Lecture 34:

• HW-6 Overview: Pub/Sub, Google Dataflow, etc.

Announcements:

• R-5 out
• HW-6 out
• Project Part 2 due

HW-6

General Theme: Pipelines with some TensorFlow

• Pub/Sub distributed message queue
• Google Cloud Scheduler
• Google Dataflow (Apache Beam implementation) ... most of HW-6
• Vertex AI Pipelines
• Tensorflow examples
(1) Pub/Sub: distributed message queues

- A topic represents a queue
- A publisher can send messages to a topic
- A subscriber can pull messages from a topic (or have them pushed)

Benefits of distributed message queues:

- Decouple service-to-service communication (e.g., many-to-many problem)
- Provides buffering, e.g., for large amounts and/or real-time data
- Provide fault-tolerance, latency/throughput guarantees, etc.

Some similar systems:

- Open source: RabbitMQ, redis (more than a message queue)
- AWS Simple Queue Service (serverless), Kafka
- Azure Queue Storage

(2) Google Cloud Scheduler: service for scheduling jobs on a recurring basis

- e.g., to extract information and send to pub/sub
- or to periodically run BigQuery or ML jobs, etc.

Example with Pub/Sub and Google Cloud Functions

- create a pub/sub topic
- setup scheduler to emit a message (e.g., every minute, hour, etc.)
- write a cloud function (serverless) to be triggered by pub/sub message
- see: https://cloud.google.com/scheduler/docs/tut-gcf-pub-sub

Schedules are based on cron (which is an old Unix command):

- allows you to set up shell commands to run on a schedule
- see https://en.wikipedia.org/wiki/Cron
(3) Google Cloud Dataflow

- implementation of Apache Beam (i.e., the dataflow paper from R-4)
- serverless (auto scaling) service for running Beam pipelines
- note: Beam is based on FlumeJava, which is a precursor to Spark
- plus adds real-time streaming support, etc.

Basic model: … note: pipeline means a “workflow” DAG

- Pipeline: the end-to-end job (e.g., read data, transform, write data)
- PCollection: distributed data set (bounded or unbounded)
- PTransform: pipeline transformation step (over PCollections)
- Various IO transforms available for reading and writing data
- Can use either direct pipeline runner (local testing) or dataflow runner

A Beam program typically involves:

- create a Pipeline object (p) with various execution options
- create an initial PCollection to get data into pipeline
- add PTransform’s to each PCollection, to form a dataflow graph
- add a file (IO) transform to save final PCollection to an external source

Examples of PTransforms:

- ParDo: similar to a Map phase in MapReduce
- GroupByKey: similar to Shuffle phase in MapReduce
- CoGroupByKey: relational join of two or more key/value PCollections
- Combine: similar to Reduce phase (e.g., apply some type of aggregate)
- Flatten: merge PCollections (w/ same data types) into one PCollection
- Partition: split PCollection into many via given function
The pipe | operator in Unix/Linux:

```bash
$ cat foods.txt | grep 'french' | sort
french dip
french dressing
french fries
french onion soup
french toast
```

- output of cat piped to input of grep
- output of grep piped to input of sort

Overloaded bitwise | operator in Python for defining Beam pipelines:

```python
word_lengths = words | beam.Map(len)
```

Similarly to add a name to each step via bitwise >> operator:

```python
lines = p | 'ReadMyFile' >> ReadFromText('gs://some/inputData.txt')
```

VertexAI Pipelines

- Another serverless pipeline approach in Google
- Two different options – Kubeflow or Tensorflow Extended (TFX)
- Can define a DAG of steps, where each step can be executed in parallel
- Workflow steps for data preprocessing, model training, model deployment

For HW-6, just do a basic pipeline (Kubeflow)

- A TFX example is an option (see assignment)

Tensorflow:

- a few labs to get started, get a taste
- choose two of a few additional labs