SPARK SUMMIT

SPARK & TENSORFLOW AS-A-SERVICE

Jim Dowling

Assoc Prof, KTH

Senior Researcher, RISE SICS

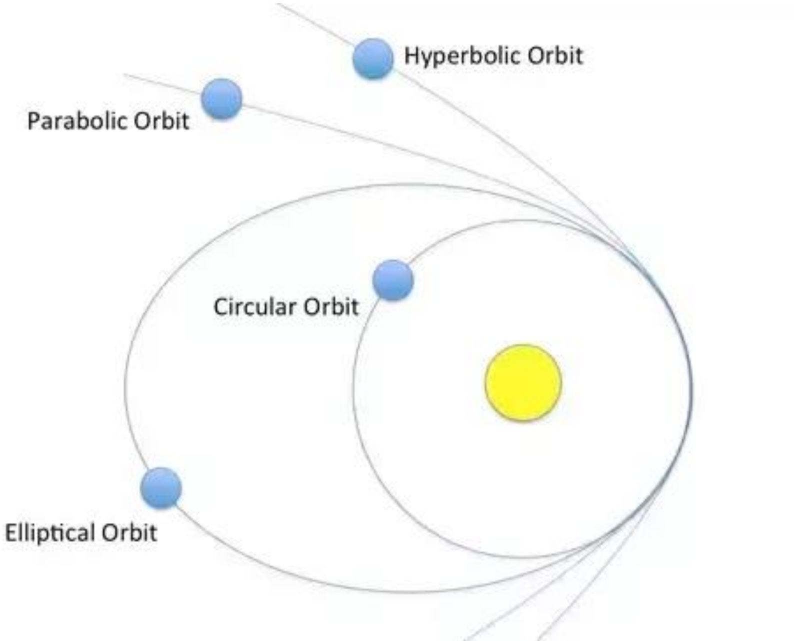
CEO, Logical Clocks AB

#EUai8

Hops

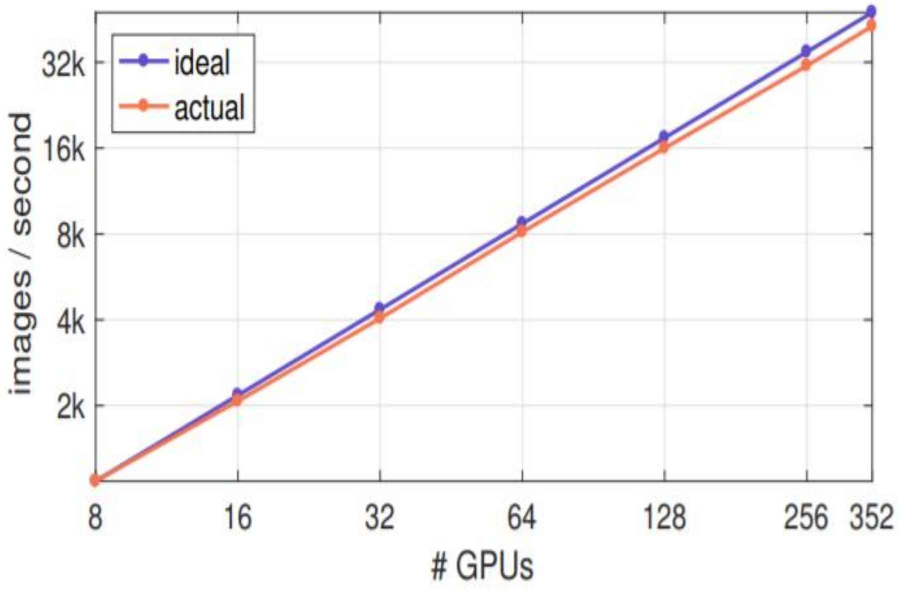
Newton confirmed what many suspected

- In August 1684, Halley visited Newton:
  “What type of curve does a planet describe in its orbit about the sun, assuming an inverse square law of attraction?”



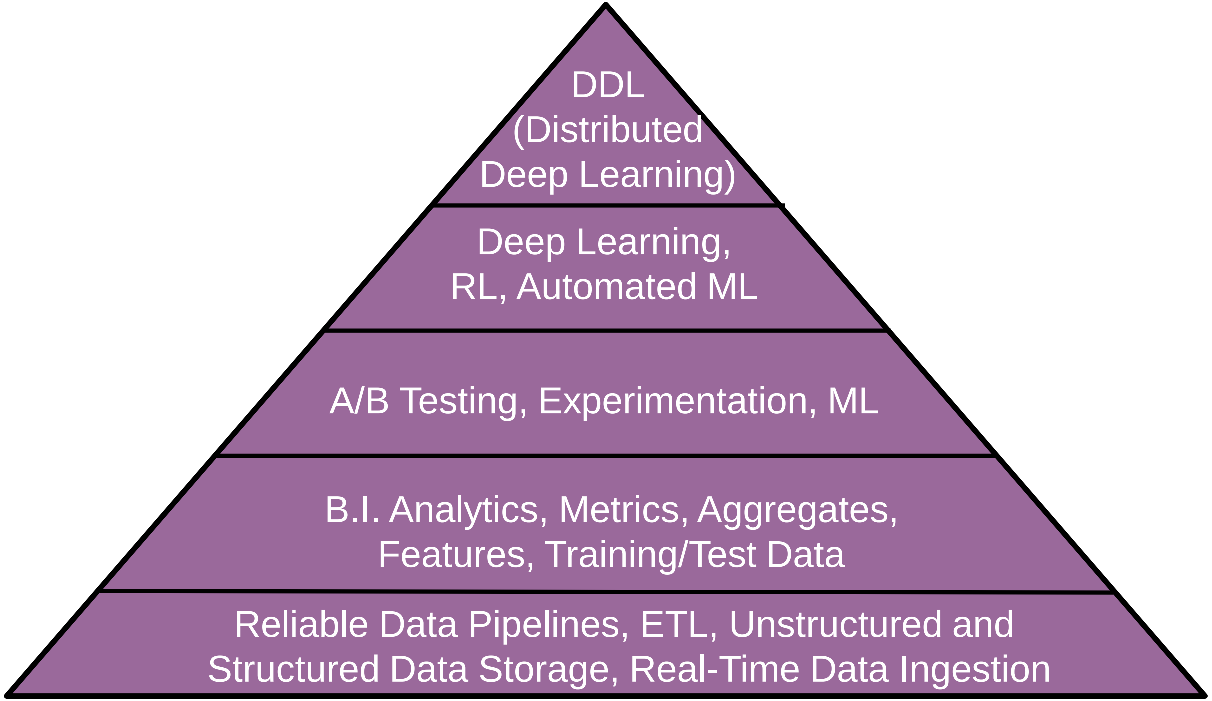
Facebook confirmed what many suspected

- In June 2017, Facebook showed how to reduce training time on ImageNet for a Deep CNN from 2 weeks to 1 hour by scaling out to 256 GPUs.



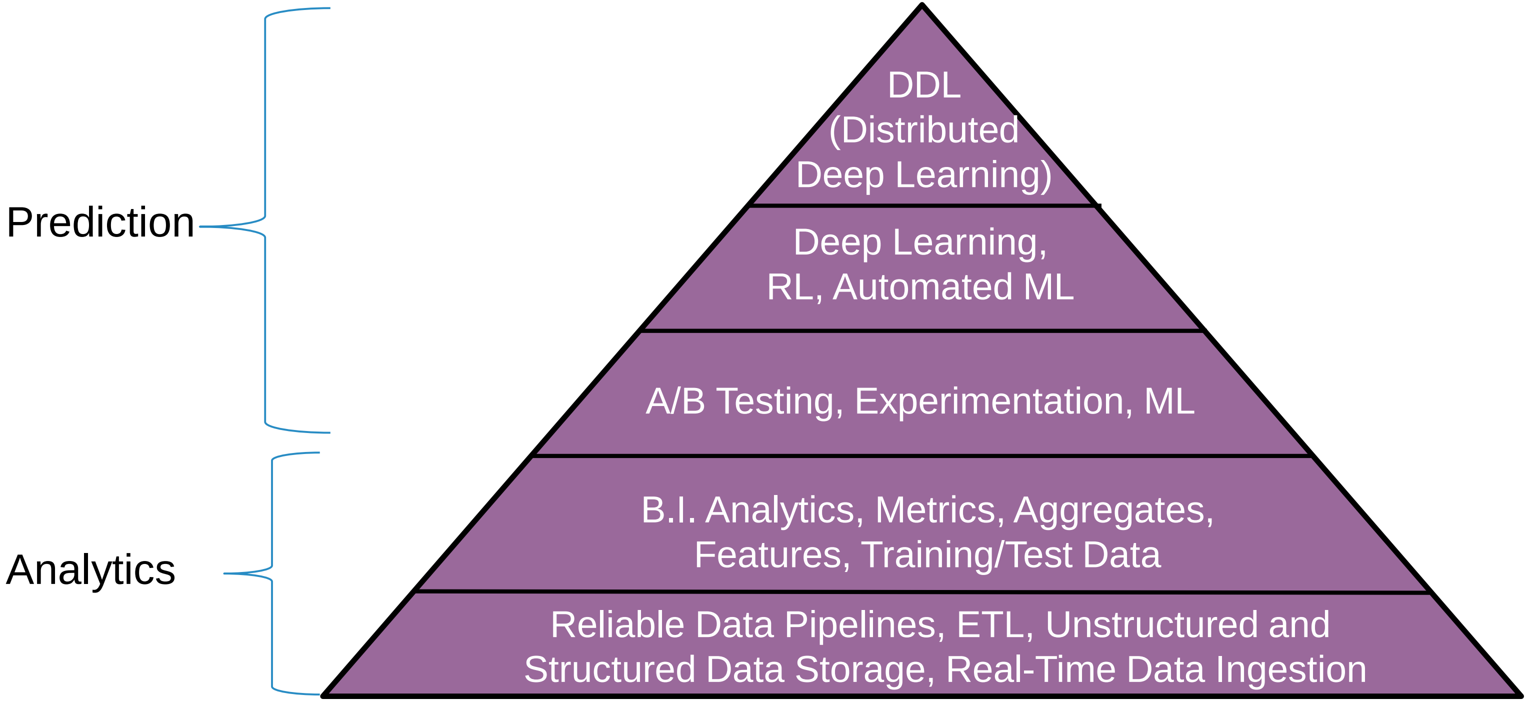
https://arxiv.org/abs/1706.02677

AI Hierarchy of Needs



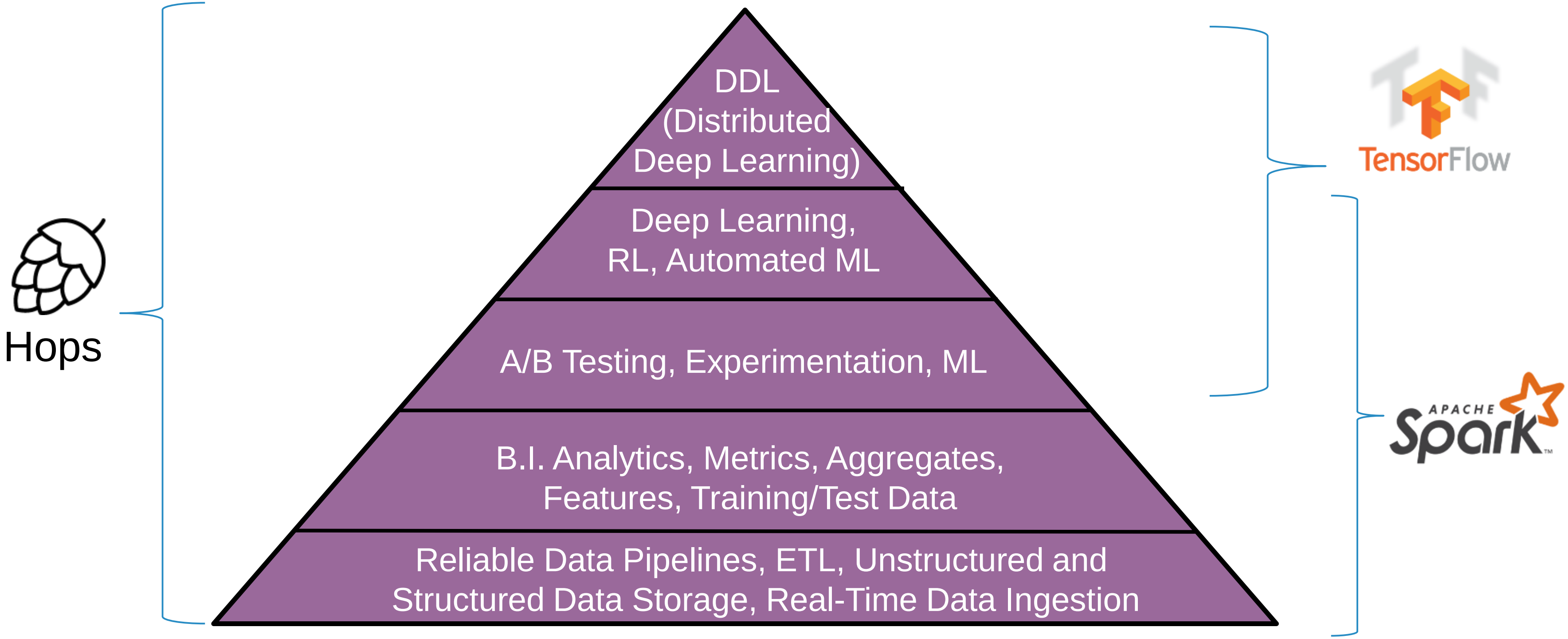
[Adapted from https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007?gi=7e13a696e469]

AI Hierarchy of Needs



[Adapted from https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007?gi=7e13a696e469]

AI Hierarchy of Needs



[Adapted from https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007?gi=7e13a696e469]

Deep Learning Hierarchy of Scale

Training Time for ImageNet

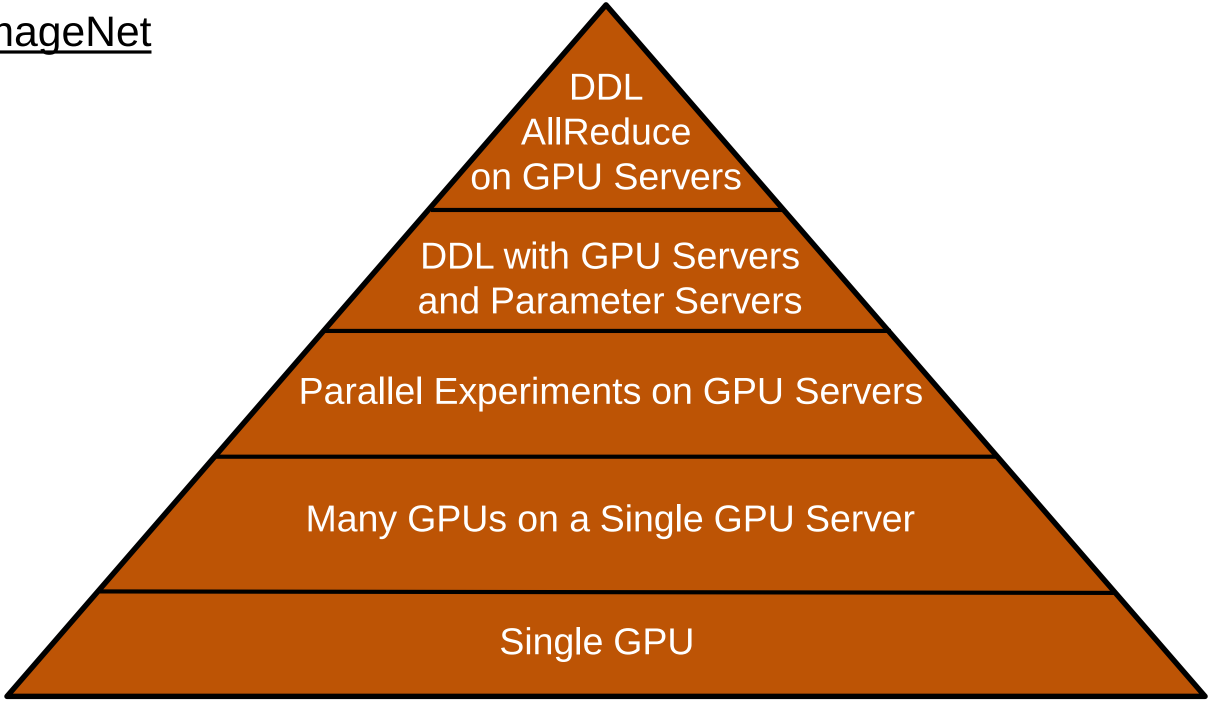
Minutes

Hours

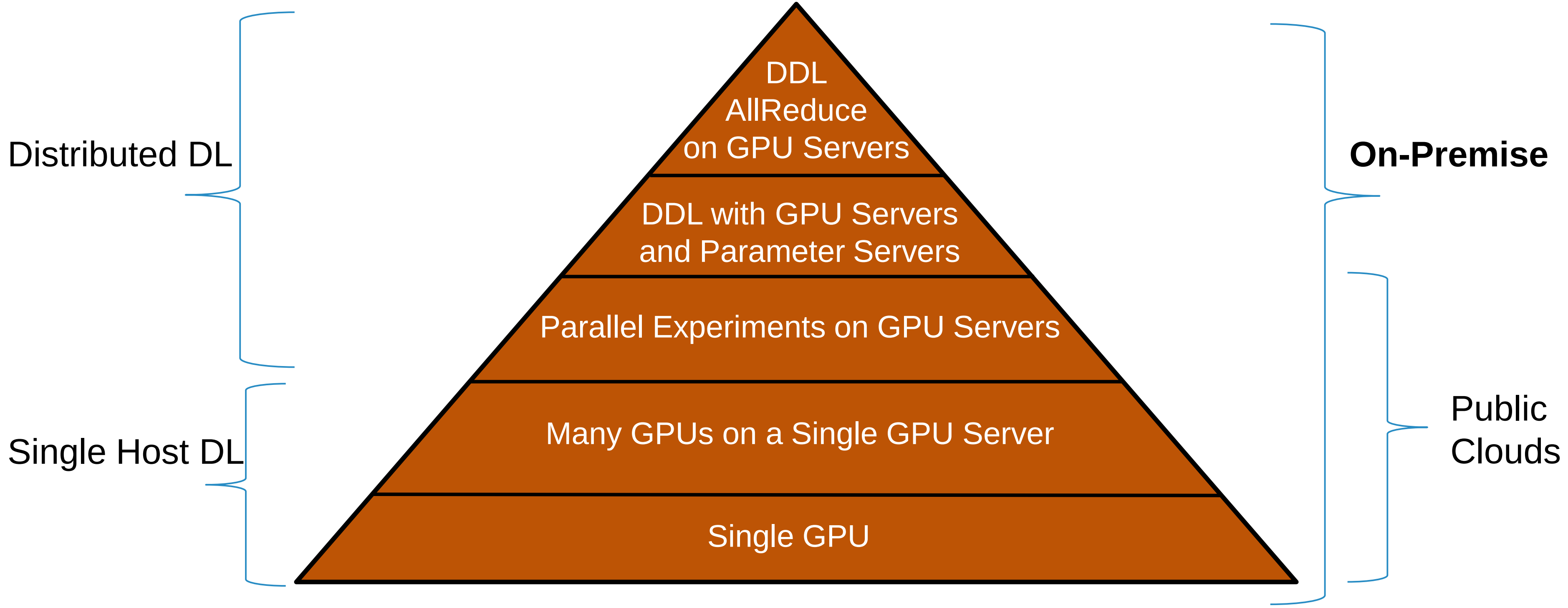
Days/Hours

Days

Weeks



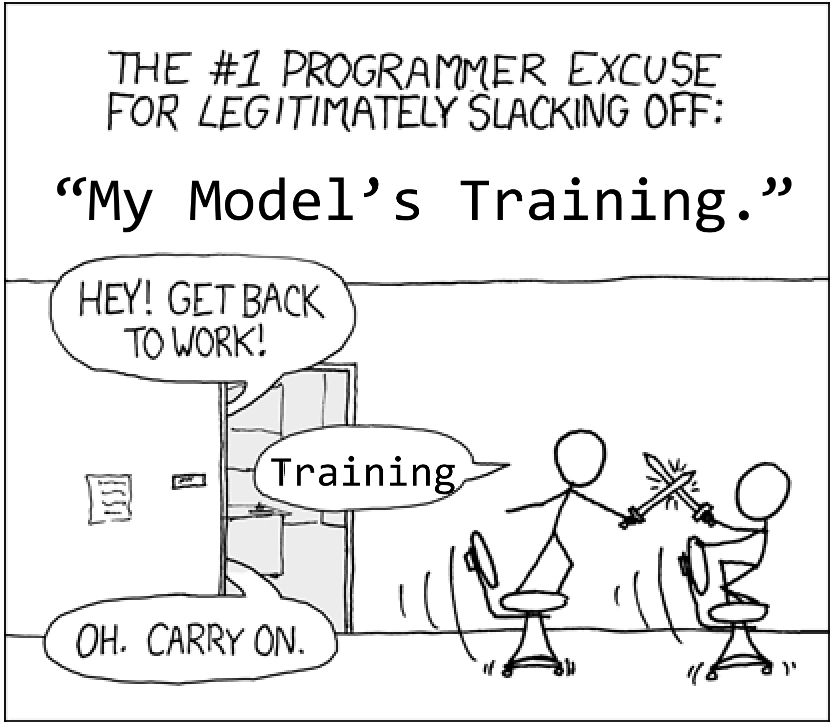
Deep Learning Hierarchy of Scale



DNN Training Time and Researcher Productivity

- Distributed Deep Learning
  - Interactive analysis!
  - Instant gratification!

- Single Host Deep Learning
  - Google-Envy



What Hardware do you Need?

- SingleRoot PCI Complex Server*
  - 10 Nvidia GTX 1080Ti
    - 11 GB Memory
  - 256 GB Ram
  - 2 Intel Xeon CPUs
  - 2x56 Gb Infiniband

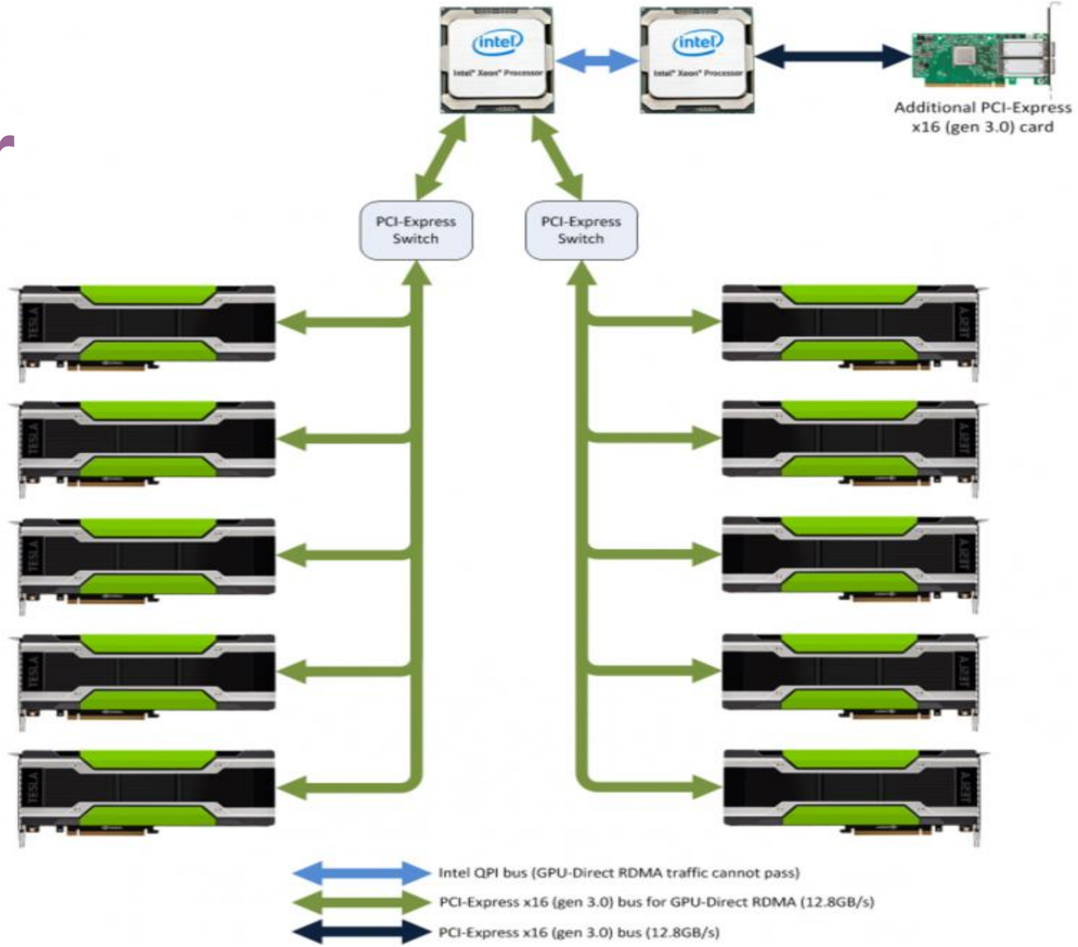
15K Euro

- Nvidia DGX-1
  - 8 Nvidia Tesla P100/V100
    - 16 GB Memory
  - 512 GB Ram
  - 2 Intel Xeon CPUs
  - 4x100 Gb Infiniband
  - NVLink**

up to **150K Euro**

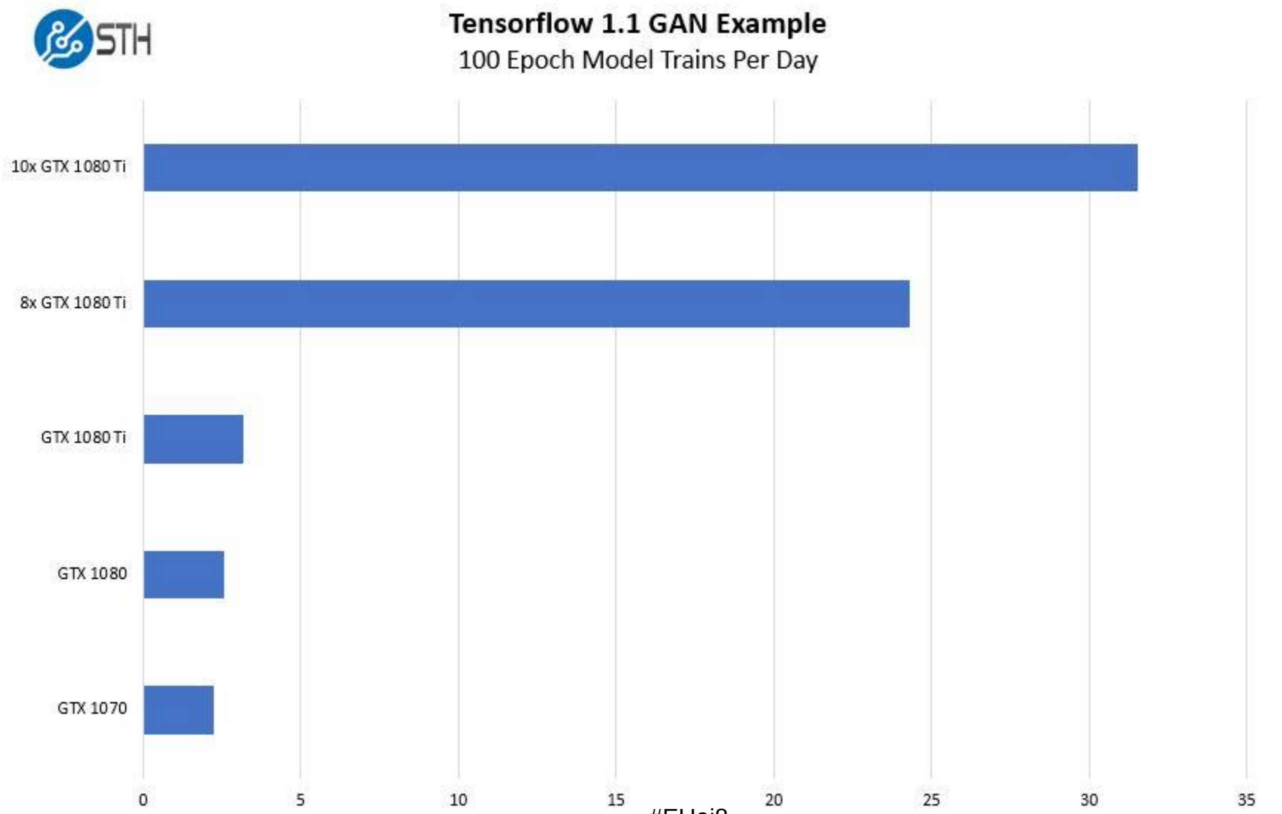
*https://www.servethehome.com/single-root-or-dual-root-for-deep-learning-gpu-to-gpu-systems
**https://www.microway.com/hpc-tech-tips/comparing-nvlink-vs-pci-e-nvidia-tesla-p100-gpus-openpower-servers/

SingleRoot Complex Server with 10 GPUs



[Images from: https://www.microway.com/product/octoputer-4u-10-gpu-server-single-root-complex/]

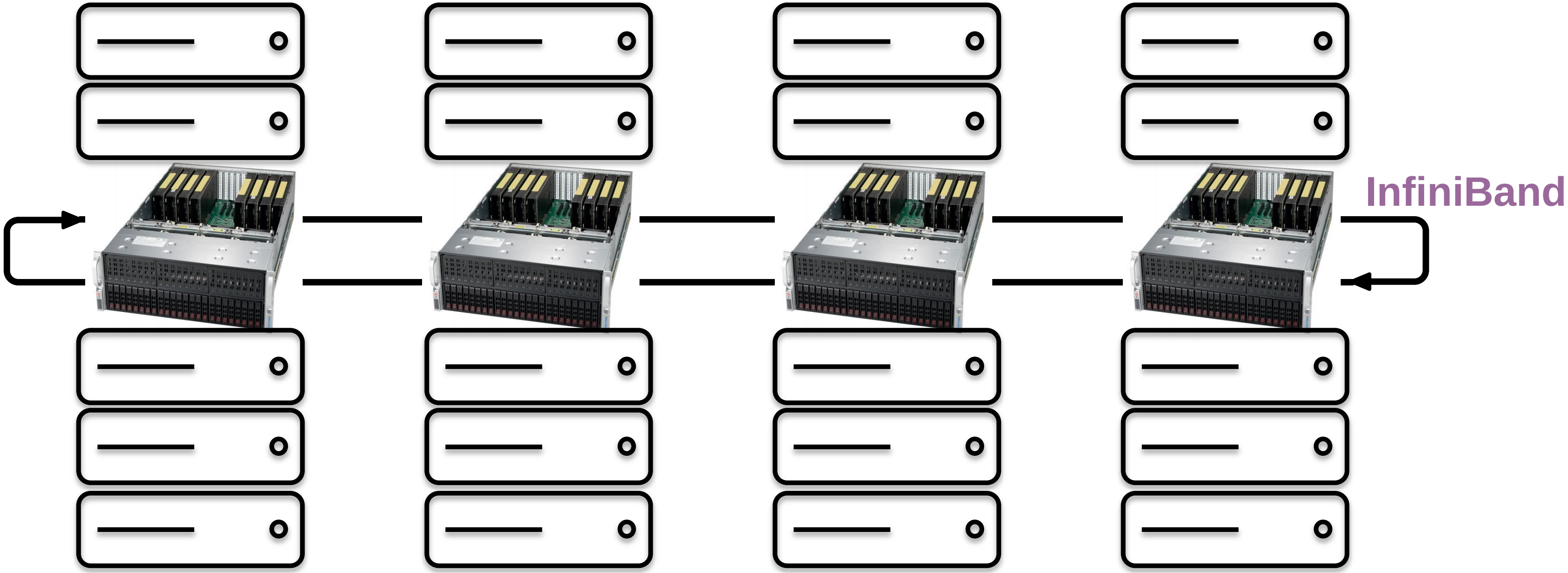
Tensorflow GAN Training Example*



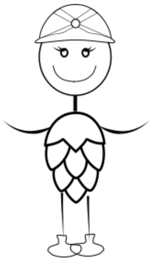
*https://www.servethehome.com/deeplearning11-10x-nvidia-gtx-1080-ti-single-root-deep-learning-server-part-1/

Cluster of Commodity GPU Servers

Max 1-2 GPU Servers per Rack (2-4 KW per server)



Spark and TF – Cluster Integration



Deprecated → **Single GPU Experiment**

Parallel Experiments (HyperParam Tuning)

Distributed Training Job

Cluster Manager

Mix of commodity GPUs and more powerful GPUs good for (1) parallel experiments and (2) distributed training

Training Data and Model Store

GPU Resource Requests in Hops

4 GPUs on any host
10 GPUs on 1 host

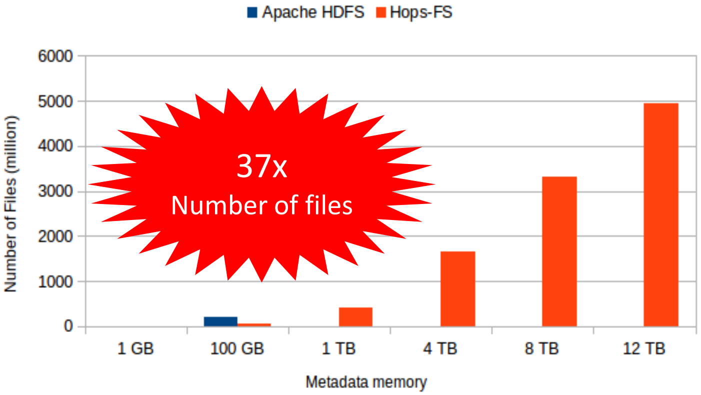
100 GPUs on 10 hosts with ‘Infiniband’
20 GPUs on 2 hosts with ‘Infiniband_P100’

HopsYARN (Supports GPUs-as-a-Resource)

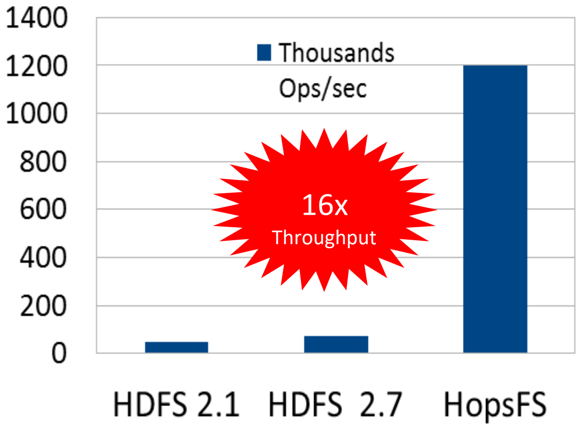
HopsFS

Hops

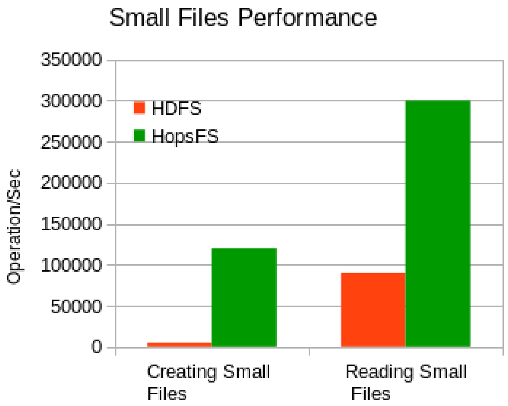
HopsFS: Next Generation HDFS*



Bigger



Faster



Small Files**

IEEE Scale Challenge Winner (2017)

SPARK SUMMIT EUROPE 2017

*https://www.usenix.org/conference/fast17/technical-sessions/presentation/niazi
**https://eurosys2017.github.io/assets/data/posters/poster09-Niazi.pdf

TensorFlow Spark API Integration

- Tight Integration
  - Databricks' Tensorframes and Deep Learning Pipelines

- Loose Integration
  - TensorFlow-on-Spark, Hops TfLauncher
    - PySpark as a wrapper for TensorFlow

Deep Learning Pipelines

```
graph = tf.Graph() with tf.Session(graph=graph) as sess:
   image_arr = utils.imageInputPlaceholder()
   frozen_graph = tfx.strip_and_freeze_until(…)
   transformer = TFImageTransformer(…)
   image_df = readImages("/data/myimages")
   processed_image_df = transformer.transform(image_df)
…
select image, driven_by_007(image) as probability from car_examples
      order by probability desc limit 6
```

Inferencing possible with SparkSQL

Hops TfLauncher – TF in Spark

```
def model_fn(learning_rate, dropout):
    import tensorflow as tf
    from hops import tensorboard, hdfs, devices
 …..
```

“Pure” TensorFlow code in the Executor

```
from hops import tflauncher
args_dict = {'learning_rate': [0.001], 'dropout': [0.5]}
tflauncher.launch(spark, model_fn, args_dict)
```

Launch TF jobs as Mappers in Spark

Hops TfLauncher – Parallel Experiments

```
def model_fn(learning_rate, dropout):
…..
```

Launches 3 Executors with 3 different Hyperparameter settings. Each Executor can have 1-N GPUs.

```
from hops import tflauncher
args_dict = {'learning_rate': [0.001, 0.005, 0.01],
             'dropout': [0.5, 0.6, 0.7]}
tflauncher.launch(spark, model_fn, args_dict)
```

New TensorFlow APIs

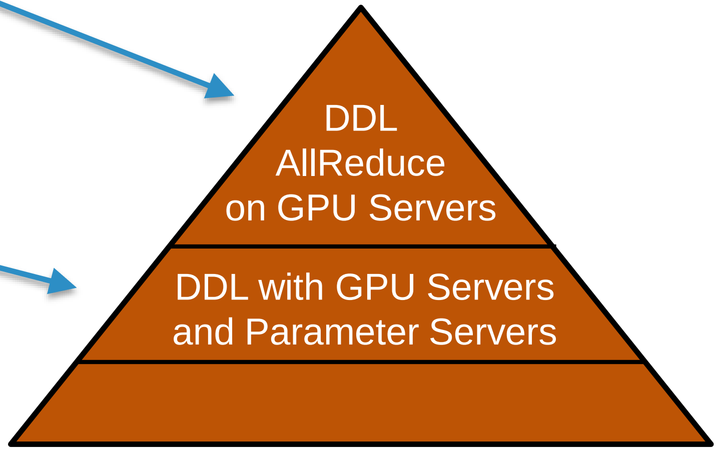
`tf.data.Dataset` `tf.estimator.Estimator` `tf.data.Iterator`

Prefer over RDDs-to-feed_dict

```
def model_fn(features, labels, mode, params):
    …
dataset = tf.data.TFRecordDataset([“/v/f1.tfrecord", “/v/f2.tfrecord"])
dataset = dataset.map(...)
dataset = dataset.shuffle(buffer_size=10000)
dataset = dataset.batch(32)
iterator = Iterator.from_dataset(dataset)
….
nn = tf.estimator.Estimator(model_fn=model_fn, params=dict_hyp_params)
```

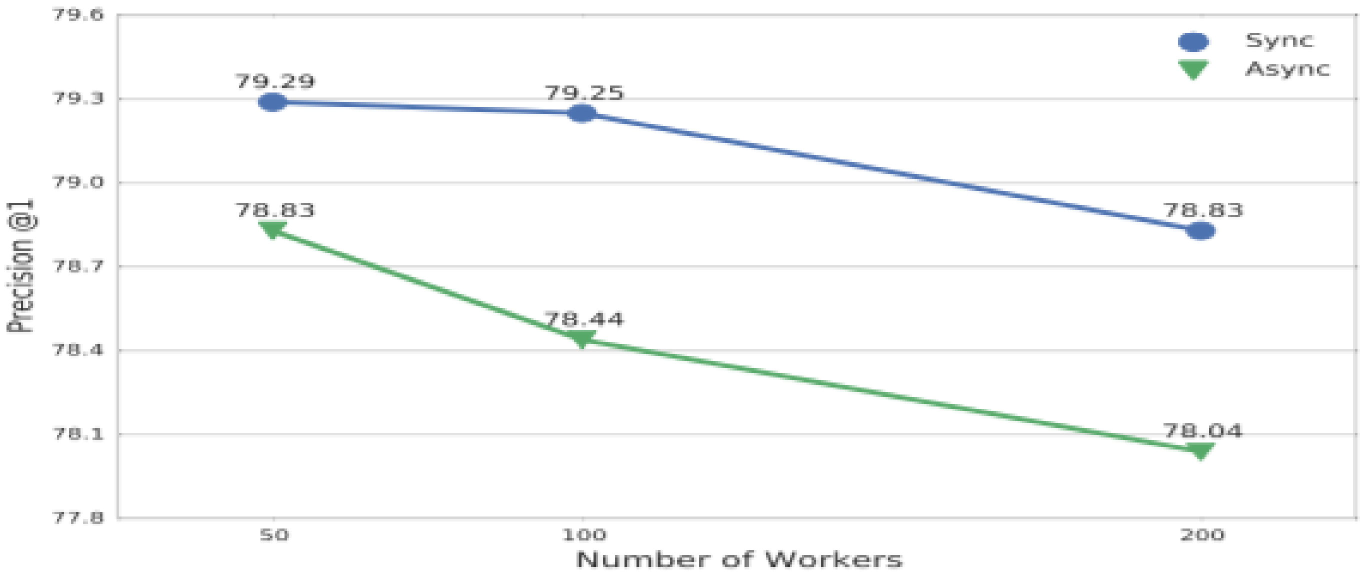
Distributed TensorFlow

- AllReduce
  - Horovod by Uber with MPI/NCCL
  - Baidu AllReduce/MPI in TensorFlow/contrib
- Distributed Parameter Servers
  - TensorFlow-on-Spark
  - Distributed TensorFlow



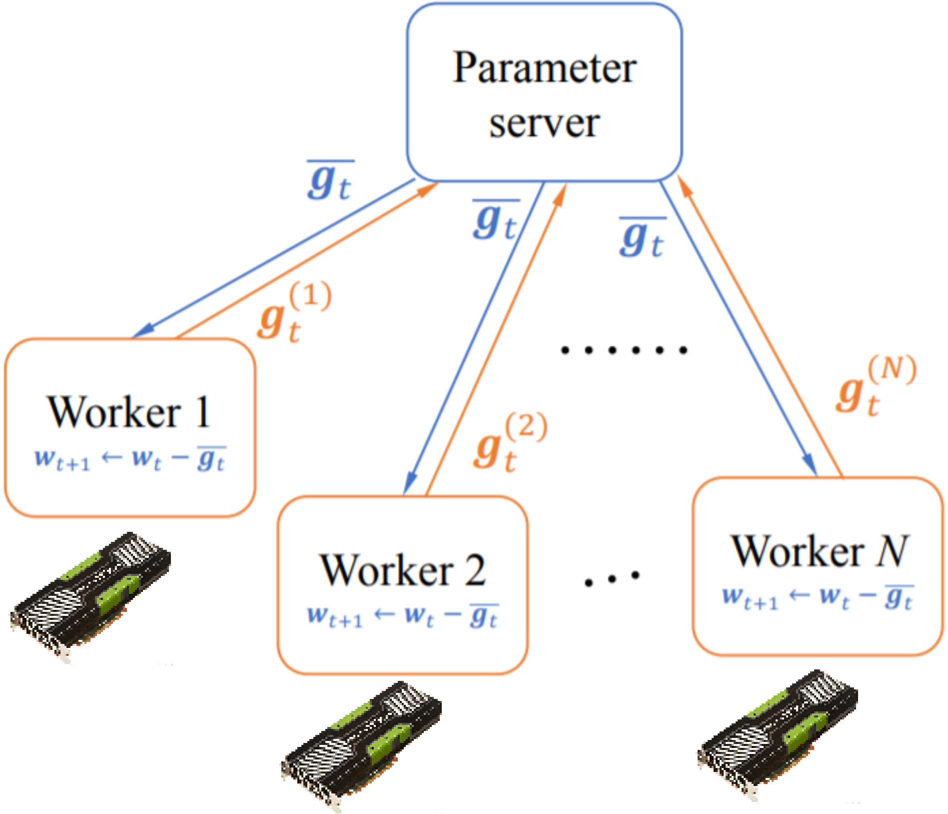
Asynchronous SGD vs Synchronous SGD

- Synchronous Stochastic Gradient Descent (SGD) now dominant, due to improved convergence guarantees:
  - **“Revisiting Synchronous SGD”**, Chen et al, ICLR 2016



Distributed TF with Parameter Servers

Synchronous SGD
with Data Parallelism



Tensorflow-on-Spark (Yahoo!)

- Rewrite TensorFlow apps to Distributed TensorFlow
- Two modes:
  1. feed_dict: RDD.mapPartitions()
  2. TFReader + queue_runner: direct HDFS access from Tensorflow

```
cluster = TFCluster.run(sc, map_fn, args, num_executors,
num_ps, tensorboard, input_mode)

cluster.train(dataRDD, num_epochs=0)

cluster.inference(dataRDD)

cluster.shutdown()
```

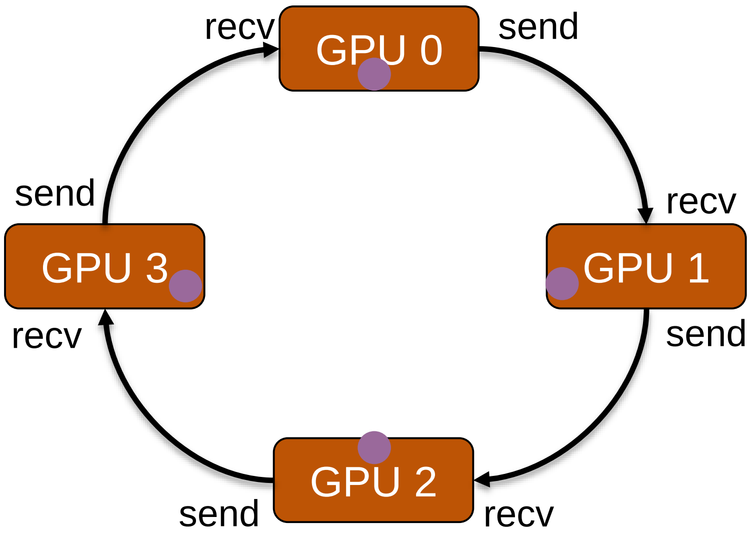
[Image from https://www.slideshare.net/Hadoop_Summit/tensorflowonspark-scalable-tensorflow-learning-on-spark-clusters]

TFonSpark with Spark Streaming

```
from pyspark.streaming import StreamingContext
ssc = StreamingContext(sc, 10)
images = sc.textFile(args.images).map(lambda ln: parse(ln)])
stream = ssc.textFileStream(args.images)
imageRDD = stream.map(lambda ln: parse(ln))
cluster = TFCluster.run(sc, map_fun, args,…)
predictionRDD = cluster.inference(imageRDD)
predictionRDD.saveAsTextFile(args.output)
predictionRDD.saveAsTextFiles(args.output)
ssc.start()
cluster.shutdown(ssc)
```

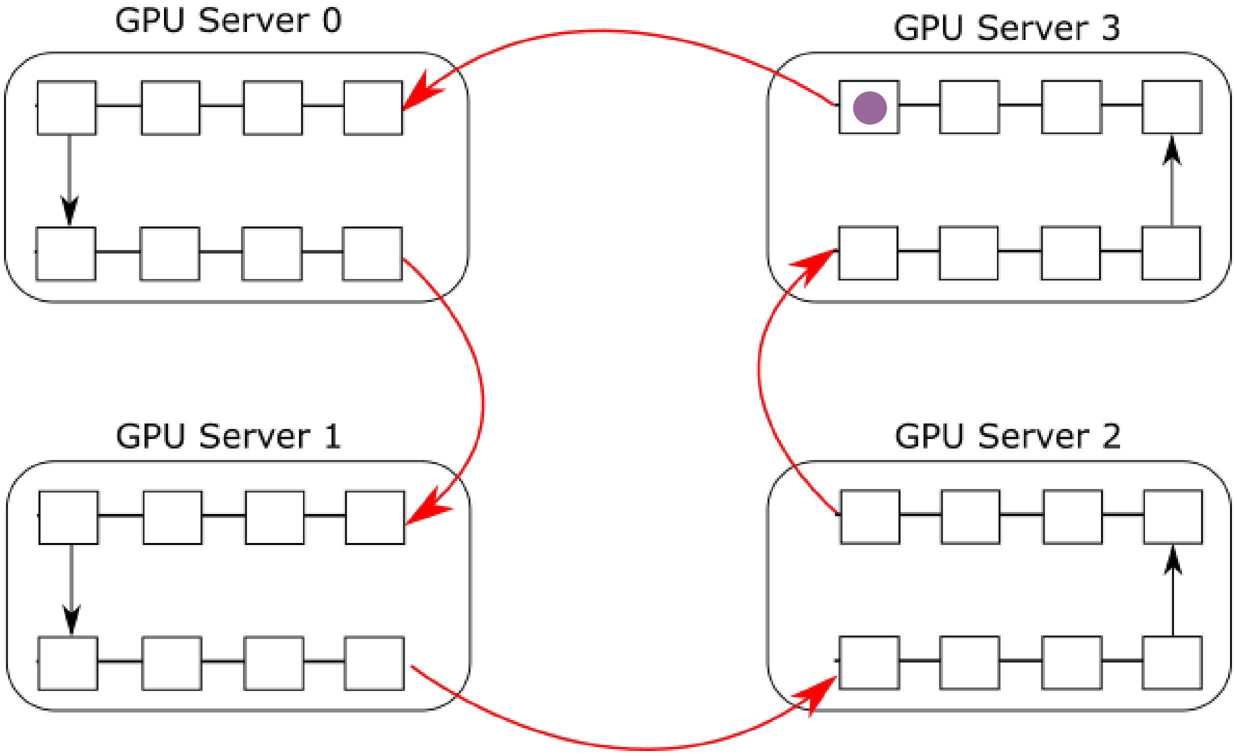
[Image from https://www.slideshare.net/Hadoop_Summit/tensorflowonspark-scalable-tensorflow-learning-on-spark-clusters]

All-Reduce/MPI



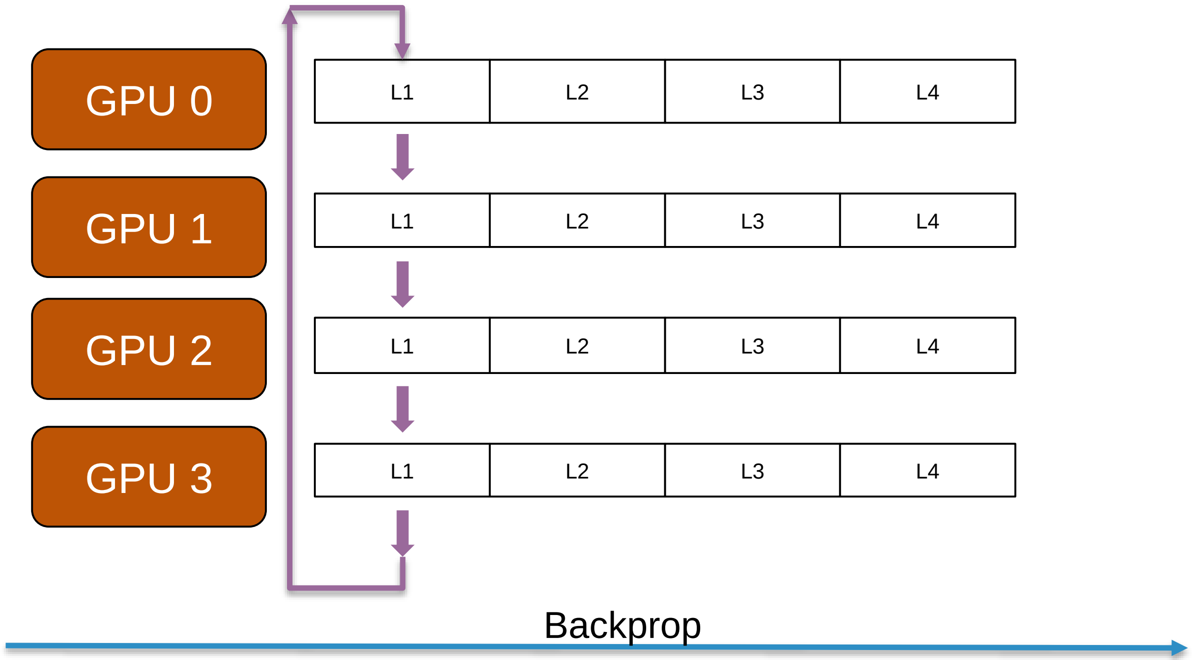
AllReduce: Minimize Inter-Host B/W

Only one slow worker or comms link is needed to bottleneck DNN training time.



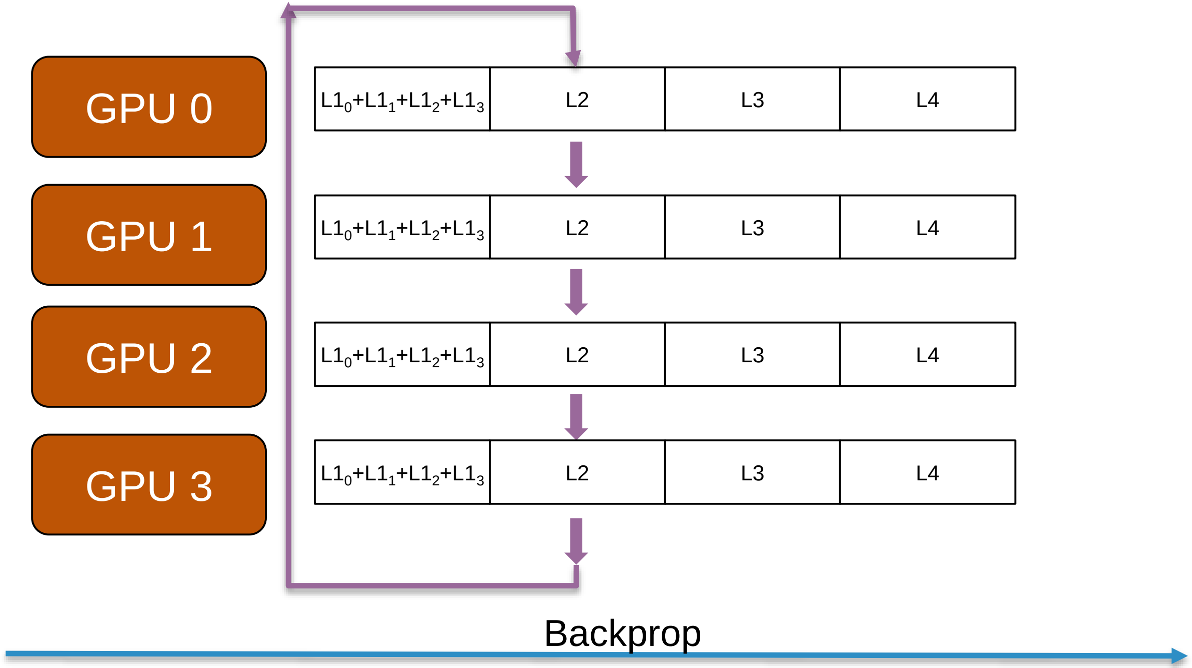
AllReduce Algorithm

- AllReduce sums all Gradients in N Layers (L1..LN) using N GPUs in parallel (simplified steps shown).



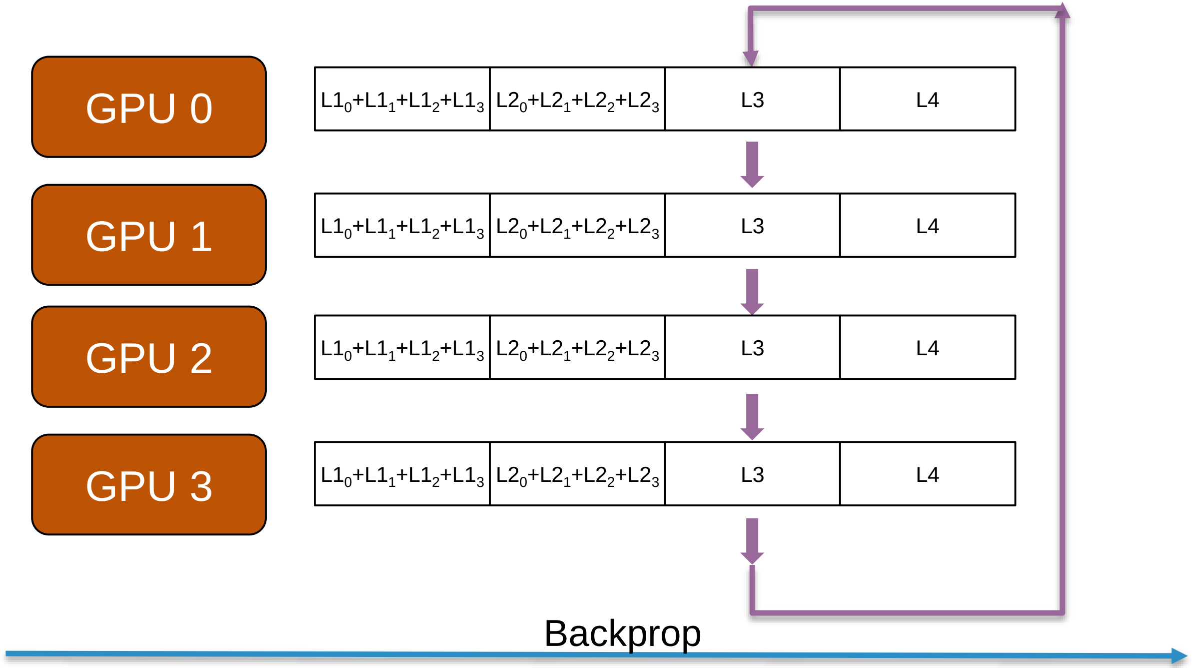
AllReduce Algorithm

- Aggregate Gradients from the first layer (L1) while sending Gradients for L2



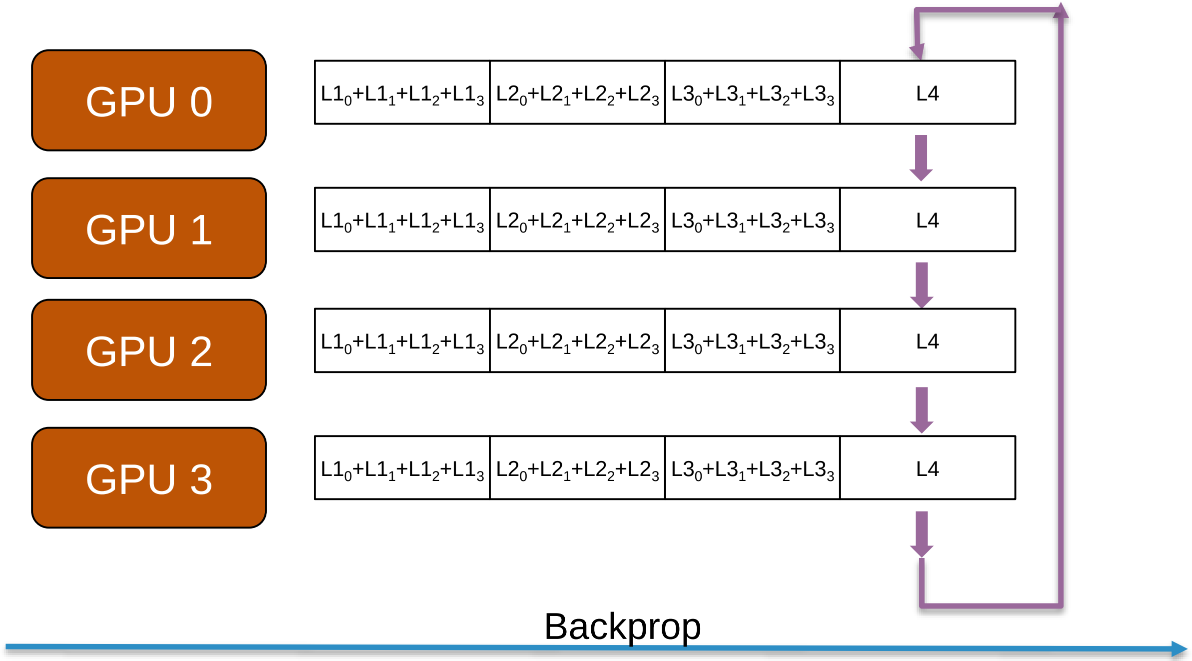
AllReduce Algorithm

- Broadcast Gradients from higher layers while computing Gradients at lower layers.



AllReduce Algorithm

- Nearly there.



AllReduce Algorithm

- Finished an iteration.

GPU 0	$L1_0+L1_1+L1_2+L1_3$	$L2_0+L2_1+L2_2+L2_3$	$L3_0+L3_1+L3_2+L3_3$	$L4_0+L4_1+L4_2+L4_3$
GPU 1	$L1_0+L1_1+L1_2+L1_3$	$L2_0+L2_1+L2_2+L2_3$	$L3_0+L3_1+L3_2+L3_3$	$L4_0+L4_1+L4_2+L4_3$
GPU 2	$L1_0+L1_1+L1_2+L1_3$	$L2_0+L2_1+L2_2+L2_3$	$L3_0+L3_1+L3_2+L3_3$	$L4_0+L4_1+L4_2+L4_3$
GPU 3	$L1_0+L1_1+L1_2+L1_3$	$L2_0+L2_1+L2_2+L2_3$	$L3_0+L3_1+L3_2+L3_3$	$L4_0+L4_1+L4_2+L4_3$

Hops AllReduce/Horovod/TensorFlow

```
import horovod.tensorflow as hvd
def conv_model(feature, target, mode)
 …..
def main(_):
   hvd.init()
   opt = hvd.DistributedOptimizer(opt)
   if hvd.local_rank()==0:
     hooks = [hvd.BroadcastGlobalVariablesHook(0), ..]
     …..
   else:
     hooks = [hvd.BroadcastGlobalVariablesHook(0), ..]
     …..
from hops import allreduce
allreduce.launch(spark, 'hdfs:///Projects/…/all_reduce.ipynb')
```

"Pure" TensorFlow code

Parameter Server vs AllReduce (Uber)*

Setup: 16 servers with 4 P100 GPUs each connected by 40 Gbit/s network (synthetic data).

Setup	Inception V3	ResNet-101	VGG-16
Baseline single-GPU (batch size=64)	148.8	136.0	149.6
On 64 GPUs	x	x	x
Distributed TensorFlow	4,225.3 (28.4x)	2,996.0 (22.0x)	97.0 (0.6x)
Distributed TensorFlow (variables on CPU)	5,297.4 (35.6x)	4,269.2 (31.4x)	100.8 (0.7x)
TCP Horovod (allreduce on CPU)	6,549.6 (44.0x)	3,761.6 (27.7x)	1,462.6 (9.8x)
TCP Horovod (allreduce on GPU with NCCL)	**7,932.1 (53.3x)**	**7,741.6 (56.9x)**	**6,084.2 (40.7x)**

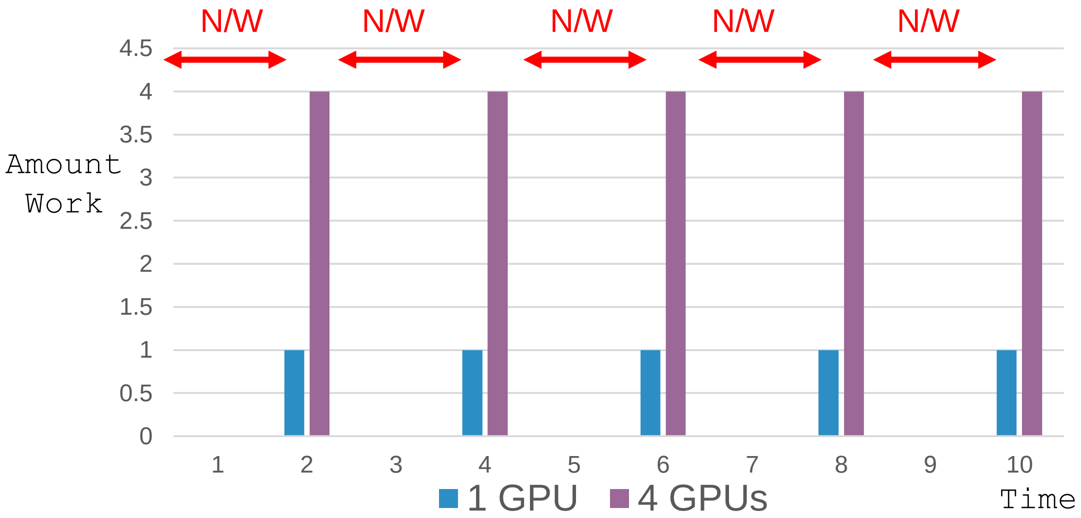
VGG model is larger

*https://github.com/uber/horovod

Dist. Synchnrous SGD: N/W is the Bottleneck

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

Amdahl's Law



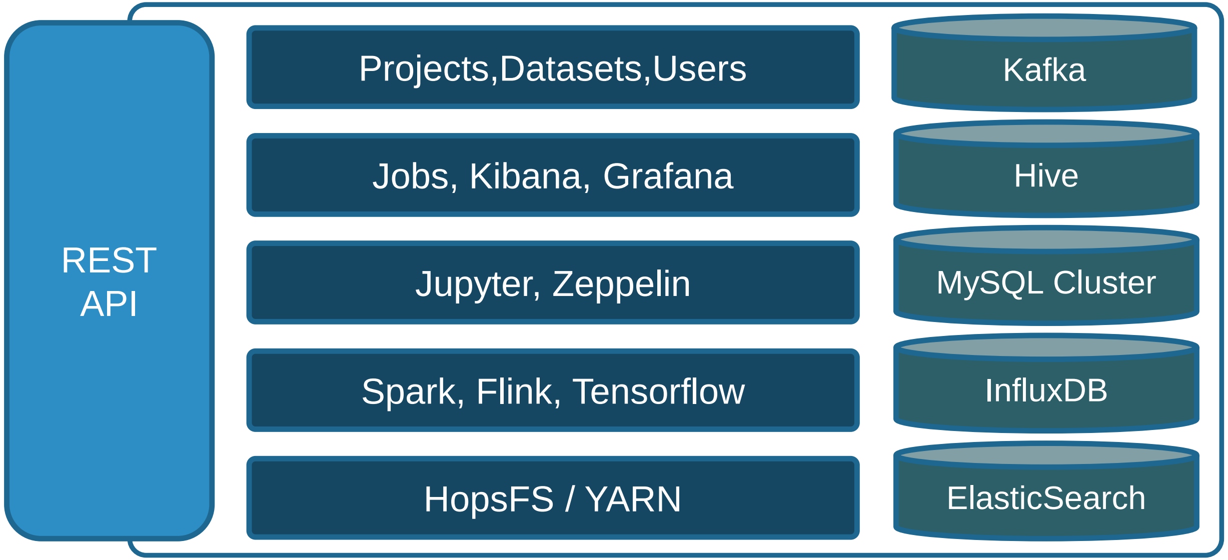
Reduce N/W Comms Time, Increase Computation Time

Hopsworks:Tensorflow/Spark-as-a-Service

Hopsworks: Full AI Hierarchy of Needs

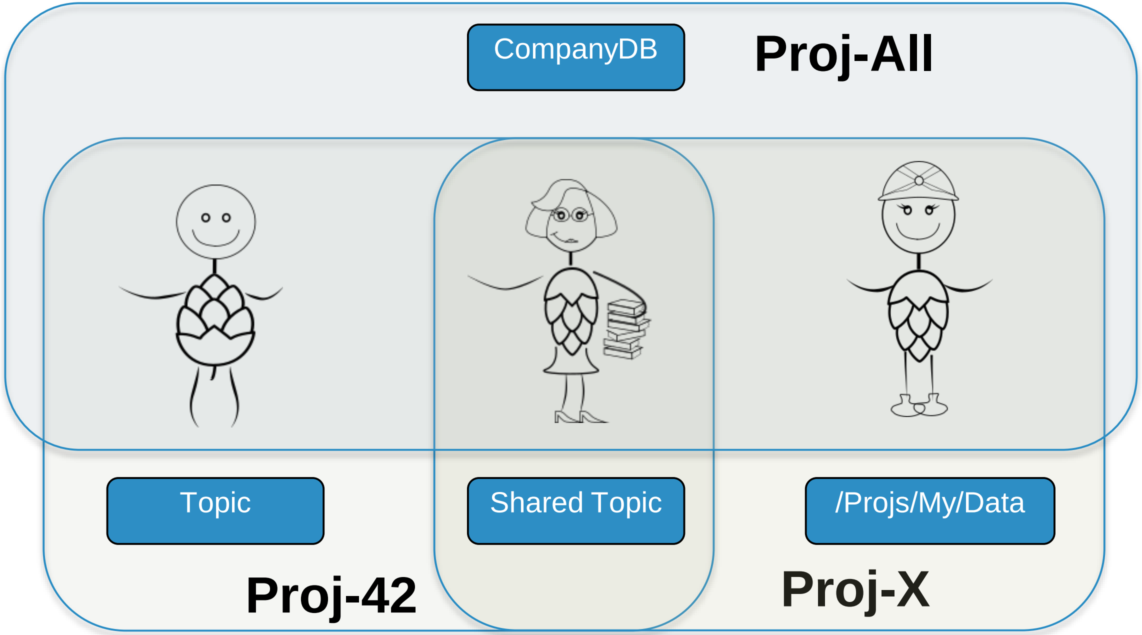
Develop → Train → Test → Deploy

Hopsworks



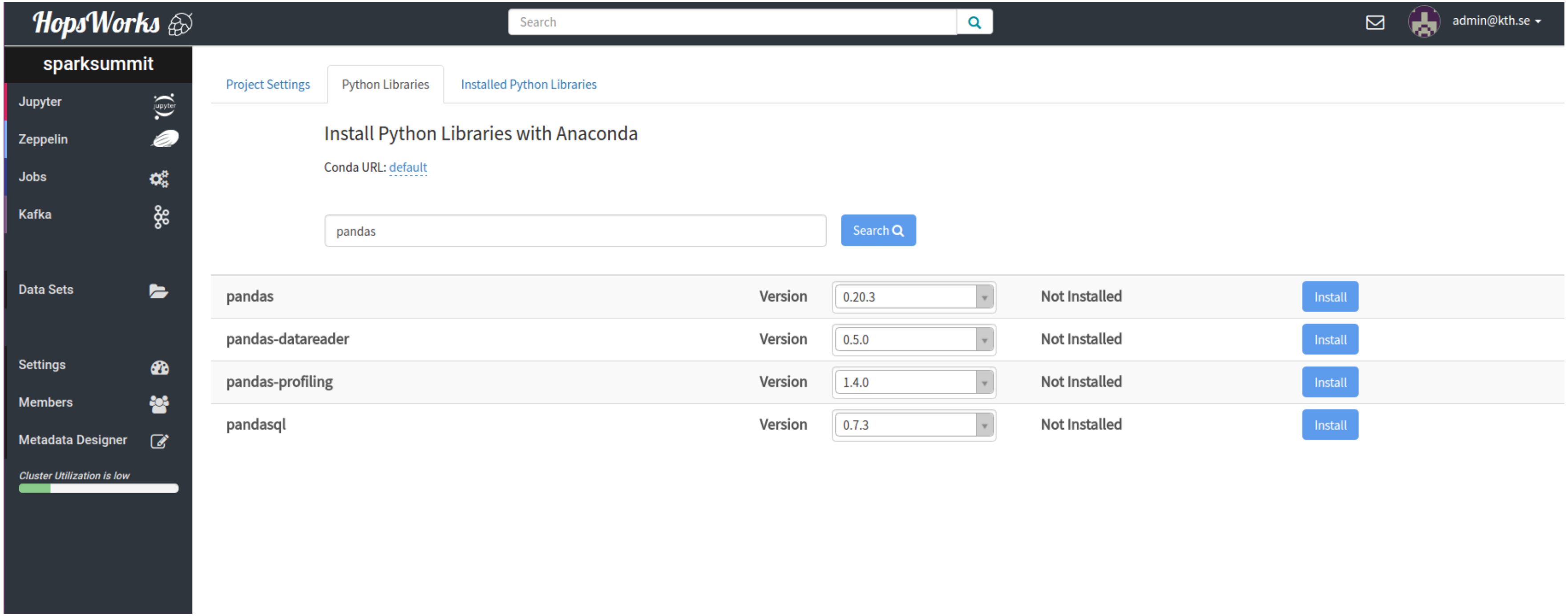
Hopsworks Abstractions

A **Project** is a Grouping of **Users** and **Data**

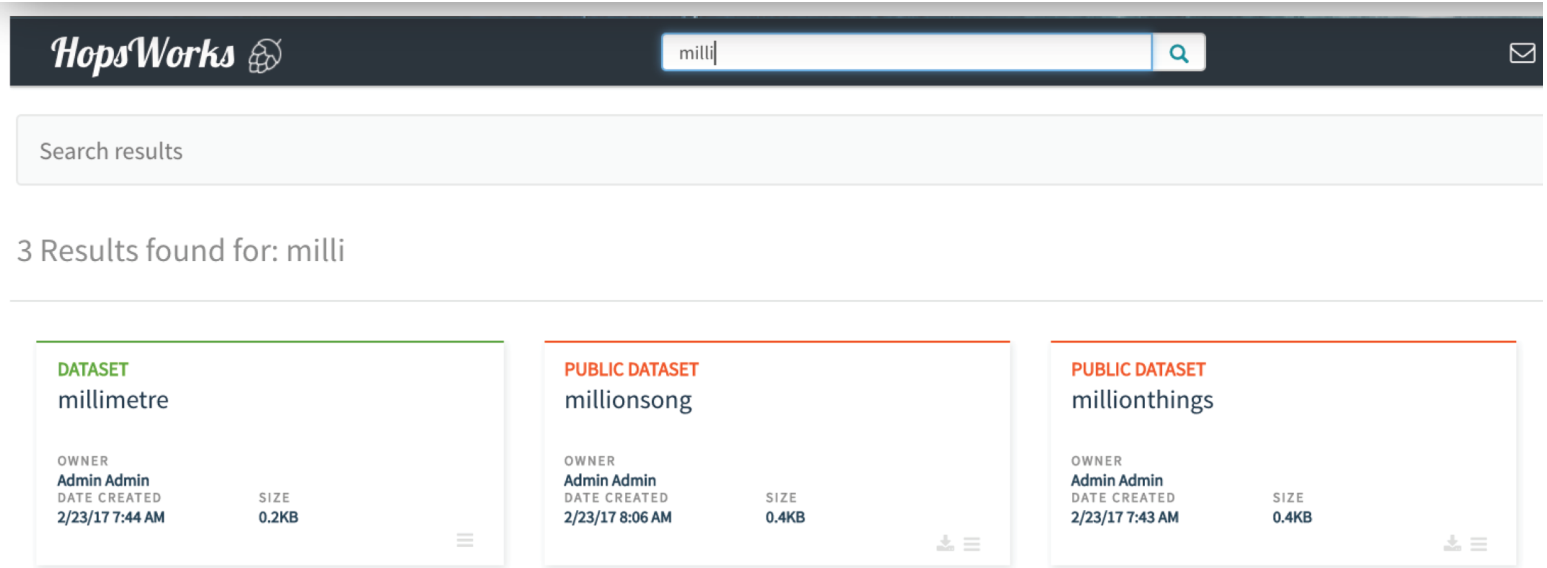


Ismail et al, Hopsworks: Improving User Experience and Development on Hadoop with Scalable, Strongly Consistent Metadata, ICDCS 2017

Per-Project Conda Libs in Hopsworks



Dela*



Peer-to-Peer Search and Download for Huge DataSets (ImageNet, YouTube8M, MsCoCo, Reddit, etc)

*http://ieeexplore.ieee.org/document/7980225/ (ICDCS 2017)

DEMO

Register and Play for today:
http://spark.hops.site

Conclusions

- Many good frameworks for TF and Spark
  - TensorFlowOnSpark, Deep Learning Pipelines
- Hopsworks support for TF and Spark
  - GPUs-as-a-Resource in HopsYARN
  - TfLauncher, TensorFlow-on-Spark, Horovod
  - Jupyter with Conda Support
- More on GPU-Servers at www.logicalclocks.com

Hops Heads

Jim Dowling, Seif Haridi, Gautier Berthou, Salman Niazi, Mahmoud Ismail, Theofilos Kakantousis, Ermias Gebremeskel, Antonios Kouzoupis, Alex Ormenisan, Fabio Buso, Robin Anderssо,n August Bonds, Filotas Siskos, Mahmoud Hamed.

Please Follow Us!

@hopshadoop

Alumni:

Roberto Bampi, ArunaKumari Yedurupaka, Tobias Johansson, Fanti Machmount Al Samisti, Braulio Grana, Adam Alpire, Zahin Azher Rashid, Vasileios Giannokostas, Johan Svedlund Nordström,Rizvi Hasan, Paul Mälzer, Bram Leenders, Juan Roca, Misganu Dessalegn, K “Sri” Srijeyanthan, Jude D’Souza, Alberto Lorente, Andre Moré, Ali Gholami, Davis Jaunzems, Stig Viaene, Hooman Peiro, Evangelos Savvidis, Steffen Grohsschmiedt, Qi Qi, Gayana Chandrasekara, Nikolaos Stanogias, Daniel Bali, Ioannis Kerkinos, Peter Buechler, Pushparaj Motamari, Hamid Afzali, Wasif Malik, Lalith Suresh, Mariano Valles, Ying Lieu.

Please Star Us!

http://github.com/hopshadoop/hopsworks