SCALABLE DISTRIBUTED DEEP LEARNING
Han Hee Song, PhD
Soft On Net
10/7/2016
BATCH PROCESSING FRAMEWORKS FOR DL

• Data parallelism provides efficient big data processing: data collecting, feeding, cleaning
• Easy integration with distributed data storage solutions: HBase, HIVE, HDFS
• Fault-tolerance providing reliable, fail-safe job scheduling

Additionally, for Apache Spark...
• In-memory data processing without storing intermediate results to disk

Nice single-framework for deep learning. However...
CHALLENGES OF USING SPARK FOR DL

**TRADITIONAL BATCH PROCESSING TASKS**

Each task *independently* mappable to mappers

*Synchronously* updatable for global updates among workers

**SHARDS OF DEEP LEARNING TASKS**

Gradient descent used in DL incur *lots of communications* among tasks

DL parameter updating is *asynchronous*
DISTRIBUTED DEEP LEARNING

Data parallel Deep Learning* on Spark

Use multiple model replicas to process different examples concurrently

* Jeff Dean, NIPS, 2013
DISTRIBUTED TRAINING ARCHITECTURE

- **Training data on HDFS**
- **Apache Oozie**
  - Schedule jobs when new training needed
- **Hadoop YARN**
  - Monitor CPU and main memory usage
- **Parameter server**
- **Spark driver**
- **Spark executor**
  - Model trainer
  - CPU
  - Shared virtual GPUs
- **Trained model on HDFS**

Control flow

Data flow
Training GoogLeNet with Batch Normalization (lower error is better)

Parallelizing training with 16 GPUs yields 5x faster convergence
DISTRIBUTED INFERENCE

Real-time processing of 1,000s of videos

Problems

• Significant performance degradation due to IPC communication delay
• Low utilization of GPU

Solution

• In each GPU card, improve GPU occupancy
• In the cluster, promote fine-grained resource allocation among machines

For operational system, promoting efficiency in inference is much more important than in training
INFERENCE PERFORMANCE

Time taken to process $k$ 60 second videos (for a single GPU)

Each GPU can only process 2 videos in real-time.
EFFICIENT USE OF CORES IN EACH GPU

Observation

• Inference on deep network does not use all 2,500 GPU cores
• GPU by default dedicates all cores to each task at a time

Solution: Nvidia Multi-Process Service (MPS)

• Launches multiple kernels concurrently
• Concurrently processes multiple tasks partially occupying GPU cores
EFFICIENT USE OF CORES IN EACH GPU

Micro-benchmark using Nvidia Visual Profiler

Source: Priyanka, Improving GPU utilization with MPS, Nvidia GTC 2015
INFERENCE PERFORMANCE WITH MPS

Time taken to process \(k\) 60 second videos (for a single GPU)

Each GPU can now process as many as 8 videos in real-time.
DISTRIBUTED INFERENCE ARCHITECTURE

Schedule jobs when New video arrives

Apache Oozie

Hadoop YARN

Monitor GPUs, main memory, CPU

YARN labeling

Video on HDFS

Video to image

Spark driver

Spark executor

Object detection

Trajectory tracking

CPU

Partial GPU

Spark executor

Object detection

Trajectory tracking

Partial GPU

CPU

Spark executor

Object detection

Trajectory tracking

CPU

Partial GPU

Spark executor

Object detection

Trajectory tracking

Partial GPU

CPU

Annotated Video on HDFS

Discovered object metadata on HBASE

Assign tasks up to 4 x 8

Assign tasks up to 1 x 8

Control flow

Data flow
SMART EYE
Deep-learning based Video Surveillance on the Cloud

Cost-effective, reliable, and mobile solution to both small and large consumers
THANK YOU

JOIN THE CONVERSATION

#GTCxKorea2016