Taming Unbalanced Training Workloads in Deep Learning with Partial Collective Operations

Shigang Li, Tal Ben-Nun, Salvatore Di Girolamo, Torsten Hoefler
ETH Zurich

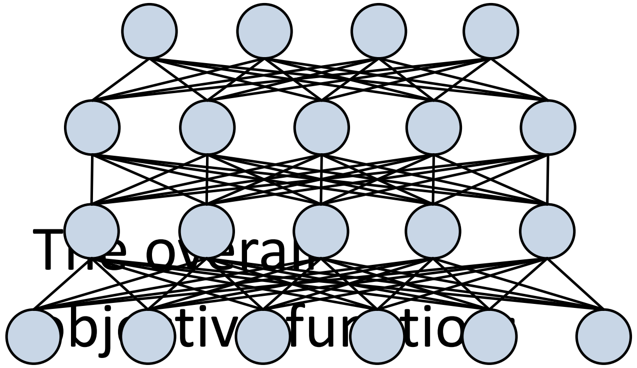
Dan Alistarh
IST Austria

PPoPP'20

Feb. 22-26, 2020
San Diego, CA, US

Deep learning training

The overall objective function



$$f(w) = \mathbb{E}_{\xi \sim D}\, F(w; \xi)$$

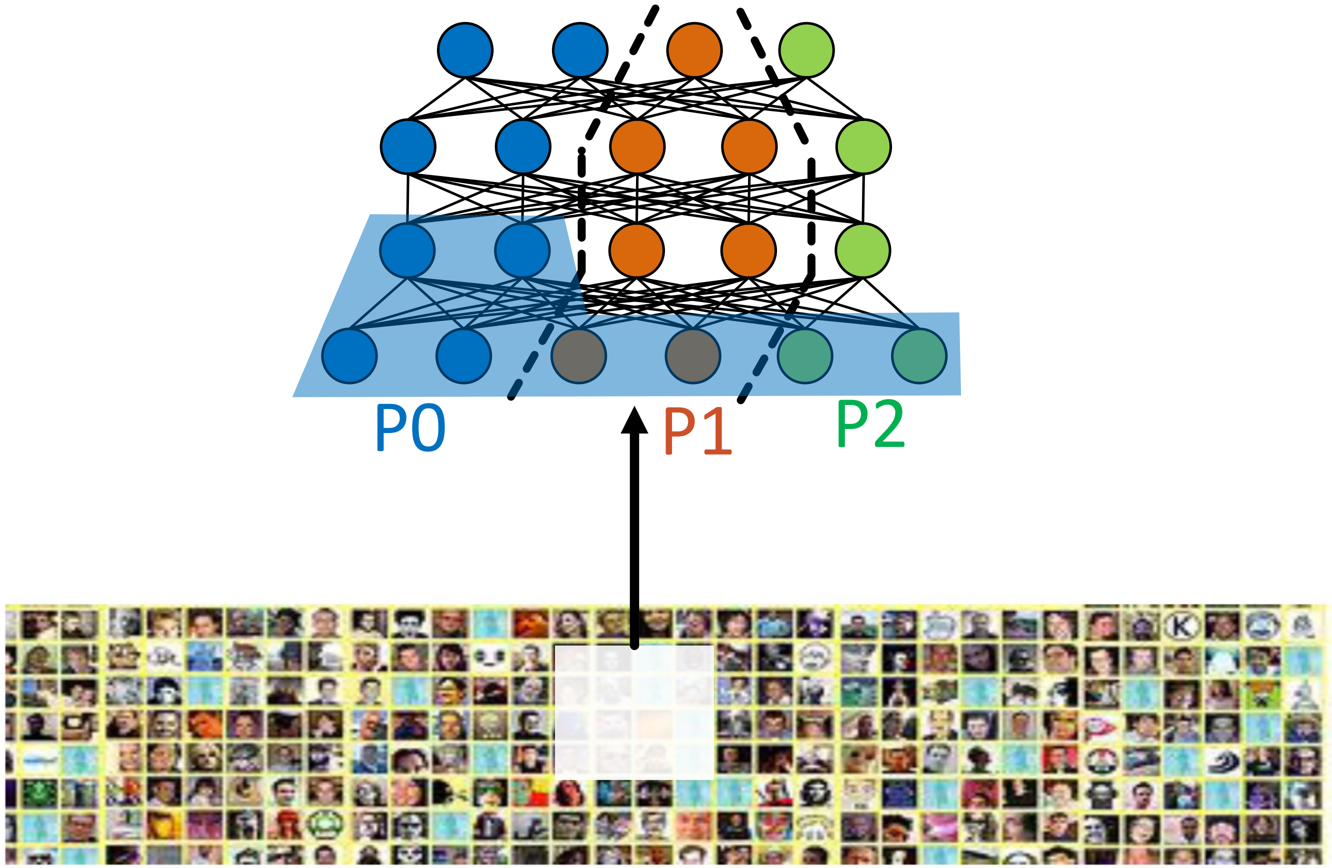
w denotes the model parameters.

F is the loss function.

ξ is a data point sampled from a distribution ***D***.

Training: optimize ***w*** to minimize ***f*** (using SGD).

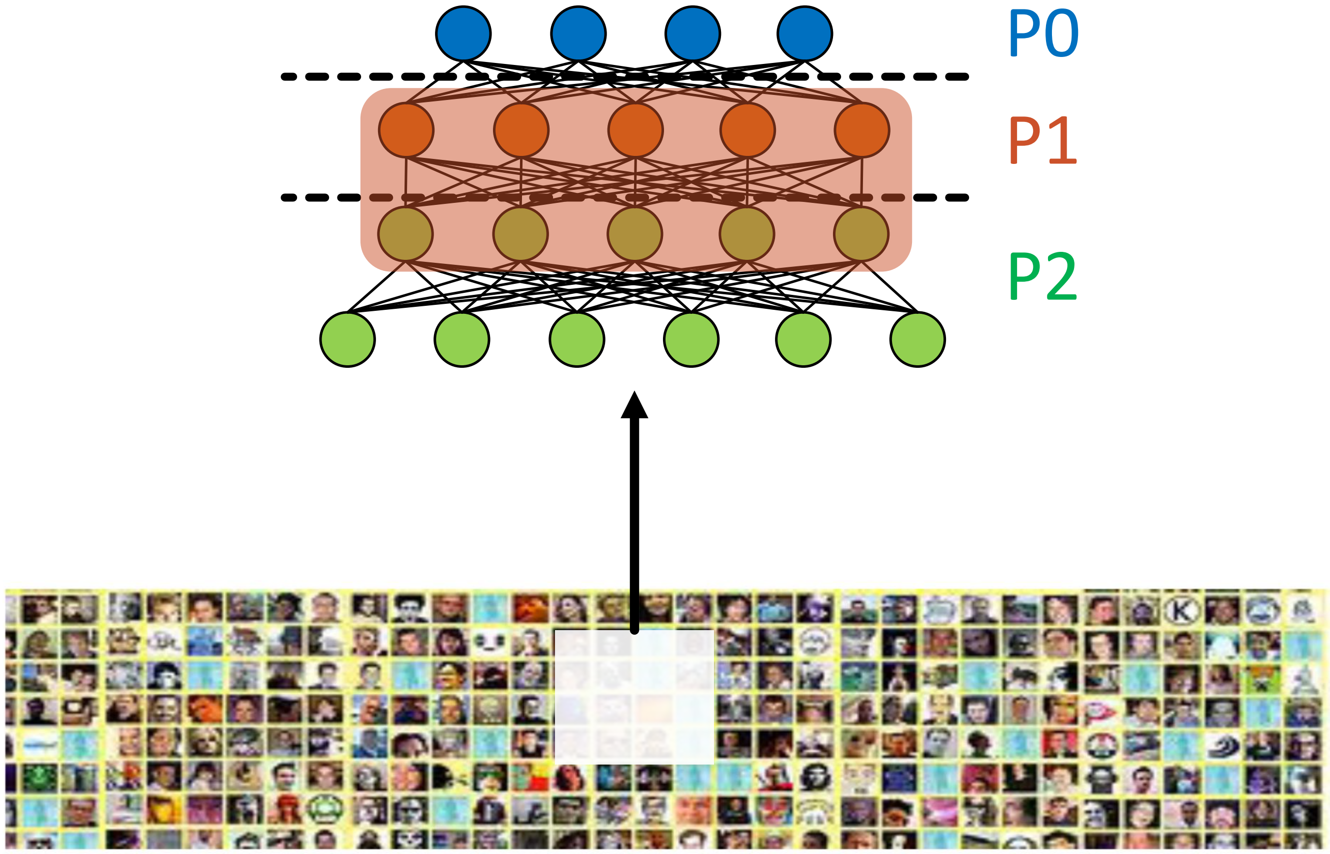
Model parallelism



Dataset

Deep learning training

Pipeline parallelism



Dataset

The overall objective function:

$$\underbrace{f(w)}_{} = \mathbb{E}_{\xi \sim D}\, \underbrace{F}_{}(w; \underbrace{\xi}_{})$$

w denotes the model parameters.

F is the loss function.

ξ is a data point sampled from a distribution ***D***.

Training: optimize ***w*** to minimize ***f*** (using SGD).

Deep learning training

The overall objective function:

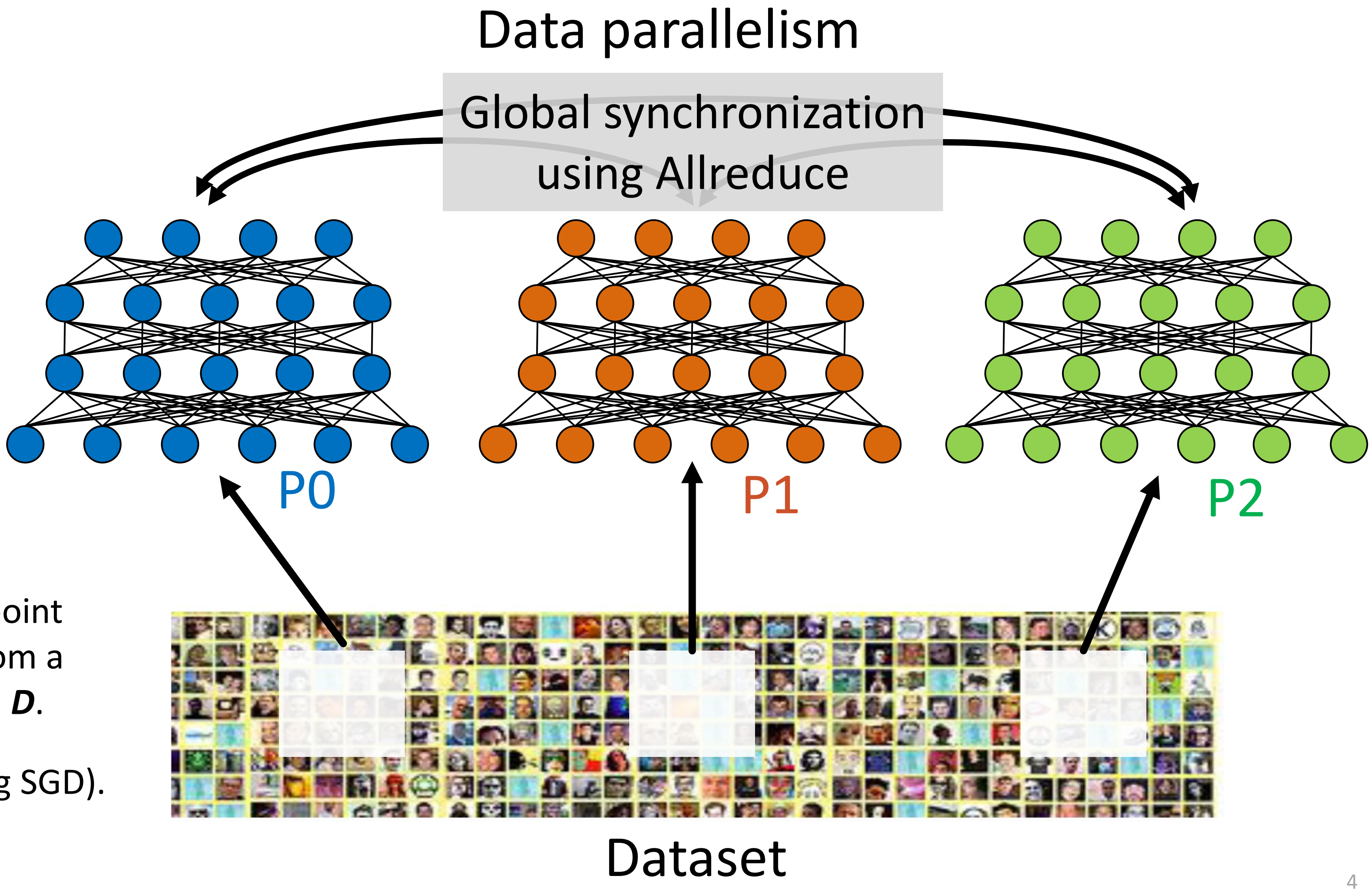
$$f(w) = \mathbb{E}_{\xi \sim D}\, F(w; \xi)$$

w denotes the model parameters.

F is the loss function.

ξ is a data point sampled from a distribution ***D***.

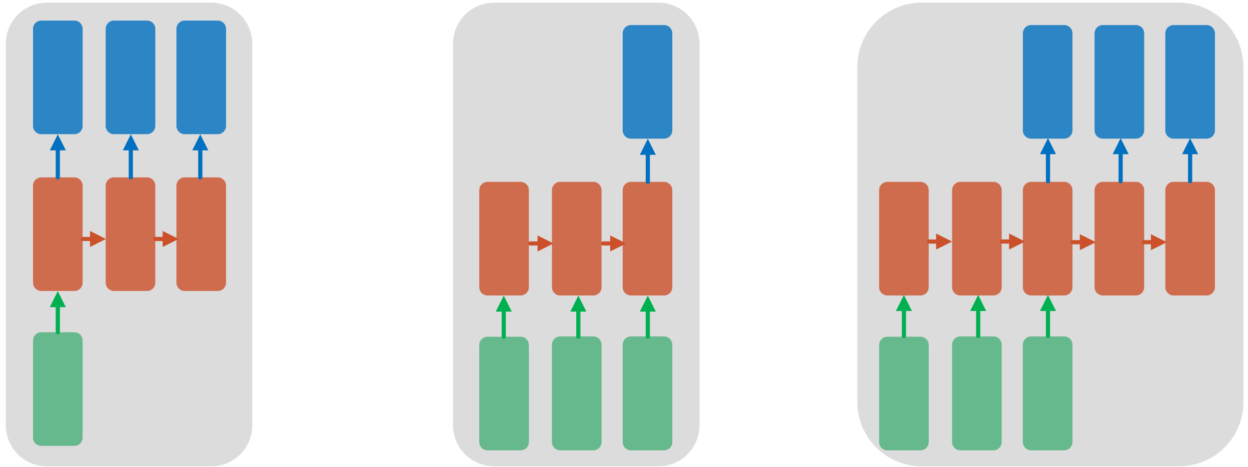
Training: optimize ***w*** to minimize ***f*** (using SGD).



Unbalanced training workloads

- **Load imbalance on application level**
  - Recurrent Neural Networks (RNN/LSTM/GRU)
  - Transformers

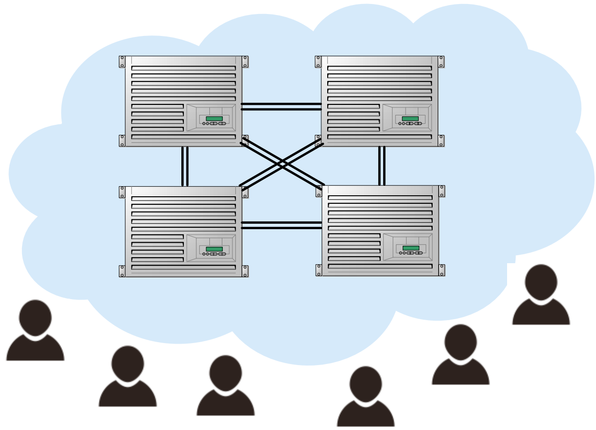
Challenge: stragglers dominate the performance.



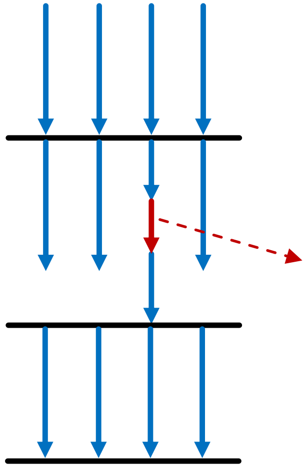
(One input multiple outputs) **(Multiple inputs one output)** **(Multiple inputs multiple outputs)**

Different types of RNNs

- **Load imbalance on system level**
  - Performance variability on multitenant cloud systems
  - System or network noise

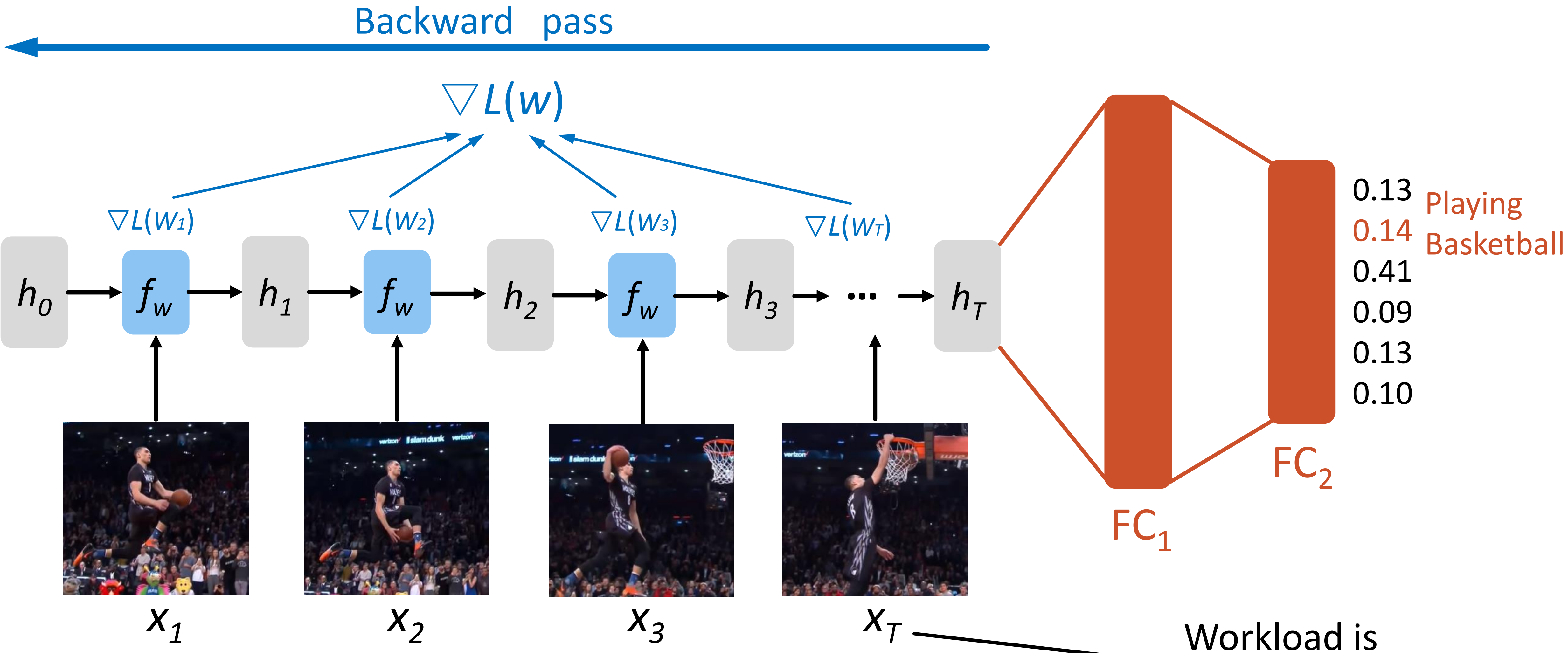


Multitenant cloud system



Interrupts, daemon, page/cache misses, et al.

Many-to-one RNN for video classification

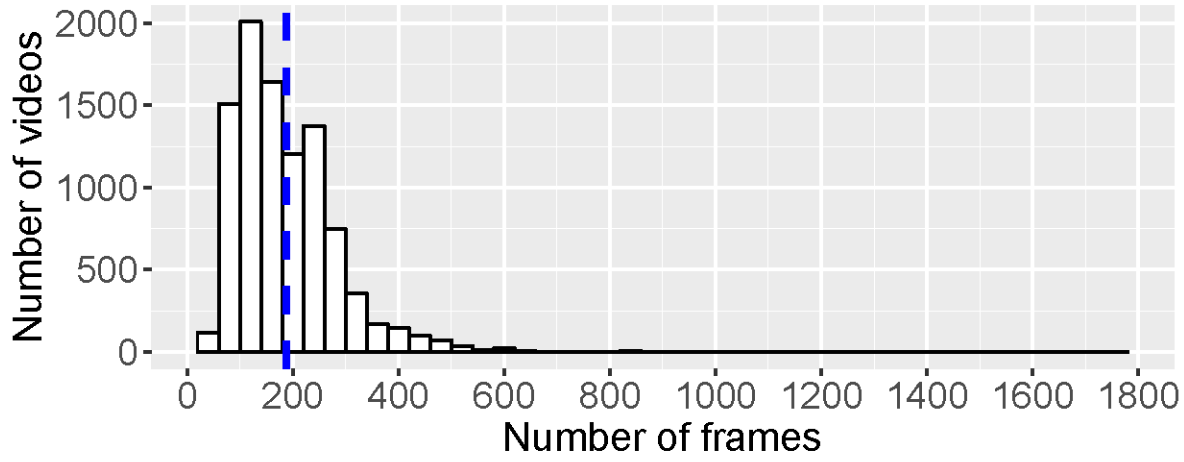


Workload is proportional to ***T***

RNN: $h_t = f_W(h_{t-1}, x_t)$

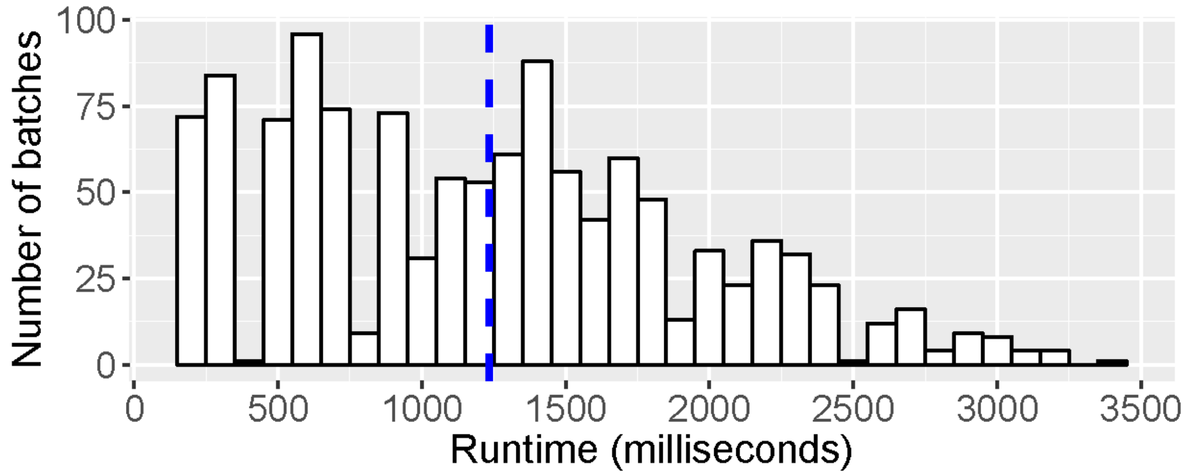
Workload statistics for video classification

Distribution: 29 ~ 1,776 frames
Mean: 187 frames
Standard deviation: 97 frames



(a) **Video length distribution** for UCF101 dataset

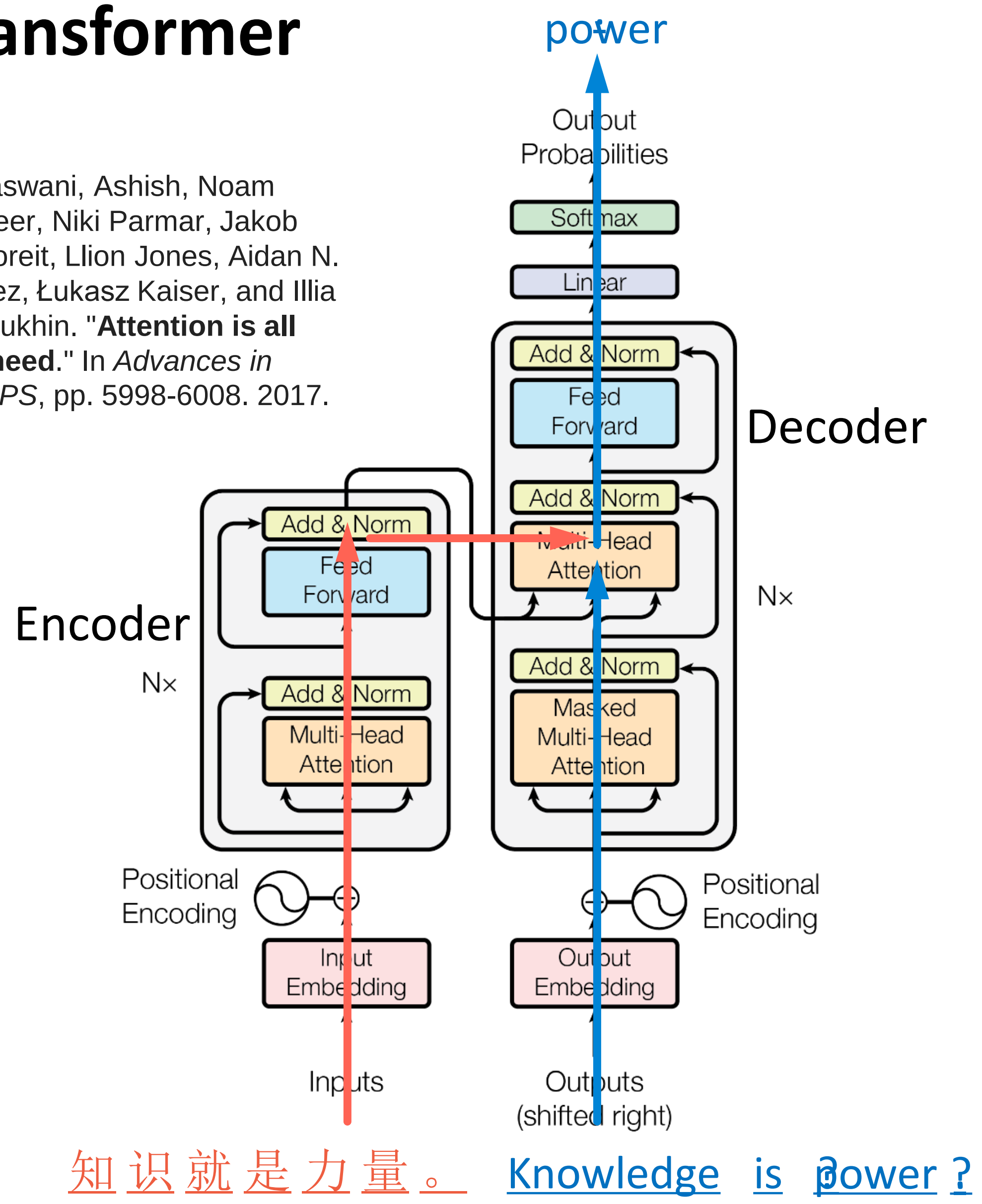
Distribution: 201 ~ 3,410 ms
Mean: 1,235 ms
Standard deviation: 706 ms



(b) **Runtime distribution** for the mini-batches to train a LSTM model on P100

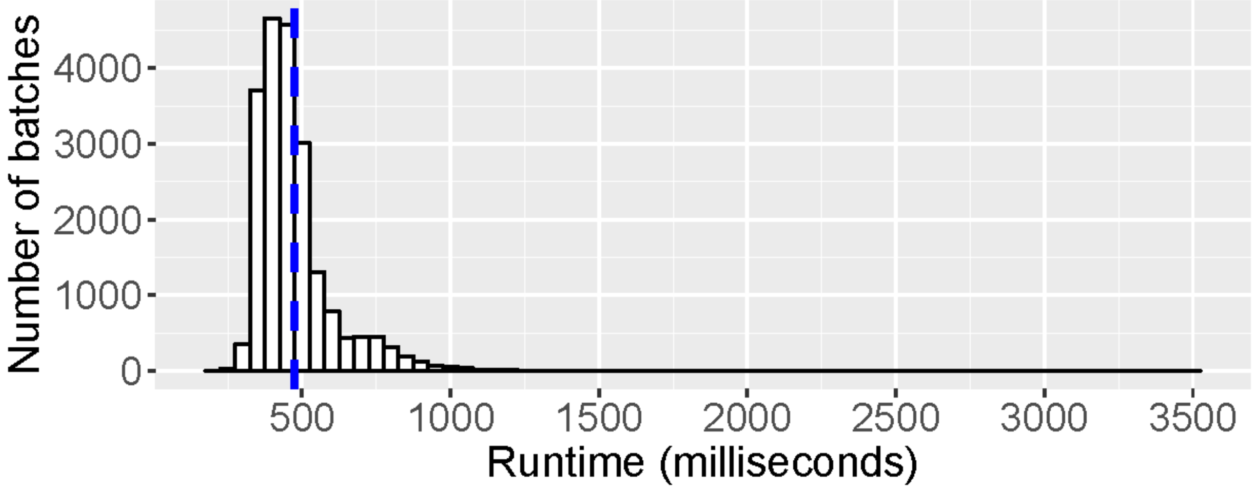
Transformer

[1] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "**Attention is all you need**." In *Advances in NeurIPS*, pp. 5998-6008. 2017.



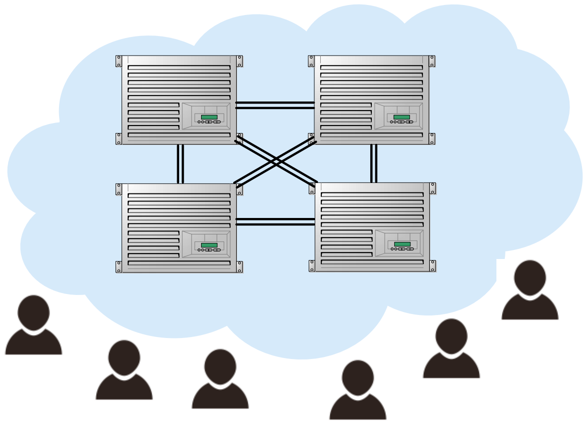
The workload is proportional to ***input_size*** * ***output_size*** .

Distribution: 179 ~ 3,482 ms
Mean: 475 ms
Standard deviation: 144 ms

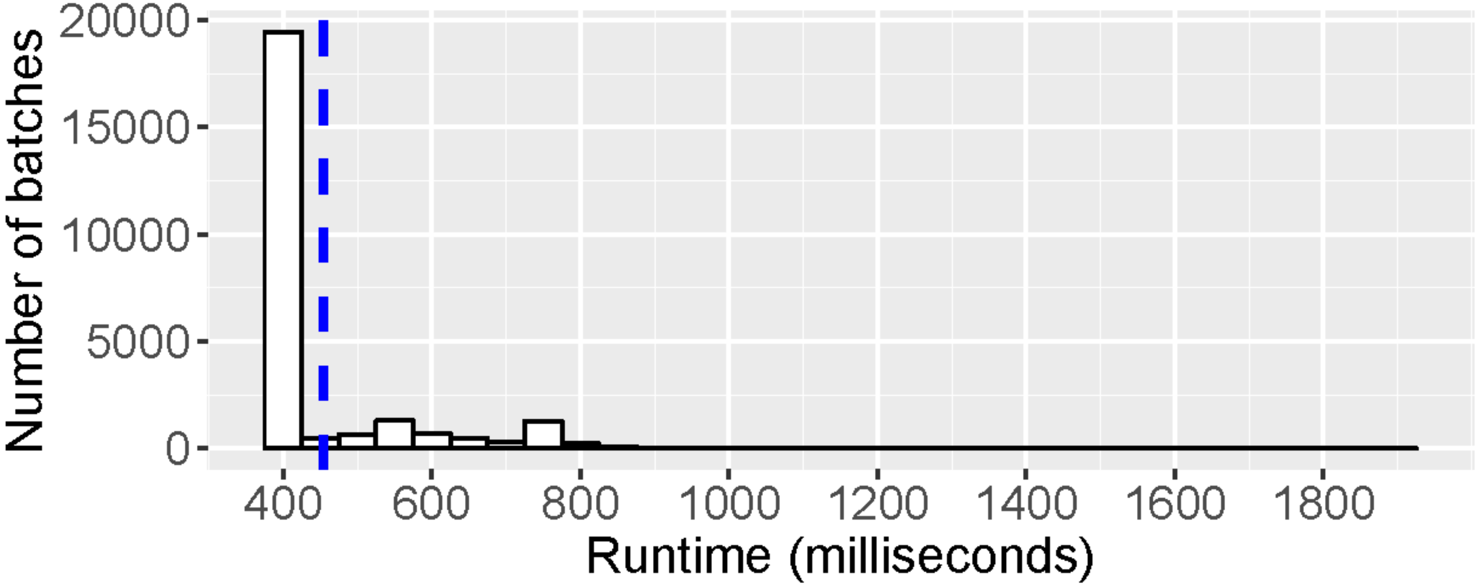


Runtime distribution for the mini-batches to train a Transformer model (using WMT16) on P100

Training on Cloud



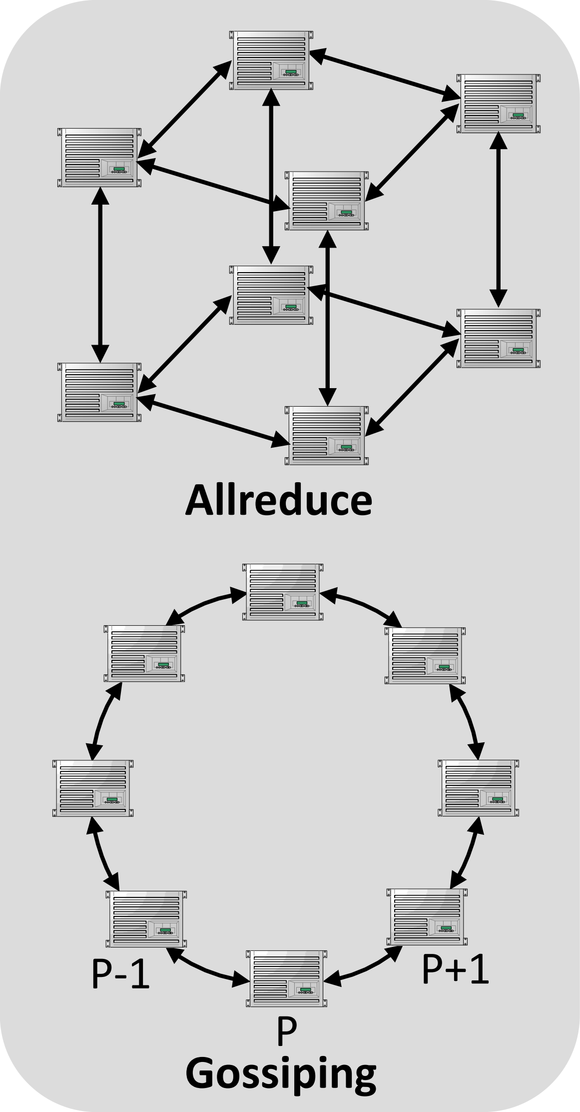
Distribution: 399 ~ 1,892 ms
Mean: 454 ms
Standard deviation: 116 ms

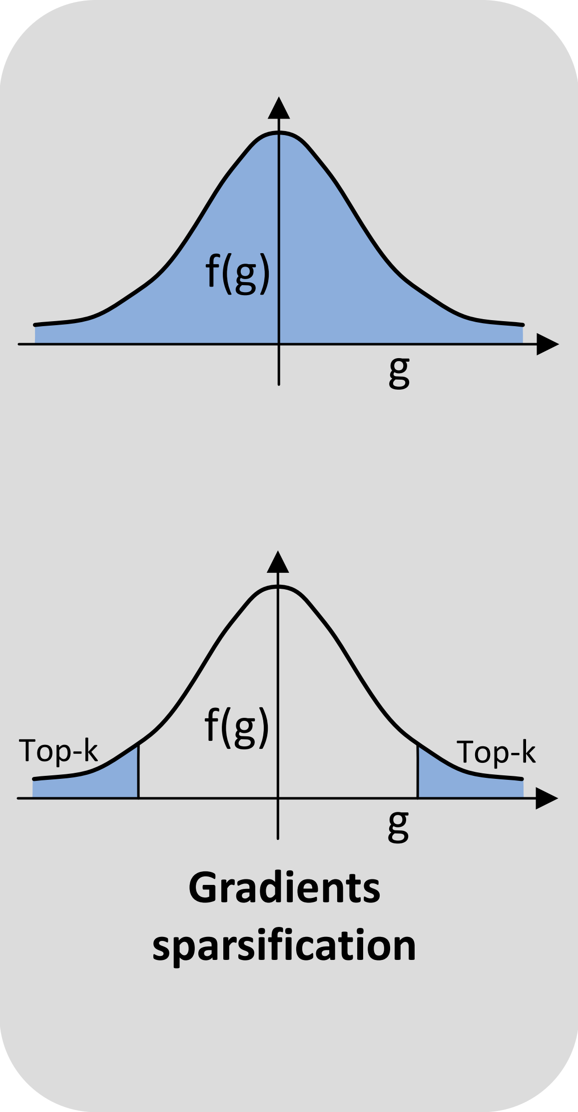


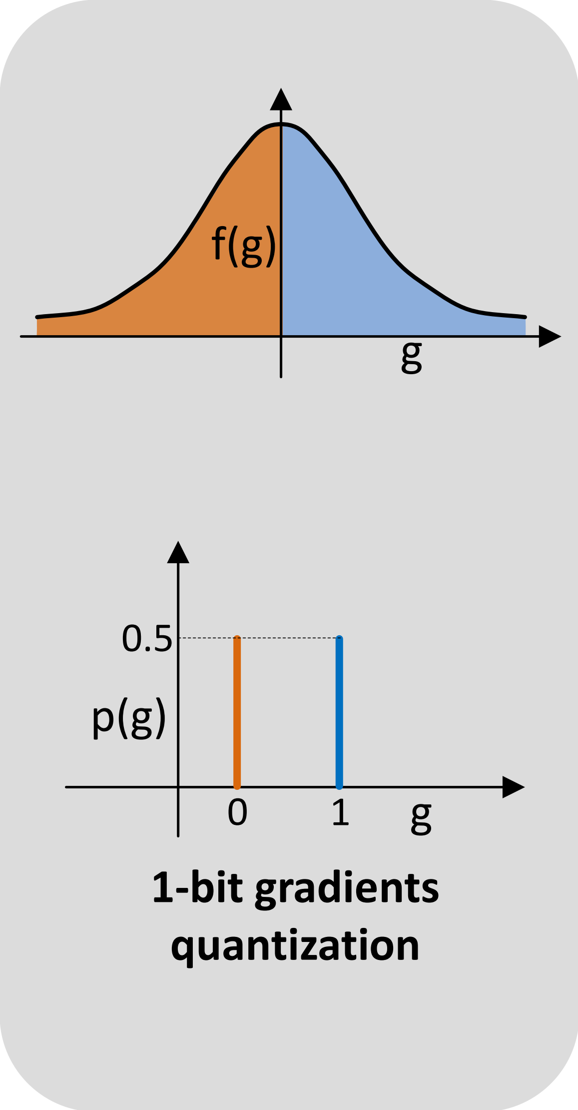
Runtime distribution on Google Cloud with 2xV100 GPUs (batch size=256, ResNet-50 on ImageNet).

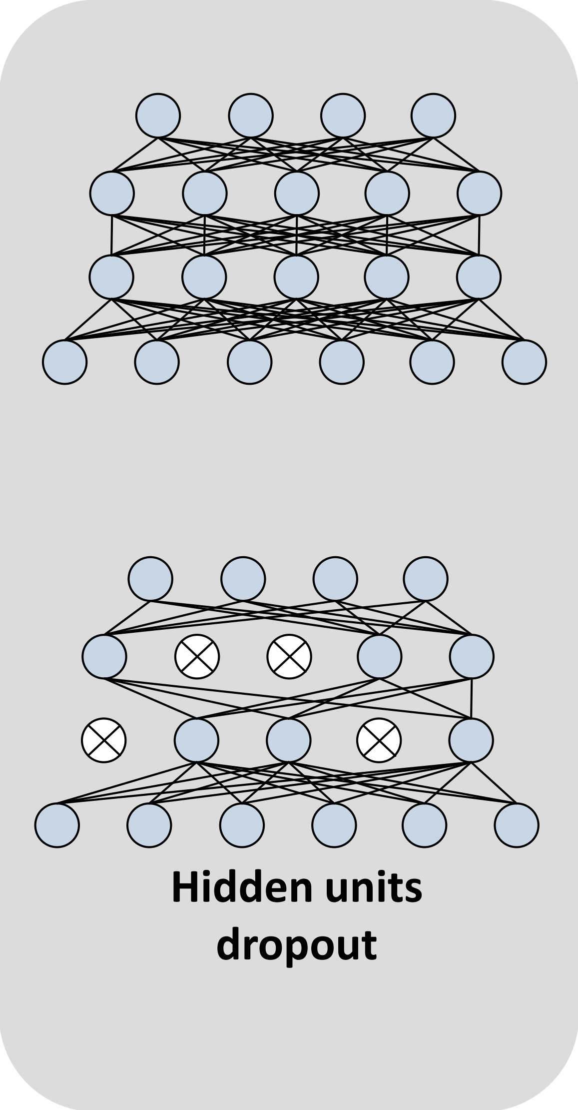
- Compared with imbalanced applications (e.g., LSTM, Transformer), the load imbalance on cloud servers is relatively light.

Deep learning training is robust

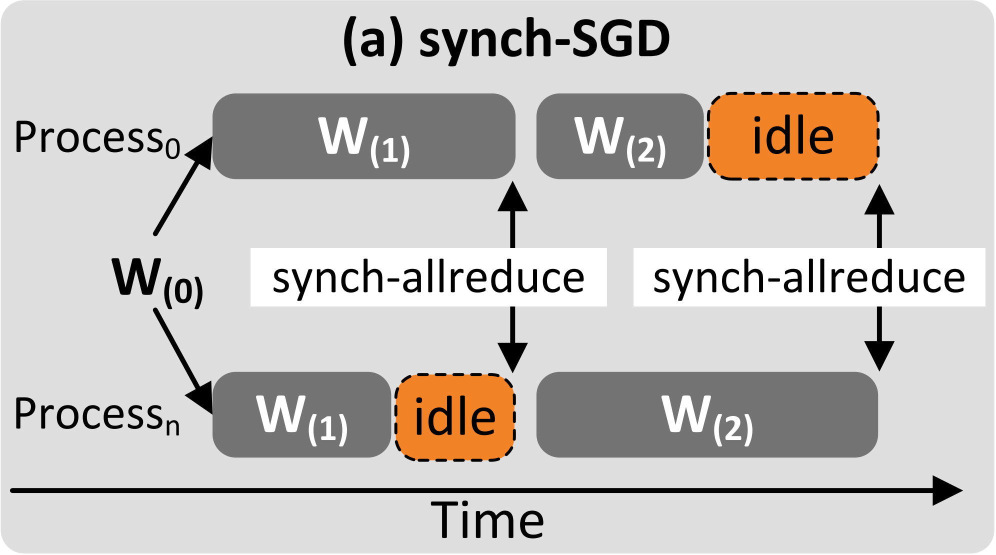


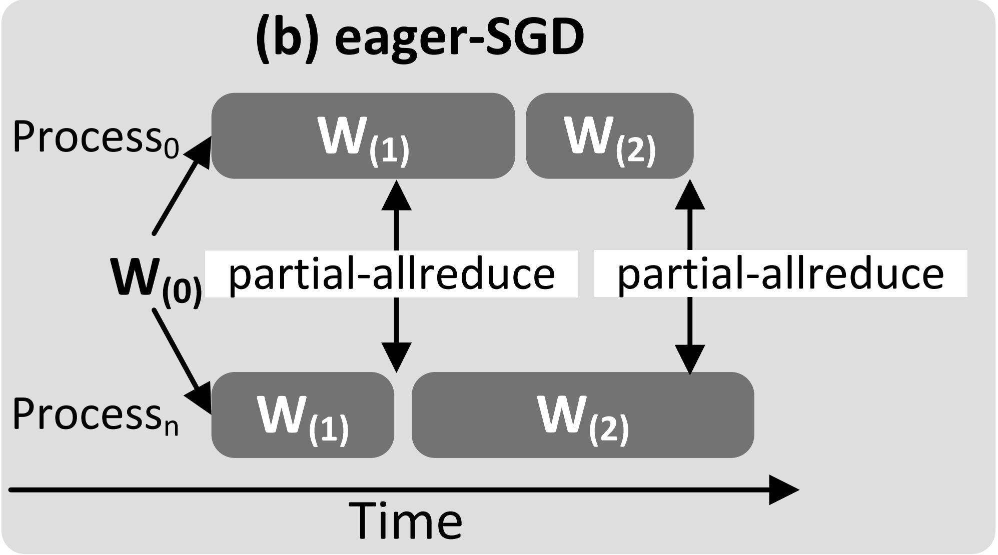






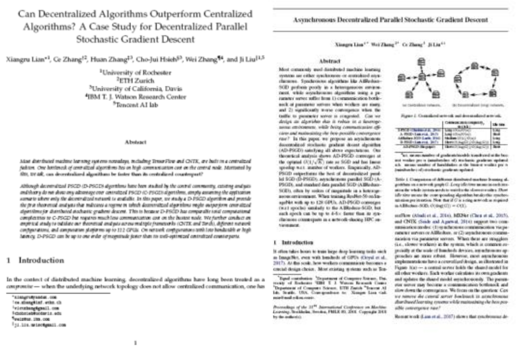
Eager-SGD to solve the load imbalance problem





Eager-SGD exploits the robustness of the training by allowing *allreduce* on stale gradients.

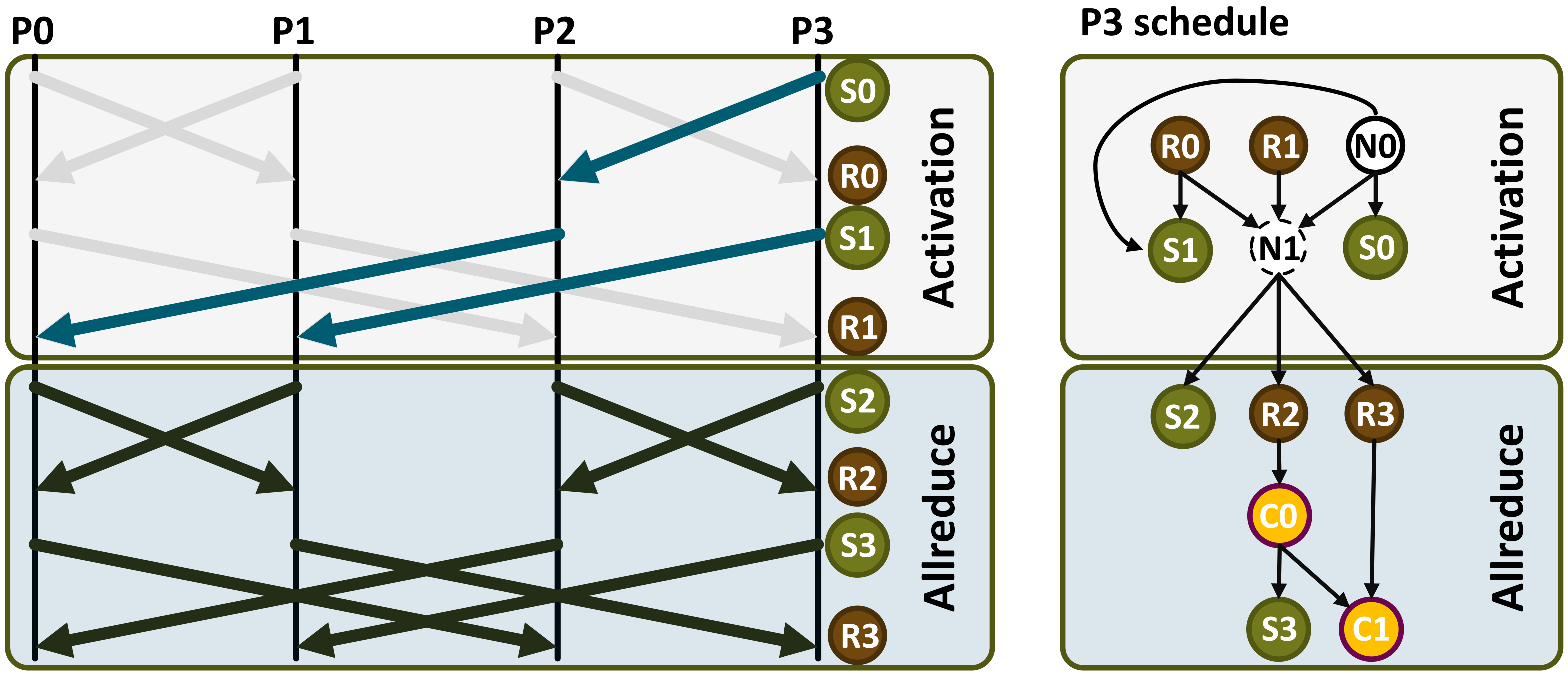
Gossip-based SGDs



	Communication participants	Number of steps for update propagation	Consistency mode
D-PSGD [1]	2	O(P)	synchronous
AD-PSGD [2]	1	O(logP)	asynchronous
eager-SGD	P	1	asynchronous

Partial Allreduce operations

- **Two phases:** the activation and the collective operation



- **Asynchronous execution**: an auxiliary thread would progress the execution (activation and collective) in the background.
- **Multiple initiators:** the same operation is only executed once even if we may have multiple initiators, i.e. multiple processes arrive at the same time.

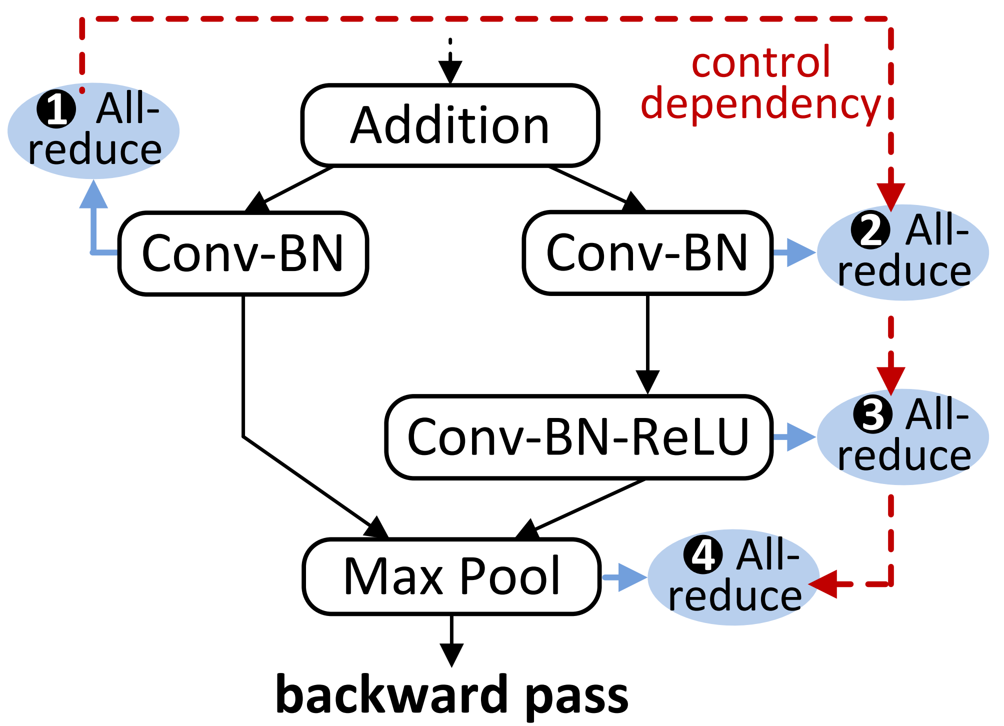
Solo allreduce and majority allreduce

- Two variants: solo allreduce [3] and majority allreduce.
- For solo, at least one process “actively” participates.
- For majority, a majority of processes must “actively” participate.

	Solo allreduce	Majority allreduce
Initiator	The fastest process	A randomly specified process
Attributes	Wait-free	Wait for the randomly specified initiator
The expectation of the participants	$\Omega(1)$	$\Omega(P/2)$

[3] Di Girolamo, Salvatore, Pierre Jolivet, Keith D. Underwood, and Torsten Hoefler. "Exploiting offload enabled network interfaces." In *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, pp. 26-33. IEEE, 2015.

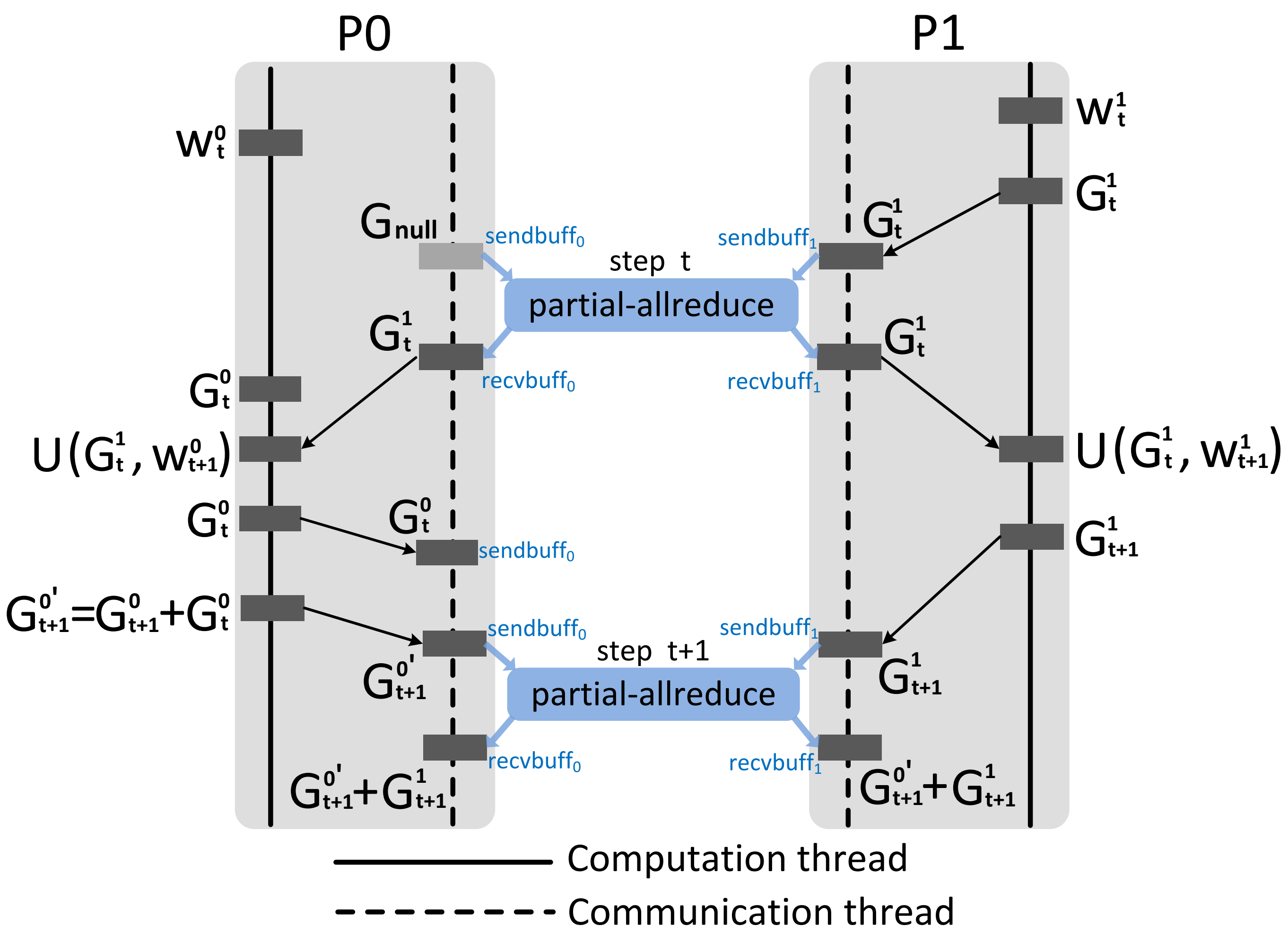
Implementation eager-SGD based on Tensorflow



Customized distributed optimizer based on Tensorflow

Eager-SGD utilizes the execution engine of TF to exploit the parallelism in the computation DAG.

Execution of eager-SGD



1. Two processes and P1 is faster.

2. P1 finishes the calculation for the gradients of **step *t***, and triggers partial-allreduce. P0 contributes NULL.

3. P0 finishes **step *t***, and discovers partial-allreduce is already done. P0 copies the stale gradients to its send buffer.

4. P0 catches up P1 in **step *t+1***. The stale gradients are combined with the latest gradients, and then commit to partial-allreduce.

Convergence of eager-SGD

- For a learning rate value

$$\alpha \le \min\left(\frac{\sqrt{\epsilon}P}{\sqrt{12L^2\tau M^2(P-Q)}}, \frac{\epsilon}{12M^2L}, \frac{\sqrt{\epsilon}P}{\sqrt{4L\tau M^2(P-Q)}}\right),$$

eager-SGD converges after

$$T = \Theta\left(\frac{f(w_0)-m}{\epsilon\alpha}\right)$$ iterations.

Staleness bound

The total number of processes

The number of processes which contribute the latest gradients

$$T \ge \Theta\left(\frac{(f(w_0)-m)\sqrt{\tau(P-Q)}}{P\epsilon^{3/2}}\right)$$

- Note the dependence in $\boldsymbol{\tau}$ (staleness bound) and $\boldsymbol{P}$-$\boldsymbol{Q}$ (the number of stale gradients) for iterations $\boldsymbol{T}$.
- Eager-SGD would converge slower if too many stale gradients are used.

Evaluation

- CSCS Piz Daint supercomputer.
- Cray Aries interconnected network.
- Cray MPICH 7.7.2 communication library.
- Each node contains a 12-core Intel Xeon E5-2690 CPU, and one NVIDIA Tesla P100 GPU.
- We compare **eager-SGD** with the allreduce-based synch-SGD (**Horovod** and **Deep500**), the asynchronous centralized SGD (**TF parameter server**), and the gossip SGDs (**D-PSGD**, **SGP**).

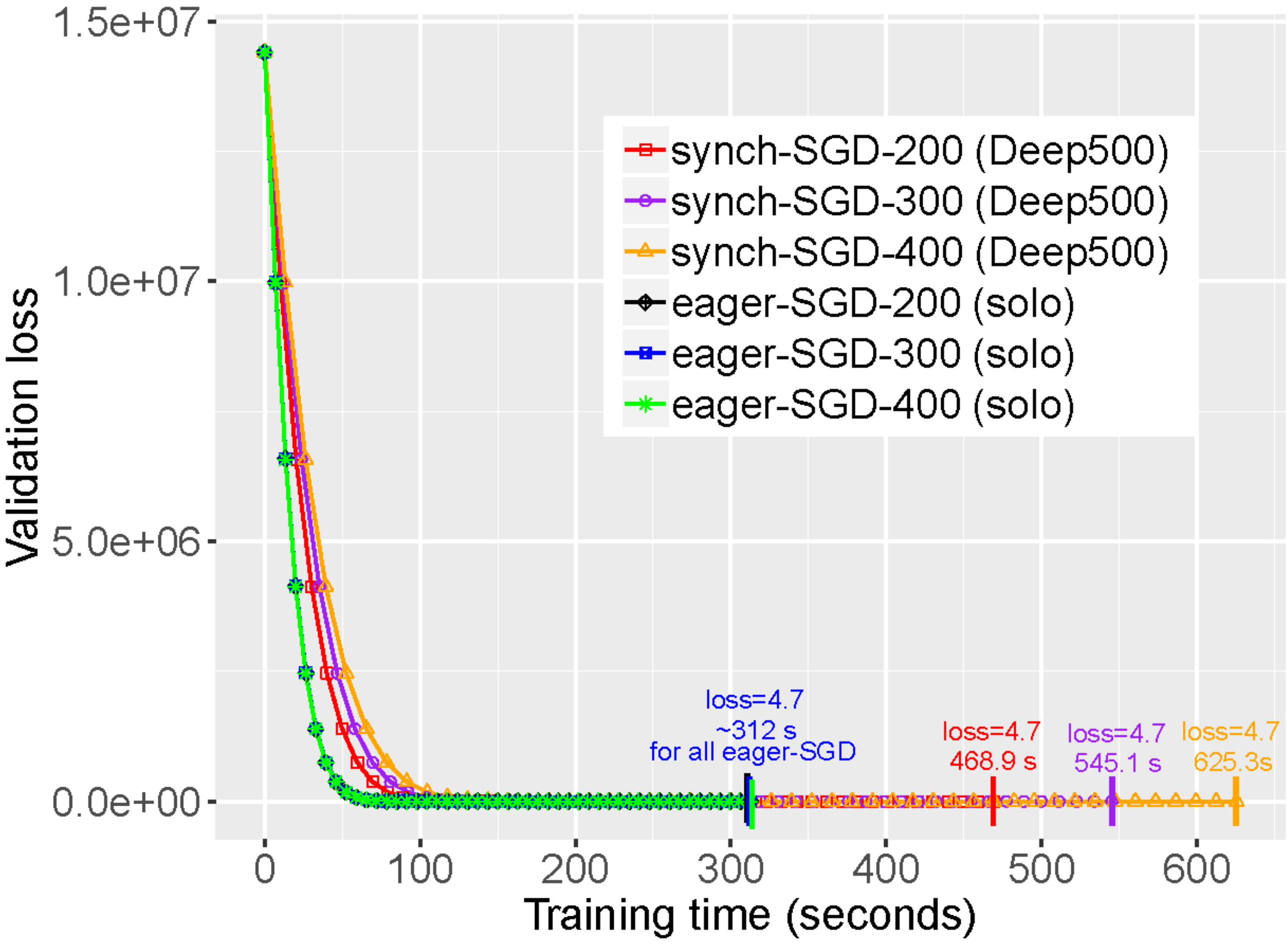
Simulated load imbalance (traces on cloud machine)

Table 1. Neural networks used for evaluation

Tasks	Models	Parameters	Train data size	Batch size	Epochs	Processes
Hyperplane regression	One-layer MLP	8,193	32,768 points	2,048	48	8
Cifar-10	ResNet-32 [21]	467,194	50,000 images	512	190	8
ImageNet [14]	ResNet-50 [21]	25,559,081	1,281,167 images	8,192	90	64
UCF101 [53]	Inception+LSTM [61]	34,663,525	9,537 videos	128	50	8

Inherent load imbalance

Hyperplane regression (light load imbalance)

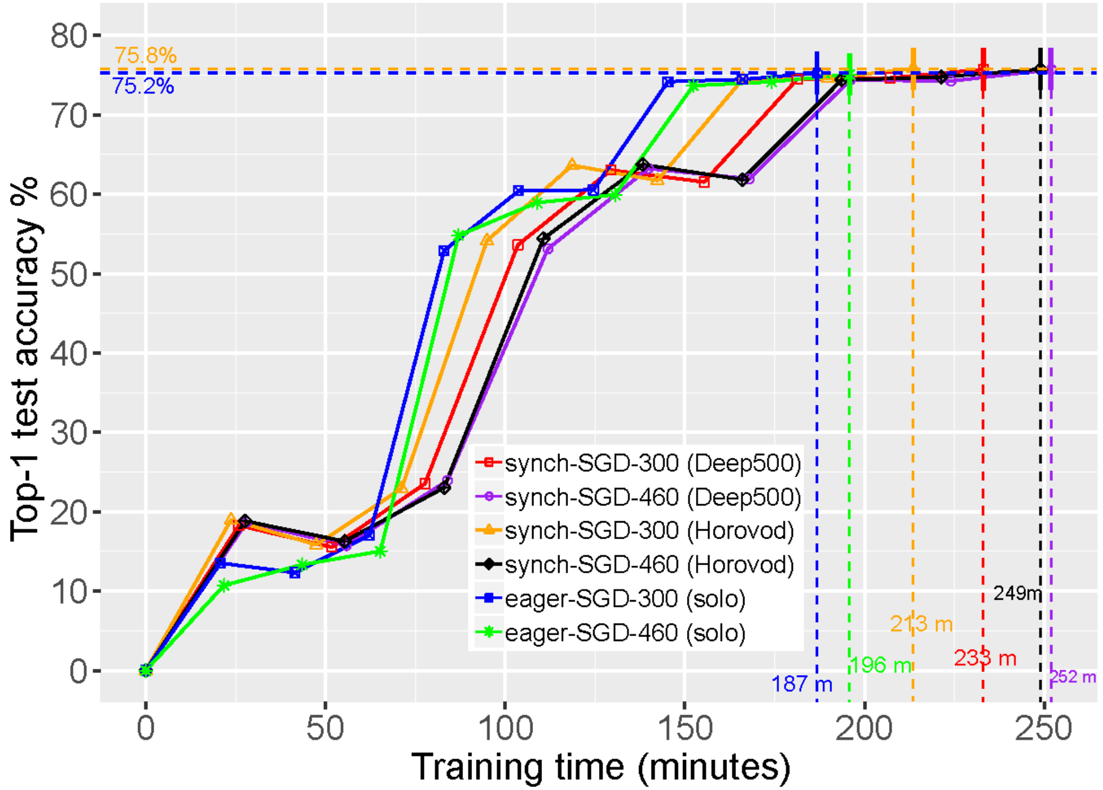


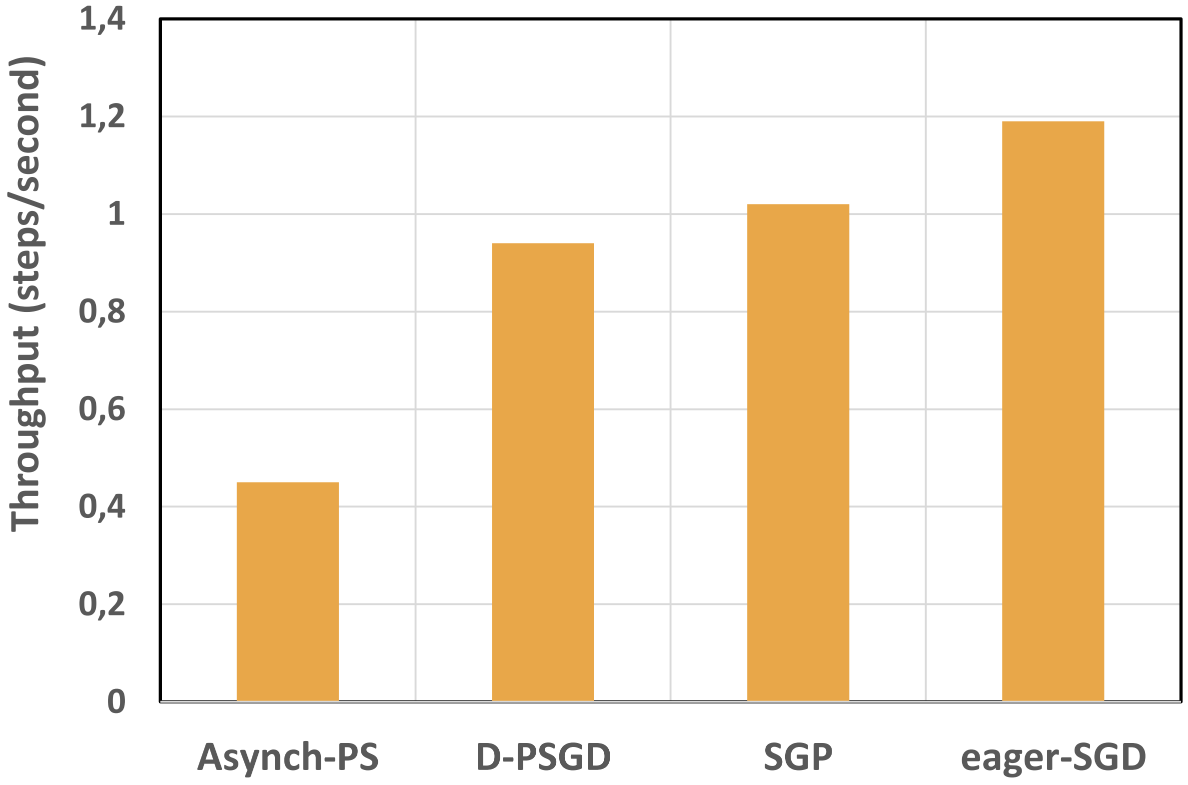
Synch-SGD vs eager-SGD for hyperplane regression using 8 GPUs. "synch/eager-SGD-200/300/400" represent 200/300/400 ms load imbalance injection for 1 out of 8 processes.

- Eager-SGD (solo) achieves **1.50x**, **1.75x**, and **2.01x** speedup over synch-SGD (Deep500), respectively.
- The loss value is equivalent with synch-SGD (Deep500).

ResNet-50 on ImageNet (light load imbalance)

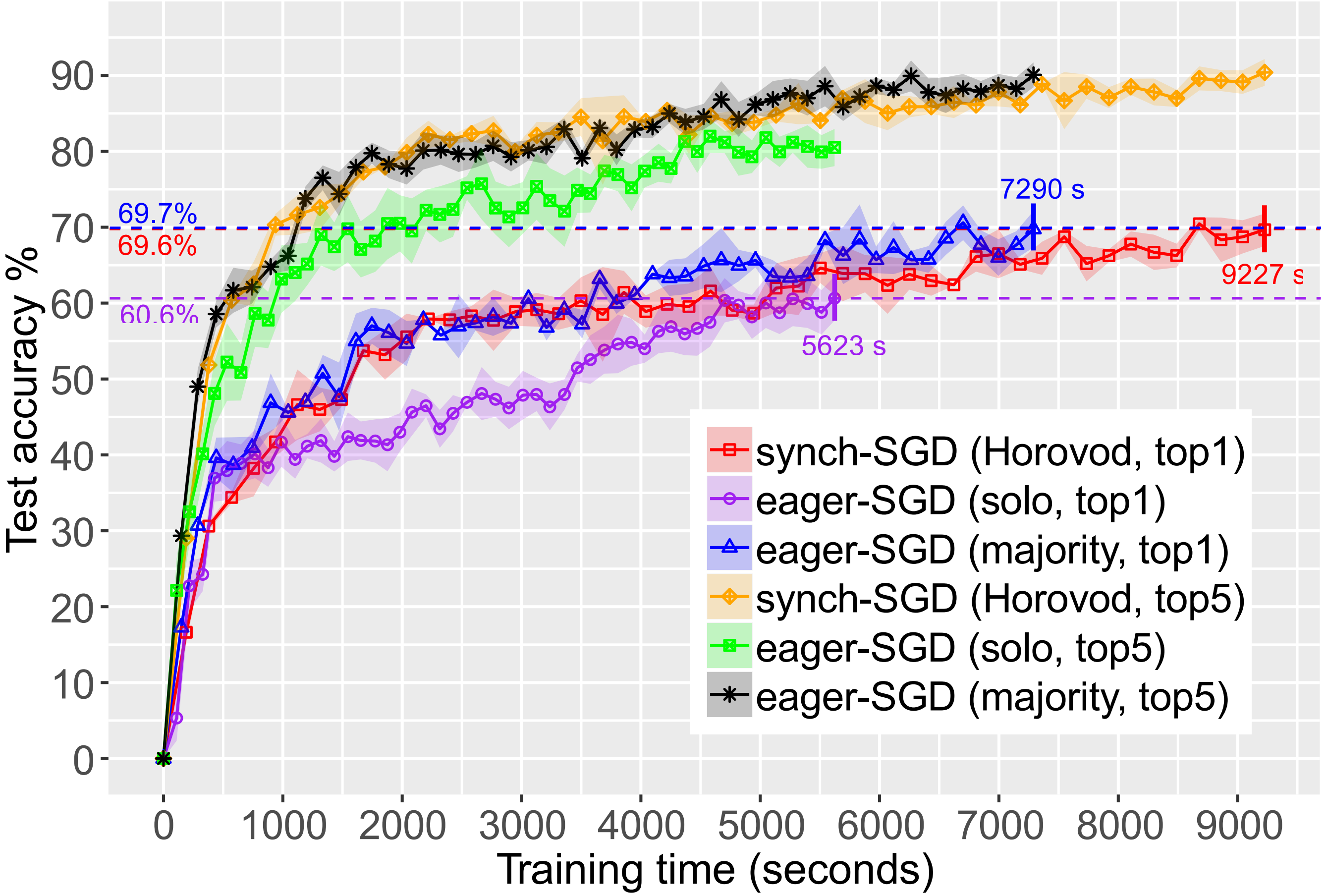
Synch-SGD vs eager-SGD for ResNet-50 on ImageNet using 64 GPUs. "synch/eager-SGD-300/460" represent 300/460 ms load imbalance injection for 4 out of 64 processes.





- Eager-SGD (solo) achieves **1.25x** and **1.29x** speedup over Deep500, respectively; **1.14x** and **1.27x** speedup over Horovod, respectively. Top-1 accuracy is almost equivalent (75.2% vs 75.8%).
- Eager-SGD (solo) achieves **2.64x**, **1.26x**, **1.17x** over aysnch-PS and gossip-based SGDs (D-PSGD, SGP) respectively.

LSTM on UCF101 (severe load imbalance)

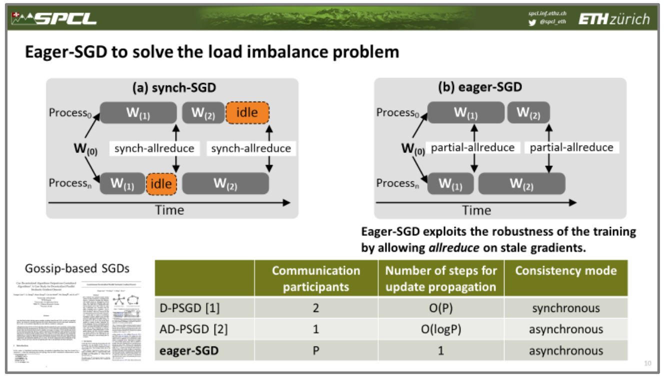


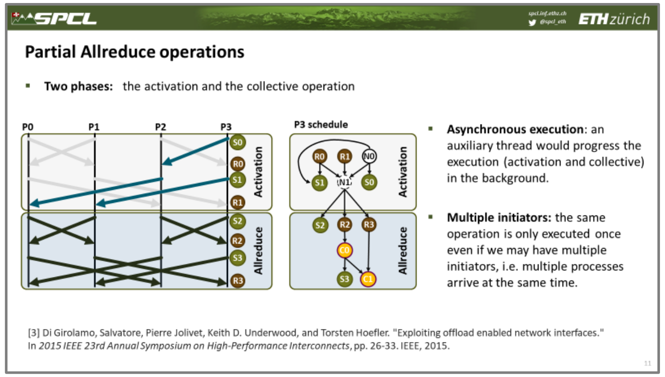
Top-1 test accuracy and runtime for LSTM on UCF101 using 8 GPUs.

	eager-SGD (solo)	eager-SGD (majority)
Speedup over Horovod	1.64x	1.27x
Top-1 test accuracy	60.6% on average, up to 70.4%	69.7% on average, up to 72.8%

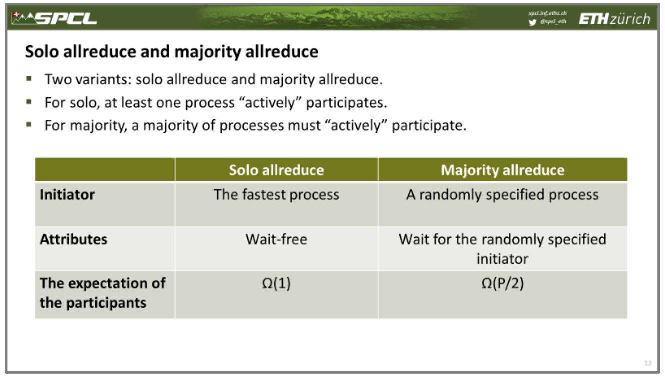
Conclusion

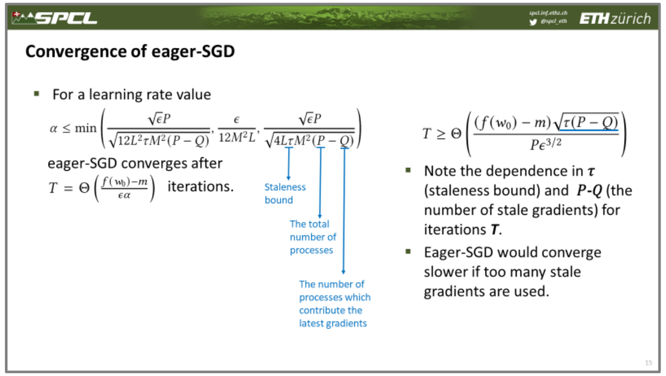
1. Eager-SGD deals with the imbalanced workloads using partial allreduce operations.



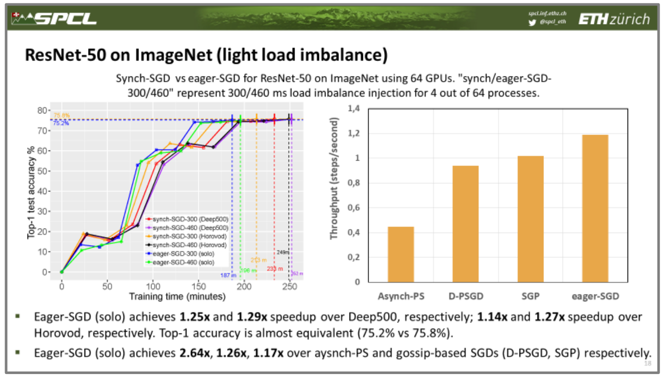


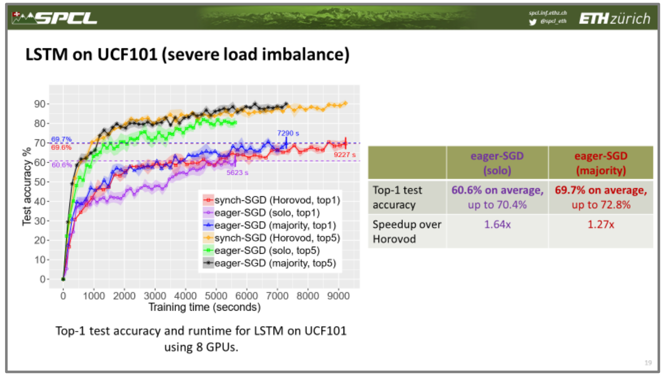
2. Eager-SGD has two variants, solo and majority.





3. Solo allreduce is suitable for light load imbalance, while majority allreduce works for severe load imbalance.





4. For the future work, we will verify the idea of eager-SGD on model-averaging SGD algorithms.

Questions: shigang.li@inf.ethz.ch