

CS341: Project in Mining Massive Datasets

Advanced ML in Google Cloud



Abhay Agarwal (MS Design '19)

Agenda

- General Notes on Pipelining
- Some History
- Distributed Processing in Tensorflow
- Cloud ML Engine in Google Cloud
- Misc. features and TPUs

Background on ML pipelines

- What is an ML pipeline?

```
1 # load a file
2 f = engine.adobe.File("/Users/josh/Downloads/collaborate-better-go-faster.jpg")
3 engine.adobe.app.load(f)
4
5 # get the active layer for the new document
6 layer = engine.adobe.app.activeDocument.activeLayer
7
8 # duplicate the active layer and adjust the brightness/contrast
9 for i in range(-50, 50, 10):
10     brightness = -1 * i
11     contrast = i
12     new_layer = layer.duplicate()
13     new_layer.adjustBrightnessContrast(brightness, contrast)
14
15
```



Background on ML pipelines

- What is an ML pipeline?
- Why do we need an ML pipeline?

Local machine is not fast enough to run the computations effectively

Require specialized hardware

Hard drive isn't large enough to store data

Want to do stream rather than batch processing

Want to parallelize tasks using multiple machines

Want to collaborate on development without replicating dev state

Want to get several of these features “for free” without changing my workflow (too much)

... a bit of history of server pipelining

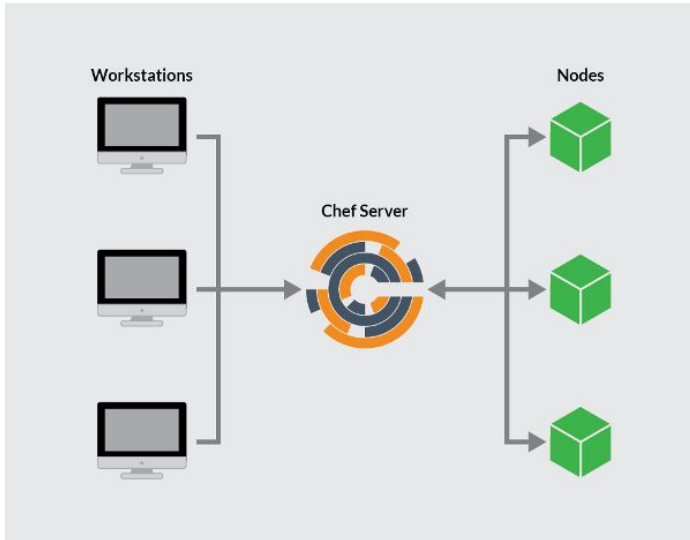
Here's a very basic way to orchestrate your servers...

What's wrong?

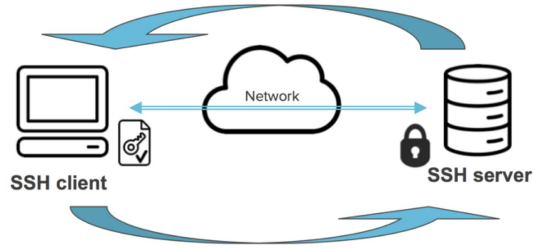
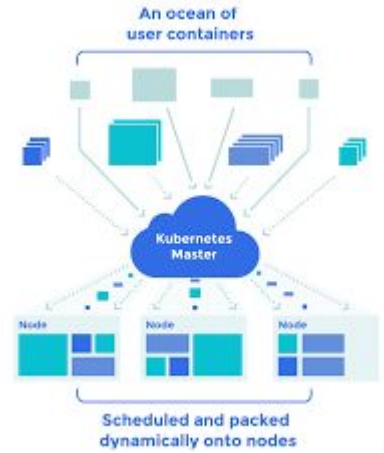
```
$ for SVR in 1 2 3
> do
> ssh root@server0$SVR.example.com -p
*****
> # DO SOMETHING
> done
```

... a bit of history of scaling datacenters

Here are slightly less basic way to orchestrate your servers:

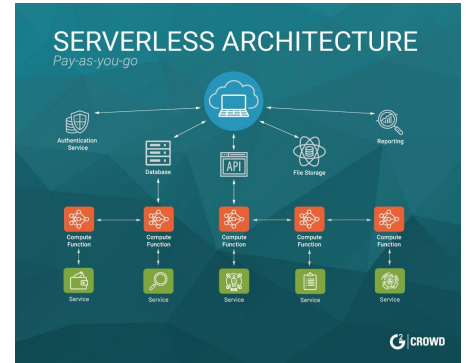
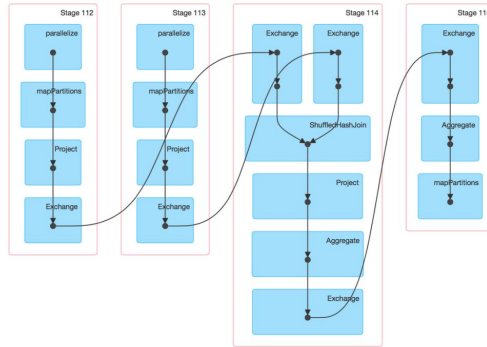


Types of pipelines



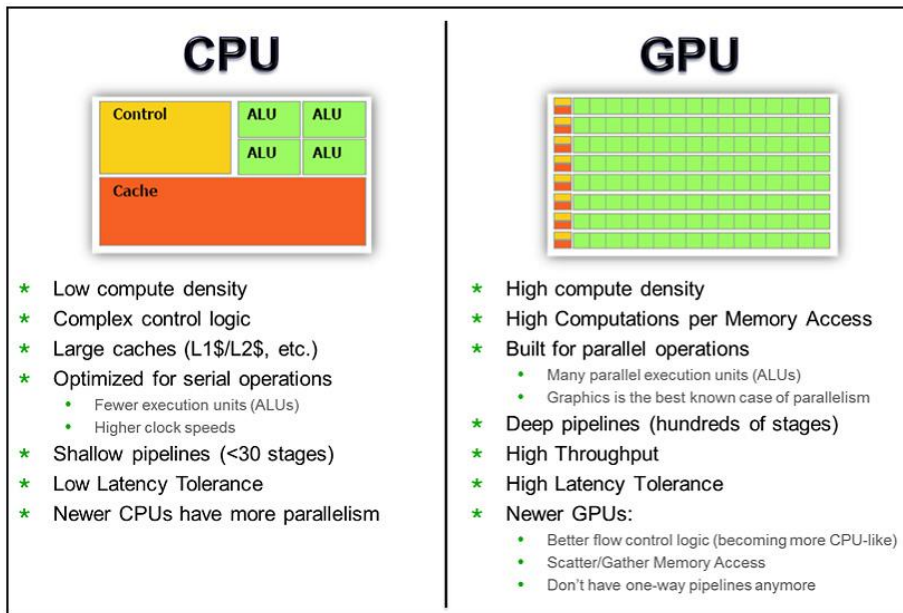
Details for Job 8

Status: SUCCEEDED
Completed Stages: 4
▶ Event Timeline
▼ DAG Visualization



A Note on GPU sharing

- GPUs are very difficult to virtualize for obvious reasons...
 - CUDA (Nvidia GPU API) is essentially written for single processes
 - GPU memory-sharing limits processing capabilities
 - Time-sharing: interleave processes in time domain (doesn't add any savings...)
- Though, this will probably change in our lifetime



Easy Mode: TensorFlow Cluster Primitive

- Create multiple tensorflow processes
- Communicate over sockets
- Can live on multiple servers (with tensorflow server daemon)
- Can live on same machine with different GPUs or with same CPUs

tf.train.ClusterSpec construction	Available tasks
<pre>tf.train.ClusterSpec({"local": ["localhost:2222", "localhost:2223"]})</pre>	 /job:local/task:0 /job:local/task:1
<pre>tf.train.ClusterSpec({ "worker": ["worker0.example.com:2222", "worker1.example.com:2222", "worker2.example.com:2222"], "ps": ["ps0.example.com:2222", "ps1.example.com:2222"] })</pre>	 /job:worker/task:0 /job:worker/task:1 /job:ps/task:0 /job:ps/task:1

Easy Mode: TensorFlow Cluster Primitive


tf.train.ClusterSpec construction

```
tf.train.ClusterSpec({"local": ["localhost:2222", "localhost:2223"]})
```

```
tf.train.ClusterSpec({  
  "worker": [  
    "worker0.example.com:2222",  
    "worker1.example.com:2222",  
    "worker2.example.com:2222"  
  ],  
  "ps": [  
    "ps0.example.com:2222",  
    "ps1.example.com:2222"  
  ]  
})
```

Available tasks

```
 /job:local/task:0  
/job:local/task:1
```

```
 /job:worker/task:0  
/job:worker/task:1  
/job:worker/task:2  
/job:ps/task:0  
/job:ps/task:1
```

So why might you want to do this?

```
with tf.device("/job:ps/task:0"):  
    weights_1 = tf.Variable(...)  
    biases_1 = tf.Variable(...)
```

```
with tf.device("/job:ps/task:1"):  
    weights_2 = tf.Variable(...)  
    biases_2 = tf.Variable(...)
```

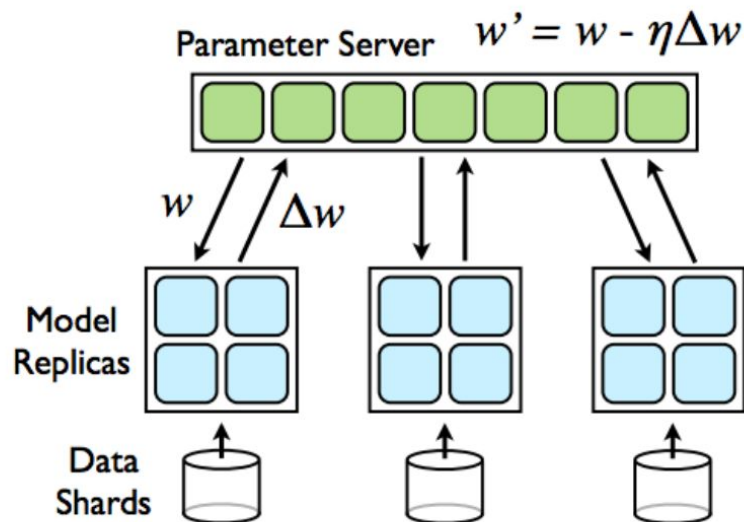
```
with tf.device("/job:worker/task:7"):  
    input, labels = ...  
    layer_1 = tf.nn.relu(tf.matmul(input, weights_1)  
+ biases_1)  
    logits = tf.nn.relu(tf.matmul(layer_1,  
weights_2) + biases_2)  
    # ...  
    train_op = ...
```

```
with tf.Session("grpc://worker7.example.com:2222")  
as sess:  
    for _ in range(10000):  
        sess.run(train_op)
```

Easy Mode: TensorFlow Cluster Primitive

So why might you want to do this?

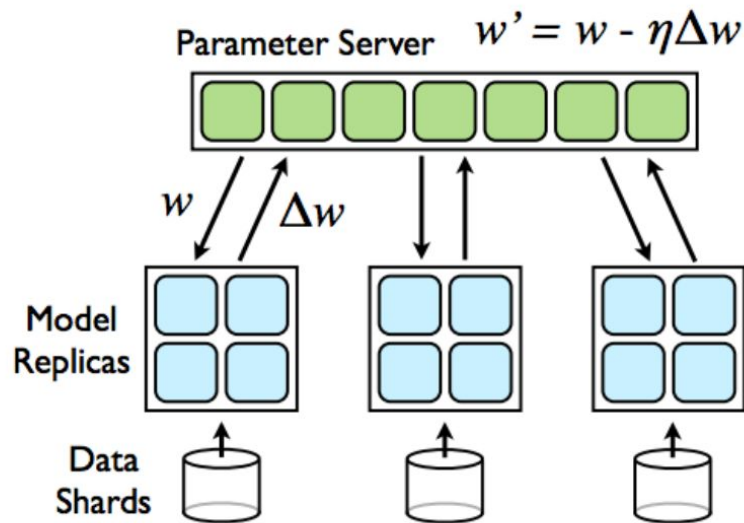
- Lots of data and lots of GPUs
- Data \gg learning rate
- Certain algorithms benefit from this kind of parallelism (e.g. A3C)
- Gradients roughly commutative



Easy Mode: TensorFlow Cluster Primitive

Merging gradients

- Synchronous: Gradient Averaging
- Asynchronous: Gradient aggregation



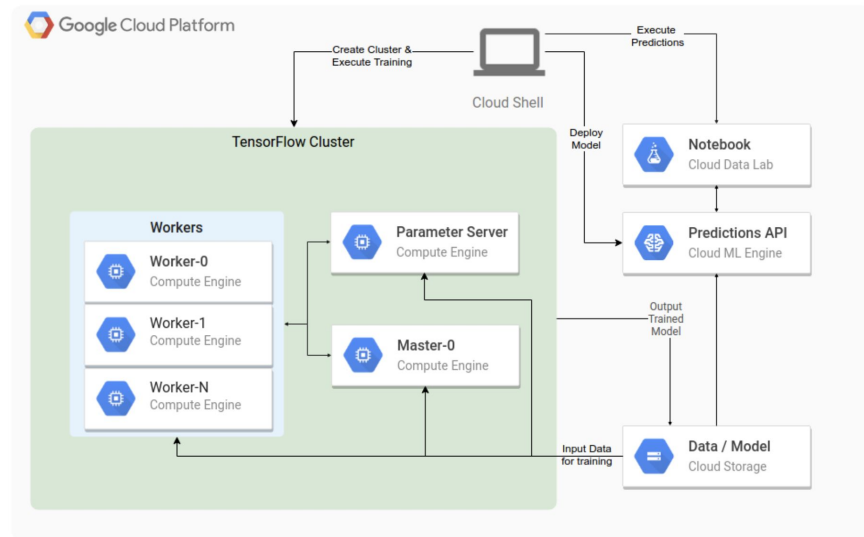
Easy Mode: TensorFlow Cluster Primitive

- Takeaways:
 - Tensorflow can abstract out between-process or between-machine communication
 - Potential massive time savings for compute-intensive network training

- Future
 - Potential for containerization (e.g. Kubernetes-style)
 - Potential for high-level software abstraction (e.g. Spark-style)

Cloud ML

- Simple API for testing and deploying tensorflow/python code
- Local development environment
 - Single node mode
 - Distributed mode
- Cloud deployment functionalities
 - Online prediction (i.e. serverless event-driven)
 - Batch prediction



Cloud ML - local testing

Specify env vars:

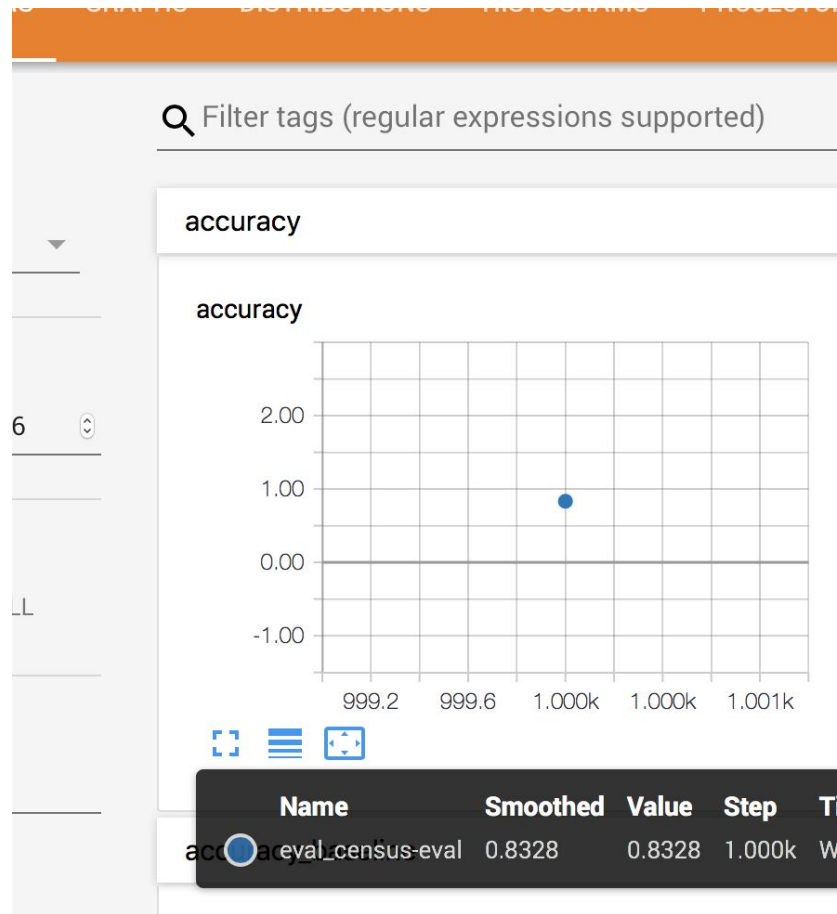
```
MODEL_DIR=output  
TRAIN_DATA=$(pwd)/data/adult.data.csv  
EVAL_DATA=$(pwd)/data/adult.test.csv
```

Build and train your model locally:

```
gcloud ml-engine local train \  
  --module-name trainer.task \  
  --package-path trainer/ \  
  --job-dir $MODEL_DIR \  
  -- \  
  --train-files $TRAIN_DATA \  
  --eval-files $EVAL_DATA \  
  --train-steps 1000 \  
  --eval-steps 100
```

Inspect results in Tensorboard:

```
tensorboard --logdir=$MODEL_DIR
```



Cloud ML - deploy remotely

Create a cloud storage bucket and upload your data:

```
gsutil mb -l $REGION gs://$BUCKET_NAME
```

Now point your environment vars to the new data:

```
TRAIN_DATA=gs://$BUCKET_NAME/data/adult.data.csv  
EVAL_DATA=gs://$BUCKET_NAME/data/adult.test.csv  
TEST_JSON=gs://$BUCKET_NAME/data/test.json  
OUTPUT_PATH=gs://$BUCKET_NAME/$JOB_NAME
```

And run a (slightly modified) command:

```
gcloud ml-engine jobs submit training $JOB_NAME \  
  --job-dir $OUTPUT_PATH \  
  --runtime-version 1.4 \  
  --module-name trainer.task \  
  --package-path trainer/ \  
  --region $REGION \  
  -- \  
  --train-files $TRAIN_DATA \  
  --eval-files $EVAL_DATA \  
  --train-steps 1000 \  
  --eval-steps 100 \  
  --verbosity DEBUG
```


Cloud ML - train remotely

Google Cloud Platform CS341-Demo

Jobs [REFRESH](#)

Filter jobs

Job ID	Type	Creation time	Elapsed time	Logs
census_single_1	Training	May 23, 2018, 7:51:19 AM	54 sec	View logs

Google Cloud Platform CS341-Demo

[CREATE METRIC](#) [CREATE EXPORT](#) [REFRESH](#) [PLAY](#)

Filter by label or text search

Cloud ML Job, census_single_1 All logs Any log level Last hour Jump to now

Showing logs from the last hour ending at 7:52 AM (PDT) [View Options](#)

No older entries found matching current filter in the last hour. [Load older logs](#)

- 2018-05-23 07:51:19.088 PDT service Validating job requirements...
- 2018-05-23 07:51:19.333 PDT service Job creation request has been successfully validated.
- 2018-05-23 07:51:19.539 PDT Waiting for job to be provisioned.
- 2018-05-23 07:51:19.557 PDT service Job census_single_1 is queued.
- 2018-05-23 07:51:21.830 PDT Waiting for TensorFlow to start.

[Load newer logs](#)

Cloud ML - deploy model

The screenshot shows the Google Cloud Platform Storage console. The top navigation bar includes 'Google Cloud Platform', 'CS341-Demo', and a search icon. The left sidebar contains 'Storage', 'Browser', 'Transfer', 'Transfer Appliance', and 'Settings'. The main content area is titled 'Bucket details' for 'cs341-demo-mlengine'. It includes buttons for 'EDIT BUCKET' and 'REFRESH BUCKET'. Below the bucket name are tabs for 'Objects' and 'Overview'. Action buttons include 'Upload files', 'Upload folder', 'Create folder', and 'Delete'. A search bar is labeled 'Filter by prefix...'. The breadcrumb path is 'Buckets / cs341-demo-mlengine / census_single_1 / export / census / 1527087194'. A table lists the bucket's contents:

<input type="checkbox"/>	Name	Size	Type	Storage class	Last modified	Share publicly	Encryption [?]
<input type="checkbox"/>	saved_model.pb	482.6 KB	-	Regional	5/23/18, 7:53 AM	<input type="checkbox"/>	Google-managed key
<input type="checkbox"/>	variables/	-	Folder	-	-		-

```
MODEL_NAME=census
MODEL_BINARIES=gs://$BUCKET_NAME/census_single_1/export/census/1527087194/
gcloud ml-engine versions create v1 \
--model $MODEL_NAME \
--origin $MODEL_BINARIES \
--runtime-version 1.4
```

Cloud ML - productize model

```
test.json — census UN
task.py model.py test.json
1 {"age": 25, "workclass": " Private", "education": " 11th", "education_num": 7, "marital_status": "
  Never-married", "occupation": " Machine-op-inspct", "relationship": " Own-child", "race": "
  Black", "gender": " Male", "capital_gain": 0, "capital_loss": 0, "hours_per_week": 40, "
  native_country": " United-States"}
2
```

```
gcloud ml-engine predict \
--model $MODEL_NAME \
--version v1 \
--json-instances \
../test.json
```

```
Creating version (this might take a few minutes).....done.
→ estimator gcloud ml-engine predict \
--model $MODEL_NAME \
--version v1 \
--json-instances \
../test.json

CLASS_IDS  CLASSES  LOGISTIC  LOGITS  PROBABILITIES
[0]        [u'0']   [0.06335537880659103]  [-2.6935441493988037]  [0.93664461374
28284, 0.06335537135601044]
→ estimator
```

Cloud ML - further features

- Distributed mode (runs multiple parallel workers)
- Hyperparameter Tuning (trains multiple concurrent models and selects best)
- Easy to connect GPUs and TPUs

Cloud ML - Using TPUs

- Disclaimer: I have not used TPUs in my work
- What is a TPU?



Cloud ML - Using TPUs

- Disclaimer: I have not used TPUs in my work
- What is a TPU?
- Results are mixed
 - Hosted GPUs are more predictable and not necessarily slower
 - TPUs are more capable for inference but not necessarily training
 - Fine tuning/optimizing DL training is key
- <https://cloud.google.com/ml-engine/docs/tensorflow/using-tpus>