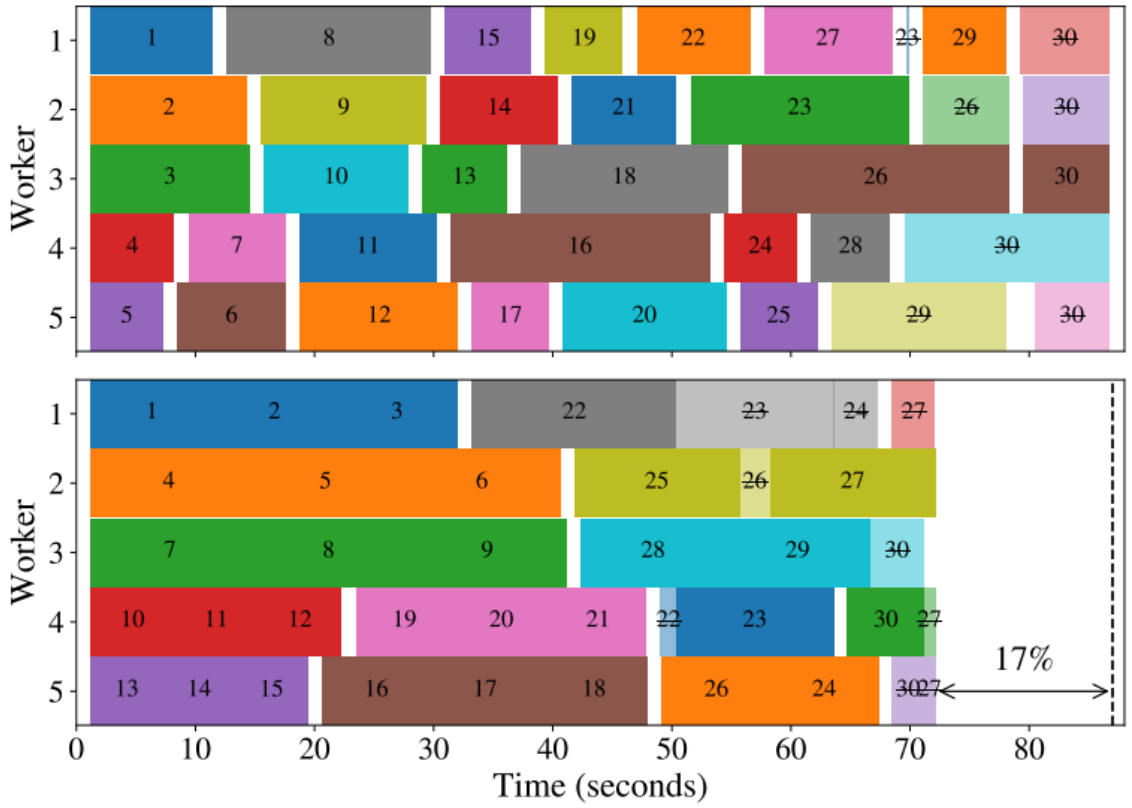
Baseline: Group size = 1 (standard replication).

Key observation: Time savings increase with the straggler severity.



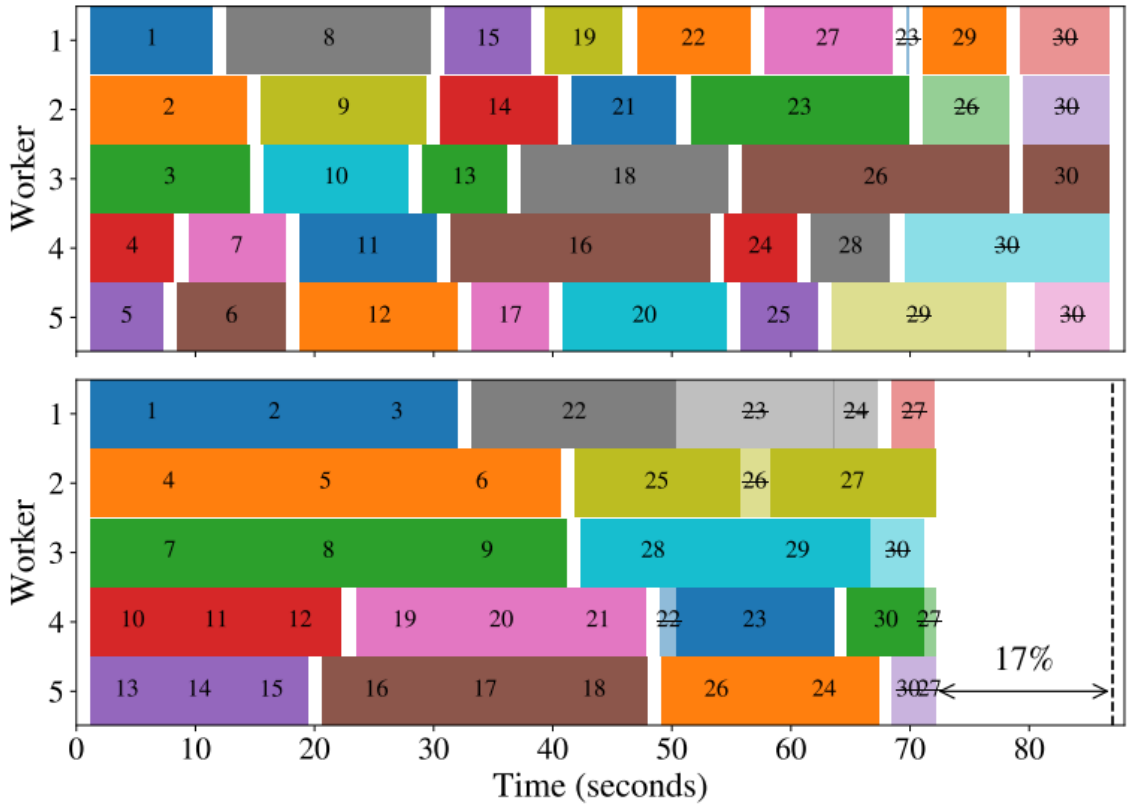


Source of Time Savings

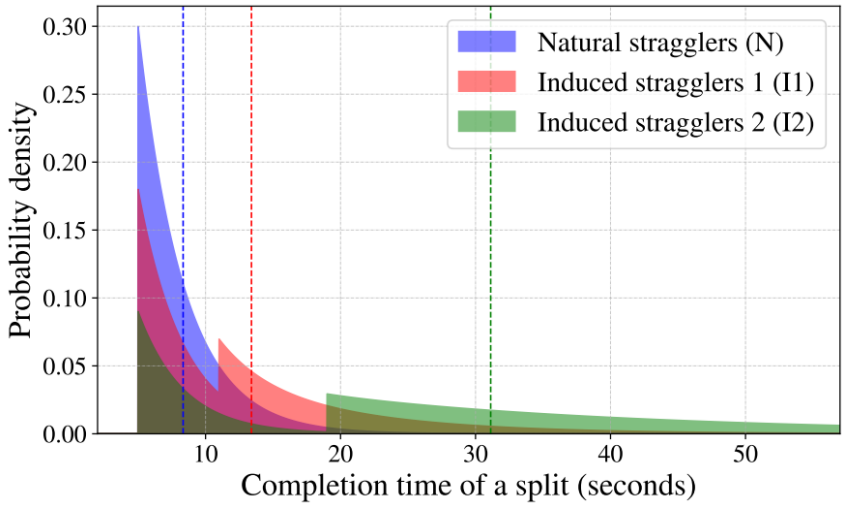


- Grouping reduces overhead (whitespace between tasks).
- How much savings is due to the reduction of overhead?
- Conduct simulation-based experiments.

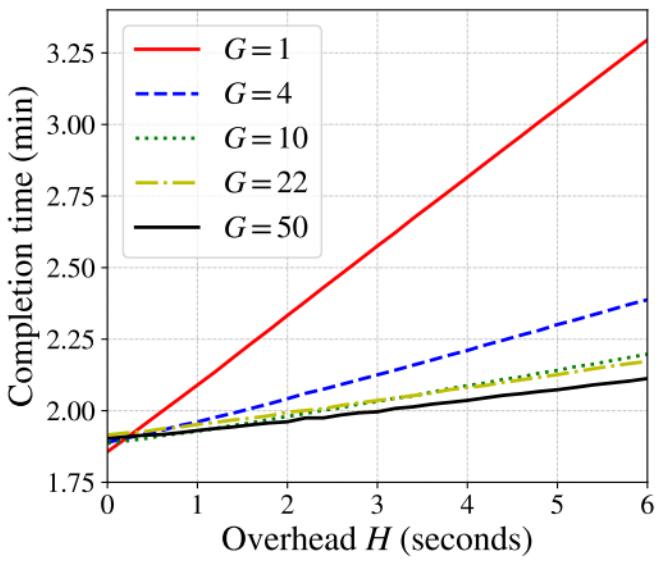
Source of Time Savings



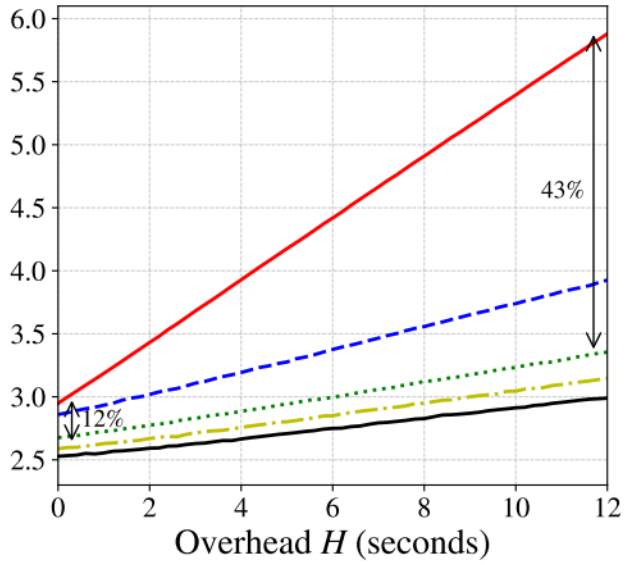
- Grouping reduces overhead (whitespace between tasks).
- How much savings is due to the reduction of overhead?
- Conduct simulation-based experiments.



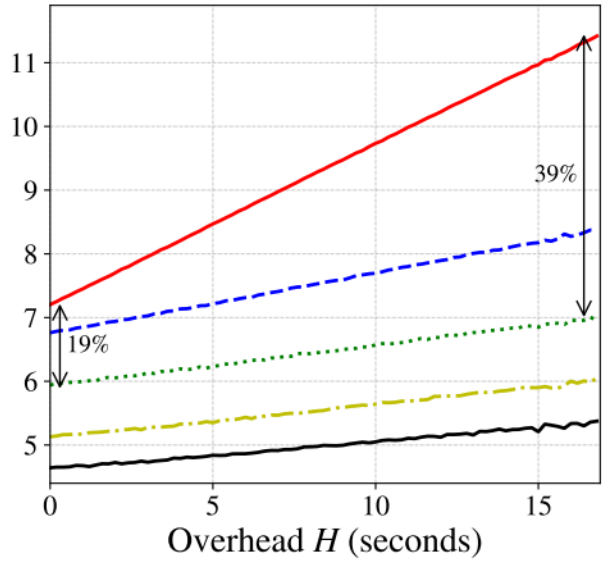
Overhead Reduction **vs.** Straggler **Exploitation**



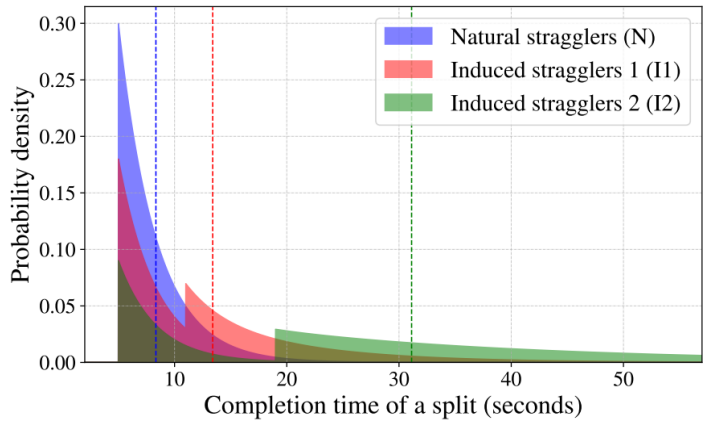
(a) With PDF (N).



(b) With PDF (I1).



(c) With PDF (I2).



Key observation: Time savings even if overhead=0, if stragglers are severe enough.

Conclusions and **Next Steps**

Talk summary:

- Proposed an algorithm for exploiting work performed by straggling workers.
- The key idea is to increase granularity of work assigned to workers.

Possible future work:

- An analysis that characterizes the performance improvement using of the group size parameter.
- A method for predicting the optimal group size for a given straggler profile/set of tasks.

Thank you! Questions?

References

1. M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, “Improving MapReduce performance in heterogeneous environments,” in OSDI, Dec 2008.
2. G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, “Reining in the outliers in MapReduce clusters using Mantri,” in OSDI, Oct 2010.
3. Q. Chen, C. Liu, and Z. Xiao, “Improving MapReduce performance using smart speculative execution strategy,” IEEE Trans. on Comp., vol. 63, no. 4, pp. 954–967, 2013.
4. H. Xu and W. C. Lau, “Resource optimization for speculative execution in a MapReduce cluster,” in IEEE Int. Conf. on Network Protocols. IEEE, 2013.
5. ——, “Speculative execution for a single job in a MapReduce system,” in IEEE Int. Conf. on Cloud Computing. IEEE, 2014, pp. 586–593.
6. ——, “Task-cloning algorithms in a MapReduce cluster with competitive performance bounds,” in IEEE Int. Conf. on Distributed Comp. Sys. IEEE, 2015, pp. 339–348.
7. A. C. Zhou, T.-D. Phan, S. Ibrahim, and B. He, “Energy-efficient speculative execution using advanced reservation for heterogeneous clusters,” in Int. Conf. on Parallel Processing, 2018, pp. 1–10.
8. G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, “Effective straggler mitigation: Attack of the clones,” in Symp. on Networked Sys. Design and Implementation, 2013, pp. 185–198.
9. N. Ferdinand, H. Al-Lawati, S. Draper, and M. Nokleby, “Anytime minibatch: Exploiting stragglers in online distributed optimization,” in Int. Conf. Learning Representations, New Orleans, May 2019.
10. H. Al-Lawati, T. B. Adikari, and S. C. Draper, “Asynchronous delayed optimization with time-varying minibatches,” IEEE J. Select. Areas Inform. Theory, vol. 2, no. 2, pp. 784–801, 2021.
11. A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, Tree gradient coding,” in Proc. Int. Symp. Inform. Theory. IEEE, 2019, pp. 2808–2812.