Let’s SMACK!
Building Real-time Data pipelines

Agenda

- SMACK
- Apache Mesos & DC/OS
- Frameworks
- Message Queues
  - Apache Kafka
- Stream Processing
  - Apache Spark
  - Apache Flink
- Storage
  - Apache Cassandra
  - Apache HDFS
- TensorFlow
Ancient Times...

MapReduce is crunching Data
Today...

We need to turn faster!
Fast Data

<table>
<thead>
<tr>
<th>Days</th>
<th>Hours</th>
<th>Minutes</th>
<th>Seconds</th>
<th>Microseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>Micro-Batch</td>
<td>Event Processing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reports what has happened using descriptive analytics

Solves problems using predictive and prescriptive analytics

Billing, Chargeback  
Product recommendations  
Real-time Pricing and Routing  
Real-time Advertising  
Predictive User Interface
The SMACK Stack

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

ANALYZE
Real-time and batch process data

STORE
Distributed & highly scalable database

ACT
Visualize data and build data driven applications

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Devices</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Kafka</td>
<td>Apache Spark</td>
<td>Apache Cassandra</td>
</tr>
</tbody>
</table>

Mesos/DC/OS
NAIVE APPROACH

Typical Datacenter
siloed, over-provisioned servers, low utilization

Industry Average
12-15% utilization

Spark/Hadoop
Kafka
mySQL
microservice
Cassandra
<table>
<thead>
<tr>
<th>Typical Datacenter</th>
<th>Mesos/ DC/OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>siloed, over-provisioned servers, low utilization</td>
<td>automated schedulers, workload multiplexing onto the same machines</td>
</tr>
</tbody>
</table>

Typical Datacenter:
- Spark/Hadoop
- Kafka
- mySQL
- microservice
- Cassandra

Mesos/ DC/OS:
- Spark/Hadoop
- Kafka
- mySQL
- microservice
- Cassandra
Apache Mesos

- A top-level Apache project
- A cluster resource negotiator
- Scalable to 10,000s of nodes
- Fault-tolerant, battle-tested
- An SDK for distributed apps
- Native Docker support
Two-level Scheduling

1. Agents advertise resources to Master
2. Master offers resources to Framework
3. Framework rejects / uses resources
4. Agent reports task status to Master
Mesos Storage Options

- Default Sandbox
  - Simple to use, Task failures
- Persistent Volumes
  - Task failures, (permanent) Node failures
- Distributed File System/External Storage
  - Node failures, non-local writes
DC/OS ENABLES MODERN DISTRIBUTED APPS

Modern App Components

- Microservices (in containers)
- Functions & Logic

- Streaming
- Batch
- Machine Learning
- Search
- Time Series
- SQL / NoSQL

Datacenter Operating System (DC/OS)

- Container Orchestration
- Security & Governance
- Monitoring & Operations
- User Interface

Distributed Systems Kernel (Mesos)

Any Infrastructure (Physical, Virtual, Cloud)
DC/OS is ...

- Open source (ASL2.0)
  + A big, diverse community

- Disclaimer:
  During the workshop we will the EE version, for the only reason of simplified login.
Lab 0

GUI and CLI
GUI

Your connection is not private

Attackers might be trying to steal your information from `joerg-tf1a7b-pub-mas-elb-1367604716.us-west-2.elb.amazonaws.com` (for example, passwords, messages or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

- Automatically send some system information and page content to Google to help detect dangerous apps and sites. Privacy Policy

HIDE ADVANCED

This server could not prove that it is `joerg-tf1a7b-pub-mas-elb-1367604716.us-west-2.elb.amazonaws.com`; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to `joerg-tf1a7b-pub-mas-elb-1367604716.us-west-2.elb.amazonaws.com` (unsafe)
First Container

Run a Service

Service
Configure your service below. Start by giving your service an ID.

SERVICE ID
Give your service a unique name within the cluster, e.g. my-service.

CONTAINER IMAGE
Enter a Docker image you want to run, e.g. nginx.

COMMAND
A shell command for your container to execute.

INSTANCES

CPU

Memory (MB)

Container Runtime
The container runtime is responsible for running your service. We support the Mesos and Docker containerizers.

- DOCKER ENGINE
  Docker's container runtime. No support for multiple containers (Pods) or GPU resources.

- MESOS RUNTIME
  Universal Container Runtime (UCR) using native Mesos engine. Supports Docker file format, multiple containers (Pods) and GPU resources.
INSTALLING THE CLI

Install DC/OS CLI

Installation

Choose your operating system and follow the instructions. For any issues or questions, please refer to our documentation.

Copy and paste the code snippet into the terminal:

```bash
curl -fLs --retry 20 -Y 100000 -y 60 https://downloads.mesosphere/ dcos mv dcos /usr/local/bin &&
sudo chmod +x /usr/local/bin/dcos &&
dcos config set core.dcos_url http://instructo-elastic
```

Close
HYPERSCALE MEANS VOLUME AND VELOCITY

Batch
Reports what has happened using descriptive analytics

Micro-Batch

Event Processing
Solves problems using predictive and prescriptive analytics

Billing, Chargeback
Product Recommendations
Real-time Pricing and Routing
Real-time Advertising
Predictive User Interface
FAST Data pipeline

**EVENTS**

Broad variety of connected sensors and apps produce constant streams of ubiquitous data — from cars, wearables, buildings to everything else.

**INGEST**

Highly scalable publish-subscribe event bus ensures ubiquitous data is captured with no loss, supporting millions of events per second.

**ANALYZE**

Data from event bus and persistent database is processed using batch, streaming, machine learning, and graph compilation to gain new insights.

**STORE**

Highly durable, available, and scalable database stores processed data from analysis and applications.

**ACT**

Present insights to a human, or trigger actions in connected devices or applications.
Frameworks Challenge
SMACK stack

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

ANALYZE
Real-time and batch process data

STORE
Distributed & highly scalable database

ACT
Visualize data and build data driven applications

<table>
<thead>
<tr>
<th></th>
<th>Apache Kafka</th>
<th>Apache Spark</th>
<th>Apache Cassandra</th>
<th>Akka</th>
<th>DC/OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensors</td>
<td>Devices</td>
<td>Clients</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MESSAGE QUEUES
MESSAGE QUEUES

- Apache Kafka
- ØMQ, RabbitMQ, Disque (Redis-based), etc.
- fluentd, Logstash, Flume
- Akka streams
- cloud-only: AWS SQS, Google Cloud Pub/Sub
- see also queues.io
APACHE KAFKA

- High-throughput, distributed, persistent publish-subscribe messaging system
- Originates from LinkedIn
- Typically used as buffer/de-coupling layer in online stream processing
fluentd
HOW TO CHOOSE?

- Scalability
- Message Type
  - Log vs
- Delivery Guarantees/Message durability
- Routing Capabilities
- Failover
- Community
- Mesos Support ;-)}
DELIVERY GUARANTEES

- **At most once**—Messages may be lost but are never redelivered.
- **At least once**—Messages are never lost but may be redelivered.
- **Exactly once**—this is what people actually want, each message is delivered once and only once.

Murphy’s Law of Distributed Systems:

*Anything that can go wrong, will go wrong ... partially!*
Routing

Simple Pipes

Routing

RabbitMQ
Kafka - Partitions

Anatomy of a Topic

Partition 0
0 1 2 3 4 5 6 7 8 9 1 0 1 1 2

Partition 1
0 1 2 3 4 5 6 7 8 9

Partition 2
0 1 2 3 4 5 6 7 8 9 1 0 1 1 2

Old —— New

 Writes
Kafka - Replication

Leader (red) and replicas (blue)
Multiple Consumer

- **Scale-out** (load sharing)
- **Fan-out** (consuming the same messages)
- Offset Handling
  - Per partition/consumer group
  - Ordering guarantees
DC/OS Kafka - CLI Operations

- DC/OS support for many operations
  - Connection details
  - Manage topics

```bash
$ dcos kafka endpoints broker
{
  "address": [
    "10.0.0.211:9843",
    "10.0.0.217:10056",
    "10.0.0.214:9689"
  ],
  "dns": [
    "broker-0.kafka.mesos:9843",
    "broker-1.kafka.mesos:10056",
    "broker-2.kafka.mesos:9689"
  ],
  "vip": "broker.kafka.14lb.thisdcos.directory:9092",
  "zookeeper": "master.mesos:2181/dcos-service-kafka"
}
```

```bash
$ dcos kafka topic create topic1
```
DC/OS Kafka - Upgrades

- HA Updates
- Update plan supports individual steps
  - Different Strategies between steps
    - Install
    - Stage
- Default Phases
  - Mesos Reconciliation
  - Update

```bash
curl -H "Authorization: token=$AUTH_TOKEN"
"$DCOS_URI/service/kafka/v1/plan"

$ curl -X PUT -H "Authorization: token=$AUTH_TOKEN"
"$DCOS_URI/service/kafka/v1/plan?cmd=continue"
```
Lab 2

Install and Configure Kafka

Once per Cluster
Add kafka client

```bash
# Create marathon app definition
$ cat <<'EOF' >> kafkaclient.json
{
    "id": "/kafka-client",
    "instances": 1,
    "container": {
        "type": "MESOS",
        "docker": {
            "image": "wurstmeister/kafka:0.11.0.0"
        }
    },
    "cpus": 0.5,
    "mem": 256,
    "cmd": "sleep 100000"
}
EOF

# Deploy marathon app definition
$ dcos marathon app add kafkaclient.json
```
Produce/comsume events

# Produce single `Hello world` event
$ dcos task exec kafka-client bash -c "export JAVA_HOME=/opt/jdk1.8.0_144/jre/; echo 'Hello, World.' | /opt/kafka_2.12-0.11.0.0/bin/kafka-console-producer.sh --broker-list broker.kafka.l4lb.thisdcos.directory:9092 --topic topic1"

# Consume events from topic1
$ dcos task exec kafka-client bash -c "export JAVA_HOME=/opt/jdk1.8.0_144/jre/; /opt/kafka_2.12-0.11.0.0/bin/kafka-console-consumer.sh --zookeeper master.mesos:2181/dcos-service-kafka --topic topic1 --from-beginning"
Hello, World.
Install Kafka

1. Install a Kafka cluster.

   $ dcos package install kafka

2. Create a new topic.

   $ dcos kafka topic create topic1
3. Find Zookeeper and broker endpoint information.

```bash
$ dcos kafka endpoints zookeeper
master.mesos:2181/dcos-service-kafka

$ dcos kafka endpoints broker
{
    "address": [
        "10.0.3.226:1000",
        "10.0.3.98:1000",
        "10.0.0.120:1000"
    ],
    "dns": [
        "kafka-2-broker.kafka.autoip.dcos.thisdcos.directory:1000",
        "kafka-0-broker.kafka.autoip.dcos.thisdcos.directory:1000",
        "kafka-1-broker.kafka.autoip.dcos.thisdcos.directory:1000"
    ],
    "vip": "broker.kafka.l4lb.thisdcos.directory:9092"
}
```
Install Kafka Cli - Everyone

$ dcos package install kafka --cli
Deploy Kafka-Client

```json
{
    "id": "/kafka-client",
    "instances": 1,
    "container": {
        "type": "MESOS",
        "docker": {
            "image": "mesosphere/kafka-client"
        }
    },
    "cpus": 0.5,
    "mem": 256,
    "cmd": "sleep 100000"
}
```
Install Kafka-Client

$ dcos task exec kafka-client bash -c "echo 'Hello, World.' | /bin/kafka-console-producer.sh --broker-list broker.kafka.l4lb.thisdcos.directory:9092 --topic topic1"

$ dcos task exec kafka-client bash -c "~/bin/kafka-console-consumer.sh --zookeeper master.mesos:2181/dcos-service-kafka --topic topic1 --from-beginning"
DATA PROCESSING AT HYPERSCALE

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

STORE
Distributed & highly scalable database

ANALYZE
Real-time and batch process data

ACT
Visualize data and build data driven applications

Apache Kafka
Apache Cassandra
Apache Spark
Akka

DC/OS
STREAM PROCESSING

- Apache Storm
- Apache Spark
- Apache Samza
- Apache Flink
- Apache Apex
- Concord
- cloud-only: AWS Kinesis, Google Cloud Dataflow
Apache Beam
**APACHE SPARK (STREAMING)**

**Typical Use:** distributed, large-scale data processing; micro-batching

**Why Spark Streaming?**
- Micro-batching creates very low latency, which can be faster
- Well defined role means it fits in well with other pieces of the pipeline
APACHE Flink

Native streaming

Why Apache Flink?
- Low Latency
- Snapshots
- Event Time
- ...

Stateful, event-driven, event-time-aware processing

Event-driven applications (event sourcing, CQRS)

Stream Processing / Analytics (data streams, windows, …)
Batch Processing (data sets)
HOW TO CHOOSE?

- Execution Model
  - Native Streaming vs Microbatch
- Fault Tolerance Granularity
  - Per record, per batch
- Delivery Guarantees
- API
  - SQL
  - Spark
- Performance....
  - Realtime ≠ Realtime
- Community
- Mesos Support ;-)

© 2016 Mesosphere, Inc. All Rights Reserved.
EXECUTION MODEL

Micro-Batching

Native Streaming

Spark

APEX

Flink

STORM

© 2018 Mesosphere, Inc. All Rights Reserved.
Checkpoint per "Batch"

Fault Tolerance

Ack-Per-Record

Batch

Checkpoint per Batch
DELIVERY GUARANTEES

At least Once
PERFORMANCE

https://github.com/yahoo/streaming-benchmarks
https://github.com/intel-hadoop/HiBench
MESOS AND DC/OS SUPPORT

Production
Experimental

Coming Soon

Spark
STORM
Flink
APEX
Lab 2

Install and Configure Spark

Once per Cluster

https://docs.mesosphere.com/service-docs/spark/v2.0.1-2.2.0-1/quickstart/
Lab 2

Install and Configure Flink
Once per Cluster

https://github.com/dcos/examples/blob/master/flink/1.10/README.md
Lab

Flink Demo

Once per Cluster

https://github.com/dcos/demos/tree/master/flink/1.9
DATA PROCESSING AT HYPERSONE

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

STORE
Distributed & highly scalable database

ANALYZE
Real-time and batch process data

ACT
Visualize data and build data driven applications

Sensors
Devices
Clients

Apache Kafka
Apache Cassandra
Apache Spark
Akka

DC/OS
Datastores
## Data Model

<table>
<thead>
<tr>
<th>Relational</th>
<th>Key-Value</th>
<th>Graph</th>
<th>Document</th>
<th>Time-Series</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Schema</td>
<td>- Simple</td>
<td>- Complex relations</td>
<td>- Schema-Less</td>
<td>- HDFS</td>
<td></td>
</tr>
<tr>
<td>- SQL</td>
<td>- Scalable</td>
<td>- Social Graph</td>
<td>- Semi-structured</td>
<td>- Quobyte</td>
<td></td>
</tr>
<tr>
<td>- Foreign</td>
<td>- Cache</td>
<td>- Recommendation</td>
<td>- queries</td>
<td>- influxdb</td>
<td></td>
</tr>
<tr>
<td>- Keys/Joins</td>
<td></td>
<td>- Fraud detections</td>
<td>- Product catalogue</td>
<td>- Ceph</td>
<td></td>
</tr>
<tr>
<td>- OLTP/OLAP</td>
<td></td>
<td></td>
<td>- Session data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relational
- MySQL
- PostgreSQL
- Oracle

### Key-Value
- Redis
- HBase
- Cassandra

### Graph
- Neo4j
- ArangoDB
- Couchbase

### Document
- MongoDB
- MemSQL
- Redis

### Time-Series
- InfluxDB

### Files
Consistency Model - Between Transactions

ACID

Atomic Operations

Couchbase

influxdb

APACHE HBASE

neo4j

redis

MEMSQL

ArangoDB

cassandra

mongoDB
Distribution Model

- Sharding vs Replication
  - Master-Slave
- CAP limitations
  - Consistency
  - Availability
  - Partition Tolerance
DC/OS Cassandra - CLI Operations

- DC/OS support for many operations
  - Connection details
  - Node status
  - Backup
- Also exposed via HTTP API

```bash
$ dcos cassandra endpoints

$ dcos cassandra --name=cassandra node list

$ dcos cassandra --name=cassandra node status

$ dcos cassandra --name=<service-name> backup start \  --backup_name=<backup-name> \  --external_location=s3://<bucket-name> \  --s3_access_key=<s3-access-key> \  --s3_secret_key=<s3-secret-key>
```
DC/OS Cassandra - Backup

- Simple DC/OS CLI command
- Backup one node at a time
- Can be implemented via scheduled DC/OS Job
- Be careful when restoring a backup of a different version
- Prior to 1.0.21 requires separate schema backup

```
$ dcos cassandra --name=<service-name> backup start \
    --backup_name=<backup-name> \
    --external_location=s3://<bucket-name> \
    --s3_access_key=<s3-access-key> \
    --s3_secret_key=<s3-secret-key>

$ dcos cassandra --name=<service-name> backup status

$ dcos cassandra --name=<service-name> backup stop

$ dcos cassandra --name=<service-name> restore start \
    --backup_name=<backup-name> \
    --external_location=azure://<container-name> \
    --s3_access_key=<s3-access-key> \
    --s3_secret_key=<s3-secret-key>
```
Cassandra - Data Consistency

- Cassandra is eventually consistent
- Multiple parameter to tweak read/write consistency
  - Write Strategies:
    - Any, One, Quorum, All, ..
  - Read Strategies:
    - One, Quorum, ALL
- Granularity: single row/key
Relational Database Modelling

- Minimize Number of writes
- Minimize Data Duplication

Cassandra Data Modelling

- Spread Data evenly across the cluster
  - Choose partition key caref
  - Minimize the number of partition reads
  - Consider query workload

4th Normal Form
Lab

Install and Configure Cassandra

Once per Cluster

https://docs.mesosphere.com/service-docs/cassandra/2.0.2-3.0.14/quick-start/
HDFS
Keep it running!
SERVICE OPERATIONS

- Configuration **Updates** (ex: Scaling, re-configuration)
- Binary **Upgrades**
- Cluster **Maintenance** (ex: Backup, Restore, Restart)
- **Monitor** progress of operations
- **Debug** any runtime blockages

More details this afternoon!
Deep Learning Overview

**Traditional Machine Learning**

- Input
- Feature extraction
- Classification
- Output

**Deep Learning**

- Input
- Feature extraction + classification
- Output
### Deep Learning Overview

#### Step 1: Training
- **Input:** Lots of Labeled Data
- **Deep neural network model**
- **Output:** Trained Model

#### Step 2: Inference
- **New Input from Camera or Sensor**
- **Trained Model**
- **Output:** Classification

- **Dog**
  - Input: Lots of Labeled Data
  - Deep neural network model
  - Output: Trained Model
  - 97% Dog
  - 3% Panda
Deep Learning Overview

```
Final Layer Objects

Layer 2 Objects

Layer 1 Objects
```
What is Tensorflow?


- **Machine Intelligence** is the broad term used to describe techniques allowing computers to “learn” by analyzing very large data sets using artificial neural networks.
What is Tensorflow?

“An open-source **software library** for Machine Intelligence” - tensorflow.org

- Tensorflow is a **software library** that makes it easy for developers to construct artificial neural networks to analyze their data of interest
Alternatives

```python
import tensorflow as tf
import numpy as np
X = tf.placeholder("float")
Y = tf.placeholder("float")
W = tf.Variable(np.random.random(), name="weight")
pred = tf.multiply(X, W)
cost = tf.reduce_sum(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
    for t in range(10000):
        x = np.array(np.random.random()).reshape((1, 1, 1, 1))
        y = x * 3
        (_, c) = sess.run([optimizer, cost], feed_dict={X: x, Y: y})
        print(c)
```

```python
import numpy as np
import torch
from torch.autograd import Variable
model = torch.nn.Linear(1, 1)
loss_fn = torch.nn.MSELoss(size_average=False)
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
for t in range(10000):
    x = Variable(torch.from_numpy(np.random.random((1,1))).astype(np.float32))
    y = x * 3
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print(loss.data[0])
```
Data Analytics Ecosystem
Lab 2

First TensorFlow Model
Typical Developer Workflow for TensorFlow (Single-Node)

- Download and install the Python TensorFlow library
- Design your model in terms of TensorFlow’s basic machine learning primitives
- Write your code, **optimized for single-node performance**
- Train your data on a single-node → Output Trained Model
1. Launch container on DC/OS using the Services UI without GPUs

```json
{
    "id": "tensorflow-no-gpus-<MYNAME>",
    "cpus": 4,
    "gpus": 0,
    "mem": 2048,
    "disk": 0,
    "instances": 1,
    "container": {
        "type": "MESOS",
        "docker": {
            "image": "tensorflow/tensorflow"
        }
    }
}
```
2. Run TensorFlow

# Exec into the tensorflow container
$ dcos task exec -it tensorflow-no-gpus-<NAME> bash

export PATH=$PATH:/usr/local/sbin:/usr/sbin:/sbin

# Download TensorFlow-Examples
$ apt-get update; apt-get install -y git
$ git clone https://github.com/aymericdamien/TensorFlow-Examples

# Run the convolutional network example
$ cd TensorFlow-Examples/examples/3_NeuralNetworks
$ time python convolutional_network.py
Training Challenges

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

Input: Lots of Labeled Data

Deep neural network model

Output: Trained Model

Dog
Training Challenges

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

Input: Lots of Labeled Data

Deep neural network model

Output: Trained Model

Dog
Training Challenges

- Compute Intensive
  - (Hopefully) Large Datasets
    - Train
    - Dev
    - Test
  - Hyperparameter
    - #Layer
    - #Units per Layer
    - Learning Rate
    - ....

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

Input: Lots of Labeled Data

Deep neural network model

Output: Trained Model
Why GPUs?

- GPUs are the tool of choice for many big-data cloud applications
  - Deep Learning
  - Natural Language Processing
  - Genome Sequencing
Apache Mesos and GPUs

Mesos Agent

- Containerizer API
- Isolator API

Nvidia GPU Isolator

- Allocates GPUs to tasks
- Isolates Access to GPUs between tasks

Mimics functionality of nvidia-docker-plugin

- Nvidia GPU Allocator
- Nvidia Volume Manager

Linux devices cgroup
Lab 3

TensorFlow meets GPU
TensorFlow meets GPU

# Exec into the tensorflow container
$ dcos task exec -it tensorflow-gpus-1-<NAME> bash

export PATH=$PATH:/usr/local/sbin:/usr/sbin:/sbin

# Download TensorFlow-Examples
$ apt-get update; apt-get install -y git
$ git clone https://github.com/aymericdamien/TensorFlow-Examples

# Run the convolutional network example
$ cd TensorFlow-Examples/examples/3_NeuralNetworks
$ time python convolutional_network.py
Typical Developer Workflow for TensorFlow (Distributed)

- Download and install the Python TensorFlow library
- Design your model in terms of TensorFlow’s basic machine learning primitives
- Write your code, optimized for distributed computation
- …
Typical Developer Workflow for TensorFlow (Distributed)

- ...  
- Provision a set of machines to run your computation  
- Install TensorFlow on them  
- Write code to map distributed computations to the exact IP address of the machine where those computations will be performed  
- Deploy your code on every machine  
- Train your data on the cluster → Output Trained Model
Challenges running distributed TensorFlow

- Hard-coding a “ClusterSpec” is incredibly tedious
  - Users need to rewrite code for every job they want to run in a distributed setting
  - True even for code they “inherit” from standard models

```python
tf.train.ClusterSpec(
    "worker": [
        "worker0.example.com:2222",
        "worker1.example.com:2222",
        "worker2.example.com:2222",
        "worker3.example.com:2222",
        "worker4.example.com:2222",
        "worker5.example.com:2222",
        ...
    ],
    "ps": [
        "ps0.example.com:2222",
        "ps1.example.com:2222",
        "ps2.example.com:2222",
        ...
    ]
)
```
Challenges running distributed TensorFlow

- Dealing with failures is not graceful
  - Users need to stop training, change their hard-coded ClusterSpec, and manually restart their jobs

"As far as we can tell, the system went down because someone stepped on a crack in the sidewalk."
Challenges running distributed TensorFlow

- Manually configuring each node in a cluster takes a long time and is error-prone
  - Setting up access to a shared file system (for checkpoint and summary files) requires authenticating on each node
  - Tweaking hyper-parameters requires re-uploading code to every node
Running distributed TensorFlow on DC/OS

- The dcos-commons SDK cleanly restarts failed tasks and reconnects them to the cluster
Running distributed TensorFlow on DC/OS

- We use DC/OS Secrets (or alternatively environment variables) to pass credentials to every node in the cluster
Running distributed TensorFlow on DC/OS

- We use a runtime configuration dictionary to quickly tweak hyper-parameters between different runs of the same model.

```json
{
  "service": {
    "name": "mnist",
    "job_url": "...",
    "job_context": "...",
  },
  "gpu_worker": {...
  },
  "worker": {...
  },
  "ps": {...
}
```
Lab 5

Distributed TensorFlow
TensorFlow meets GPU

https://github.com/dcos/examples/tree/master/tensorflow
Special Thanks to All Collaborators

Sam Pringle
springle@mesosphere.com
Primary Developer of the DC/OS TensorFlow Package

Ben Wood
Oh
Evan Lezar
Rand
Gabriel Hartmann
Bo Hu

Robin
Art
Chris Lambert
JVM & Container (internals)
What is this about?
Containers
Write Once Run Any Where
Virtual Machines vs Container

Virtual Machine

- Application
- Bins/Libs
- Guest Os
- Hypervisor
- Host Os
- Server

Container

- Application
- Bins/Libs
- DockerEngine
- Host Os
- Server
Virtual Machines vs Container

**Virtual Machine**
- Application
- Bins/Libs
- Guest Os
- Hypervisor
- Host Os
- Server

**Container**
- Application
- Bins/Libs
- DockerEngine
- Host Os
- Server

- **Weaker isolation in container**
- **Container run near-native speed CPU/IO**
- **Container launch in around 0.1 second (libcontainer)**
- **Less storage and memory overhead**
```
$ ps aux
USER   PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root   1     0  0.2  33636  2960 ?        Ss   Oct17   0:00 /sbin/init
...  
root  12972  0.0  3.9 757236 40704 ?        Ssl  01:55   0:18 /usr/bin/dockerd
root  12981  0.0  0.9 299096 9384 ?        Ssl  01:55   0:01 `docker-containerd` -l unix:///var/run/docker/libcontainerd/docker-
root  13850  0.0  0.4 199036 4180 ?        Ssl  01:58   0:00 `docker-containerd-shim` 2f86cbc34/var/run/docker/l
root  13867  0.0  0.2  31752 2884 ?        Ss   01:58   0:01 \_ nginx: master process nginx -g daemon off;
sshd  13889  0.0  0.1  32144  1664 ?        S   01:58   0:00 \_ nginx: worker process
root  17642  0.0  0.4 199036 4188 ?        Ssl  11:54   0:00 \_ docker-containerd-shim /var/run/docker/l
root  17661  0.0  0.2  1172   4 ?        Rs  11:54   23:37 \_ md5sum /dev/urandom
root  18340  0.0  0.4 199036 4144 ?        Ssl  12:16   0:00 \_ docker-containerd-shim 4121c6479262112b /var/run/docker/l
vagrant 18353  0.0  0.4  1164   4 ?        Ss   12:16   0:00 \_ sleep 1000
```
• container runtime* != container image != container instance
• beyond docker runtime
  • Universal Container Runtime
    • supports docker images
• CRI{-o}
  • ...
{  
  "id": "/springboot-demo",
  "cmd": "$JAVA_HOME/bin/java -jar MyApp.jar",
  "instances": 1,
  "fetch": [
    { 
      "uri": "http://.../MyApp.jar",
    },
    { 
      "uri": "https://.../jre-8u121-linux-x64.tar.gz",
    }
  ],
}
Isolation
Namespaces VS. Cgroups

**Namespaces provide isolated views:**
- pid (processes)
- net (network interfaces, routing...)
- ipc (System V IPC)
- mnt (mount points, filesystems)
- uts (hostname)
- user (UIDs)

**Control groups control resources:**
- cpu (CPU shares)
- cpuacct
- cpuset (limit processes to a CPU)
- memory (swap, dirty pages)
- blkio (throttle reads/writes)
- devices
- net_cls, net_prio: control packet class and priority
- freezer
Control Groups
Control groups (v1)

- `/sys/fs/cgroup`
- Each subsystem (memory, CPU...) has a hierarchy (tree)
- Each process belongs to exactly 1 node in each hierarchy
- Each hierarchy starts with 1 node (the root)
- Each node = group of processes (sharing the same resources)

[cgroups V2]
Memory cgroup: limits

• Each group can have **hard** and **soft** limits
• Soft limits are not enforced
• Hard limits will trigger a per-group OOM killer
  • No `OutOfMemoryError`
• Limits can be set for physical, kernel, total memory

```
docker run -it --rm -m 128m fedora bash
```
Cpu cgroup

- Simple Accounting
  - Metrics: cpuacct.stats user | system

- Limitations
  - CPU Shares
  - CPU Sets
CPU Shares

- Priority Weighting across all the cores
  - default 1024
- **Use CFS for hard limit**

```bash
sudo cgcreate -g cpu:A
sudo cgcreate -g cpu:B

cgroup A:  sudo cgset -r cpu.shares=768 A  75%
cgroup B:  sudo cgset -r cpu.shares=256 B  25%
```

docker run -it --rm -c 512 stress …
CPU Sets

- **Pin** groups to **specific CPU(s)**
- **Reserve CPUs** for specific apps
- Avoid processes bouncing between CPUs
- Also relevant for NUMA systems

```bash
docker run -it -cpuset=0,4,6 stress
```
Namespaces
Namespaces

- Provide container (= process groups) with their own view of the system
- Multiple namespaces:
  - pid, net, mnt, uts, ipc, user
- Each process is in one namespace of each type
Pid namespace

- Processes within a PID namespace only see processes in the same PID namespace
- Each PID namespace has its own numbering (starting at 1)
- When PID 1 goes away, the whole namespace is killed

```
ps aux
USER   PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root  1  0.0  0.1 106628 31480 ?        Ss   09:00   0:00 /opt/mesosphere/active/mesos/libexec/mesos/mesos
       -containerizer
root  6  0.0  0.2 776872 37712 ?        Sl   09:00   0:01 mesos-executor --launcher_dir=/opt/mesosphere/active/mesos/libexec/mesos --sandbox_directory=/m
root 16  0.0  0.0   4512   792 ?        Ss   09:00   0:00 sh -c sleep 100000
root 17  0.0  0.0   4384   664 ?        S    09:00   0:00 sleep 100000
root 264 0.2  0.1 106492 31016 ?       Ss   09:32   0:00 /opt/mesosphere/active/mesos/libexec/mesos/mesos-containerizer launch
root 265 0.0  0.0  18240  3316 ?        S    09:32   0:00 /bin/bash
root 276 0.0  0.0  34428  2872 ?        R+   09:33   0:00 ps aux
```
Java
Java Language + Java Specification + Java Runtime
Java Memory Impact

- Native JRE
- Heap
- Perm / meta
- JIT bytecode
- JNI
- NIO
- Threads
From Perm to Metaspace
JRE default initializations

- Based on core count*
  - JIT compiler threads
  - HotSpot thresholds and optimizations
  - Default # threads for GC
  - Number of thread in the common fork-join pool
  - …
Java meets Container

- Development
  - Java App packaged in a container
Java meets Container

• Development
  • Java App packaged in a container

• Production
  • 10 JVM container on a 32 core box
    – 10 * (32 cores are seen by each JRE)
    – 10 * (32 threads set by default for ForkJoinPool)
    – 10 * (32 threads ....)
Java meets Cgroups

Where Java retrieve the core count?

- JDK 7/8 - resources from `sysconf`
  ```c
  sysconf(_SC_NPROCESSORS_ONLN);
  ```

- JDK 9 - `sched_getaffinity`
  - accounts for cpusets
Java with CPU Set

- CPUSSET
  - pin to specific CPUs

- `Runtime.getRuntime().availableProcessors(); == # cores assigned`

```
docker run -ti --cpuset=0,4,6
```
Java with CPU Share

• CPU Share
  – Priority Weighting across all the cores
  – `Runtime.getRuntime().availableProcessors();` == # cores on node

`docker run -ti -c 512`
How about memory?

- `/proc/meminfo`
- `/proc/vmstat`
- `/proc/PID/smaps`
Java 9

Java SE support for Docker CPU and memory limits

By: Donald Smith | Sr. Director of Product Management
• CPU
  • Considers CPU sets
  • Not aware of CPU shares…

• Memory
  • -XX:+UseCGroupMemoryLimitForHeap
  • -XX:+UnlockExperimentalVMOptions
THANK YOU!

ANY QUESTIONS?

@dcos
chat.dcos.io
users@dcos.io
/groups/8295652
/dcos
/dcos/examples
/dcos/demos
GREAT INDIAN DEVELOPER SUMMIT 2019

Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com
@greatindiandev
bit.ly/gidslinkedin
facebook.com/gids19
bit.ly/saltmarchyoutube
flickr.com/photos/saltmarch/