TensorLights

End-Host Traffic Scheduling for Distributed Deep Learning

Xin Sunny Huang
Ang Chen
T. S. Eugene Ng

Rice University
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)

Classic AI problems

Language processing

Image Recognition

Also used for ...

Power Scheduling [1]

System Security [2]

Network Routing [3]

Database Index [4]

The Rise of Deep Learning (DL)

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

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

Parameter Server (PS)

worker 1

model update

barrier

step=1

step=2

worker 2

gradient update

steps per job: 1,000s to 1,000,000s[1]

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.

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.

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.

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.

Contention among collocated PSes!
How would PS contention impact the performance of distributed DL jobs?

Measurement Setup

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

\footnotesize

\textsuperscript{[1]} https://www.tensorflow.org/
\textsuperscript{[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

PS_1 → PS_2 → host machine → to workers

model update sent

FIFO

1

2

time
Stragglers under Contention

Possible stragglers detected!
Workers (of PS₁) receiving the tail part will delay the progress of the whole job.
Stragglers under Contention

Possible stragglers detected!
Workers (of PS₁) 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<sub>1</sub>) 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

With priority “1>2”
Mitigate Stragglers with Traffic Priority

With priority “1>2”

One priority for one job’s model update (from PS)

model update sent

time

host machine

PS1  PS2

to work

With priority “1>2”
Mitigate Stragglers with Traffic Priority

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

With priority “1>2”

One priority for one job’s model update (from PS)

model update sent
time

host machine

PS₁ PS₂

to work
Reducing Stragglers with TensorLights

- **PS$_1$**
- **PS$_2$**

FIFO model update sent to workers

- **TensorLights - One**
- **TensorLights - RoundRobin**

Rotate priority assignments of “1>2” and “2>1”
Reducing Stragglers with TensorLights

Reducing stragglers with priority while achieving fair progress among concurrent jobs!

TensorLights - Round Robin

TensorLights

FIFO

model update sent

time

PS1

PS2

host machine

to workers

Rotate priority assignments of “1>2” and “2>1”
Scheduling Model with TensorLights

FIFO

TensorLights - RoundRobin

TensorLights - One
<table>
<thead>
<tr>
<th></th>
<th>TensorLights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource scheduling</td>
<td>✓ Work conserving</td>
</tr>
<tr>
<td>Scheduling overhead</td>
<td>✓ Local, light-weight</td>
</tr>
<tr>
<td>Deployment</td>
<td>✓ No change to app, cluster scheduler, or hardware</td>
</tr>
</tbody>
</table>
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 Placement

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

![Distribution of Barrier Wait Time](image)

(a) Average in one barrier

(b) Standard variance in one barrier

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**

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!