Georgia Tech

DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2018 // CHRISTINE HERLIHY

LECTURE #08:

TENSORFLOW: A SYSTEM FOR LARGE-SCALE MACHINE LEARNING

CREATING THE NEXT®

TODAY'S PAPER

- **TensorFlow: A system for large-scale machine learning**
  - **Authors:**
    - Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
  - **Affiliation:** Google Brain (deep-learning AI research team)
    - Published in 2016
  - **Areas of focus:**
    - Machine learning at scale; deep learning

TODAY'S AGENDA

- Problem Overview
- Context: Background Info on Relevant Concepts
- Key Idea
- Technical Details
- Experiments
- Discussion Questions

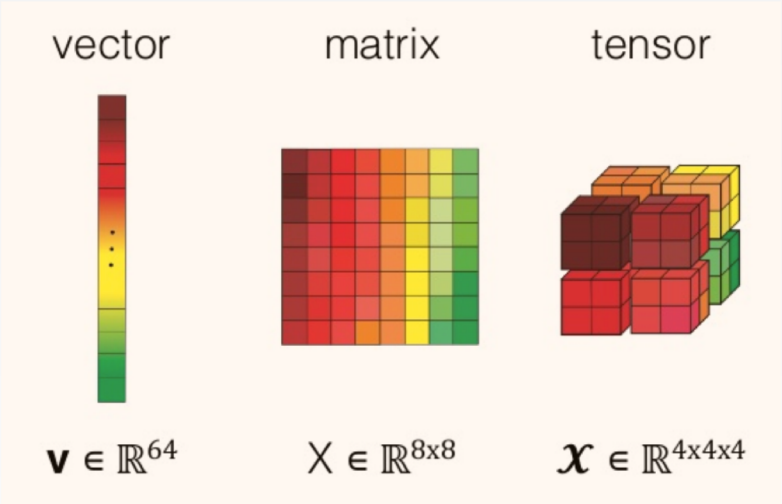
PROBLEM OVERVIEW

- **Status Quo Prior to Tensor Flow:**
  - A less flexible system called DistBelief was used internally at Google
    - Primary use case: training DNN with billions of parameters using thousands of CPU cores
- **Objective:**
  - Make it easier for developers to efficiently develop/test new optimizations and model training algorithms across a range of distributed computing environments
  - Empower development of DNN architectures in higher-level languages (e.g., Python)
- **Key contributions:**
  - TF is a flexible, portable, open-source framework for efficient, large-scale model development

Sources: https://ai.google/research/pubs/pub40565

CONTEXT: TENSORS

- **Tensor:** "Generalization of scalars, vectors, and matrices to an arbitrary number of indices"
  - (e.g., potentially higher dimensions)
- **Rank:** number of dimensions
- **TF tensor attributes:** data type; shape



Rank	Math entity
0	Scalar (magnitude only)
1	Vector (magnitude and direction)
2	Matrix (table of numbers)
3	3-Tensor (cube of numbers)
n	n-Tensor (you get the idea)

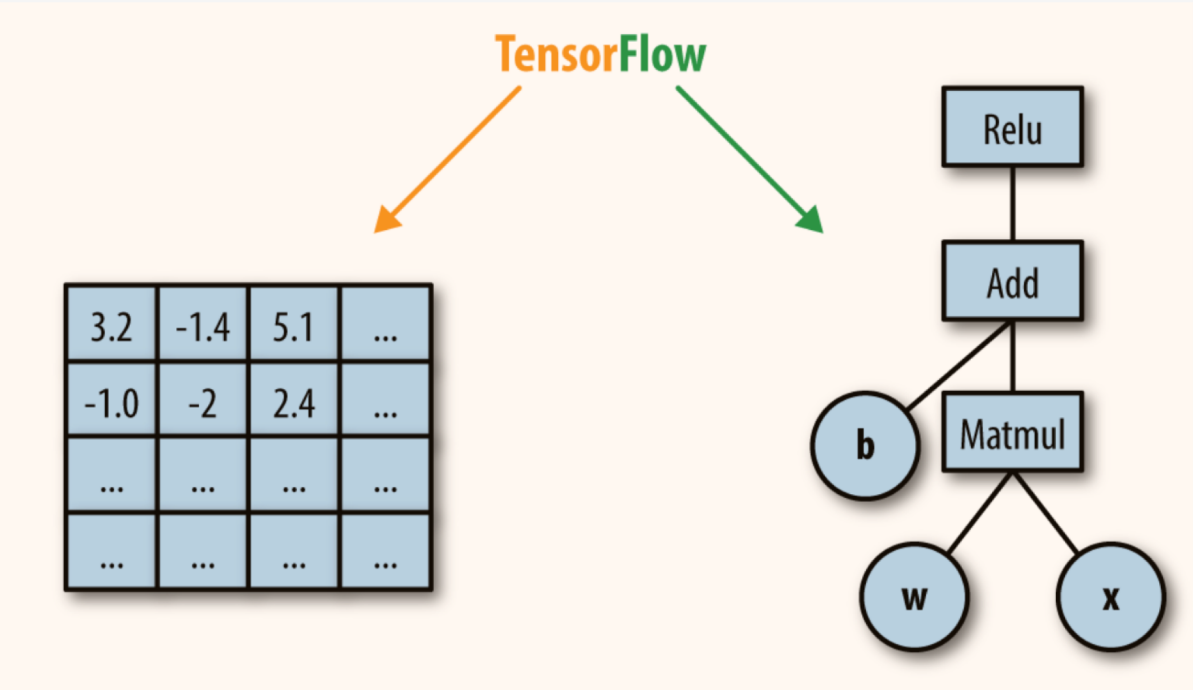
Sources: http://www.wolframalpha.com/input/?i=tensor; https://www.tensorflow.org/guide/tensors; https://www.slideshare.net/BertonEarnshaw/a-brief-survey-of-tensors

CONTEXT: STOCHASTIC GRADIENT DESCENT (SGD)

- SGD: an iterative method for optimizing a differentiable objective function
- Stochastic because samples are randomly selected

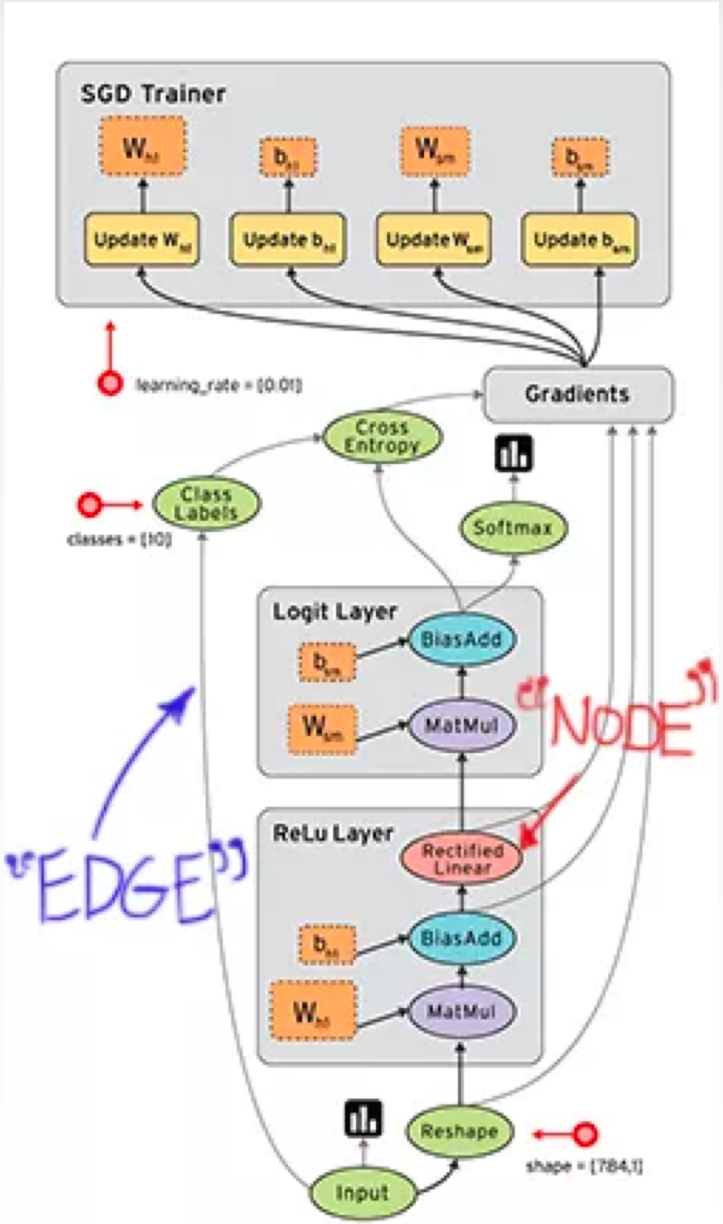
CONTEXT: DATAFLOW GRAPHS

- **Nodes:** represent units of computation
- **Edges:** represent data consumed/produced by a computation



Source: https://www.safaribooksonline.com/library/view/learning-tensorflow/9781491978504/ch01.html

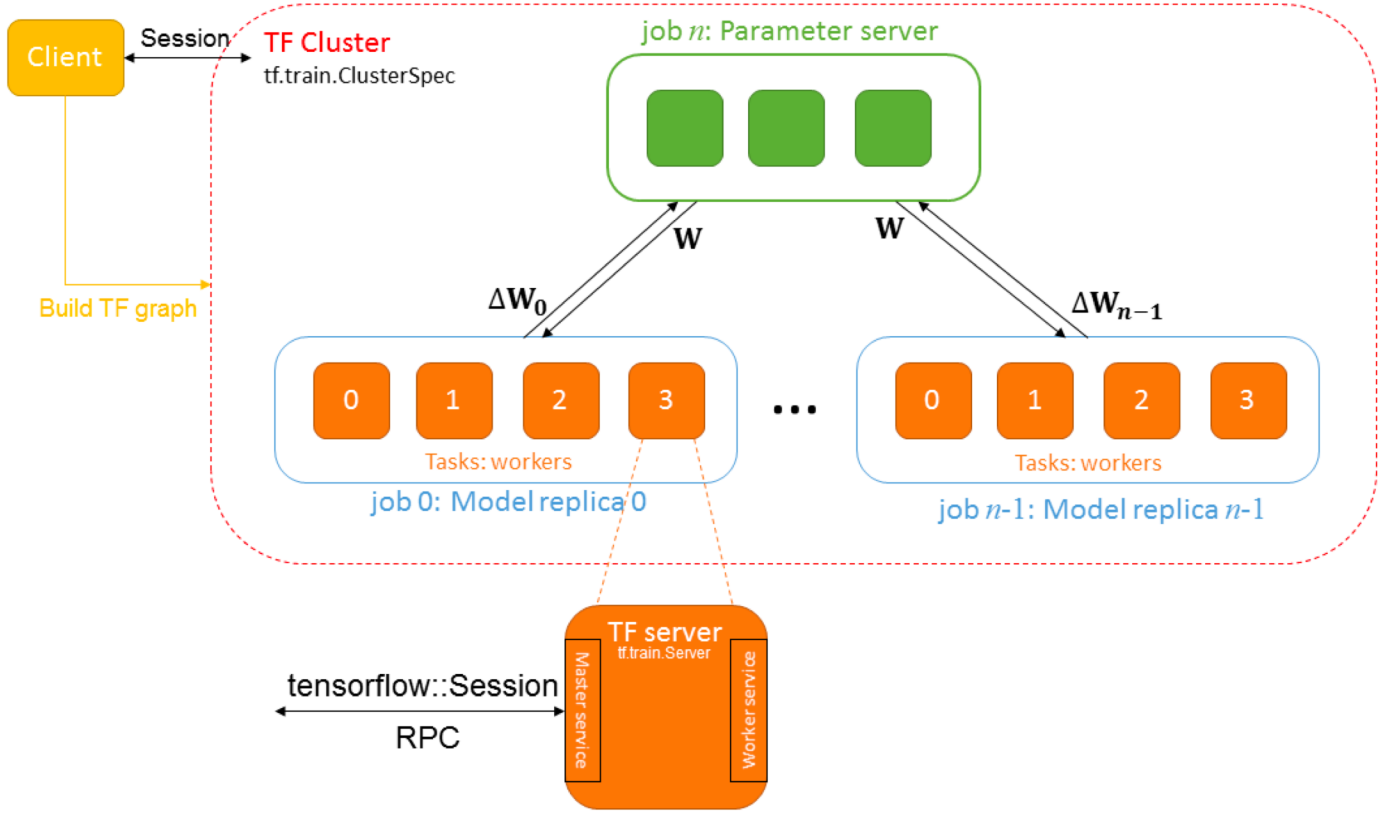
Example of a more complex TF dataflow graph:



Taken from http://tensorflow.org/

CONTEXT: PARAMETER SERVER ARCHITECUTRE

- **Parameter server:** a centralized server that distributed models can use to share parameters (e.g., get/put operations and updates)



Source: http://www.pittnuts.com/2016/08/glossary-in-distributed-tensorflow/

CONTEXT: MODEL PARALLELISM

- **Model parallelism:** single model is partitioned across machines
- Communication required between nodes whose edges cross partition boundaries

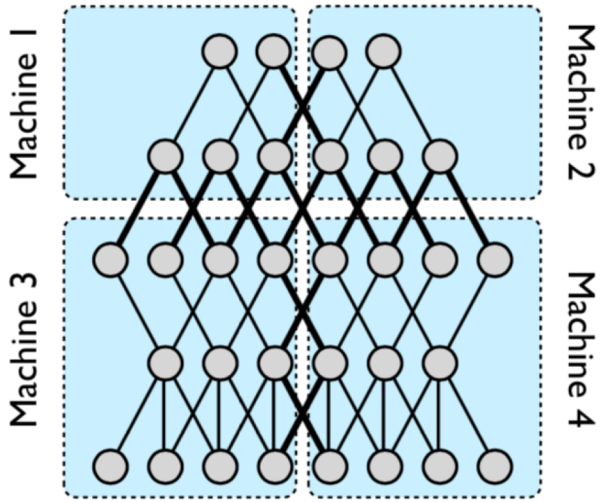


Figure 1: An example of model parallelism in DistBelief. A five layer deep neural network with local connectivity is shown here, partitioned across four machines (blue rectangles). Only those nodes with edges that cross partition boundaries (thick lines) will need to have their state transmitted between machines. Even in cases where a node has multiple edges crossing a partition boundary, its state is only sent to the machine on the other side of that boundary once. Within each partition, computation for individual nodes will the parallelized across all available CPU cores.

Source: https://ai.google/research/pubs/pub40565

CONTEXT: DATA PARALLELISM

- Multiple replicas (instances) of a model are used to optimize a single objective function

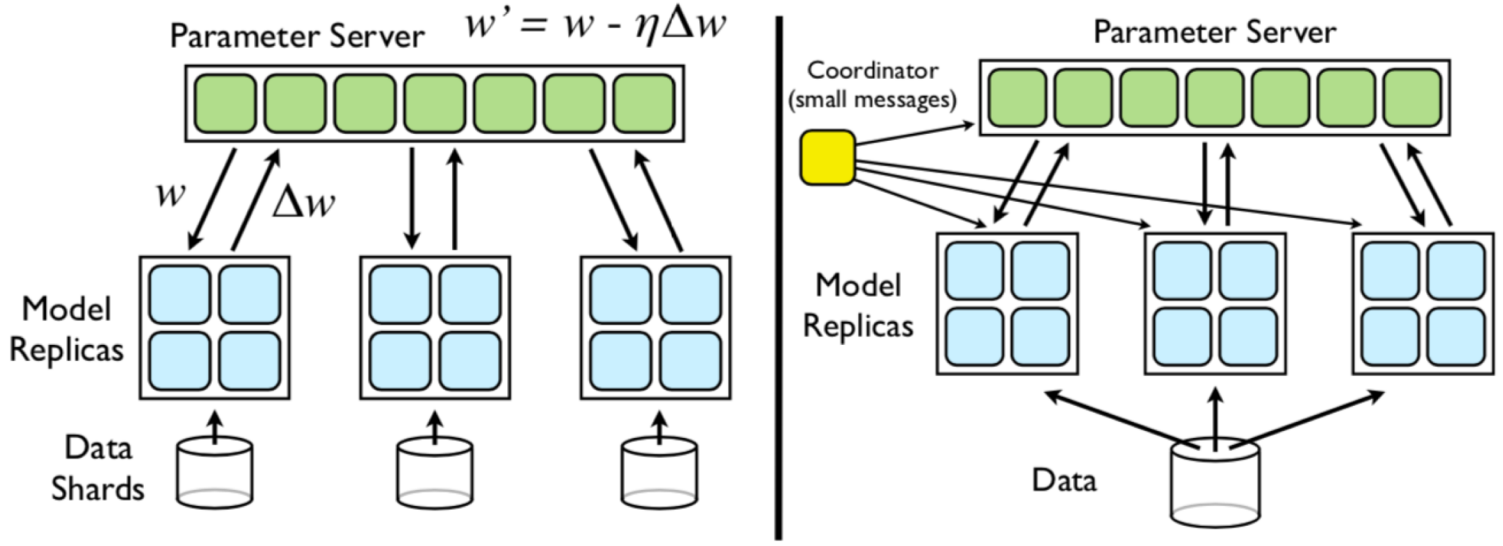


Figure 2: Left: Downpour SGD. Model replicas asynchronously fetch parameters w and push gradients Δw to the parameter server. Right: Sandblaster L-BFGS. A single 'coordinator' sends small messages to replicas and the parameter server to orchestrate batch optimization.

Source: https://ai.google/research/pubs/pub40565

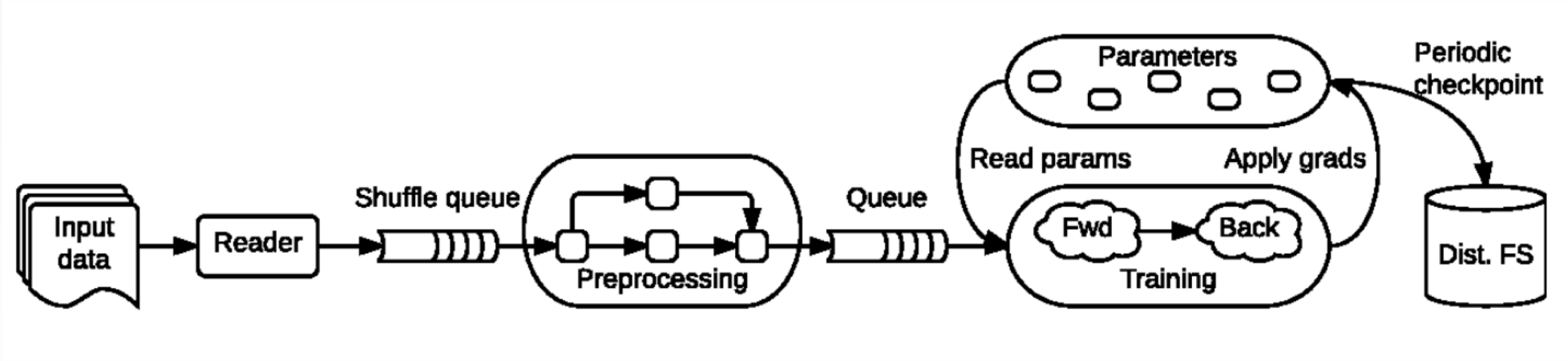
CONTEXT: DISTBELIEF

- **DistBelief was the pre-cursor to TF:**
  - Distributed system for training DNNs
  - Uses parameter-server architecture
  - NN defined as an acyclic graph of layers that terminates with a loss function
- **Limitations:**
  - Layers were C++ classes; researchers wanted to work in Python when prototyping new architectures
  - New optimization methods required changes to the PS architecture
  - Fixed execution pattern that worked well for FFNs was not suitable for RNNs, GANs, or RL models
  - Was designed for large cluster environment; hard to scale down

KEY IDEA

- **Objective:**
  - Empower users to efficiently implement and test experimental network architectures and optimization algorithms at scale, in a way that takes advantage of distributed resources and/or parallelization opportunities when available

- **How?**



Source: https://ai.google/research/pubs/pub40565

TECHNICAL DETAILS: EXECUTION MODEL

- A single dataflow graph is used to represent all computation and state in a given ML algorithm
  - Vertices represent (mathematical) operations
  - Edges represent values (stored as tensors)
- Multiple concurrent executions on overlapping subgraphs of overall graph are supported
- Individual vertices can have mutable state that can be shared between different executions of the graph (allows for in-place updates to large parameters)

TECHNICAL DETAILS: EXTENSIBLILITY (1/4)

- **Use case 1:** Differentiation and optimization
- TF includes a user-level library that differentiates symbolic expression for loss function and produces new symbolic expression representing gradients
- Differentiation algorithm performs BFS to identify all backward paths, and sums partial gradient contributions
- Graph structure allows for conditional and/or iterative control flow decisions to be (re)played during forward/backward passes
- Many optimization algorithms implemented on top of TF, including: Momentum, AdaGrad, AdaDelta, RMSProp, Adam, and L-BFGS

Source: https://ai.google/research/pubs/pub45381

TECHNICAL DETAILS: EXTENSIBLILITY (2/4)

- **Use case 2:** Training very large models
- **Example:** Given high-dimensional text data, generate lower-dimensional embeddings
  - Multiply a batch of *b* sparse vectors against an *n*d* embedding matrix to produce a dense *b*d* representation; *b* << *n*
  - The *n*d* matrix may be too large to copy to a worker or store in RAM on a single host
- TF lets you split such operations across multiple parameter server tasks

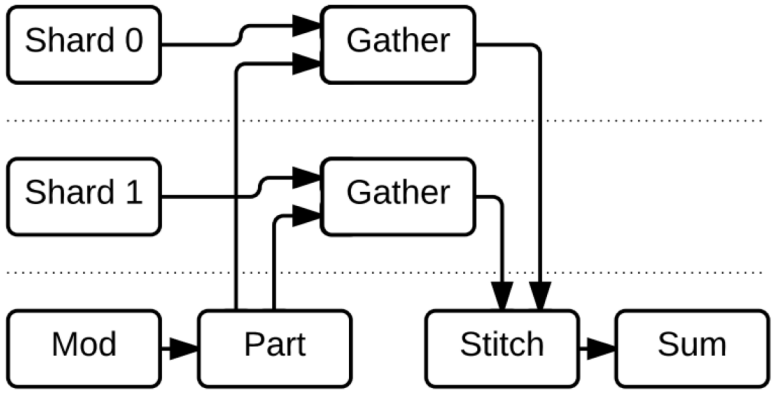


Figure 3: Schematic dataflow graph for a sparse embedding layer containing a two-way sharded embedding matrix.

Source: https://ai.google/research/pubs/pub45381

TECHNICAL DETAILS: EXTENSIBLILITY (3/4)

- **Case study 3:** Fault tolerance
- Training long-running models on non-dedicated machines requires fault tolerance
- Operation-level fault tolerance is not necessarily required
  - Many learning algorithms have only weak consistency requirements
- TF uses user-level checkpointing (save/restore)
- Checkpointing can be customized (e.g., when a high score is received on a specified evaluation metric)

Source: https://ai.google/research/pubs/pub45381

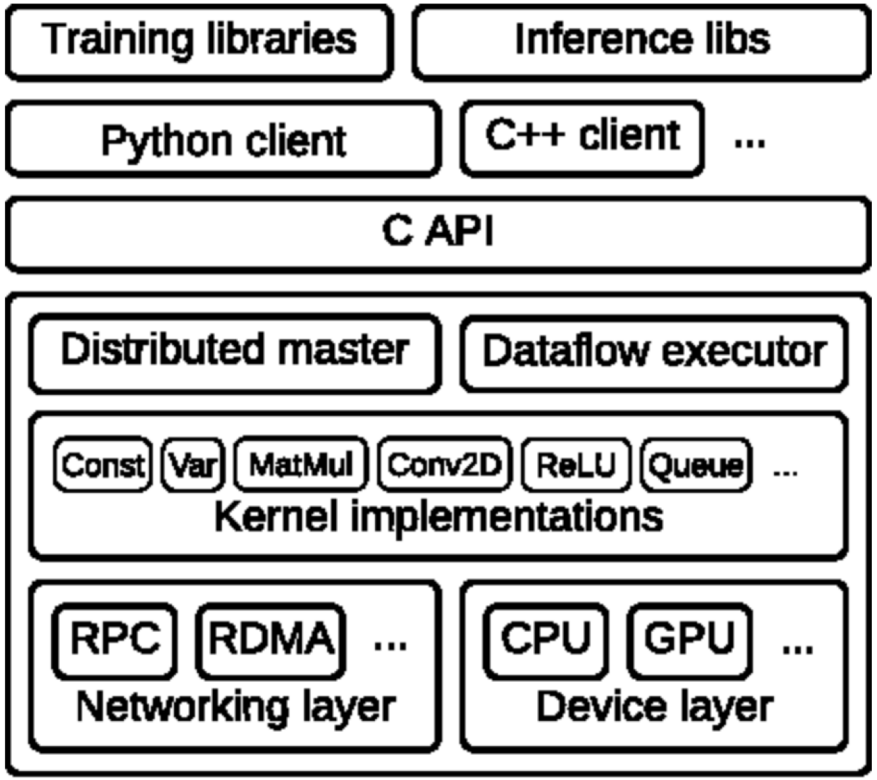
TECHNICAL DETAILS: EXTENSIBLILITY (4/4)

- **Case study 4:** Synchronous replica coordination
- Synchronous parameter updates have the potential to be a computational bottleneck
  - Only as fast as slowest worker
- GPUs reduce the number of machines required, making synchronous updates more feasible
- TF implements proactive backup workers to mitigate stragglers during synchronous updates
  - Aggregation takes first m of n updates produced; works for SGD since batches are randomly selected rather than sequentially

Source: https://ai.google/research/pubs/pub45381

TECHNICAL DETAILS: SYSTEM ARCHITECTURE

- Core library is implemented in C++
- C API connects this core runtime to higher-level user code in different languages (focus on C++; Python)
- Portable; runs on many different OS and architectures, including:
  - Linux; Mac OSX; Windows; Android, iOS
  - x86; various ARM-based CPU architectures
  - NVIDIA's Kepler, Maxwell, and Pascal GPU microarchitectures
- Runtime has > 200 operations
  - Math ops; array; control flow; state management



Source: https://ai.google/research/pubs/pub45381

EXPERIMENTS: GENERAL APPROACH

- TensorFlow is compared to similar frameworks, including Caffe, Neon, and Torch; self-referential benchmarks also established

- **Evaluation tasks:**
  - Single-machine benchmarks
  - Synchronous replica microbenchmark
  - Image classification
  - Language modeling

- **Evaluation metrics:**
  - System performance
  - Could have evaluated on the basis of model learning objectives instead
  - Why choose system performance?

EXP. 1: SINGLE-MACHINE BENCHMARKS

- **Question investigated:**
  - Do the design decisions that allow TensorFlow to be highly scalable impede performance for small-scale tasks that are essentially kernel-bound

- **Results:**
  - TensorFlow generally close to Torch
  - Neon often beats all 3; they attribute this to the performance gains associated with Neon's convolutional kernels, which are implemented in assembly

- **Dataset:**
  - Each of the comparison systems are used to train a 4 different CNN models using a single GPU

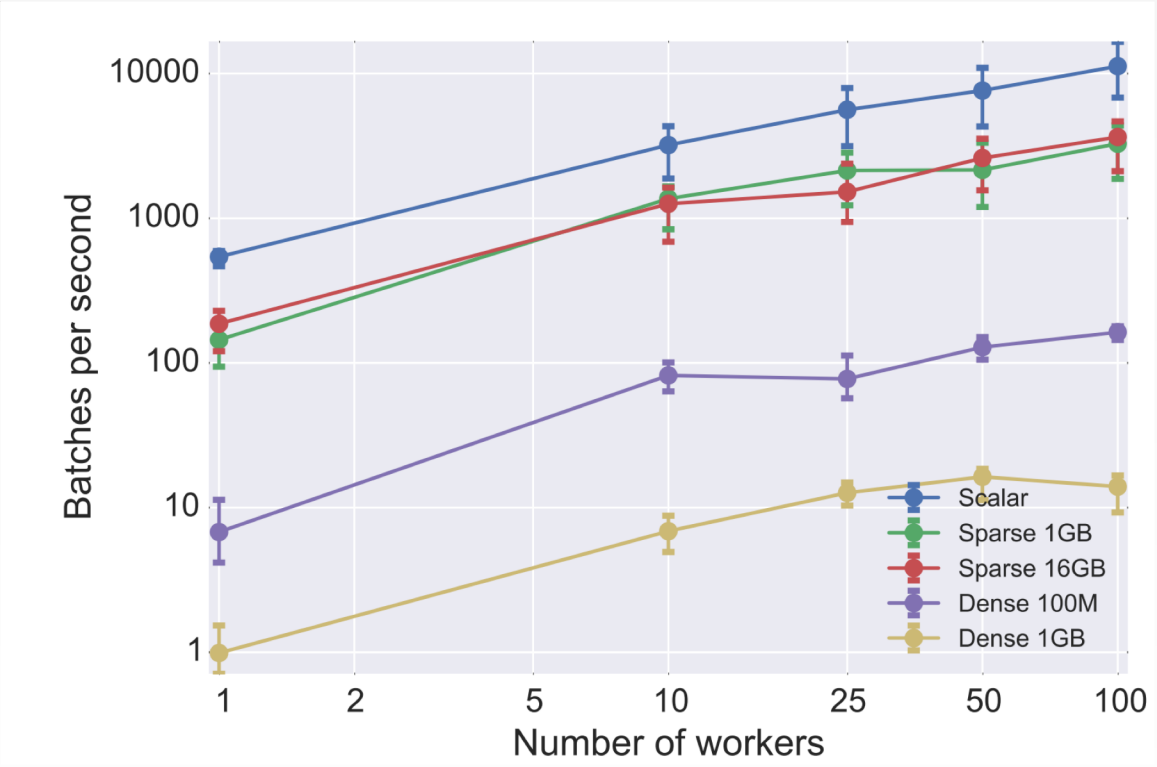
Library	AlexNet	Overfeat	OxfordNet	GoogleNet
	Training step time (ms)			
Caffe	324	823	1068	1935
Neon	87	**211**	**320**	**270**
Torch	**81**	268	529	470
TensorFlow	**81**	279	540	445

Source: https://ai.google/research/pubs/pub45381

EXP. 2: SYNCH. REPLICA MICROBENCHMARK

- **Question investigated:**
  - Investigate how the performance of their coordination implementation for synchronous training scales as workers are added to the device pool
- **Dataset:**
  - They compare the number of null training steps per second that TF can perform for models of different sizes as the number of synchronous works is increased
  - **Null step:** a worker fetches shared model parameters from 16 PS tasks, performs trivial computation, and sends updates to the parameter

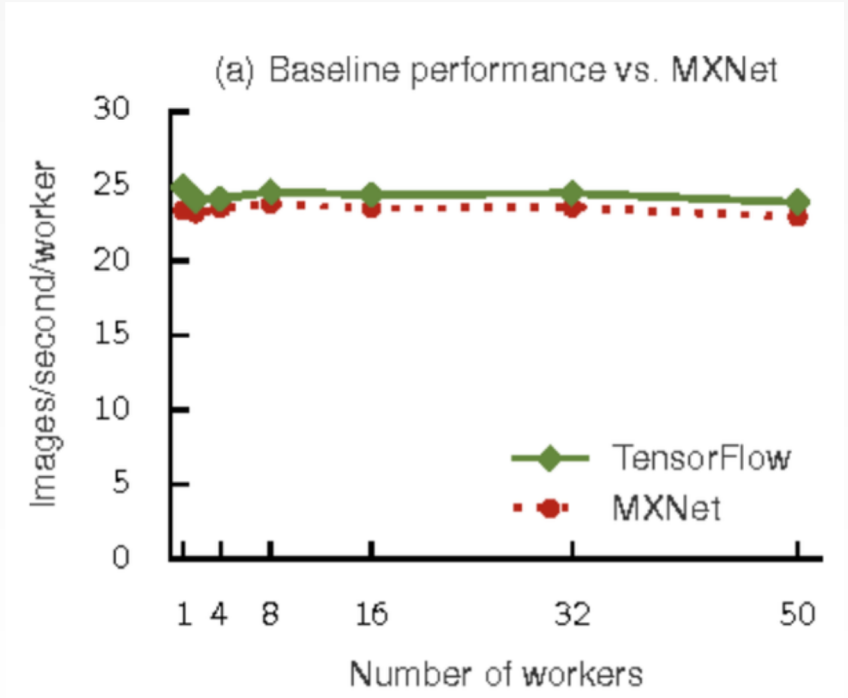
- **Results:**



Source: https://ai.google/research/pubs/pub45381

EXP. 3: IMAGE CLASSIFICATION (1/2)

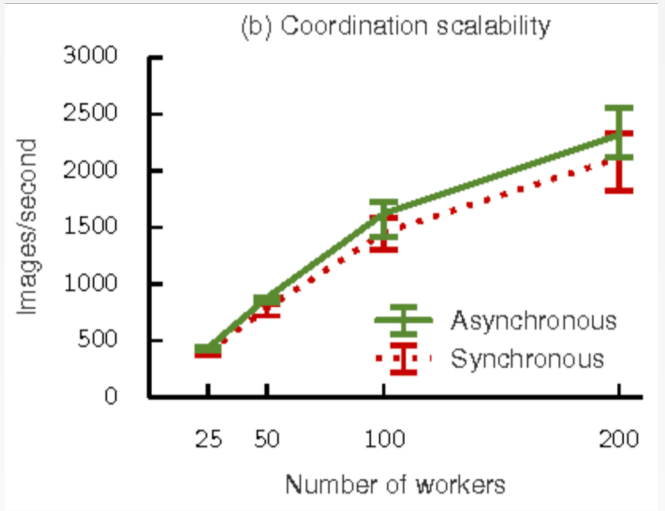
- **Questions investigated:**
  - Can TF facilitate scalable training of Inception-v3 using multiple replicas?
- **Dataset:**
  - They compare the performance achieved while training the Inception model using asynchronous SGD on TF and Apache MXNet (modern DL framework that uses parameter server architecture)
- **Results:**
  - Results are bound by single-GPU performance; both TF and MXNet use cuDNN version 5.1 :. results are similar

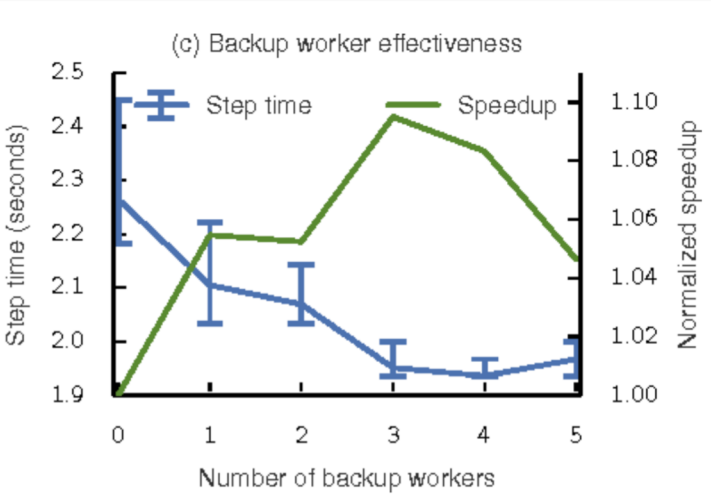


Source: https://ai.google/research/pubs/pub45381

EXP. 3: IMAGE CLASSIFICATION (2/2)

- **Questions investigated:**
  - How does coordination effect training performance?
  - For synchronous training, can adding backup workers reduce overall step time?
- **Dataset:**
  - Inception model trained on larger internal cluster
- **Results:**
  - Training throughput improves for async and sync as workers are added, but within diminishing returns due to resulting competition for PS network resources
  - Adding up to 4 backup workers reduces median step time; > 4 degrades performance



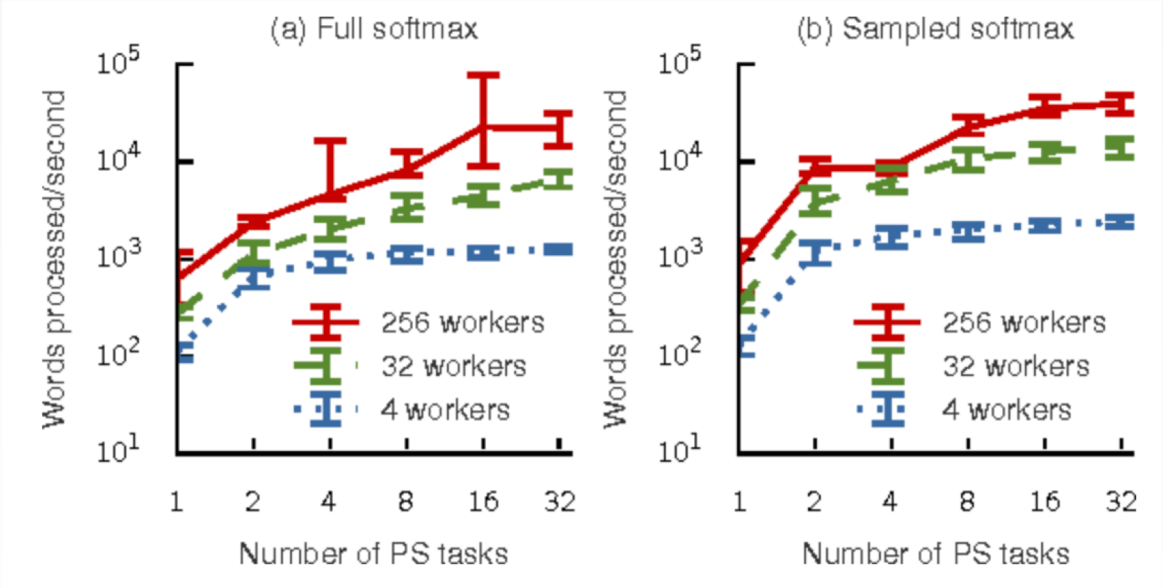


Source: https://ai.google/research/pubs/pub45381

EXP. 4: LANGUAGE MODELING

- **Questions investigated:**
  - Can TF facilitate the training of a recurrent neural network that can be used to develop a language model for the text in the One Billion Word Benchmark?
- **Dataset:**
  - Benchmark set contains ~800K unique words
  - Cardinality of the vocabulary |V| bounds training performance, so they use 40K most common words
  - They vary the number of PS and worker tasks, and softmax implementations

- **Results:**
  - Adding more PS tasks increases throughput
  - Sampled softmax reduces data transfer and computation required for PS tasks



Sources: https://ai.google/research/pubs/pub45381; http://www.statmt.org/lm-benchmark/

DISCUSSION QUESTIONS

- What are key strengths of this approach?
- What are key weaknesses/limitations?
- If you have experience working with TensorFlow, how does it compare to other high-scale ML frameworks you've worked with?
- In your opinion, is using a dataflow graph to represent ML/DL tasks an intuitive/well-suited design choice? Are there alternatives?
- How could TensorFlow be further improved?
- Could we design a system to "learn" how to represent certain types of problems using TensorFlow graphs as input?

BIBLIOGRAPHY

- https://arxiv.org/abs/1312.3005
- Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large scale distributed deep networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 1. Curran Associates Inc., USA, 1223-1231.
- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: a system for large-scale machine learning. In Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'16). USENIX Association, Berkeley, CA, USA, 265-283.
- http://www.memdump.io/2015/11/09/tensorflow-googles-latest-machine-learning-software-is-open-sourced/
- http://www.pittnuts.com/2016/08/glossary-in-distributed-tensorflow/
- https://www.safaribooksonline.com/library/view/learning-tensorflow/9781491978504/ch01.html
- http://www.statmt.org/lm-benchmark/
- http://www.wolframalpha.com/input/?i=tensor