Accelerating Distributed Deep Learning Inference on multi-GPU with Hadoop-Spark

Hunmin Yang, Se-Yoon Oh, Ki-Jung Ryu

March 2019

GTC Silicon Valley



- Introduction
- GPU-accelerated DL Inference with Hadoop-Spark
 - Fast & Scalable DL Inference
 - DL System Optimization
 - Use case: Intelligent Video Analytics
- Conclusions



Introduction

- AI Training & Inference
- Which one is more important for real-world applications?



Introduction

- Especially for industrial applications, large-scale AI inference is needed.
- Then, large-scale AI inference is easy?
 - Not really.
 - Linear scalability is not guaranteed for multiple GPUs and Servers.
 - Furthermore, most works are rather focused on AI training.



Defense Developmen

Motivation

• Need for developing a large-scale production-level deep learning inference platform for intelligent video analytics



5/82



- Real-world AI systems are complex big data analytics pipelines
 - Only a small fraction of real-world ML systems is composed of the ML code.
 - The required surrounding infrastructure is vast and complex.





- Real-world AI systems are complex big data analytics pipelines
 - Only a small fraction of real-world ML systems is composed of the ML code.
 - The required surrounding infrastructure is vast and complex.





• Hadoop becoming the center of data storage (DB)

Why an Enterprise Data Hub? Single place for all enterprise data... (unedited hi-resolution history of everything) Reduces Application Integration Costs Connect once to Hub (N vs N² connections) Lowest unit cost data processing & storage platform Open source S/W on commodity H/W (reliability in S/W not H/W) Can mix H/W vendors means every expansion is competitively tendered Fast Standardised Provision No custom design task, re-use Active Directory account/password processes Reduces Shadow IT Secure (audited, E2E visibility/auditing, encryption) Eliminate need for one off extracts

#StrataHadoop

Strata Hadoop

Goldman Sachs

Everyone is building Data Lakes

- Universal data acquisition makes all big data analytics and reporting easier
- Hadoop provides a scalable storage with HDFS
- How will we scale consumption and curation of all this data?

Phillip Radley, BT Group Strata + Hadoop World 2016 San Jose

Matthew Glickman, Goldman Sachs Spark Summit East 2015 Copyright: Jason Dai, CVPR 2018



BIND

Hadoop and Spark becoming the center of data processing (analytics)



Research objective

 Develop a fast, scalable, and optimal GPU-accelerated distributed deep learning inference platform with Hadoop-Spark



Al application deployment at scale



Outline

Introduction

• GPU-accelerated DL Inference with Hadoop-Spark

- Fast & Scalable DL Inference
- DL System Optimization
- Use case: Intelligent Video Analytics
- Conclusions



GPU-accelerated DL Inference with Hadoop-Spark

Streaming data analytics Static data analytics essing processing Spark oordination oordination Spark storage 🝭 NVIDIA. CUDA. CUDA. HBASE Streaming Streaming HBASE Spa Spar pro Data ZooKeeper ZooKeeper Data Data Offline kafka of kafka **Data streaming** Data streaming Online Data Source H.264 MPEG-4/AVC H.264 Thousands of RTSP cameras

Challenge: Fast & Scalable DL Inference

- Why do we need 'fast' DL inference?
 - Low latency for high-quality service
 - Especially critical for defense industry
- Why do we need 'scalable' DL inference?
 - Good scalability means fast computing
 - 'Scale up' one server with higher spec.
 - 'Scale out' many servers with clustering



DISTRIBUTED

SCALE UP

SCALE OUT

NODE

NODE

NODE

NODE

Fast service



13/82

Challenge: Fast & Scalable DL Inference

SOTA distributed computing: Apache Hadoop



Single-use platform

Batch processing

General-purpose platform

- Batch processing
- Streaming / Online / Interactive processing



Challenge: Fast & Scalable DL Inference

- SOTA distributed computing: Apache Spark
 - Fast & General-purpose cluster computing engine
 - With Hadoop YARN, Apache Mesos, Standalone, etc.
 - High-level APIs in Java, Scala, Python, R, and SQL



Fast

General-purpose

Runs-everywhere

16/82

- Our solution
 - Hadoop-Spark: efficient distributed computing
 - NVIDIA GPUs: accelerating DL inference





Our solution

GPU-accelerated DL Inference with Hadoop-Spark



=> Experiment #1

=> Experiment #2

Outline

Introduction

• GPU-accelerated DL Inference with Hadoop-Spark

- Fast & Scalable DL Inference
 - Experiment #1: Streaming data analytics
 - Experiment #2: Static data analytics
- DL System Optimization
- Use case: Intelligent Video Analytics
- Conclusions



- Experiment #1: Streaming data analytics
 - Purpose
 - Measure the scalability for multi-GPU & multi-node
 - Method
 - Increasing the number of clustered servers (including 1GPU each)





Test system	CPU	GPU	Memory	Storage	Network
31-node cluster w/ 30 GPUs	Intel Xeon E5-2697 v4 18-cores @ 2.3 GHz	NVIDIA Tesla M60 (1GPU/node)	256GB DDR4	24TB HDD	10Gbps Ethernet

CentOS	CUDA	cuDNN	Tensorflow	HDP	HBase	Kafka	Spark	
7.4	8.0	6.0	1.4.0-gpu	2.6.3	1.1.2	0.10.1	2.2	

- Experiment #1: Streaming data analytics
 - Distributed deep learning inference



20/82

- Experiment #1: Streaming data analytics
 - Distributed deep learning inference



GPU Fractional scaling



30 X 4 = 120

- 30 : # of GPUs (= # of Nodes)
- 4 : # of YOLOv2 models in each GPU (= # of Spark executors in each GPU)
- 120 : Total # of YOLOv2 models

22/82



- Experiment #1: Streaming data analytics
 - Results





- Experiment #1: Streaming data analytics
 - Results (normalized)



- Experiment #1: Streaming data analytics (more details)
 - Apache Kafka integration with Spark



- Experiment #1: Streaming data analytics (more details)
 - Apache Spark 'Cluster' mode



Term	Meaning
Application	User program built on Spark (Driver Program + Executors)
Application jar	Jar containing user's Spark Application
Driver program	Process running the main() of the application and creating the SparkContext
Cluster manager	External service for acquiring resources on the cluster (e.g. standalone, Mesos, YARN, etc.)
Deploy mode	 Distinguishes where the driver process runs. 'Cluster' mode : driver inside of cluster 'Client' mode : driver outside of cluster
Worker node	Any node that can run application.
Executor	<u>A process</u> that runs tasks and keeps data in memory or disk. Each application has its own executors.
Task	A unit of work that will be sent to one executor
Job	A parallel computation consisting of multiple tasks
Stage	Each job gets divided into smaller sets of tasks that depend on each other



Storage Environment Executors Streaming

Spark

User: root Total Uptime: 20 min

Spark Jobs (?)

Scheduling Mode: FAIR

Jobs

Stages

• Experiment #1: Streaming data analytics (examples)

A parallel computation consisting of **multiple tasks**

Active Jobs: 1 Completed Jobs: 1180, only showing 1000 Event Timeline Active Jobs (1) Job Id 🔻 Description Submitted Duration Stages: Succeeded/Total Tasks (for all stages): Succeeded/Total 1180 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 (kill) 2019/03/04 17:28:54 0.8 s 0/1 0/112 Completed Jobs (1180, only showing 1000) 10 Pages. Jump to 1 Show 100 items in a page. Go Page: 1 2 3 4 5 6 7 8 9 10 > Job Id 🔻 Description Submitted Duration Stages: Succeeded/Total Tasks (for all stages): Succeeded/Total 1179 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:53 0.6 s 1/1 112/112 1178 1/1 112/112 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:52 0.7 s 1177 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-sro.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:51 0.7 s 1/1 1176 112/112 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:50 0.7 s 1/1 1175 1/1 112/112 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:49 0.7 s 1174 112/112 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:48 0.7 s 1/1 1173 1/1 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:47 0.7 s 112/112 1172 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-sro.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:46 0.7 s 1/1 112/112 1171 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:45 0.7 s 1/1 1170 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:44 0.7 s 1/1 1169 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:43 0.7 s 1/1 112/112 1168 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:42 0.7 s 1/1 112/112 1167 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:41 0.7 s 1/1 112/112 1166 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:40 0.7 s 1/1 112/112 1165 0.7 s 1/1 112/112 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:39 1164 call at /usr/hdp/2.6.2.0-205/spark2/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 2019/03/04 17:28:38 1/1 112/112 0.7 s

1 Job = 120 Tasks

Agency for Defense Development

sparkdetect application UI

• Experiment #1: Streaming data analytics (examples)

Active Stages Completed St 1 Fair Scheo	: 1 ages: 1201, only duler Pools	showing 1000	1 Stage	= 120	Tasks	= 1	Job (No	t d	ivic	ded)			
Pool Name	Vool Name Minimum Share		Pool Weight	Active Stages		Running	lasks	Sch	nedulingMo	ode			
default		0	1	1		105			FIFO				
Active Stage	es (1)												
Stage Id 🔻	Pool Name	Description			Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write		
1201	default	call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	(kill) +details	2019/03/04 17:29:15	0.3 s							
Page: 1 2	3 4 5	6 7 8 9 10 >					10 P	ages. Jump	to 1	. Show 100 i	items in a page. G		
						-			-				
Stage Id ▼ 1200	Pool Name default	Description call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details	Submitted 2019/03/04 17:29:14	Duration 0.7 s	Tasks: Succeeded/Total 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id ▼ 1200 1199	Pool Name default default	Description call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13	Duration 0.7 s 0.6 s	Tasks: Succeeded/Total 112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id ▼ 1200 1199 1198	Pool Name default default default default	Description call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13 2019/03/04 17:29:12	Duration 0.7 s 0.6 s 0.5 s	Tasks: Succeeded/Total 112/112 112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id ▼ 1200 1199 1198 1197	Pool Name default default default default default	Description call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details +details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13 2019/03/04 17:29:12 2019/03/04 17:29:11	Duration 0.7 s 0.6 s 0.5 s 0.6 s	Tasks: Succeeded/Total 112/112 112/112 112/112 112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id ▼ 1200 1199 1198 1197 1196	Pool Name default default default default default default default	Description call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details +details +details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13 2019/03/04 17:29:12 2019/03/04 17:29:11 2019/03/04 17:29:10	Duration 0.7 s 0.6 s 0.5 s 0.6 s 0.7 s	Tasks: Succeeded/Total 112/112 112/112 112/112 112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id ▼ 1200 1199 1198 1197 1196 1195	Pool Name default	Description call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details +details +details +details +details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13 2019/03/04 17:29:12 2019/03/04 17:29:11 2019/03/04 17:29:10 2019/03/04 17:29:09	Duration 0.7 s 0.6 s 0.5 s 0.6 s 0.7 s 0.6 s	Tasks: Succeeded/Total 112/112 112/112 112/112 112/112 112/112 112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id ▼ 1200 1199 1198 1197 1196 1195 1194	Pool Name default	Description call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details +details +details +details +details +details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13 2019/03/04 17:29:12 2019/03/04 17:29:11 2019/03/04 17:29:10 2019/03/04 17:29:09 2019/03/04 17:29:08	Duration 0.7 s 0.6 s 0.5 s 0.6 s 0.7 s 0.6 s 0.7 s 0.6 s	Tasks: Succeeded/Total 112/112 112/112 112/112 112/112 112/112 112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		
Stage Id • 1200 • 1199 • 1198 • 1197 • 1196 • 1195 • 1194 •	Pool Name default	Description call at /usr/hdp/2.6.2.0-205/spark2/python call at /usr/hdp/2.6.2.0-205/spark2/python	/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230 /lib/py4j-0.10.4-src.zip/py4j/java_gateway.py:2230	+details +details +details +details +details +details +details +details +details +details	Submitted 2019/03/04 17:29:14 2019/03/04 17:29:13 2019/03/04 17:29:12 2019/03/04 17:29:11 2019/03/04 17:29:10 2019/03/04 17:29:09 2019/03/04 17:29:08 2019/03/04 17:29:07	Duration 0.7 s 0.6 s 0.5 s 0.6 s 0.7 s 0.6 s 0.7 s 0.6 s 0.7 s 0.6 s 0.7 s	Tasks: Succeeded/Total 112/112 112/112 112/112 112/112 112/112 112/112 112/112 112/112 112/112 112/112 1112/112 112/112	Input	Output	Shuffle Read	Shuffle Write		

• Experiment #1: Streaming data analytics (examples)

Streaming

Jobs Stages Storage Environment Executors

sparkdetect application UI

Search:

Executors

SPORK 2112.6.2.0-205

<u>A process</u> that runs tasks and keeps data in memory or disk.

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(113)	5763	44.6 MB / 107.5 GB	0.0 B	112	112	0	108640	108752	13.5 h (37 s)	0.0 B	0.0 B	0.0 B
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Total(113)	5763	44.6 MB / 107.5 GB	0.0 B	112	112	0	108640	108752	13.5 h (37 s)	0.0 B	0.0 B	0.0 B

Executors

Show 20 • entries

120 Executors = 120 Processes = 120 Tasks

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	192.168.0.103:38020	Active	51	394.6 KB / 384.1 MB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
0	192.168.0.129:39638	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.0 min (0.3 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
1	192.168.0.129:37996	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.1 min (0.5 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
2	192.168.0.129:42337	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.2 min (0.5 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
3	192.168.0.129:37734	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.2 min (0.4 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
4	192.168.0.101:46083	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.0 min (0.4 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
5	192.168.0.101:46116	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.5 min (0.6 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
6	192.168.0.101:39364	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.6 min (0.2 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
7	192.168.0.101:40732	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.3 min (0.5 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
8	192.168.0.123:39128	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.1 min (0.4 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
9	192.168.0.123:40578	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.4 min (0.4 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
10	192.168.0.123:45122	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.4 min (0.5 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
11	192.168.0.123:41428	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.0 min (0.3 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
12	192.168.0.102:40079	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.1 min (0.5 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
13	192.168.0.102:37370	Active	51	394.6 KB / 956.6 MB	0.0 B	1	1	0	970	971	7.4 min (0.4 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump



• Experiment #1: Streaming data analytics (examples)

Environment

Spork 2.1.1.2.6.2.0-205

Jobs Stages Storage

Executors Streaming

Streaming Statistics

Running batches of 1 second for 3 minutes 19 seconds since 2019/03/04 17:09:12 (197 completed batches, 76198 records)



Input rate = 1000 records/s (1000 images/s)

Scheduling delay \cong 0 s

Processing time < 1 s

Total delay < 1 s



• Experiment #1: Streaming data analytics (examples)

Active Batches (1)			D	0.4.40	1.07.41	C (
Batch Time	Input Size	Scheduling Delay (*/	Processing Time (*)	Output Ops: Succee	eded/ lotal	Status	ACTIVE
2019/03/04 17:12:31	1022 records	0 ms	-		U/ I	processing	
Completed Batches (last 197	out of 197)						
Batch Time	Input Size	Scheduling Delay (?)	Processing Time (?)	Total Delay (?)	Output Ops: Succeeded/To	tal	
2019/03/04 17:12:30	1009 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:29	1026 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:28	1021 records	1 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:27	1015 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:26	1019 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:25	1019 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:24	1012 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:23	1026 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:22	1018 records	1 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:21	1016 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:20	1017 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:19	1016 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:18	1000 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:17	1036 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:16	1020 records	1 ms	0.7 s	0.7 s		1/1	Completed
2019/03/04 17:12:15	1010 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:14	1011 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:13	1027 records	1 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:12	1024 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:11	1018 records	1 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:10	1009 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:09	1015 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:08	1015 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:07	1018 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:06	1017 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:05	1019 records	0 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:04	1015 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:03	1018 records	1 ms	0.7 s	0.7 s		1/1	
2019/03/04 17:12:02	1019 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:01	1012 records	1 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:12:00	1016 records	0 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:11:59	1008 records	1 ms	0.8 s	0.8 s		1/1	
2019/03/04 17:11:58	1021 records	0 ms	0.8 s	0.8 s		1/1	

Outline

Introduction

• GPU-accelerated DL Inference with Hadoop-Spark

- Fast & Scalable DL Inference
 - Experiment #1: Streaming data analytics
 - Experiment #2: Static data analytics
- DL System Optimization
- Use case: Intelligent Video Analytics
- Conclusions



- Experiment #2: Static data analytics
 - Purpose
 - Measure the scalability for Spark multi-processors
 - Method
 - Increasing the number of Spark executors











Test system			CPU				M	lemory	S	torage	Network
<mark>4-node</mark> clus w∕ 6 GPU	ster S	Inte 6-c	Intel Core i7-6850K 6-cores @ 3.6 GHz		NVIDI Titan > (2GPU/n	A (p ode)		32GB DDR4	1TB SSD		1 Gbps Ethernet
CentOS	CentOS CUDA		cuDNN Ten		sorflow HD		P	HBase)	Kafka	Spark
7.4	8	.0	6.0	1.4	1.4.0-gpu		5.3	.3 1.1.2		0.10.1	2.2



Defense Developmen

- Experiment #2: Static data analytics
 - Distributed deep learning inference



- Experiment #2: Static data analytics
 - Distributed deep learning inference





Node #1

Node #2

Node #3



- Experiment #2: Static data analytics
 - Results




Solution: NVIDIA GPU & Hadoop-Spark

- Experiment #2: Static data analytics
 - Results (normalized)



Agency for Defense Development

Part 1 - Summary (Fast & Scalable DL inference)

- We presented an efficient way to attain the scalable deep learning inference on multi-GPU & multi-node using Hadoop-Spark.
- We demonstrated the usage of Spark multi-processors (executors) on cluster environment to exploit the GPU resources efficiently.
- We applied the suggested approaches on both streaming and static data and achieved the high-quality scalability performance.

Hadoop-Spark Cluster

1 Nodes with 30 GPUs

Number of GPU



Streaming data analytics

84% efficiency



Static data analytics



Part 1 - Summary (Fast & Scalable DL inference)



Outline

Introduction

• GPU-accelerated DL Inference with Hadoop-Spark

- Fast & Scalable DL Inference
- DL System Optimization
- Use case: Intelligent Video Analytics
- Conclusions







Agency for Defense Development

Good Design

How to collect useful information ?



Dennis Lin @ PSU



- How can we optimize the DL system?
 - Reduce design costs by speeding up the design process and reducing labor complexity.







• DL Parameter

Dependent

Variable

- DL Model Parameter
 - Labeled Data + DL Model = weight(β_n), bias(β_0)

 $\mathbf{Y} = \mathbf{\beta}_0 + \mathbf{\beta}_1 \mathbf{X}_1 + \mathbf{\beta}_2 \mathbf{X}_2 + \dots + \mathbf{\beta}_n \mathbf{X}_n + \mathbf{\varepsilon}$

Explanatory

Variables

 $\beta_n X_n + (\varepsilon$

Random Error

Term/Residuals

- Hyper-Parameter (Algorithm Design)
 - Different Learning Structure(DNN)

Coefficients



Agency for Defense Development

• Hyper-Parameters • Learning rates Cost function (1) Grid Search Regularization parameter • Batch size (2) Random Search Training epochs • No. of layers Hyper-Parameter Optimization (3) Bayesian Optimization Convolutional filters Convolutional kernel size (4) Gradient-based Optimization Weight initialization Dropout fractions (5) Statistical DOE Search . . .



- Experiments
 - As a test or series of tests
 - Purposeful changes are made to the input variables of system
 - Identify the reasons for the changes in the output response





- Strategy of Experimentation
 - 1. "Best-guess" experiments
 - Used a lot / Non-optimal solution with poor initial guess
 - 2. One-Factor-At-a Time (OFAT) experiments
 - Fail to consider any possible interaction
 - Always less efficient than statistics-based methods

3. Statistically designed experiments

- Design of Experiments (DOE)
- Based on Fisher's factorial concept







Experimental Design





48/82

Challenge: DL System Optimization

- Design of Experiments (DOE)
 - Maximum information with minimum resources
 - Develop mathematical models (factors-responses)
 - Optimize the responses
 - Mathematical response models helps in output maximization





Response Surface Methodologies (RSM/DOE)









Case study 1: Space Filling Design

- Cifar-100 Dataset Subset
 - 20 categories
 - 50,000 color images
- Factors as design variables
 - No. Epoch(X1)
 - Batch Size(X2)
 - Learning Rate(X3)
 - LR Step(X4)
- Responses
 - Accuracy(Y1)
 - Learning Time(Y2)



RUN No	Epoch(x1)	Bach Size(x2)	Learning Rate(x3)	LR Step(x4)	Accuracy(%)	Time(min)
1	22	122	0.014	33	51.6	12
2	24	147	0.007	20	43.0	13
3	21	46	0.007	35	55.6	16
4	13	40	0.011	27	40.1	11
5	29	91	0.009	37	59.4	17
6	14	109	0.005	26	38.6	8
7	17	135	0.012	23	40.6	9
8	25	59	0.013	29	54.4	16
9	12	97	0.010	32	41.9	7
10	28	154	0.011	34	55.6	15
11	15	65	0.013	38	52.6	10
12	26	103	0.006	28	52.8	14
13	11	84	0.009	21	36.1	7
14	27	78	0.014	40	61.7	16
15	23	53	0.008	24	51.3	16
16	16	72	0.015	22	42.2	10
17	19	160	0.008	31	45.3	10
18	30	116	0.012	25	55.5	16
19	18	128	0.010	39	51.7	9
20	10	141	0.006	36	37.3	5











51/82

Challenge: DL System Optimization

- Case study 2: D-Optimal Design
 - Cifar-100 Dataset Subset
 - 20 categories
 - 50,000 color images
 - Factors as design variables
 - No. Epoch(X1)
 - Dataset Size(X2)
 - Responses
 - Accuracy(Y1)
 - Learning Time(Y2)
 - Optimum Results
 - Active learning to help in dataset selection $_{\text{\tiny Time}}$
 - Accuracy : 50 % @ min. Learning Time





- System design parameters
 - X1 : Number of Spark executors per GPU
 - X2 : Video decoding rate (Hz)
 - X3 : Batch size for DL model inference
- Responses (Performance metrics)
 - Y1 : Throughput in processed images per second (images/sec)
 - Y2 : Processing time (sec)





- FCCD experiments in our solution
 - Special case of ordinary CCD
 - Second-order regression model
 - very flexible / wide variety function form
 - easy estimation / work well in real problems



2 variable FCCD

$$y = \beta_0 + \sum_{i=1}^k \beta_i x_i + \sum_{i=1}^k \beta_{ii} x_i^2 + \sum_{i=1}^{k-1} \sum_{j=2}^k \beta_{ij} x_i x_j + \varepsilon$$

where $E(\varepsilon) = 0$

$$Var(\varepsilon) = \sigma^2$$

 $\{\varepsilon\}$ are uncorrelate random variables



3 variable factorial design 3 v





- DOE/RSM approach for system optimization
 - Factor levels for the FCCD experiments

Design	No.	Remark
Central Composite Design (FCCD)		
Total Runs	20	pts
Factorial	8	pts
Center	6	pts
Axial	6	pts
Factors	Factorial values	
X1 : Executer/GPU	1~3	
X2 : Batch Size	1~15	
X3 : Decoding Rate	1~9	Hz
Response		
Y1 : Throughput	$1,800 \le Y1 \le 2,500$	Images/sec
Y2 : Processing Time	$0.5 \le Y2 \le 1.0$	sec



FCCD matrix and measured results

RUN No	Executor(x1)	Bach Size(x2)	Decoding Rate(x3)	Throughput(y1) [images/sec] Y1	Processing Time(y2) [sec] Y2
1	1	8	5	1315	4.00
2	3	8	5	2640	2.00
3	2	8	5	2565	2.07
4	2	8	1	1396	0.75
5	2	8	5	2490	2.10
6	2	8	9	2430	3.90
7	3	1	9	3168	3.00
8	2	8	5	2600	2.03
9	2	15	5	2545	2.07
10	3	15	9	3114	3.03
11	3	15	1	1325	0.80
12	3	1	1	1832	0.58
13	1	15	9	1530	6.20
14	2	8	5	2605	2.03
15	1	1	9	1422	6.67
16	2	8	5	2565	2.07
17	1	15	1	1052	1.00
18	2	1	5	2645	2.00
19	1	1	1	1062	1.00
20	2	8	5	2645	2.00



Defense Development





• Optimization results for other design requirements





Part 2 - Summary (DL system optimization)

- We presented an efficient way to attain the optimal design parameters for distributed deep learning system with a statistical DOE method
- We applied the suggested approaches through various experiments according to the DOE/RSM based configurations.
- We demonstrated the effect of design parameters such as the number of Spark executors per GPU, batch size and video decoding rate for deep learning inference system in regards of performance metrics such as throughput and processing time.



Outline

Introduction

• GPU-accelerated DL Inference with Hadoop-Spark

- Fast & Scalable DL Inference
- DL System Optimization
- Use case: Intelligent Video Analytics
- Conclusions





• 'D-NET': GPU-accelerated big data analytics platform





NINFT

• Big data server room





Integrated control room



- Main functions
 - Video stream simulator
 - Real-time video streaming
 - DL-based object detection
 - Heterogeneous model deployment
 - GIS-based object tracking
 - Object tracking by face recognition
 - DB-based object search
 - Cluster resource monitoring
 - Integrated multi-video control



Real-time Intelligent Video Analytics for 1000Ch CCTV streams NVIDIA Tesla M60 GPU-accelerated DL Inference on 31-node & 30-GPU Hadoop-Spark Cluster





Video stream simulator

						IvsServerMng				$\sim \sim \times$			
	Simulator Preset												
	🖃 · Simulator		CPU Usag	ge : 3.47% / 75.98	% (Process / Server)								
	Server (192.	168.0.139:7201) 168.0.139:7202)	No.	CameralP	CameraType	CameraSource	CameraTitle	RoutelP	RoutePort View	wLocation Status]		
		168.0.139:7203)	1	192.168.4.1	CCTV	1.avi	model_1	192.168.0.165	7301	0,0 30			
	Server (192.	168.0.139:7204)	2	192.168.4.10	CCTV	10.avi	model_1	192.168.0.165	7302	1,0 30			
	Server (192.)	168.0.139:7205)	3 4	192.108.4.100	CCTV	100.avi	model_1	192.108.0.105	7303	2,0 30 3.0 30			
	Server (192.	168.0.139:7206)	15	192.168.4.12	CCTV	12.avi	model_1	192.168.0.165	7305	4,0 31			
		168.0.139:7207)	6	192.168.4.13	CCTV	13.avi	mode_1	192.168.0.165	7306	5,0 30			
	Server (192.	168.0.139:7208	7	192.168.4.14	CCTV	14.avi	model_1	192.168.0.165	7307	6,0 31			
	Server (192.)	168.0.139.7210)	8	192.168.4.15		15.avi	model_1	192.168.0.165	7308	7,0 <u>30</u> 8.0 20			
			10	192.168.4.17	CCTV	17.avi	model_1	192.168.0.165	7310	9,0 30		\mathbf{N}	
			11	192.168.4.18	CCTV	18.avi	mode_1	192.168.0.165	7311	0,1 31			
			12	192.168.4.19	CCTV	19.avi	model_1	192.168.0.165	7312	1,1 31			
			13	192.168.4.2	CCTV	2.avi	model_1	192.168.0.165	7313	2,1 31			
			15	192.168.4.20	CCTV	20.avi 21.avi	model 1	192.168.0.165	7315	4.1 30			
			16	192.168.4.22	CCTV	22.avi	model_1	192.168.0.165	7316	5,1 31			
			17	192.168.4.23	CCTV	23.avi	model_1	192.168.0.165	7317	6,1 30			
			18	192.168.4.24	CCTV	24.avi	model_1	192.168.0.165	7318	7,1 30			
			20	192.168.4.25		25.aVI 26 avi	model_1	192.168.0.165	7319	8,1 30 0.1 20			
			21	192.168.4.27	CCTV	27.avi	model 1	192.168.0.165	7321	0,2 31			
			22	192.168.4.28	CCTV	28.avi	mode_1	192.168.0.165	7322	1,2 30			
			23	192.168.4.29	CCTV	29.avi	model 1	192.168.0.165	7323	2.2 31			
CameralP	CameraType		Came	eraSource			CameraTitle		RouteIP	Route	Port	ViewLocation	Status
192.168.4.1	CCTV			1.avi			model_1	19	2.168.0.16	55 730	1	0,0	30
192.168.4.10	CCTV		1	l0.avi			model_1	19	2.168.0.16	55 730	2	1,0	30
192.168.4.100	CCTV		1	00.avi			model_1	192	2.168.0.16	55 730	3	2,0	30
192.168.4.11	CCTV		1	1.avi			model_1	19	2.168.0.16	55 730	4	3,0	30
			36	192.168.4.40	CCTV	40.avi	model 1	192.168.0.165	7336	5.3 29			
			37	192.168.4.41	CCTV	41.avi	mode_1	192.168.0.165	7337	6,3 <u>31</u>			
			38	192.168.4.42	CCTV	42.avi	model_1	192.168.0.165	7338	7,3 31			
			39 40	192.168.4.43	CCTV	43.3VI 44 avi	model_1	192.168.0.165	7339 7340	8,3 <u>31</u> 9.3 <u>30</u>			
			41	192.168.4.45	CCTV	45.avi	model_1	192.168.0.165	7341	0,4 30			
			42	192.168.4.46	CCTV	46.avi	model_1	192.168.0.165	7342	1,4 29			
			43	192.168.4.47	CCTV	47.avi	model_1	192.168.0.165	7343	2,4 30			
			44	192.108.4.49	CCTV	40.avi	model 1	192.168.0.165	7345	4.4 30			
			46	192.168.4.5	CCTV	5.avi	model_1	192.168.0.165	7346	5,4 30			
			47	192.168.4.50	CCTV	50.avi	model_1	192.168.0.165	7347	6,4 30			
			48	192.168.4.51	CCTV	51.avi	model_1	192.168.0.165	7348	7,4 31			
			50	192.168.4.52	CCTV	53.avi	model 1	192.168.0.165	7350	9.4 30			
			51	192.168.4.54	CCTV	54.avi	model_1	192.168.0.165	7351	0,5 31			
			52	192.168.4.55	CCTV	55.avi	model_1	192.168.0.165	7352	1,5 31 -		Age	ency for ense Developmer

Real-time video streaming







• Deep learning based object detection (CCTV)



• Deep learning based object detection (aerial view)



• Heterogeneous model deployment



GIS-based object tracking



Object tracking by face recognition





71/82

Agency for Defense Development

• Database based object search for cropped image



Defense Development
• Cluster resource monitoring





• Cluster resource monitoring

(<) (i) http://192.168.	0.137:8080/dNet/#/syste	em	+ م	C D-NET	×							() 分 ()
	LIVE	SEARCH	TRACKING	ADV.U	SER A	ADMIN.					Adn	ninistator(관리자)님 Logout
데이터 관리	데이터 처리 시간 🔻						GPU / Spa	ark 사용 현황 ▼	_			
사용자 관리	총 걸린 시간 평균			중지			GPU 평	Usage Mem Us 군 평균	sage	중지		
시스템 현황	0.22 [±]					측정 주기 : <u>5초</u> ✔ 남은 시간 : 2 초	71.	67 % 6.3	5 %			측정 주기 : [5초 V] 남은 시간 : 4 초
클러스터 관리	No	CCTV	Kafka	Spark	HBase	총 걸린 시간	No	서버	GPU	GPU Usage	Memory	Memory Usage
	1	192.168.0.1	0.04 초	0.13 초	0.04 초	0.21 초	1	192.168.0.101	Tesla M60	86 %	251 GB	8.2 %
	2	192.168.0.2	0.01 초	0.11 초	0.03 초	0.16 초	2	192.168.0.102	Tesla M60	86 %	251 GB	6.7 %
	3	192.168.0.3	0.01 초	0.21 초	0.07 초	0.29 초	3	192.168.0.103	Tesla M60	86 %	251 GB	8.1 %
	4	192.168.0.4	0.01 초	0.12 초	0.04 초	0.17 초	4	192.168.0.104	Tesla M60	86 %	251 GB	6.7 %
	5	192.168.0.5	0.02 초	0.21 초	0.05 초	0.29 초	5	192.168.0.105	Tesla M60	D %	251 GB	3.5 %
	6	192.168.0.6	0.02 초	0.13 초	0.02 초	0.16 초	6	192.168.0.106	Tesla M60	86 %	251 GB	6.6 %
	7	192.168.0.7	0.03 초	0.16 초	0.05 초	0.23 초	7	192.168.0.107	Tesla M60	86 %	251 GB	6.8 %
	8	192.168.0.8	0.01 초	0.11 초	0.03 초	0.16 초	8	192.168.0.108	Tesla M60	86 %	251 GB	6.7 %
	9	192.168.0.9	0.01 초	0.16 초	0.02 초	0.19 초	9	192.168.0.109	Tesla M60	86 %	251 GB	6.7 %
	10	192.168.1.1	0.02 초	0.18 초	0.01 초	0.21 초	10	192.168.0.110	Tesla M60	86 %	251 GB	6.8 %
	11	192.168.1.2	0.0 <mark>1</mark> 초	0.11 초	0.04 초	0.16 초	11	192.168.0.111	Tesla M60	86 %	251 GB	6.7 %
	12	192.168.1.3	0.02 초	0.14 초	0.05 초	0.21 초	12	192.168.0.112	Tesla M60	86 %	251 GB	6.7 %
	13	192.168.1.4	0.01 초	0.12 초	0.02 초	0.15 초	13	192.168.0.113	Tesla M60	86 %	251 GB	6.7 %
	14	192.168.1.5	0.01 초	0.1 초	0.03 초	0.14 초	14	192.168.0.114	Tesla M60	86 %	251 GB	6.7 %
	15	192.168.1.6	0.01 초	0.16 초	0.04 초	0.21 초	15	192.168.0.1 <mark>1</mark> 5	Tesla M60	86 %	251 GB	6.8 %
	16	192.168.1.7	0.01 초	0.22 초	0.05 초	0.28 초	16	192.168.0.116	Tesla M60	86 %	251 GB	6.7 %
	17	192.168.1.8	0.02 초	0.1 <mark>1</mark> 초	0.03 초	0.15 초	17	192.168.0.117	Tesla M60	D %	251 GB	7.9 %
	18	192.168.1.9	0.02 초	0.12 초	0.04 초	0.18 초	18	192.168.0.118	Tesla M60	86 %	251 GB	6.8 %
	19	192.168.2.1	0.01 초	0.16 초	0.06 초	0.24 초	19	192. <mark>1</mark> 68.0.119	Tesla M60	86 %	251 GB	6.7 %

Integrated multi-video control

Dent				IvsServerMng			(v x x
ESET-03 (4x3,10) + 💼	Cols: 10 👻 R	ows: 10 👻 🔽	3			c	Apply
→ Server → Server (192.168.0.161) → Server (192.168.0.162) → Server (192.168.0.163) → Server (192.168.0.164)	[0,0] model_2 (192.168.0.4)	[1,0] model_3 (192.168.0.11)	[2,0] model_1 (192.168.0.12)	[3,0] model_2 (192.168.0.13)	[4,0] model_3 [5,0] model_1 [6,0] model_2	[7,0] model_3 [8,0] model_1 [9,0] moder_z
B Server (192.168.0.165) B Server (192.168.0.166) B Server (192.168.0.167) B Server (192.168.0.168) B Server (192.168.0.169)	[0,1] model_3 (192.168.0.20)	[1,1] model_1 (192.168.0.21)	[2,1] model_2 (192.168.0.22)	[3,1] model_3 (192.168.0.23)	Cols: 10 Y Ro	í	
Server (192.168.0.170) Server (192.168.0.140)	[0,2] model_1 (192.168.0.3)	[1,2] model_2 (192.168.0.28)	[2,2] model_3 (192.168.0.29)	[3,2] model_1 (192.168.0.30)	[0,0] model_2 (192.168.0.4)	[1,0] model_3 (192.168.0.11)	[2,0] model_1 (192.168.0.12)
	10,3] model_2 (192.168.0.10)	[1,3] model_1 (192.168.0.100)	[2,3] model_1 (192.168.0.32)	[3,3] model_1 (192.168.0.33)			
	[0,4] model_ (192.168.0.40)	[1,4] model_1 (192.168.0.41)	[2,4] model_1 (192.168.0.42)	[3,4] model_1 (192.168.0.43)	[0,1] model_3 (192.168.0.20) [0,2] model_1 (192.168.0.3)	[1,1] model_1 (192.168.0.21)	[2,1] model_2 (192.168.0.22)
	[0,5] model_1 (192.168.0.50)	[1,5] hoodel_1 (192.163.0.51)	[2,5] model_1 (192.168.0.52)	[3,5] model_1 (192.168.0.53)			
	[0,6] model_1 (192.168.0.60)	[1,6] model_1 (192.168.0.61)	[2,6] model_1 (192.168.0.62)	[3,6] model_1 (192.168.0.63)			
	[0,7] model_1 (192.168.0.70)	[1,7] model_1 (192.168.0.71)	[2,7] model, 1 (192.168.0.72)	[3,7] model_1 (192.168.0.73)		[1,2] model_2 (192.168.0.28)	[2,2] model_3 (192.168.0.29)
	[0,8] model_1 (192.168.0.80)	[1,8] model_1 (192.168.0.81)	[2,8] model_1 (192.168.0.82)	[3,8] model_1 (192.168.0.83)			
	[0,9] model_1 (192.168.0.90)	[1,9] model_1 (192.168.0.91)	[2,9] model_1 (192.168.0.92)	[3,9] model_1 (192.168.0.93)	(192.168.0.94) (192.168.0.95) (192.168.0.96)) (192.168.0.97) (192.168.0.98) (192	2.168.0.99)

Integrated multi-video control



Integrated multi-video control



77/82

Integrated multi-video control



• Large-scale Intelligent Video Analytics



Real-time Intelligent Video Analytics for 1000Ch CCTV streams

NVIDIA Tesla M60 GPU-accelerated DL Inference on 31-node & 30-GPU Hadoop-Spark Cluster

Outline

- Introduction
- GPU-accelerated DL Inference with Hadoop-Spark
 - Fast & Scalable DL Inference
 - DL System Optimization
 - Use case: Intelligent Video Analytics
- Conclusions



Conclusions

- Hadoop-Spark DL Inference works
- DOE gives significant efficiency gains
- Exciting IVA works more to come!





Thanks!



Hunmin Yang hmyang@add.re.kr Agency for Defense Development



Se-Yoon Oh

syoh@add.re.kr

Agency for Defense Development



Ki-Jung Ryu taurus@add.re.kr

Acknowledgements



GIT 굿모닝아이텍(주) <u>www.goodmit.co.kr</u>