TensorLights

End-Host Traffic Scheduling for Distributed Deep Learning



Xin Sunny Huang　Ang Chen　**T. S. Eugene Ng**

Rice University

RICE

Big Data and Optical Lightpaths Driven Lab

This Work

The Parameter Server (PS) architecture is the most popular approach for distributed Deep Learning.

Disadvantage: traffic contention at PS introduces harmful stragglers.

TensorLights mitigates stragglers with improved application performance and machine utilization.

The Rise of Deep Learning (DL)

Language processing

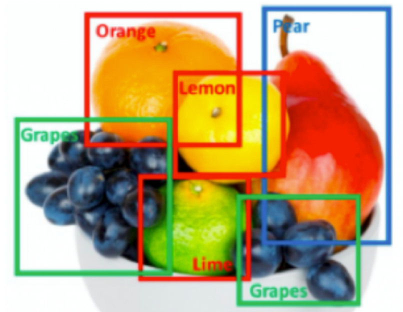
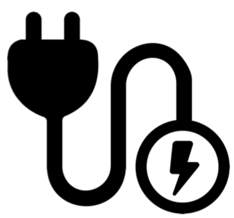


Image Recognition

Classic AI problems

Also used for ...



Power Scheduling [1]

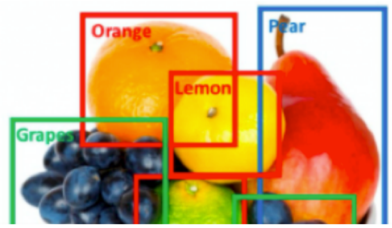
System Security [2]

Network Routing [3]

Database Index [4]

[1] Deepmind AI reduces Google data centre cooling bill by 40%. (2016)
[2] Abadi, M. et al. Learning to protect communications with adversarial neural cryptography. (arXiv 2016)
[3] Valadarsky, A. et al. Learning to route. (HotNets 2017)
[4] Kraska, T. et al. The case for learned index structures. (SIGMOD 2018)
[5] Gu, J. et al. Tiresias: A GPU Cluster Manager for Distributed Deep Learning (NSDI 2019)

The Rise of Deep Learning (DL)



La
pro

10.5× increase of DL training jobs in Microsoft [5]

Also u
for ...

Power Scheduling [1]

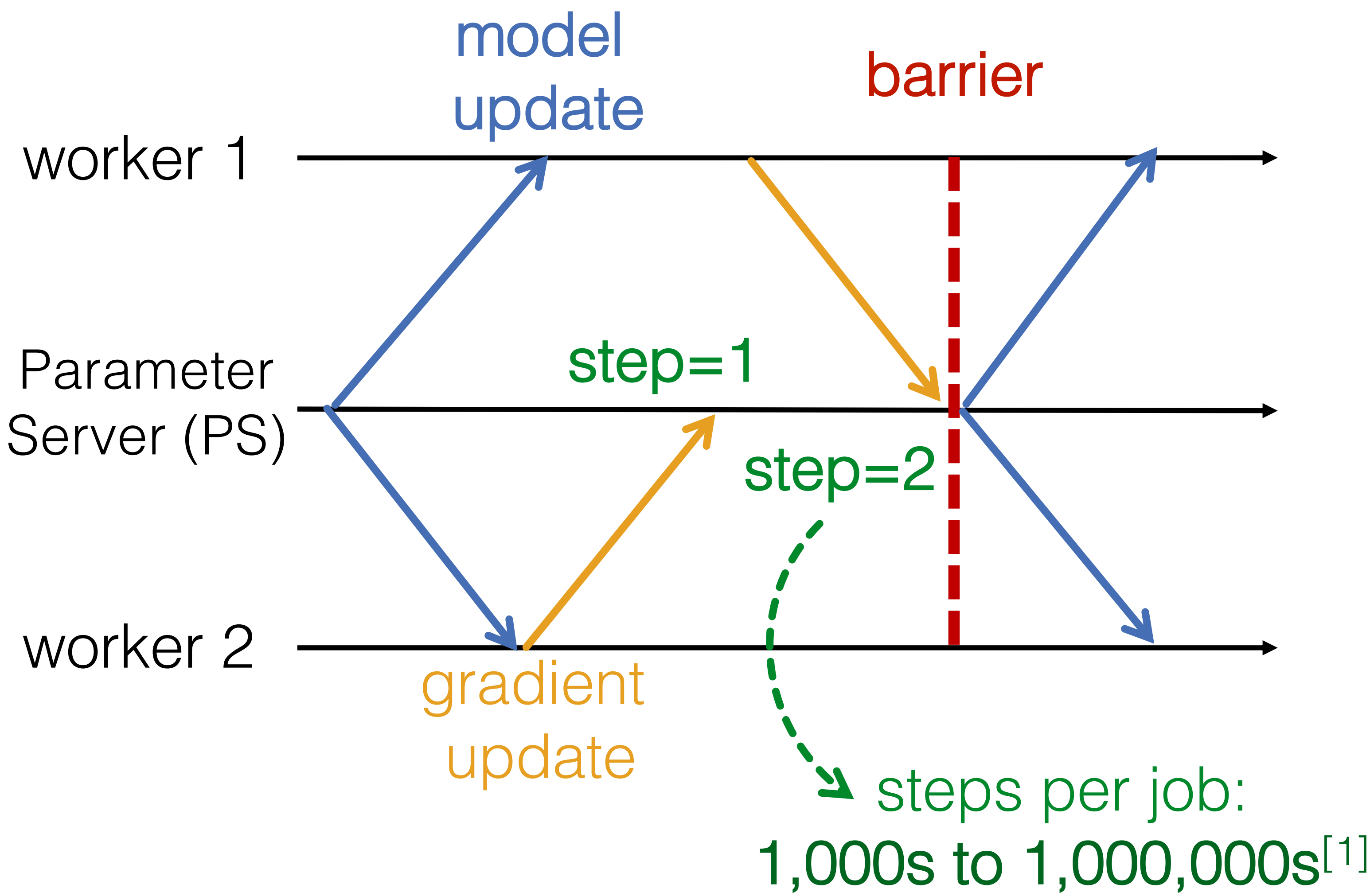
System Security [2]

Network Routing [3]

Database Index [4]

[1] Deepmind AI reduces Google data centre cooling bill by 40%. (2016)
[2] Abadi, M. et al. Learning to protect communications with adversarial neural cryptography. (arXiv 2016)
[3] Valadarsky, A. et al. Learning to route. (HotNets 2017)
[4] Kraska, T. et al. The case for learned index structures. (SIGMOD 2018)
[5] Gu, J. et al. Tiresias: A GPU Cluster Manager for Distributed Deep Learning (NSDI 2019)

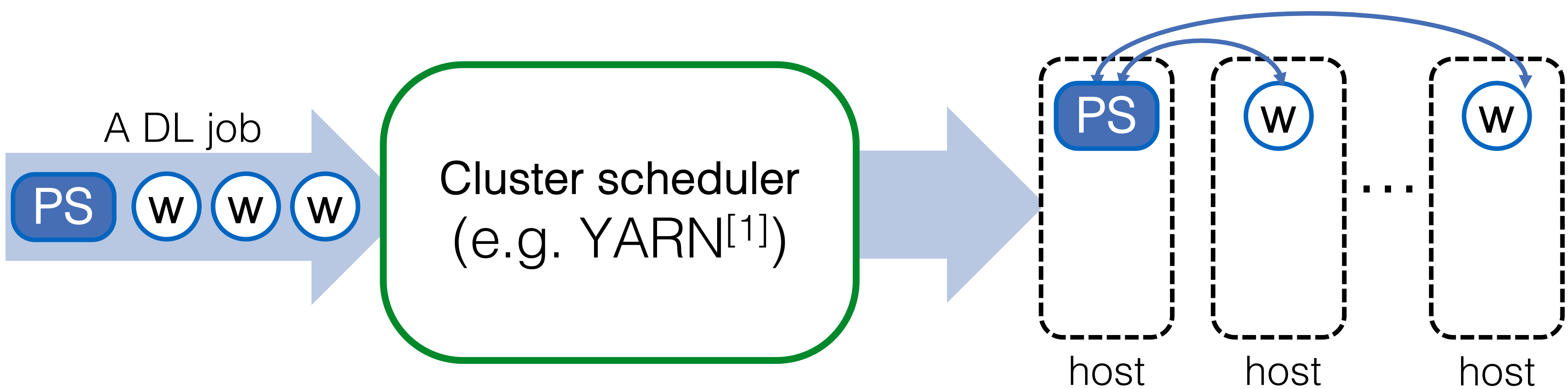
Distributed Deep Learning (DL) with Parameter Server (PS)



[1] Szegedy, C. et al. Going Deeper with Convolutions (CVPR '15)

Supporting DL at Scale

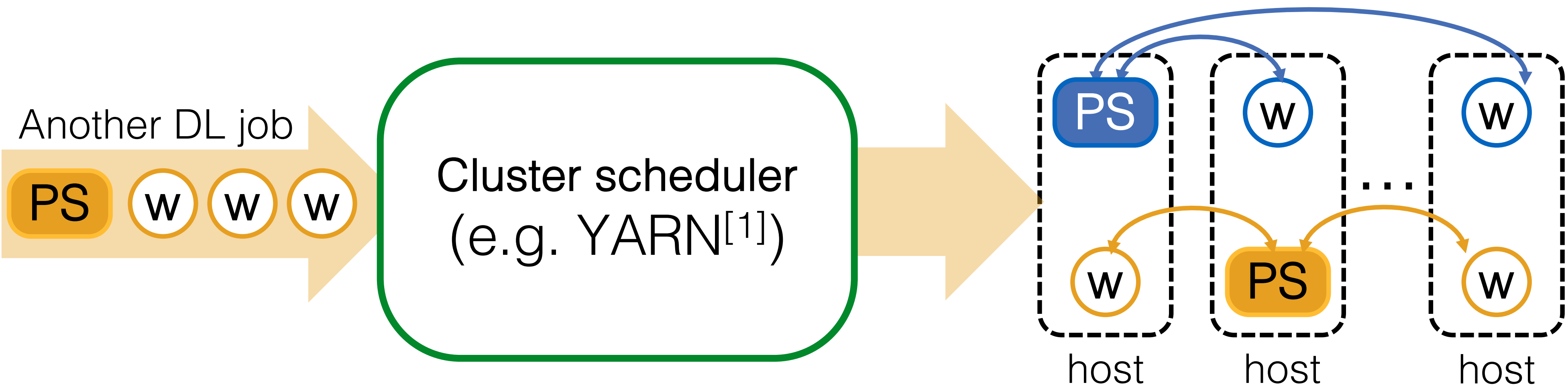
- **Cluster scheduler** to manage the lifecycles of DL jobs.
- **Grid Search**: run many DL jobs to train the same model of different hyperparameter configurations (e.g. model initialization methods) to find the best set of model configurations.



[1] Vavilapalli, V. K. et al. Apache Hadoop YARN: Yet another resource negotiator. (ACM SoCC 2013)

Supporting DL at Scale

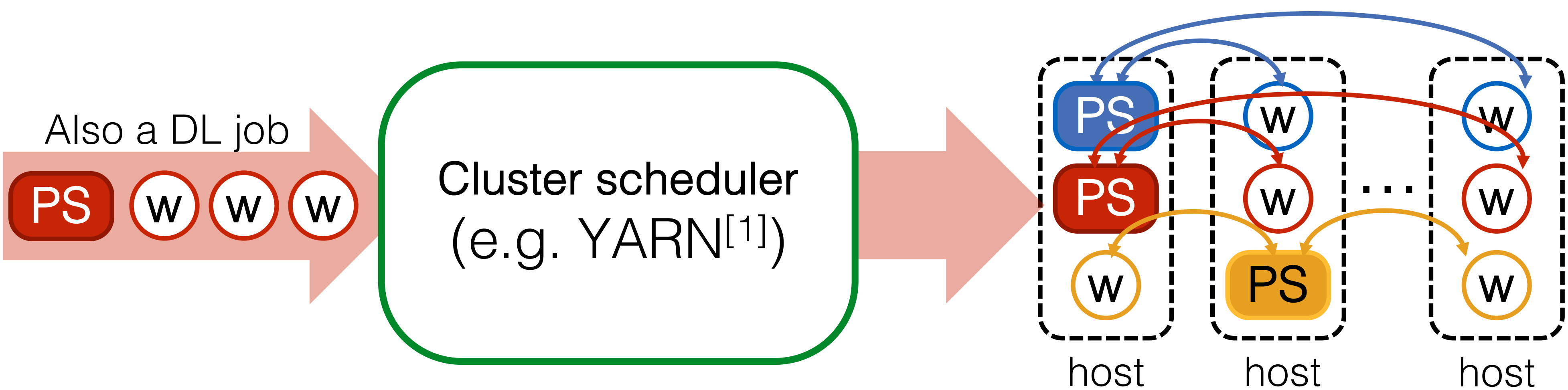
- **Cluster scheduler** to manage the lifecycles of DL jobs.
- **Grid Search**: run many DL jobs to train the same model of different hyperparameter configurations (e.g. model initialization methods) to find the best set of model configurations.



[1] Vavilapalli, V. K. et al. Apache Hadoop YARN: Yet another resource negotiator. (ACM SoCC 2013)

Supporting DL at Scale

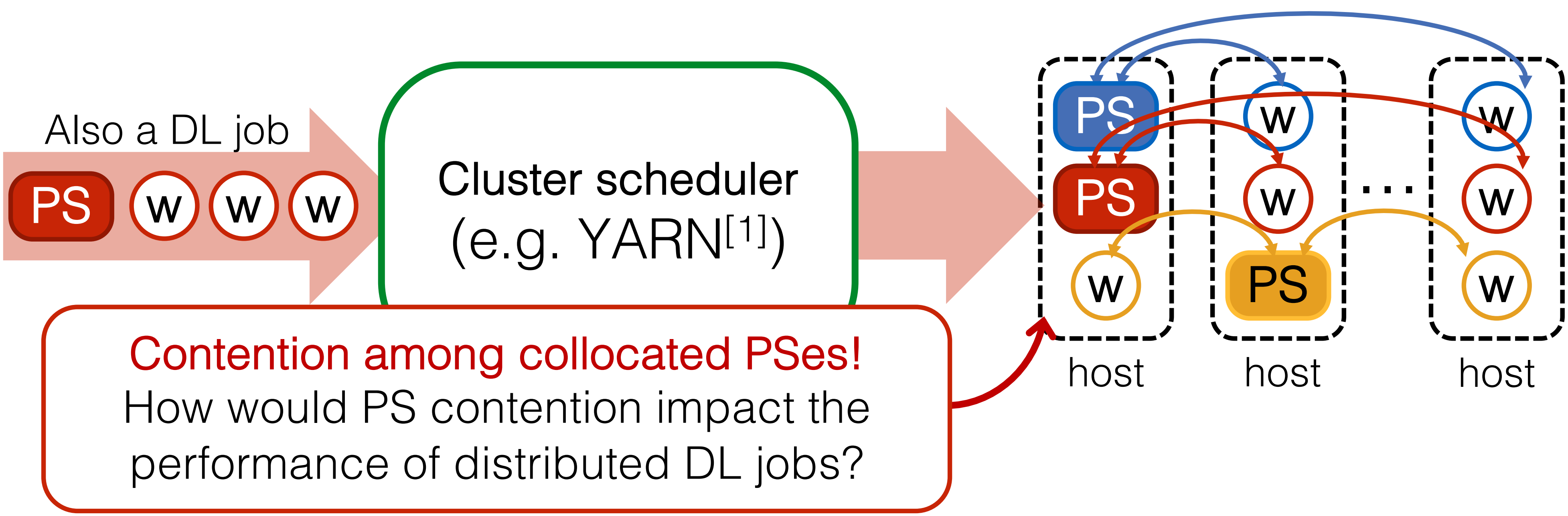
- **Cluster scheduler** to manage the lifecycles of DL jobs.
- **Grid Search**: run many DL jobs to train the same model of different hyperparameter configurations (e.g. model initialization methods) to find the best set of model configurations.



[1] Vavilapalli, V. K. et al. Apache Hadoop YARN: Yet another resource negotiator. (ACM SoCC 2013)

Supporting DL at Scale

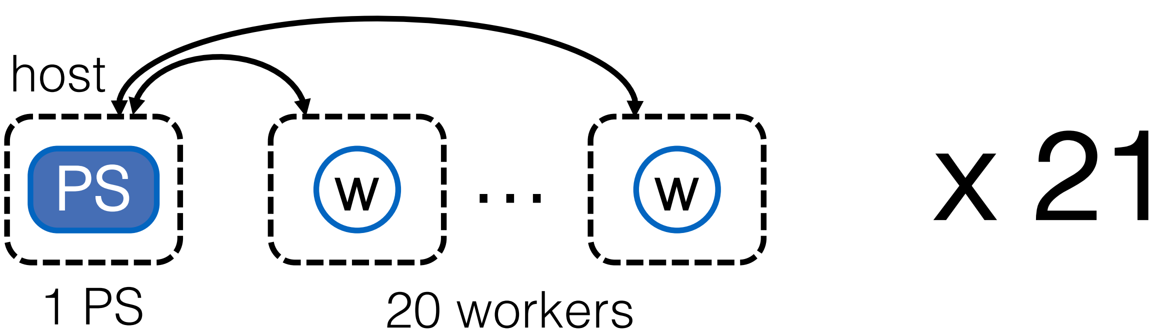
- **Cluster scheduler** to manage the lifecycles of DL jobs.
- **Grid Search**: run many DL jobs to train the same model of different hyperparameter configurations (e.g. model initialization methods) to find the best set of model configurations.



[1] Vavilapalli, V. K. et al. Apache Hadoop YARN: Yet another resource negotiator. (ACM SoCC 2013)

Measurement Setup

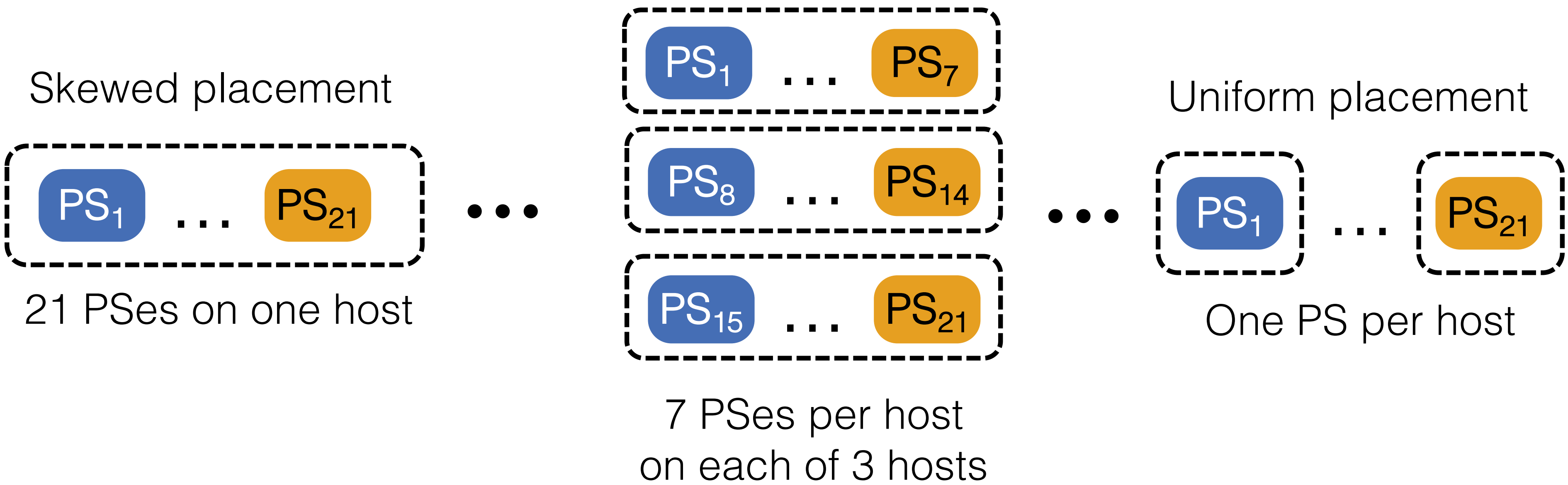
- **Workload**:
  - Each TensorFlow[1] job: 1 parameter server (PS) and 20 workers, all tasks on a different machine.
  - Each job trains the ResNet-32[2] model on the Cifar10[3] dataset until 30,000 global step is reached.
  - Total 21 concurrent jobs.



[1] https://www.tensorflow.org/
[2] He, K. et al. Deep residual learning for image recognition. (IEEE CVPR 2016)
[3] Krizhevsky, A. Learning multiple layers of features from tiny images. (University of Toronto Technical Report 2009)

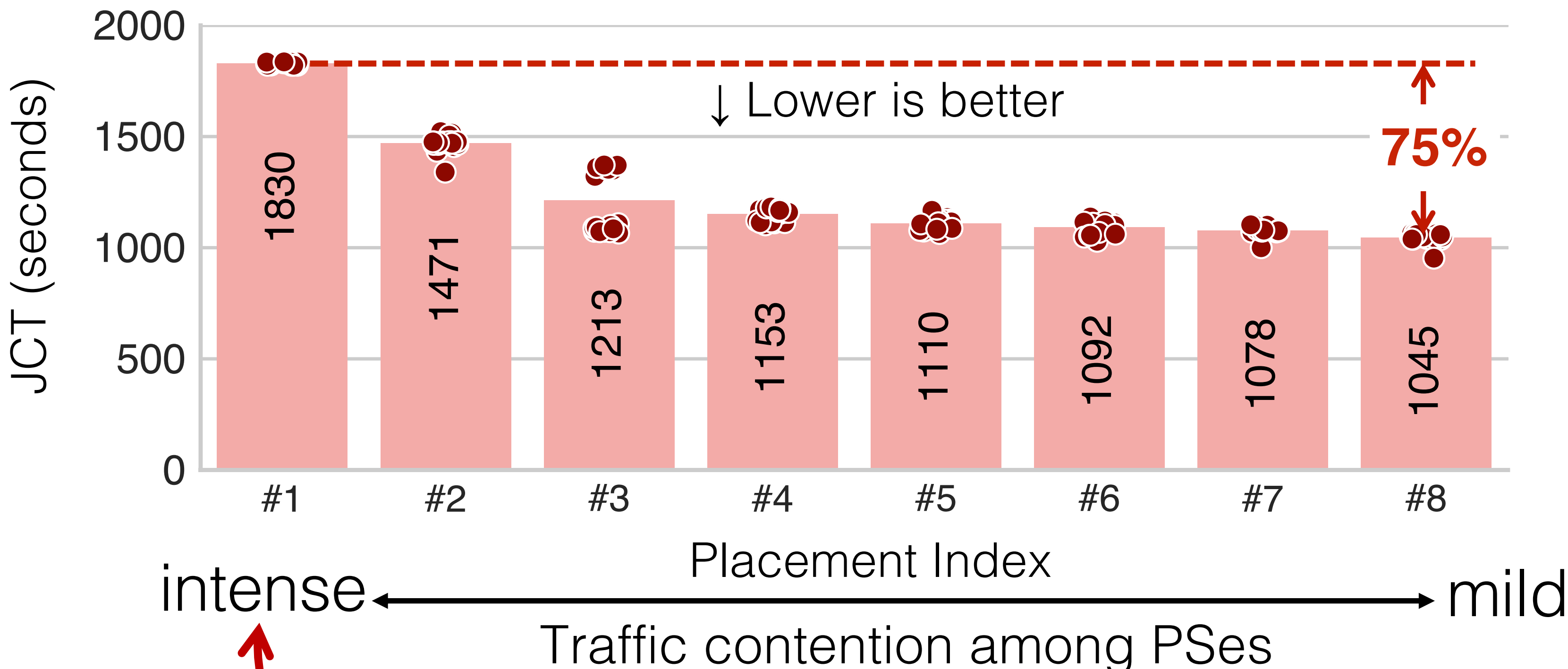
Measurement Setup (cont.)

- **Testbed**: CPU cluster with 21 hosts, all connected to an Ethernet switch with 10 Gbps link rate.
- **Task placement**: Each job's 21 tasks are on a different host. A range of PS placements from skewed to uniform.



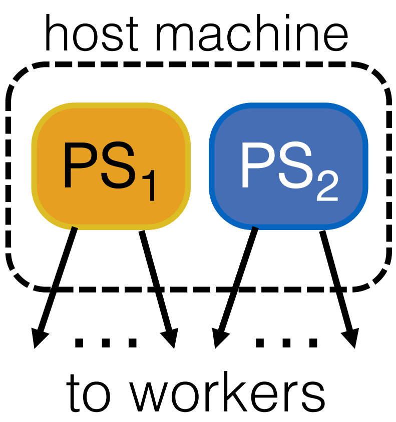
Impact of PS Placements

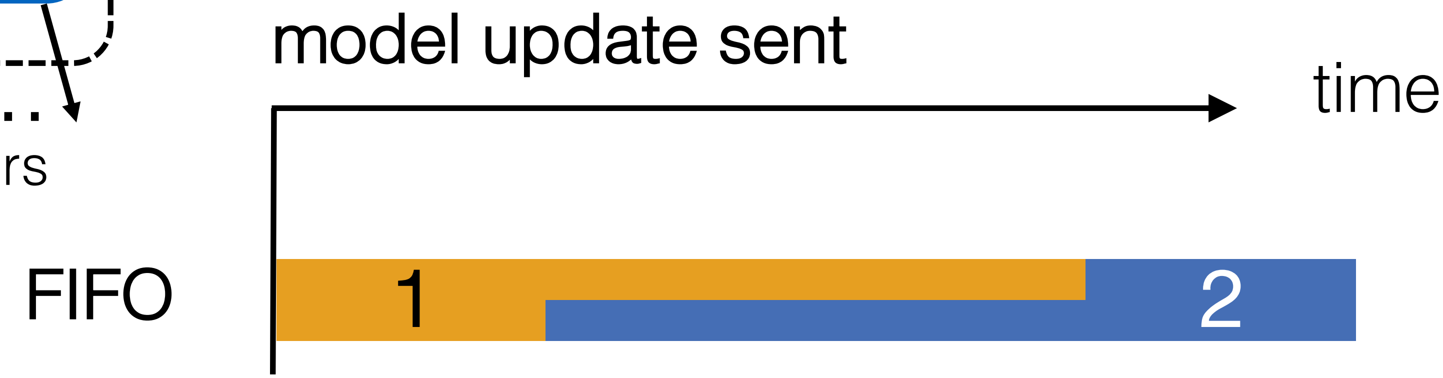
Job Completion Time (JCT) under various PS placements



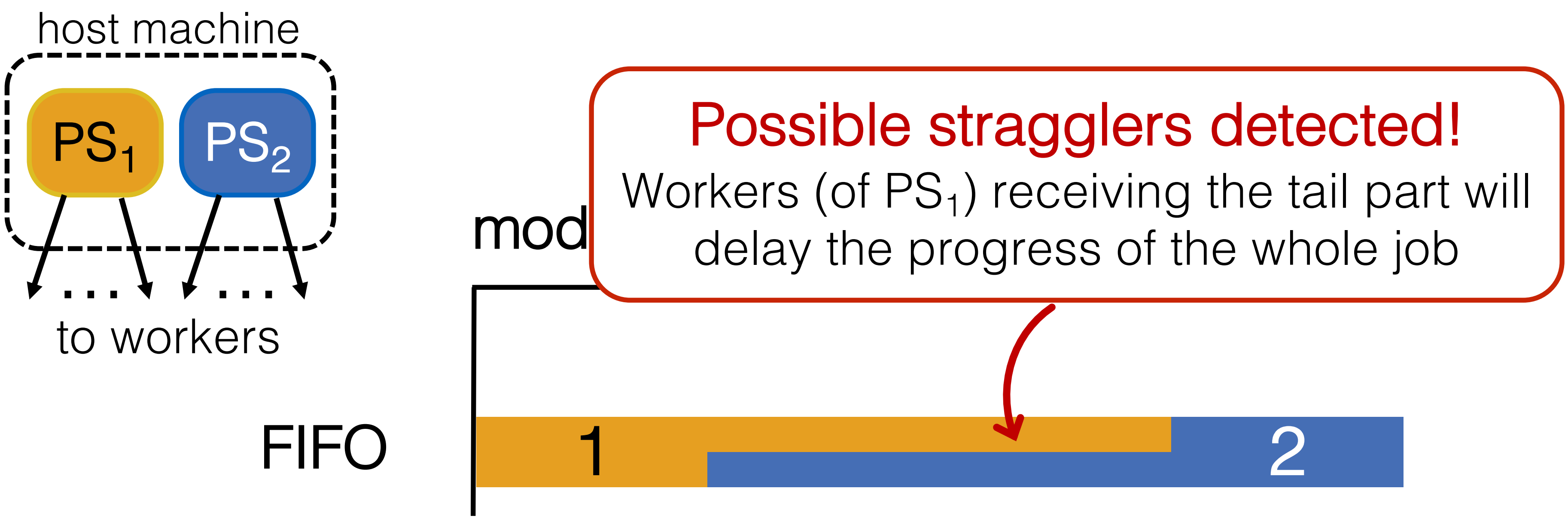
Application performance degrades due to contention at PS.

Stragglers under Contention

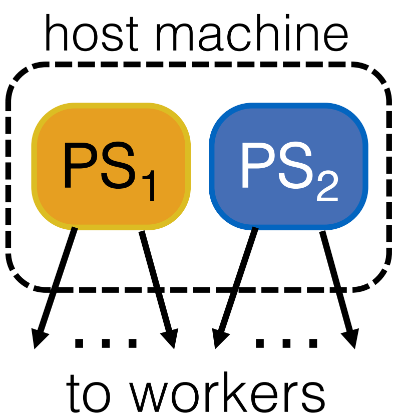




Stragglers under Contention



Stragglers under Contention

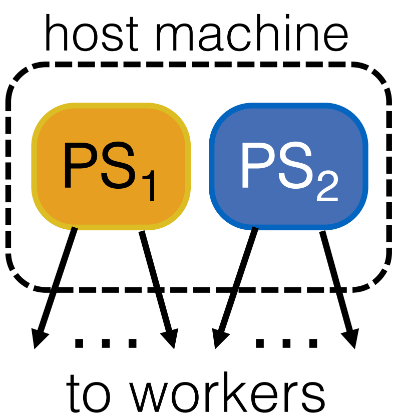


Possible stragglers detected!
Workers (of PS_1) receiving the tail part will delay the progress of the whole job

Intra-job level:
One straggling worker delays **the whole job** including other workers.

Inter-job level:
Multiple jobs are delayed simultaneously if each job has a few stragglers.

Stragglers under Contention



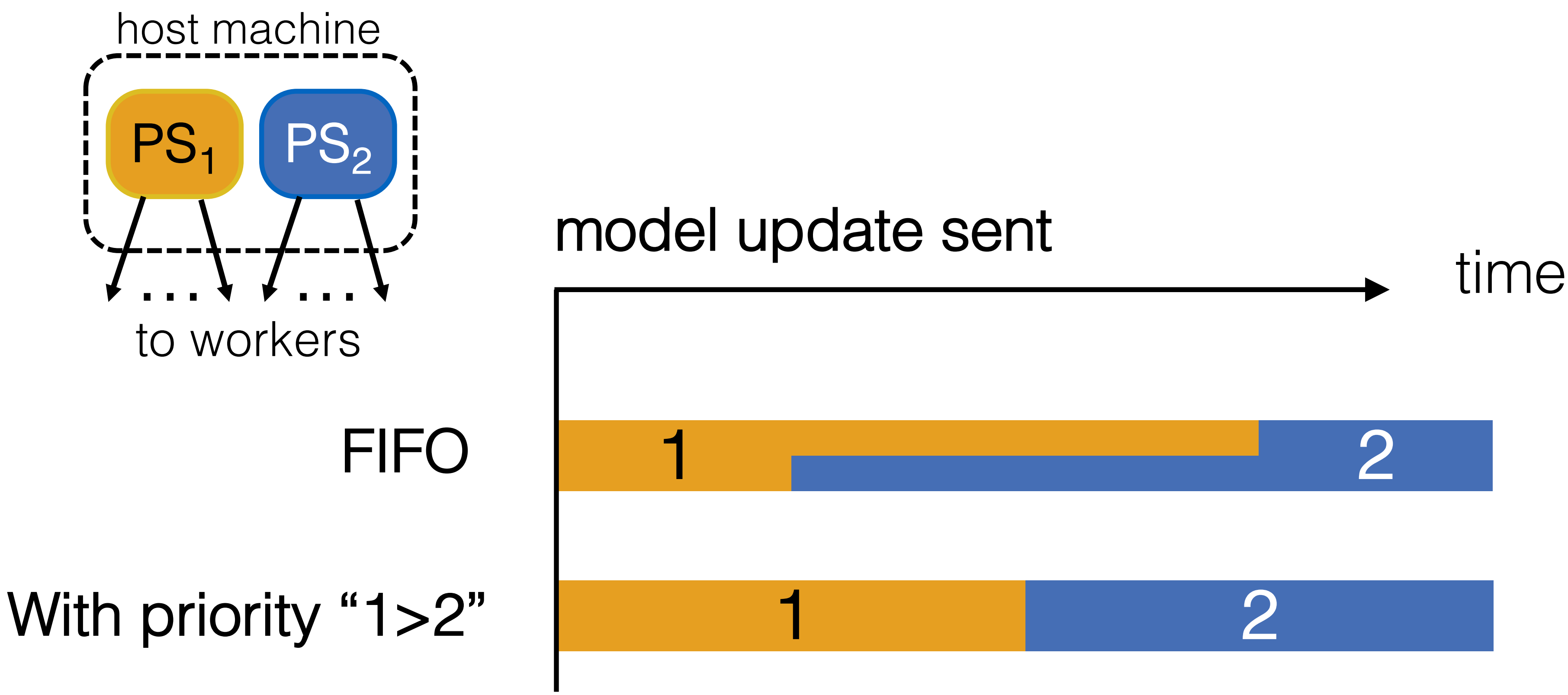
Possible stragglers detected!
Workers (of PS_1) receiving the tail part will delay the progress of the whole job

Intra-job level:
One straggling worker delays **the whole job** including other workers.

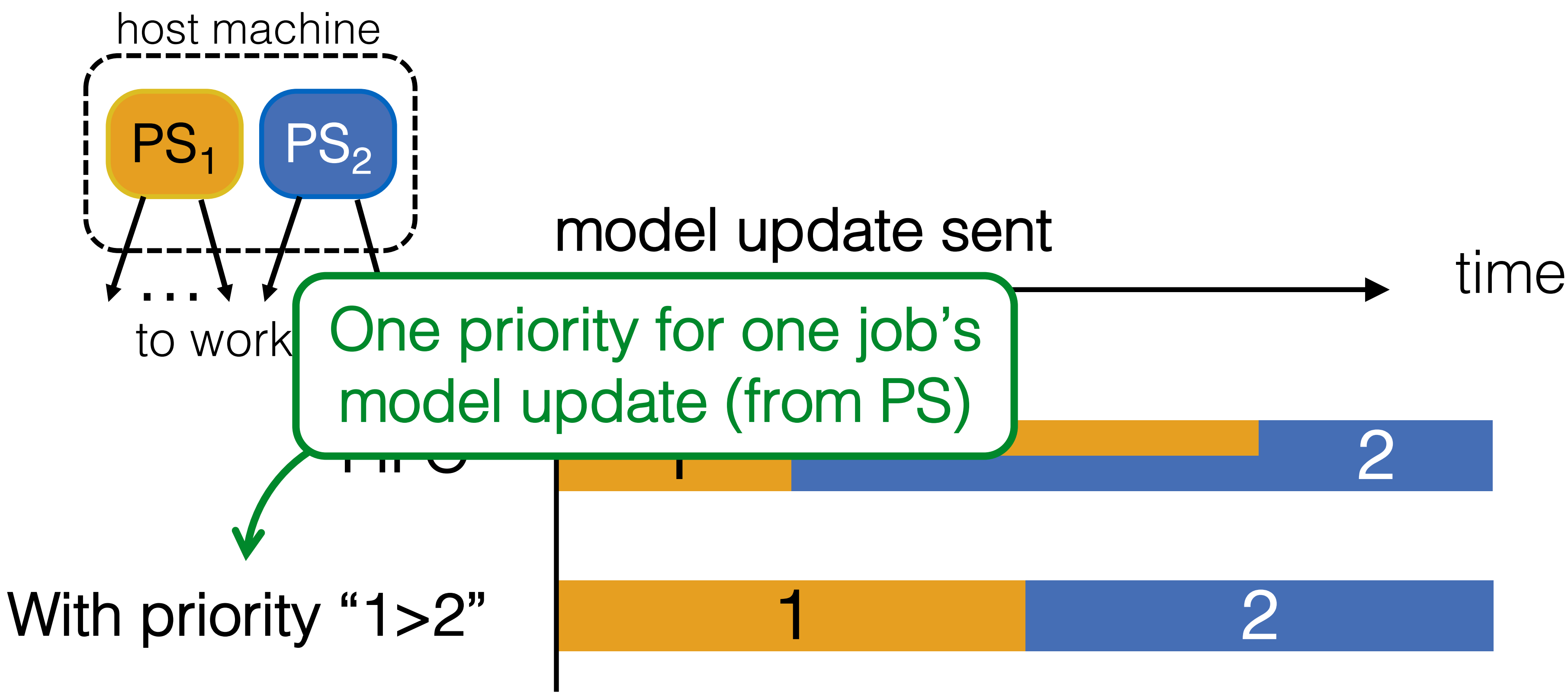
Inter-job level:
Multiple jobs are delayed simultaneously if each job has a few stragglers.

Application performance degradation and machine underutilization.

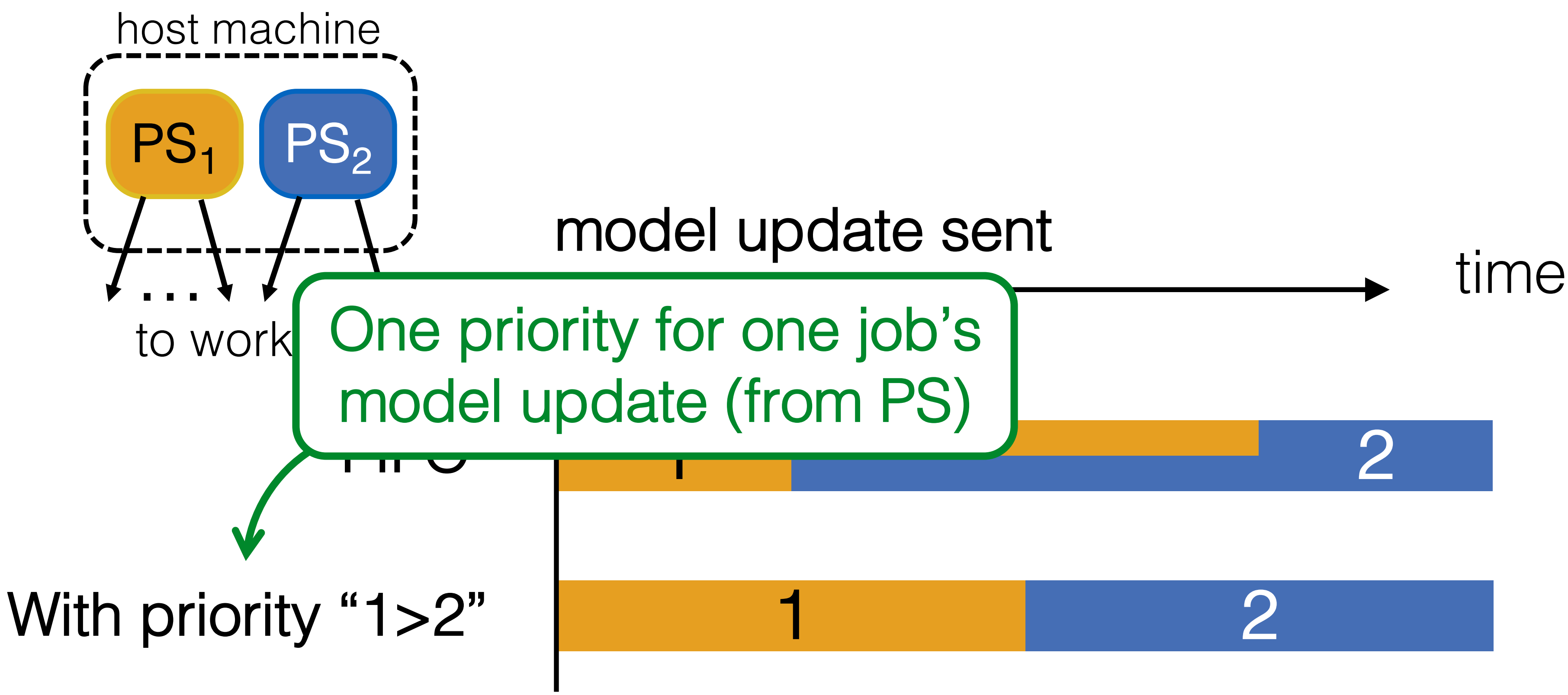
Mitigate Stragglers with Traffic Priority



Mitigate Stragglers with Traffic Priority

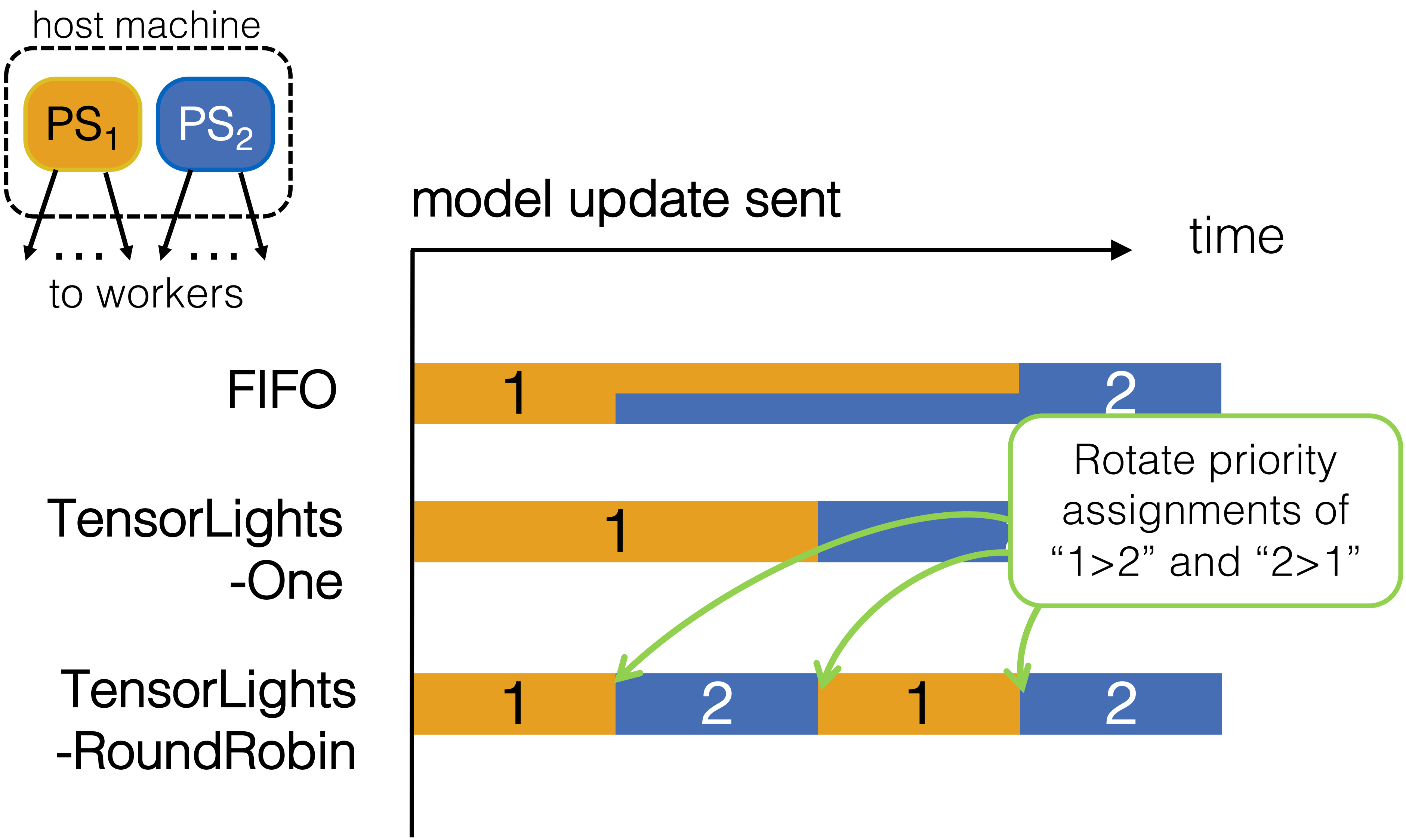


Mitigate Stragglers with Traffic Priority

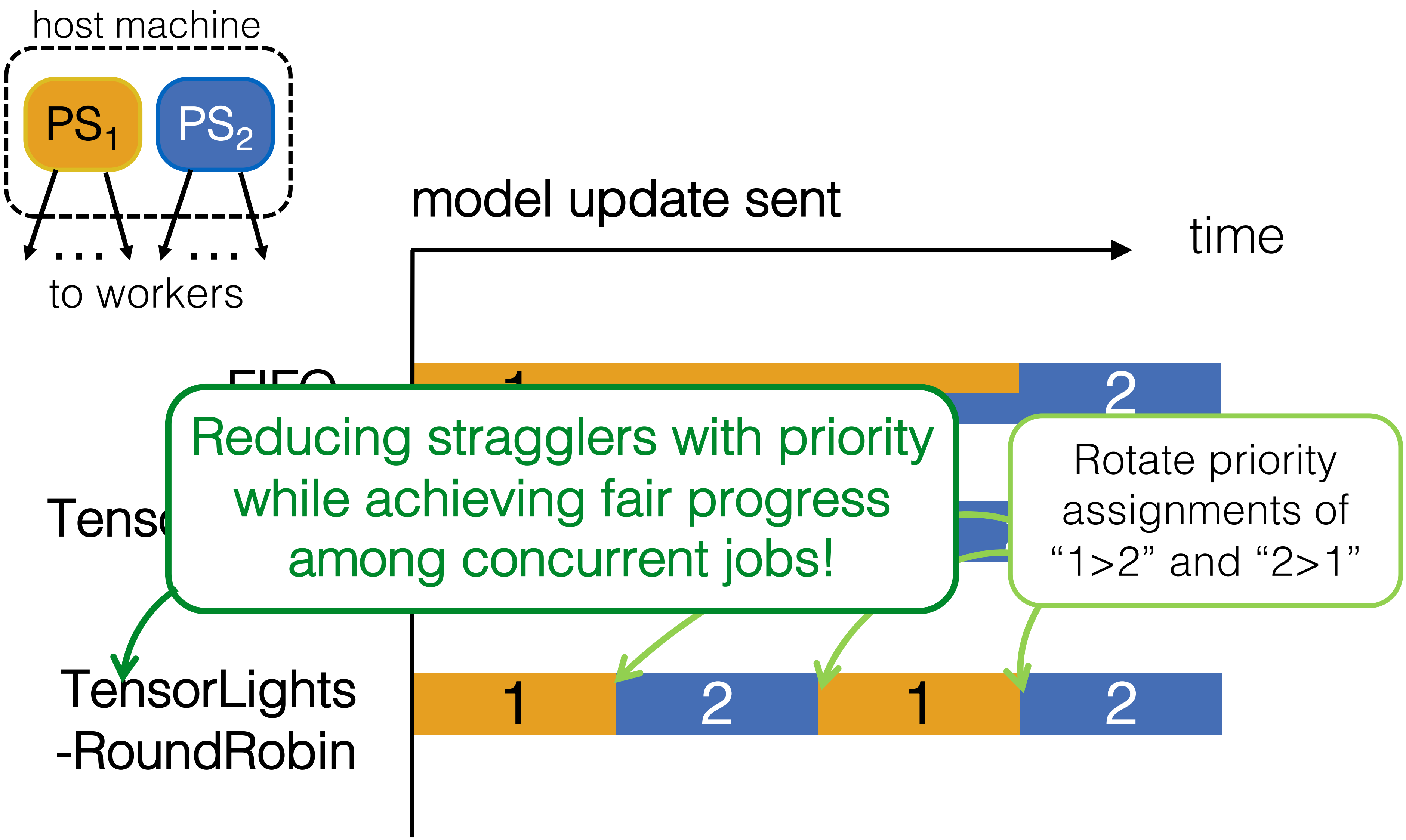


Traffic prioritization mitigates stragglers:
workers of the same job are expected to wait for similar lengths of time.

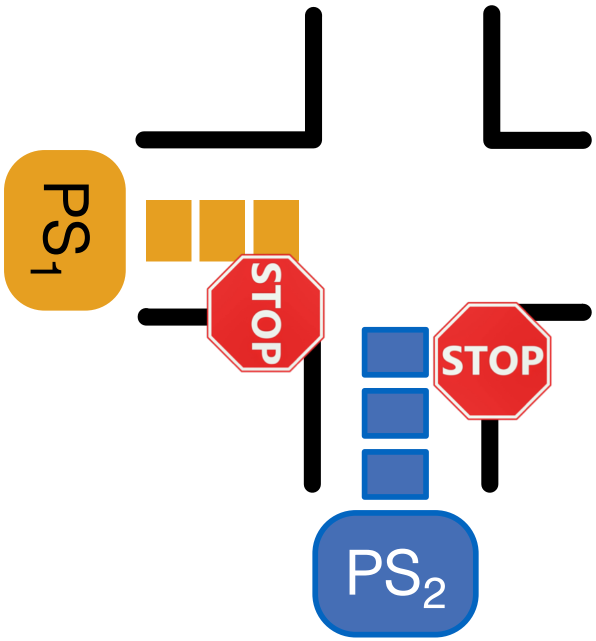
Reducing Stragglers with TensorLights



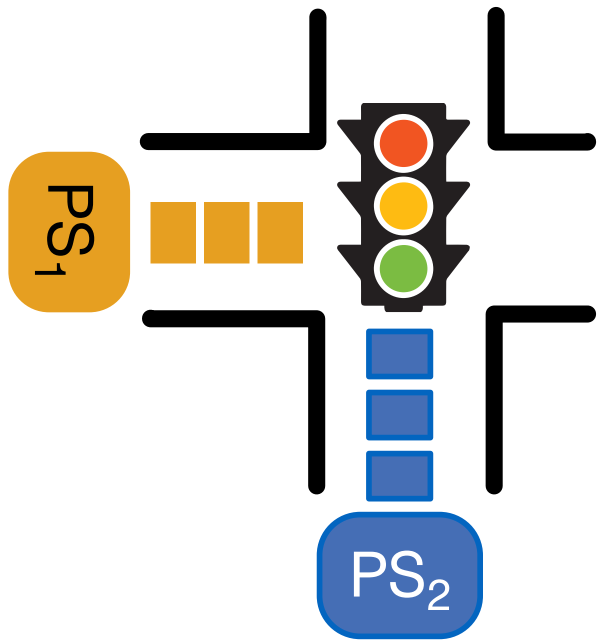
Reducing Stragglers with TensorLights



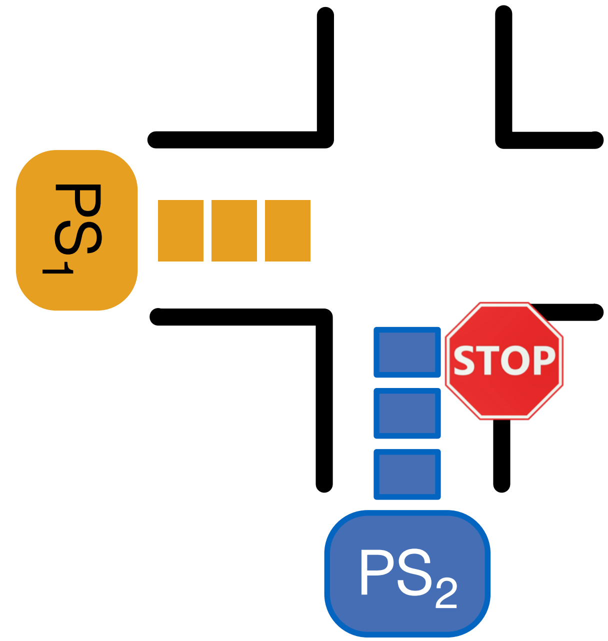
Scheduling Model with TensorLights



FIFO



TensorLights -RoundRobin



TensorLights -One

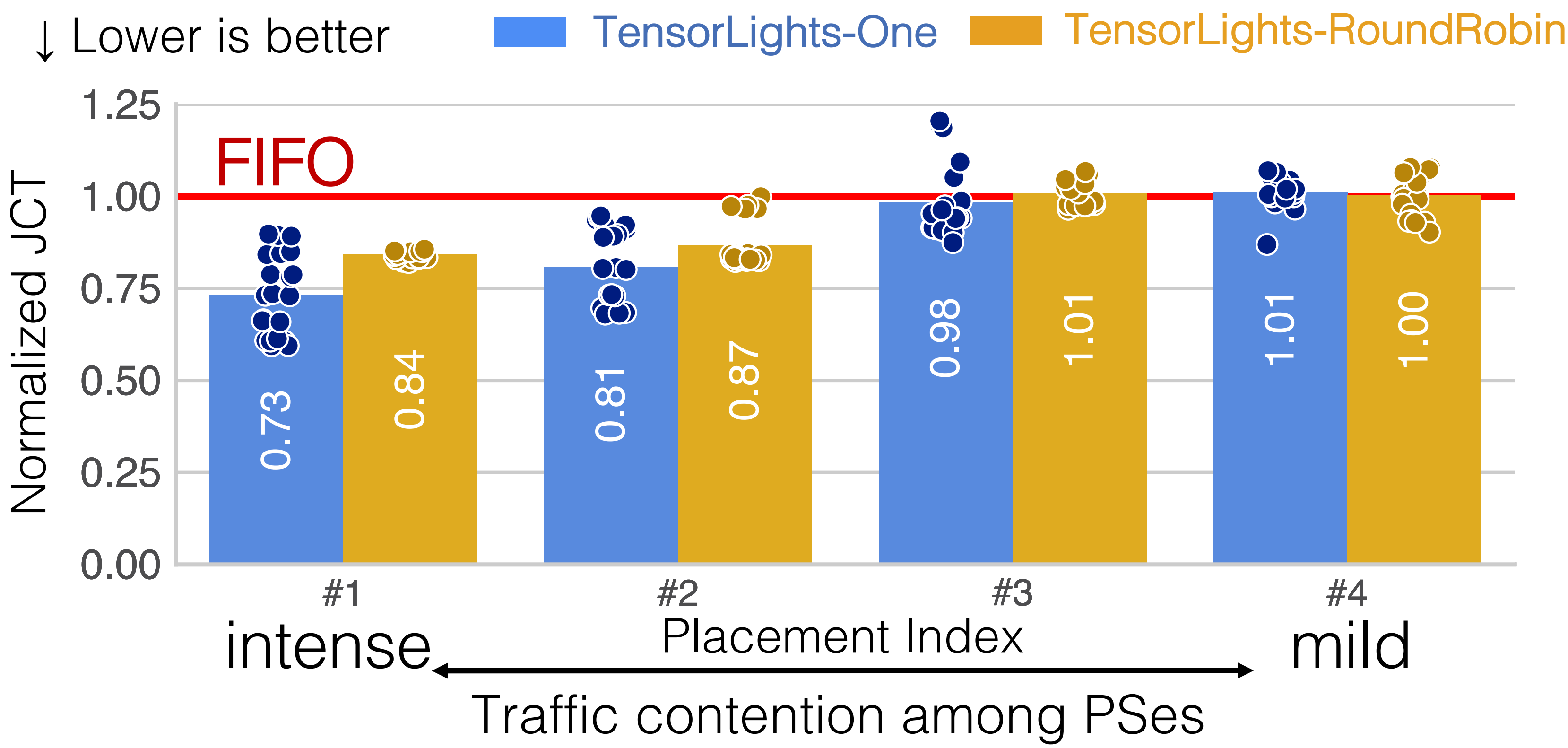
	TensorLights
Resource scheduling	✓ Work conserving
Scheduling overhead	✓ Local, light-weight
Deployment	✓ No change to app, cluster scheduler, or hardware

Evaluation

- Workload, testbed, and task placement: same as the previous measurement study.
- TensorLights implementation: Hierarchical token bucket (htb) in the traffic control (tc) module under Linux. Deployed at local host that has concurrent PSes.
- Results:
    - Improvement in job completion time
    - Improvement in barrier waiting efficiency
    - Improvement in machine utilization
    - Sensitivity to traffic contention intensity

Improvement in Job Completion Time

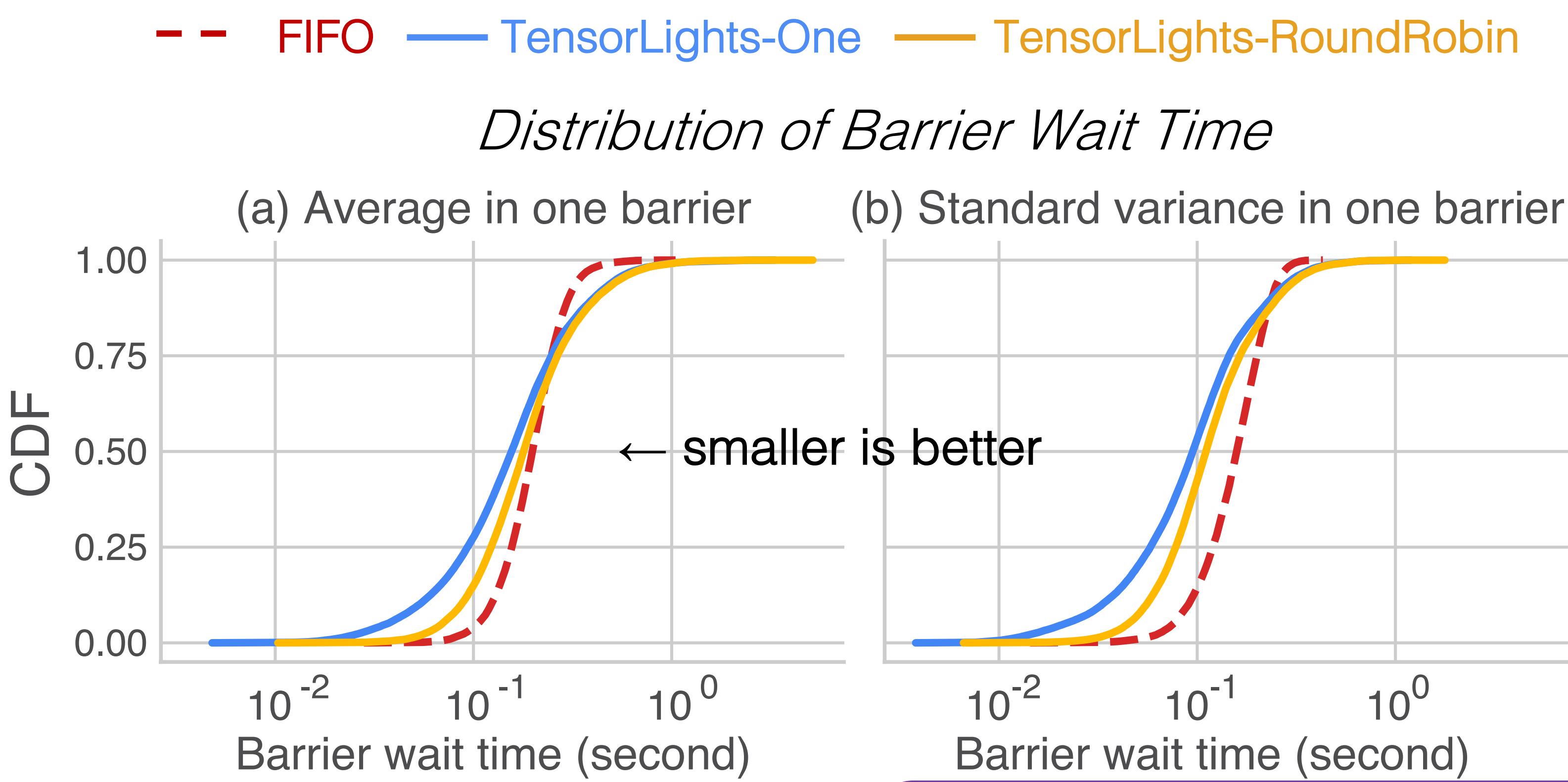
Normalized Job Completion Time (JCT) under Various PS Placementts



TensorLights is more effective for high contention case.
TensorLights improves the average JCT by up to 27%.

Reduction in Synchronization Overhead

- **Metrics**: Average (or standard variance) of elapsed waiting time for the same barrier among workers of the same job

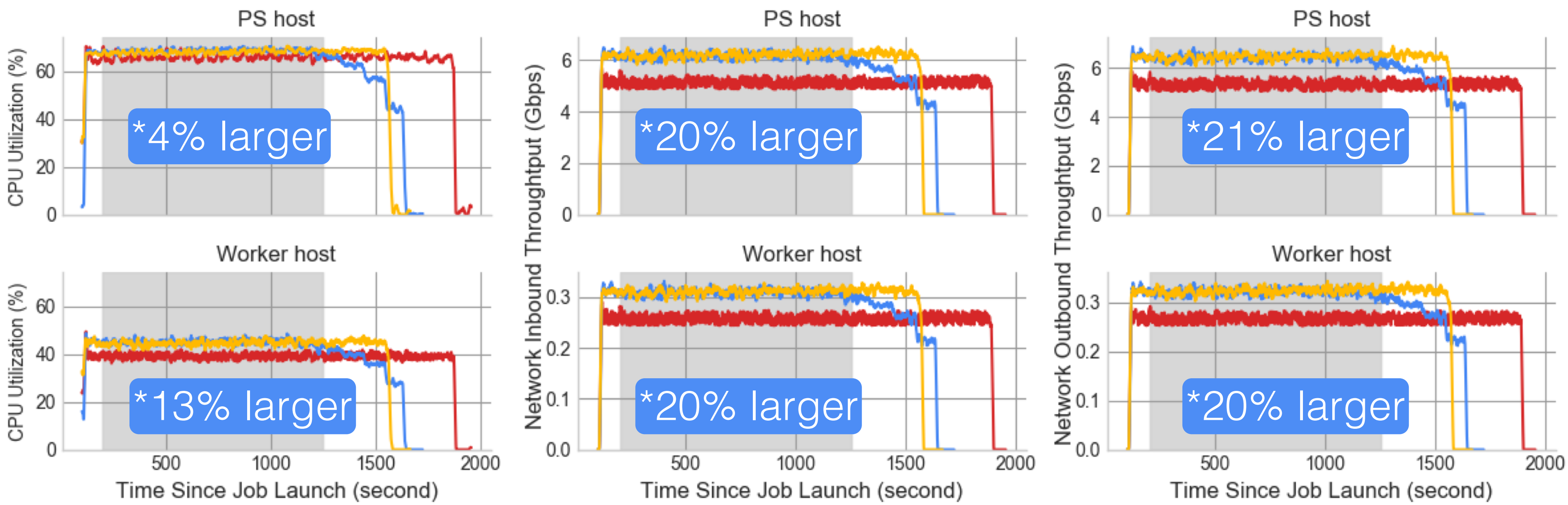


Comparable average under all policies.

TensorLights-One reduces variance by 26% on average. (TLs-RoundRobin is 15%)

Improvement in Utilization

↑ Higher is better — FIFO — TensorLights-One — TensorLights-RoundRobin



CPU | Network Inbound | Network Outbound

With more efficient barrier waiting,
TensorLights also improves machine utilization

* Presented number is for TensorLights-One. TensorLights-RoundRobin has similar results.

Conclusions

- Trends to scale up DL applications continue to introduce **more network traffic contention**.
- **Job-level traffic prioritization** is helpful to manage traffic contention.
- **TensorLights** leverages traffic prioritization to mitigate stragglers, accelerate DL jobs and increase resource utilization.

Open Source Code & Benchmark
https://github.com/TensorLights

Thank You!