

MELTINGCON  
COMMUNITY  
ALLIANCE

# BigData Scale Deep learning on Spark & Horovod , etc...

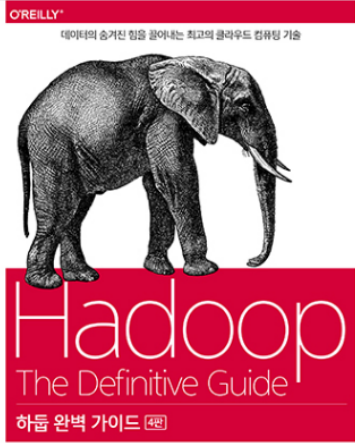
Hoon Dong Kim

Korea Spark User Group

hoondongkim@gmail.com

# I am ...

- 김훈동
- 스사모(Korea Spark User Group) 운영진
- 신세계 그룹 온라인 포털 SSG.COM 빅데이터파트 리더
- Hadoop, Spark, Machine Learning, Azure ML 분야 Microsoft MVP(Most Valuable Professional)
- <http://hoondongkim.blogspot.kr>
- <https://www.facebook.com/kim.hoondong>



하둡 완벽 가이드(4판)  
데이터의 숨겨진 힘을 끌어내는 최고의 클라우드 컴퓨팅 기술

저자 : 톰 화이트  
번역 : 강형석, 강정호, 임상배, 김훈동  
출간 : 2017-03-01  
페이지 : 876 쪽  
ISBN : 9788968484599  
물류코드 : 2459

TAG 스파크, Spark, cloud, bigdata, 빅데이터, 분산 시스템, 가상화

초급    초중급    중급    **중고급**    고급

# 오늘의 주제

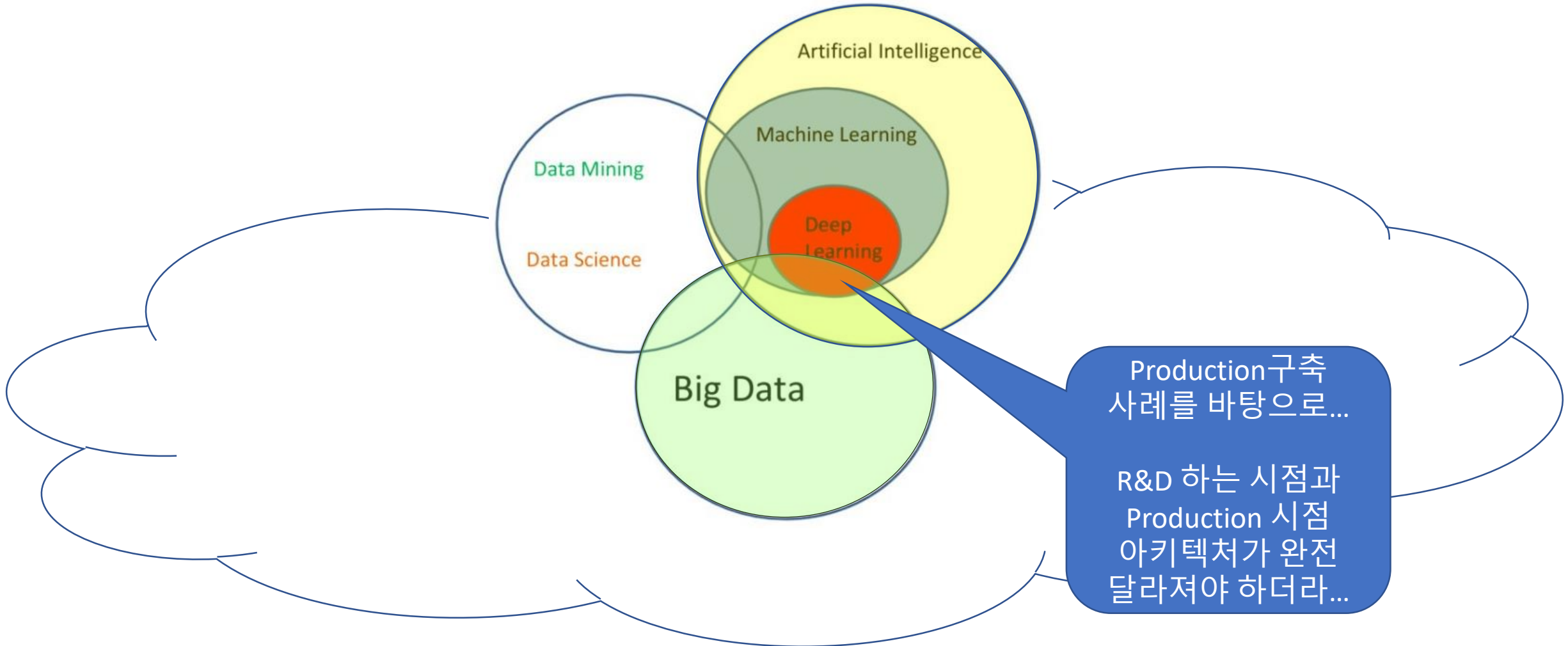
[1] Public Cloud

[2] BigData

[3] AI > Machine Learning > Deep Learning

- [1]을 만난 [2] 에 대하여
- [2]와 결합한 [3]에 대하여

# 오늘의 주제를 다시보자



# BigData Scale Deep Learning을

논하고자 한다면....

# BigData Scale Deep Learning!

1. Scale Out Training 에 대하여...
2. Scale Out Serving 에 대하여...

# 차례

- 기
  - 오늘의 주제
  - Key Word
  - BigData Scale Deep Learning 에 대하여
- 승
  - 모티브
- 전
  - Tech Pyramid
  - Traditional Architecture
  - BigData Scale Deep Learning Training 에 대하여
  - BigData Scale Deep Learning Serving 에 대하여
  - Serverless BigData + AI Architecture
- 결

# Motive!

그건 왜 하려고 하는데??



# Motive 및 우리가 목표하는 것들...

## 알리바바, 광군제 주문 폭주 'AI·로봇'으로 대응했다

송고시간 | 2017/11/12 12:15



주문 단계에서 알리바바의 AI는 개인 맞춤형 추천 상품을 제시해 소비자의 결정을 돕고 재고를 관리한다.

시스템 개발을 책임지는 엔지니어 차이샤오우는 "우리는 소매업자가 판매량을 늘리는 데 AI가 더 효율적일 것이라고 확신한다"며 "수많은 브랜드와 변수를 고려해 추천 상품을 선정하는 데 있어 노련한 패션업계 전문가보다 빅데이터와 AI가 더 탁월한 능력을 보인다"고 말했다.

알리바바는 고객 상담에도 AI를 적극적으로 활용하고 있다.

알리바바의 고객 상담용 챗봇인 '디엔샤오미'(電小秘)는 고객이 문의하는 내용의 90% 이상을 이해할 수 있으며, 하루에 350만 명의 고객을 상담할 수 있다.

알리바바의 제품 관리자 류지엔퉁은 "모든 상담을 AI가 대신할 수는 없지만, 광군제처럼 단시간에 문의가 급증할 때는 큰 도움이 된다"며 "최신 버전은 상담 과정에서 나타난 고객의 감정까지 읽을 수 있다"고 전했다.

고객의 제품 주문이 이뤄지면 포장과 운송은 로봇이 담당한다.

알리바바의 물류 자회사 차이나오(菜鸟)가 중국 남부 선전(深천<土+川>) 인근 휘저우(徽州)에 새로 개장한 자동화 물류 창고에서는 약 200대의 로봇이 24시간 일하고 있다.

## 알리바바 광군절 매출 27.5조, 8억 5,120만 건 주문

Posted 2017. 11. 13 오전 8:59:58



14만 개 이상의 브랜드에서 1,500만 가지 상품이 알리바바 광군절 이벤트에 참가했다. 알리바바는 인공지능 상품 추천 알고리즘, 알리페이 결제, 알리 클라우드 결제 시스템 등 자사 기술을 총 동원했다.

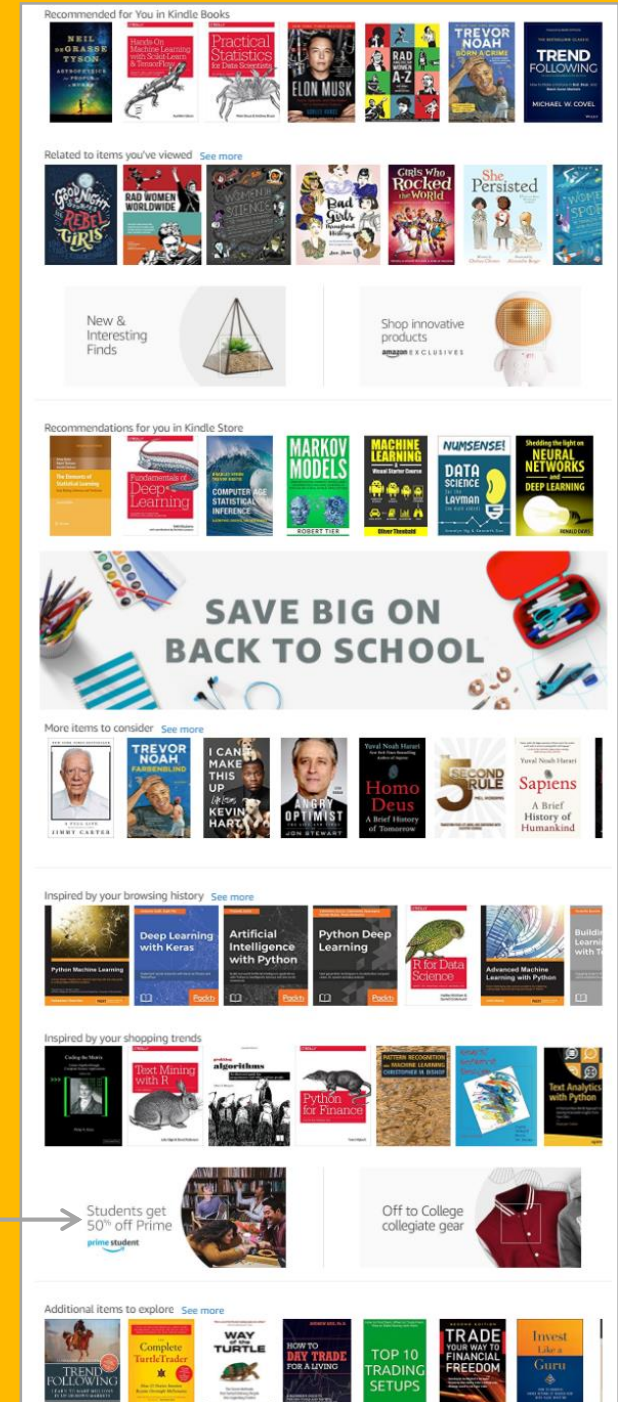
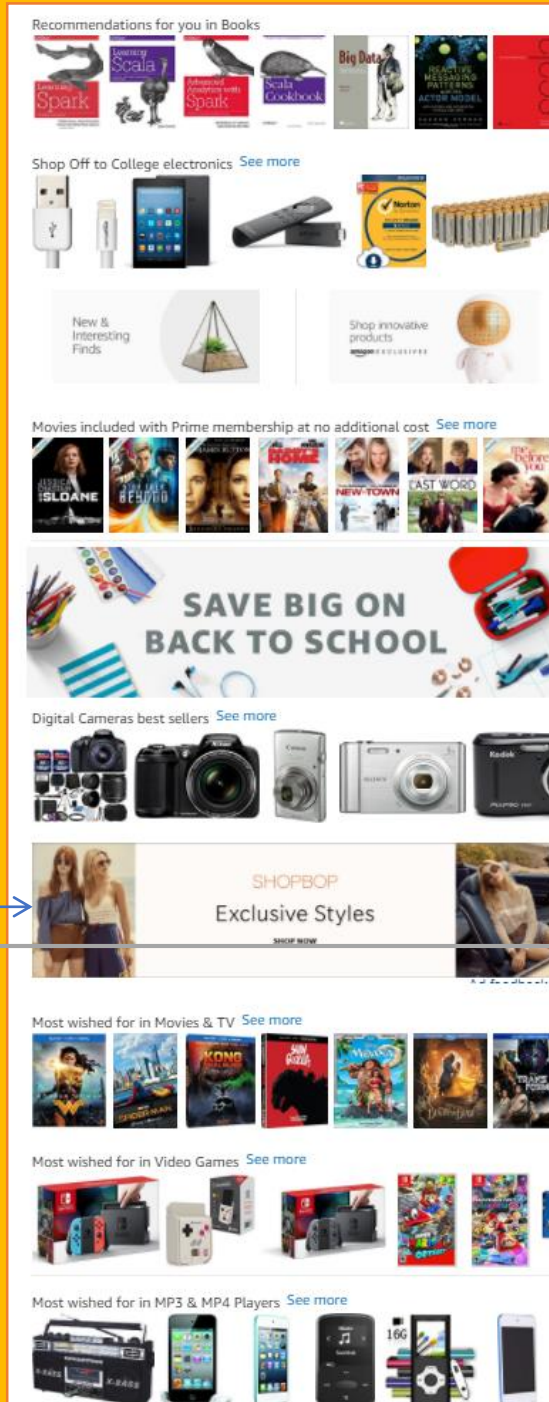
# 아마존 개인화

김\*\* 로그인

동일 시점

박\*\* 로그인

아마존 첫 페이지는 각 영역들의 순서가 개인화 되어 있고, 해당 영역에 노출되는 Item 도 개인화 되어 있음.  
광고 위치 광고 종류 또한 다름.



CON  
ITY  
CE

# Youtube 개인화

김\*\* 로그인

동일 시점

박\*\* 로그인

유튜브 첫 페이지 또한 각 영역들의 순서가 개인화 되어 있고, 해당 영역에 노출되는 Item도 개인화 되어 있음.

This screenshot shows the YouTube homepage for user 김\*\*. The interface is personalized, displaying a grid of video recommendations. The top row features videos related to toys and construction, such as '나중에 볼 동영상' and 'Kids Construction Vehicles Toys'. The second row includes educational content like '도티 TV' and 'Vocaloid Studio'. The third row shows gaming and entertainment videos, including '채재 ChaeChae' and '태경 TV'. The fourth row features technical and AI-related content from 'Siraj Raval' and 'CarrieAndToys'. The bottom row displays more gaming and entertainment videos. The layout is consistent across the different user profiles shown in the image.

This screenshot shows the YouTube homepage for user 박\*\*. The interface is personalized, displaying a grid of video recommendations. The top row features educational and technical content, such as '맞춤 동영상' and 'Learning TensorFlow'. The second row includes gaming and entertainment videos, including 'TF-KR Conf 2' and '어쌔신 크리드'. The third row shows gaming and entertainment videos, including '어쌔신 크리드' and '마인크래프트'. The fourth row features technical and AI-related content from 'Google Developers' and 'New Lokai Water Bracelet'. The bottom row displays more gaming and entertainment videos, including 'BATTLEROUNDS' and '유인상'. The layout is consistent across the different user profiles shown in the image.

NYE

# 흑자는 말한다!

1. BigData 거품론!
2. AI 거품론!
3. 그런 기술은 실제 돈을 벌어주는 기술이 아니다???

# 지피지기(知彼知己)!

우리 경쟁사들의 기술의 위치를  
줄 세워 보고  
우리를 가늠해보자.

# Technology Comparison(예시)

(절대적인 수치는 아님. 편차 설명을 위한 주관적 예시 수치 임)

- 안 중요한 것이 아님.
- 그냥 이것은 기본임.
- 이걸 잘해야 하는 것은 당연한 일.
- [필요조건 or 필수조건]
- But, 차별요소가 아님.

1. 글로벌 선두 권  
- First Mover (Google, Amazon, Alibaba, Facebook, etc)

2. 글로벌 상위권, 국내 최상위권  
- Fast Follower 혹은 차별화 선두 권(Naver, Kakao, etc)

3. Local(국내) 상위권

4. Local(국내) 중위권

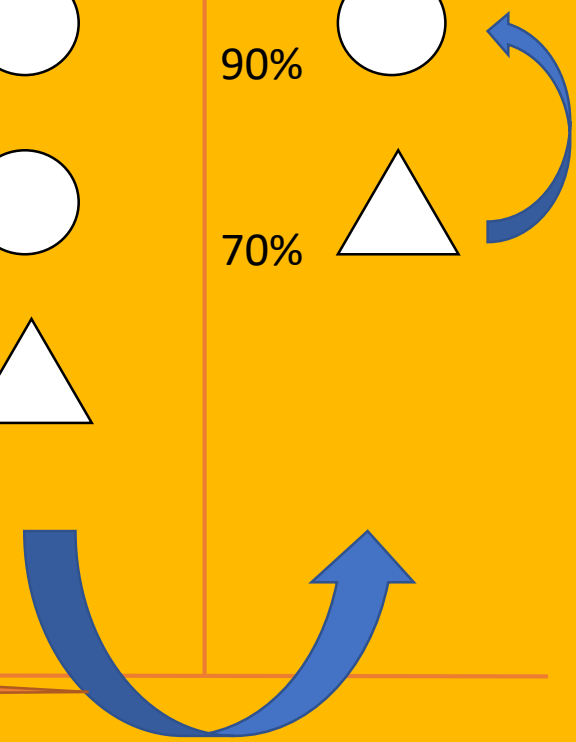
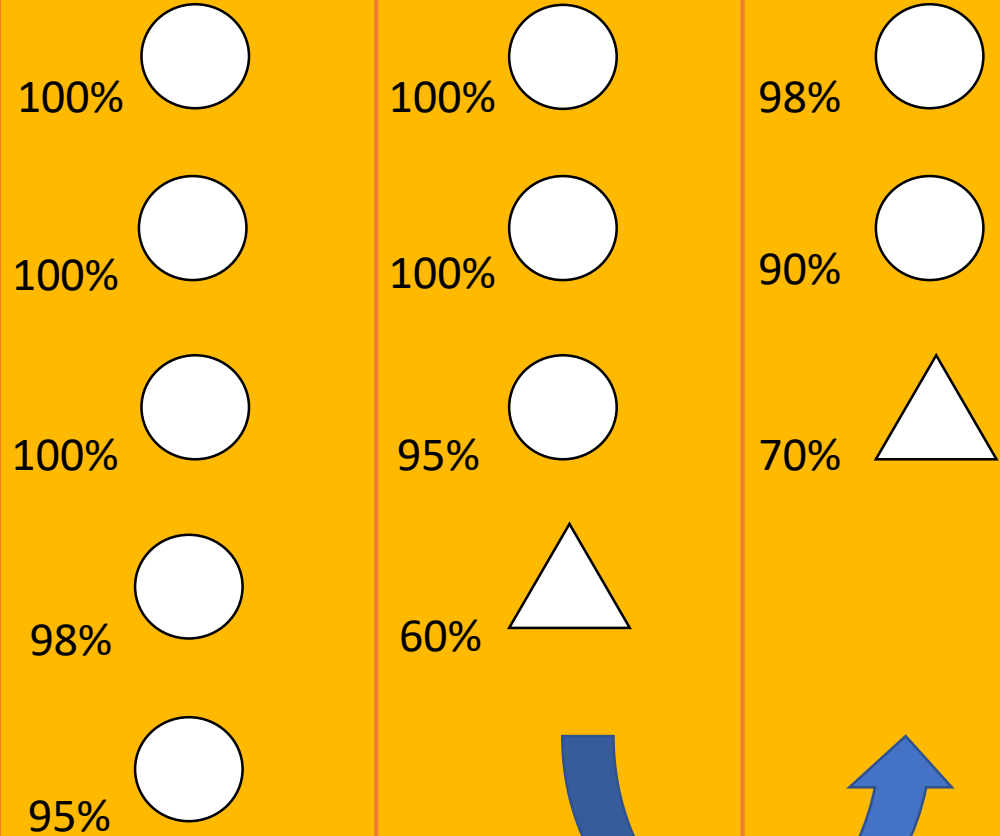
5. Local(국내) 하위권

- 오늘 말하고자 하는 것
- [충분조건]의 부분집합
- 차별요소를 극대화 하고
- ...
- 고도화 선도기술을 쉽고 빠르게 도입해보는...
- Fast Follower 전략에 관한 내용 임...

Web/DB/App  
[기본기술]

BigData/NoSQL/  
샤딩DB/DataAnalytics  
[차별기술]

개인화/ML/DL/MAB  
DevOps/Microservice  
BigData Scale AI  
[고도화선도기술]



**글로벌 선두권 들은  
어떻게 하고 있나??!!**

# Reinforcement Learning to Rank in E-Commerce Search Engine

- By Alibaba & Taobao
- <https://arxiv.org/abs/1803.00710>
- 해당 논문에 대한 블로깅.

실시간성...  
개인화...  
Training & Serving 이 동시에...



Figure 1: A typical search session in Taobao. A user starts a session from a query, and has multiple actions to choose, including clicking into an item description, buying an item, turning to the next page, and leaving the session.



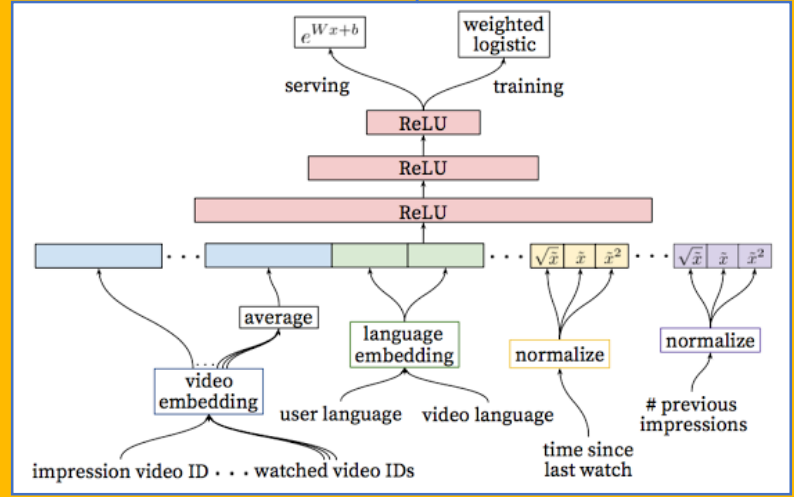
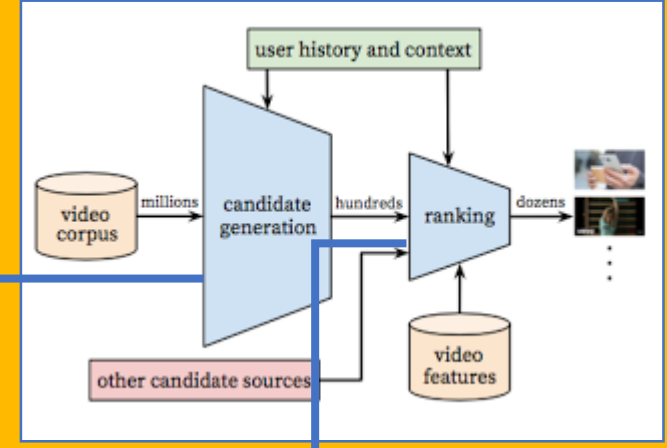
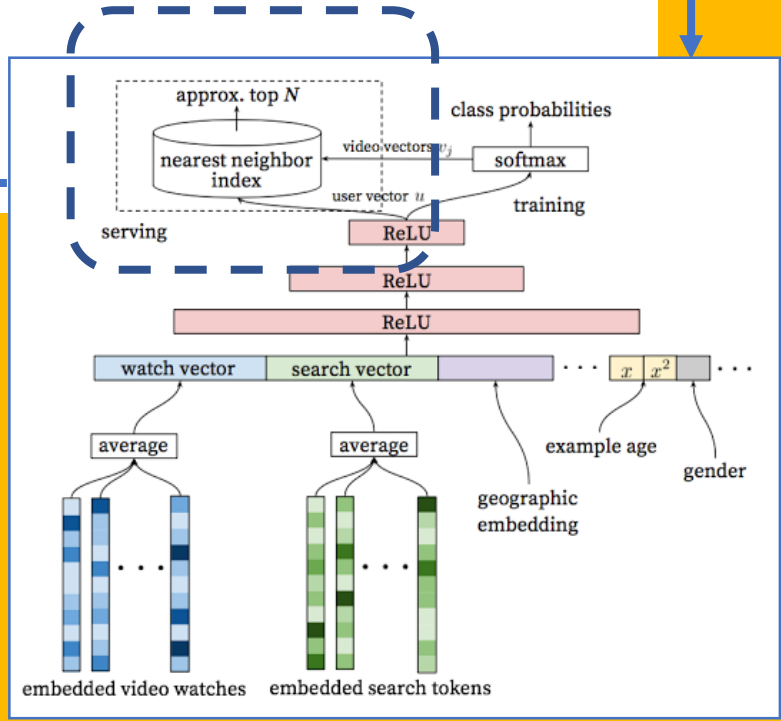
# YouTube Recommendation

## Deep Neural Networks for YouTube Recommendations

- 세 가지 중요한 잣대: Scale, Freshness, Noise
  - Scale: 데이터가 매우 많으므로 scalability가 중요하다.
  - Freshness: 새로운 비디오가 추가되었을 때 추천에 바로바로 반영되어야 한다.
  - Noise: 데이터의 sparsity, ground truth의 부재, 거의 항상 implicit feedback만 갖고있다, meta data가 엉망인 점 등을 고려해야함.
- 구현
  - Tensorflow를 사용함
  - 전체 시스템은 10억개정도의 파라미터 존재
  - 트레이닝 데이터는 수천억개정도.

Serving Layer

Freshness  
Scalability



# 실무 Production 에서의 Chatbot 이란?

- Google Smart Reply (이메일 문의 처리) 시스템의 예
  - <https://arxiv.org/abs/1606.04870>
  - **Smart Reply: Automated Response Suggestion for Email (by Google)**
  - LSTM
  - 10% 정도를 커버.(2016년 기준)
  - Production 상황의 Pain Point 를 잘 설명.
    - Diversity
    - Scalability
    - 25% 가 20 token 미만
    - Coherent
    - Response Quality
    - Utility
    - Privacy

Semi-supervised  
Learning 방식 채용

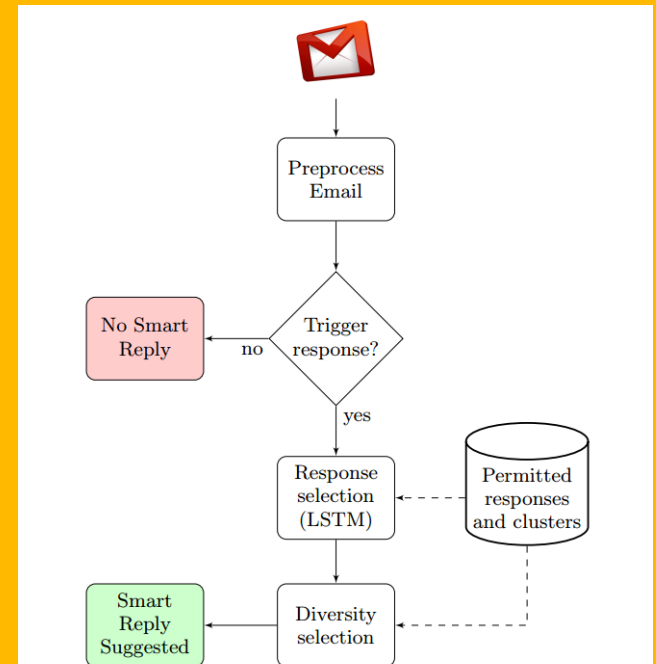


Figure 2: Life of a message. The figure presents the overview of inference.

# 실무 Production 에서의 Chatbot 이란?

- Sequential Model + Semi Supervised Learning Model 챗봇
  - Smart Reply
    - <https://arxiv.org/abs/1606.04870>
    - **Smart Reply: Automated Response Suggestion for Email (by Google)**
    - LSTM

Semi-supervised  
Learning 방식 채용

- [Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation \(by Google\)](#)
- <https://arxiv.org/pdf/1512.01752.pdf>

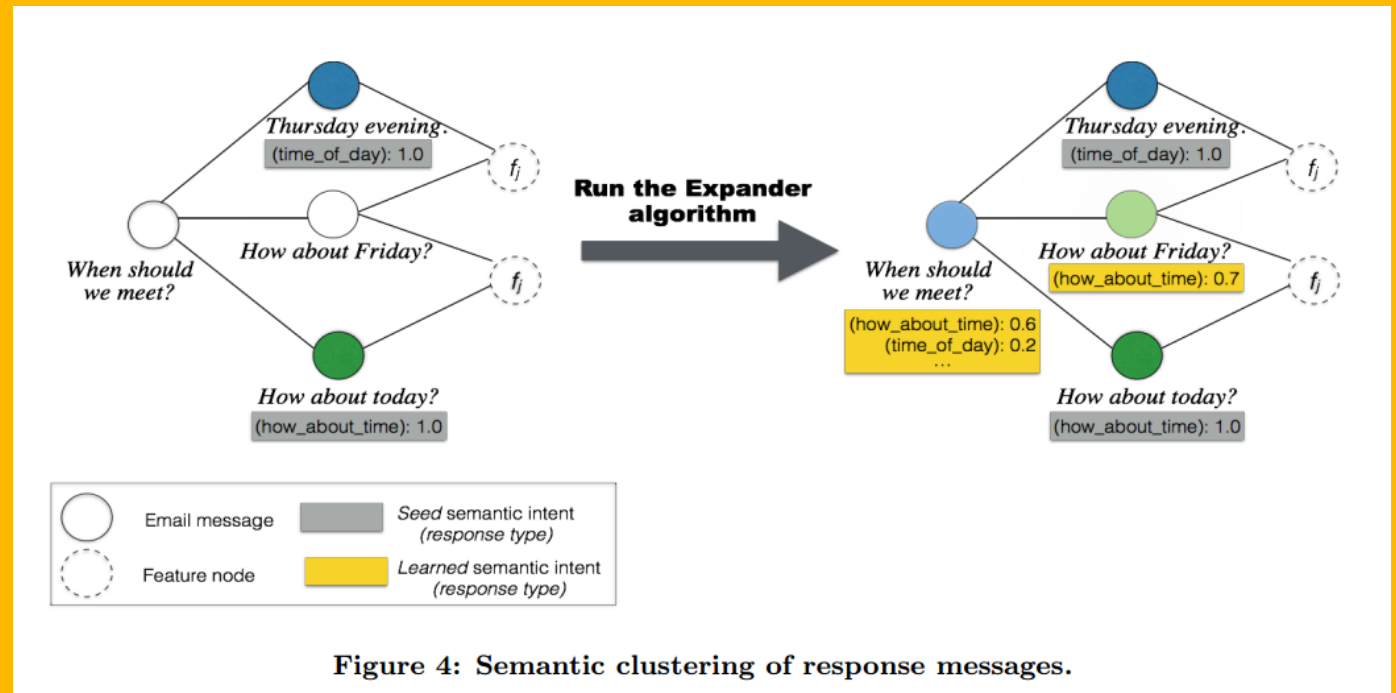
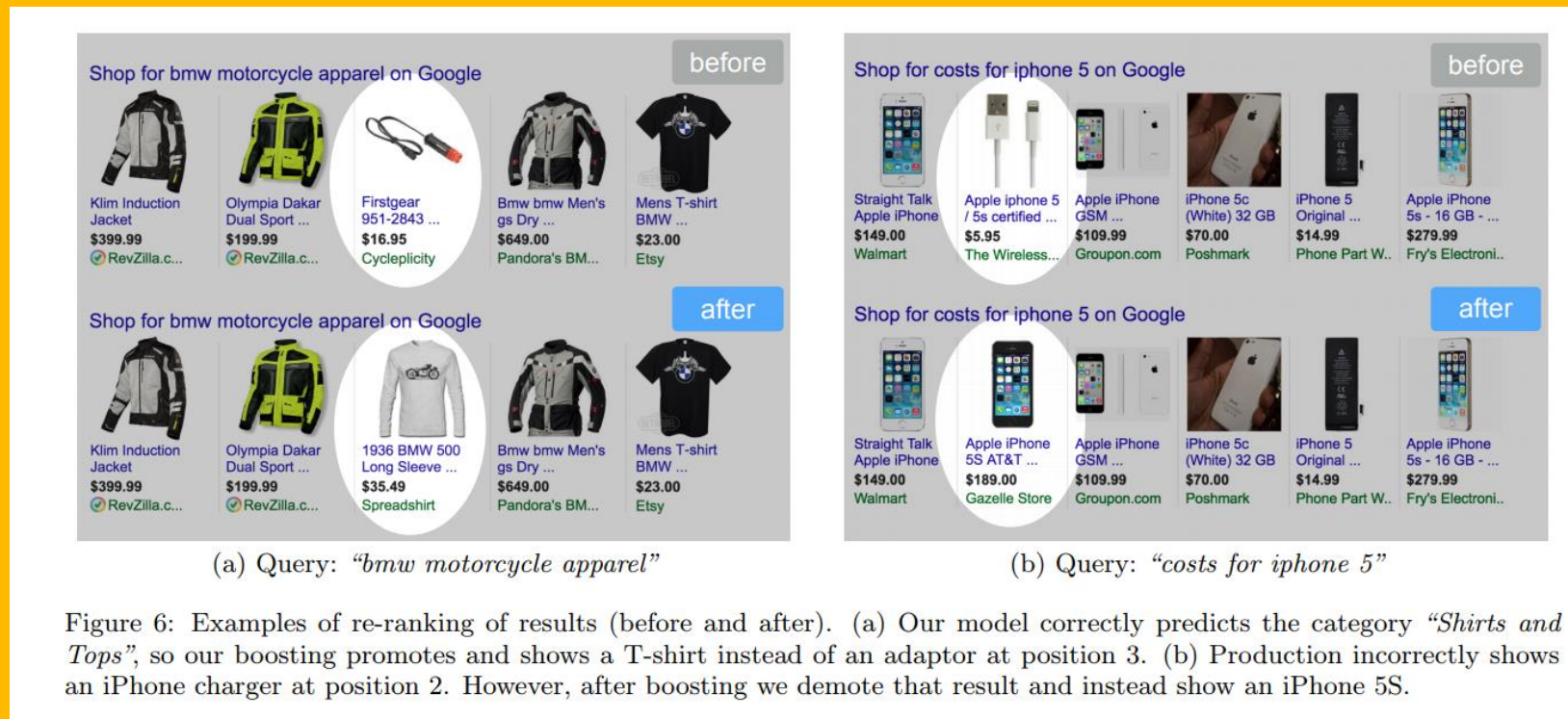


Figure 4: Semantic clustering of response messages.

# Deep Learning for Semantic Search ( by Google )

- Predicting Latent Structured Intents from Shopping Queries
- Hybrid LSTM
- [https://www.cs.utexas.edu/~cywu/www2017\\_shopping\\_query.pdf](https://www.cs.utexas.edu/~cywu/www2017_shopping_query.pdf)

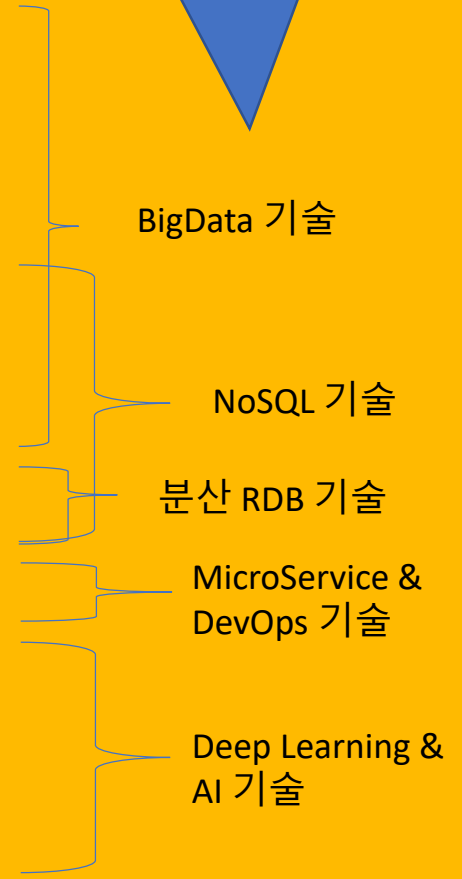


# Google Did!

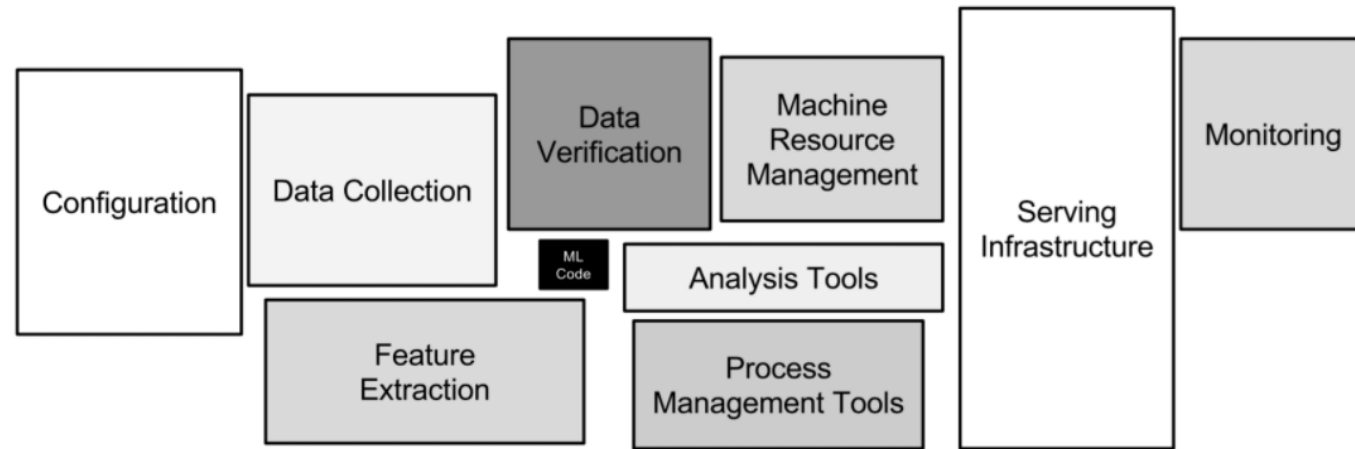
- 구글 년도 별 주요 발표 내용

Deep Learning 기술 이전에 BigData, NoSQL, MicroService 등에 대한 Core 기술 경쟁력이 뛰어남.

년도	Google 발표 기술	OpenSource 진영 기술
2003년	GFS	Hadoop HDFS
2004년	MapReduce	Hadoop MapReduce
2006년	Chubby	Zookeeper
2006년	BigTable	HBase
2010년	Pregel	Neo4J, Spark Graph-X
2010년	Dremel	Spark
2011년	Tenzing	Hive, Spark SQL
2012년	Spanner, F1	RDB Sharding + Kafka + Redis : RDB 분산 Scale Out 및 동기화 시스템
2013년	Omega	Docker(Mesos)
2014년	Word2Vec 외 다수	Word2Vec (NLP 요소 기술) 외 다수
2015년	Tensorflow 외 다수	Tensorflow (Deep Learning Framework) 외 다수
2016년	AlphaGo, DeepQN 외 다수	복합문제를 해결하는 다수의 Deep Learning 방법론
2017년	PathNet 외 다수	PathNet(Deep Learning 학습의 진화방법) 외 다수



# Production 에서는 Model 개발 외에도 많은 부분의 Engineering Art 가 필요함.



*Only a tiny fraction of the code in many ML systems is actually devoted to learning or prediction.*

Schulley et al, [Hidden Technical Debt in Machine Learning Systems](#), In NIPS 2015.

Model 개발에 소요되는 시간 및 난이도는 오히려 다른 부분에 비하여 쉬운 편.(예외도 있음)  
다수 동접자 Serving Layer 나 고객의 반응을 실시간 학습 시켜 반영 하는 Realtime inference, 모델의 무중지 Rolling Upgrade 및 각종 BigData Scale Data Pipelining 은 매우 어려운 문제.

적은 인원과, 적은 기회와, 적은 시간으로...



거인의 어깨에 올라서서  
더 넓은 세상을 바라보라  
-아이작 뉴턴



**Fast Follow 해보기.**

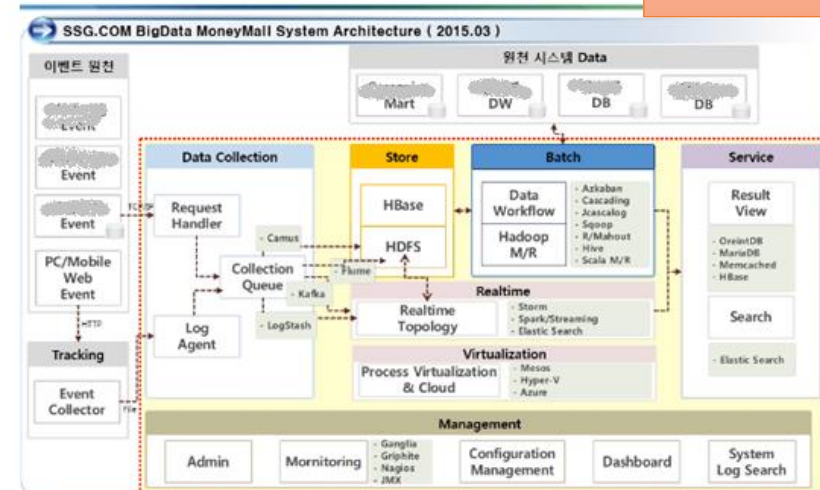
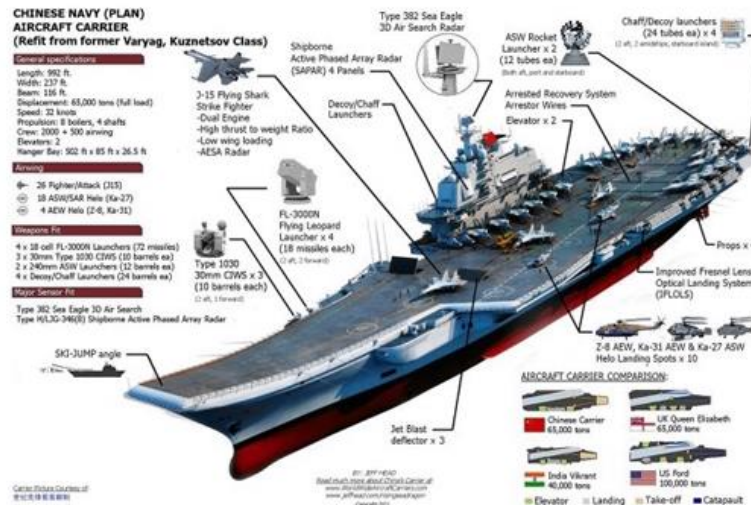
# Google 따라하기! + 알파

## Hadoop Eco System & NoSQL

- Volume, Variety, Velocity 를 Legacy 대비 100 배 끌어 올리기까지 다양한 Eco System 의 도움이 필요 하였음.

	Google	Open Source 진영	비고
2003년	GFS	HDFS	Hadoop
2004년	MapReduce	MapReduce	
2006년	Chubby	Zookeeper	클러스터 관리
2006년	BigTable	Hbase( Casandra, MongoDB, OrientDB)	NoSQL
2010년	Dremel	Drill, Impala, Tajo, Hwaq( vs Spark )	SQL on Hadoop ( + In-Memory )
2013년	Omega	Docker( vs Mesos )	Process Virtualization

+ TensorFlow





# 하는 일에 단계를 밟아가고 있음!

## - 과거 현재 미래

- 과거를 분석 한다. (BigData Eco System Infra)
  - 트래킹 로그를 남긴다.
  - 빅데이터 수집 저장 분석을 위한 인프라를 만든다.
  - 빅데이터 배치로 과거 시계열 분석을 하고 시각화를 한다.
- 현재에 반응 한다. (RealTime Layer / Spark Streaming / ELK)
  - 실시간으로 데이터 스트림을 분석한다.
  - FDS, 보안관제, 모니터링 등 즉각적으로 현재 상황에 대처하여 현재를 능동적으로 대비한다.
- 미래를 예측 한다. (Mining / Machine Learning / Deep Learning)
  - 고객이 관심 갖고 있고 곧 살 것 같은 것을 추천한다.
  - 미래에 집행할 광고 및 제휴 채널 예산을 보다 ROI 높게 배분한다.
  - 발주를 예측한다.
  - 최적의 트럭 경로를 예측한다.
  - 가격을 올릴지 말지 얼마나 세일할지 최적의 가격을 예측한다.
- 미래를 대비 한다. (Machine Learning / Deep Learning)
  - Chatbot
  - 자연어 활용, 이미지 활용 -> 검색 및 추천 고도화
  - BigData Scale Deep Learning.
  - 다중 접속자를 위한 개인화 Deep Learning 서비스.
  - Auto Scale Out, 무중지 AI 모델의 진화 및 원순환 배포.

4~5년 전

2~3년 전

요즘

순서가 있고,  
단계가 있고,  
아랫 기술은 윗 기술로  
부터 시너지가 나더라....

# Production 에서 중요한 것!



Sample Data, Selected  
Feature 정교한 모델

주1회 or 일1회

VS

더 많은 Data(or 전수 Data)  
에 적용하는 Simple 모델

최신성  
(매 시간 or 준실시간)

모두에게 적용하는 정교  
한 단일 모델

최고 정확도

개인화 모델

덜 정확해도 빠르고,  
다수에게, 실시간으로...

# BigData Scale Deep Learning!

MELTINGCON  
COMMUNITY  
ALLIANCE

Deep Dive ~~

[차례]

1. 국내 타 업체들의 방식
2. 국외 대표 업체들의 접근 방식
3. 기존에 시도해본 방식(on-premise)
4. 최근에 시도한 방식(serverless public cloud)

# BigData Scale Deep Learning!

## Deep dive~~

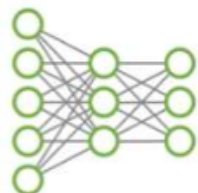
### 1. 국내 타 업체들의 방식

# 국내 업체들의 방식

## NVIDIA DEEP LEARNING SDK UPDATE

Up to 3x Faster Deep Learning on Volta

### GPU-accelerated DL Primitives



cuDNN 7

**2.5x** Faster Training of CNNs for computer vision

**3x** Faster Training of RNNs for speech and machine translations

Leading frameworks support

### Multi-GPU & Multi-node



NCCL 2

Multi-node distributed training on up-to 8 servers

Near-linear scaling on PCIe and Nvlink interconnect

Leading frameworks support

### Inference Optimizer and Runtime Engine



TensorRT 3

**3.5x** Faster Inference

Optimize TensorFlow or Caffe trained models

Linux, Windows, QNX and Android support

# 국내 업체들의 방식

## NVIDIA Collective Communications Library (NCCL) 2

Multi-GPU and multi-node collective communication primitives

High-performance multi-GPU and multi-node collective communication primitives optimized for NVIDIA GPUs

Fast routines for multi-GPU multi-node acceleration that maximizes inter-GPU bandwidth utilization

Easy to integrate and MPI compatible. Uses automatic topology detection to scale HPC and deep learning applications over PCIe and NVLink

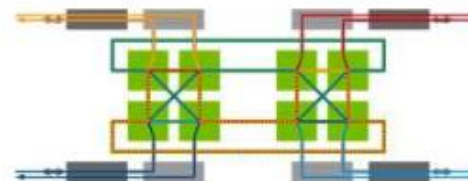
Accelerates leading deep learning frameworks such as Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch and more



Multi-GPU:  
NVLink  
PCIe



Multi-Node:  
InfiniBand verbs  
IP Sockets

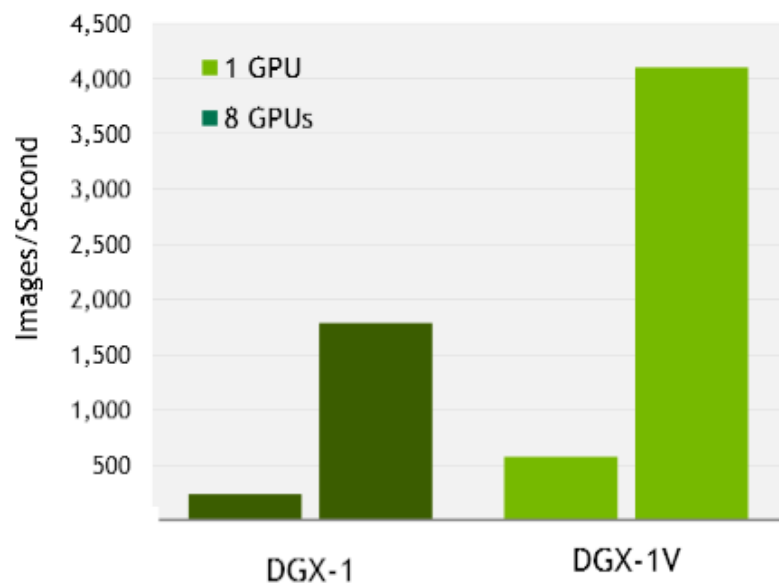


Automatic  
Topology  
Detection

# 국내 업체들의 방식

## NCCL: MULTI-GPU AND MULTI-NODE SCALING

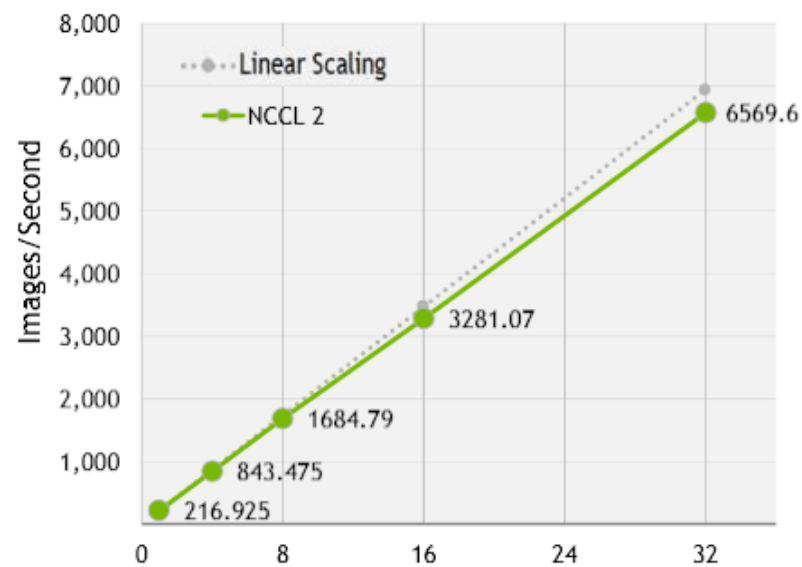
7x Faster Training on DGX vs. single GPU



Caffe2 multi-GPU performance (images/sec) DGX-1 + cuDNN 6 (FP32), DGX-1V + cuDNN 7 (FP16). ResNet50, Batch size: 64

[developer.nvidia.com/nccl](http://developer.nvidia.com/nccl)

Near-Linear Multi-Node Scaling



Microsoft Cognitive Toolkit multi-node scaling performance (images/sec), NVIDIA DGX-1 + cuDNN 6 (FP32), ResNet50, Batch size: 64

# 국내 업체들의 방식

## Job scheduler system, GPU and Container

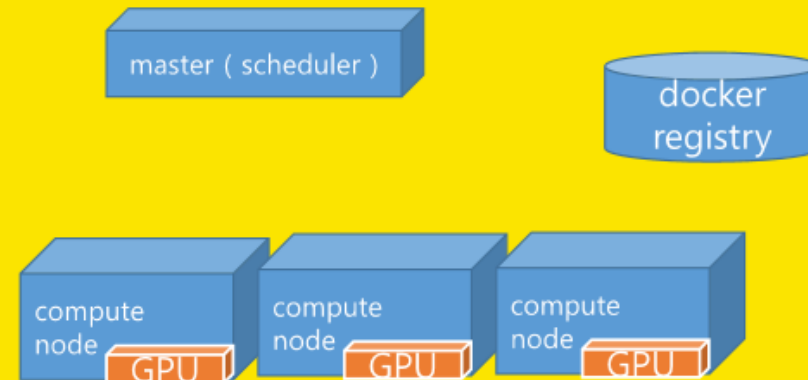
---

add GPU resource to Job Script.  
use NVIDIA Docker for the command

...

then scheduler will do the job

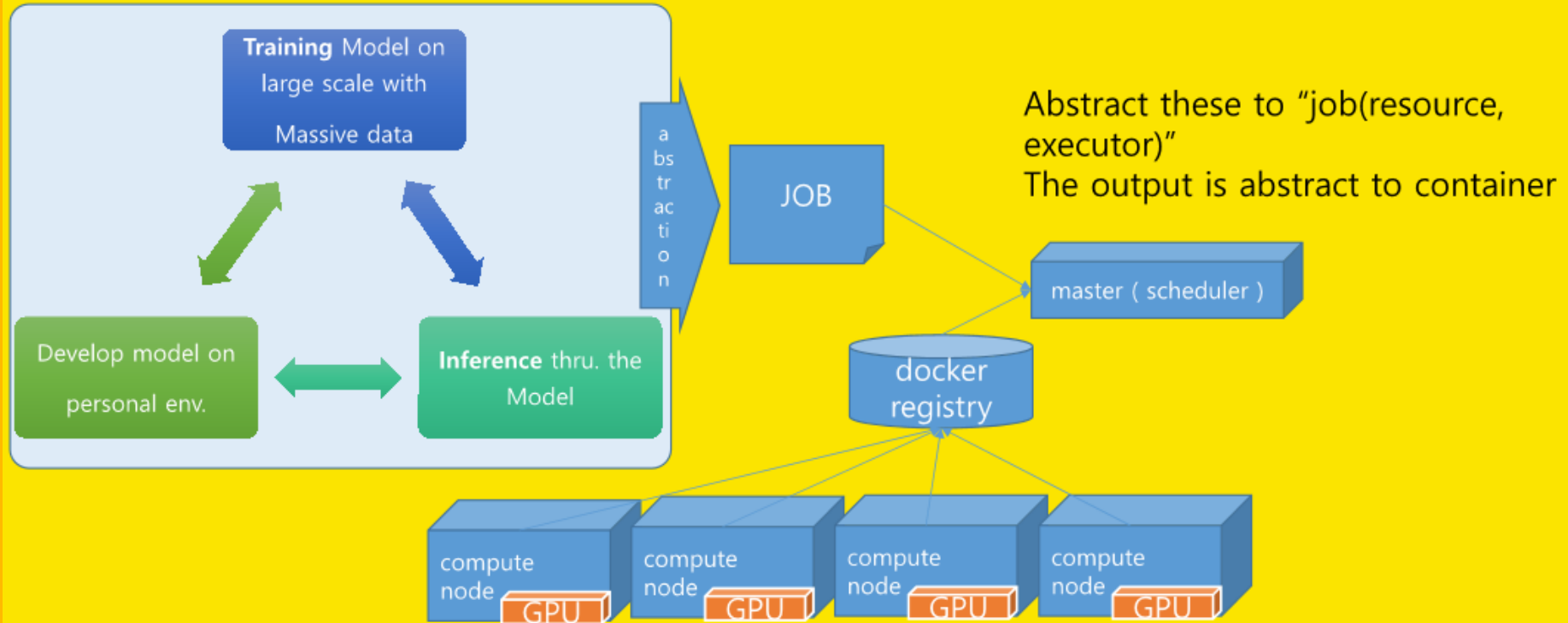
```
#!/bin/bash
#PBS -l nodes=1:ppn=2
#PBS -l walltime=00:00:59
#PBS -l gpus=8
NV_GPU=$NV_GPU nvidia-docker run --net
host -e PASSWORD=root -e USERNAME=root
-e PORT=$PORT
idock.daumkakao.io/dkos/nvidia-cuda-
sshd:dev
```





# 국내 업체들의 방식

## AI Development Cycle over compute resource



# BigData Scale Deep Learning!

## Drill down~~

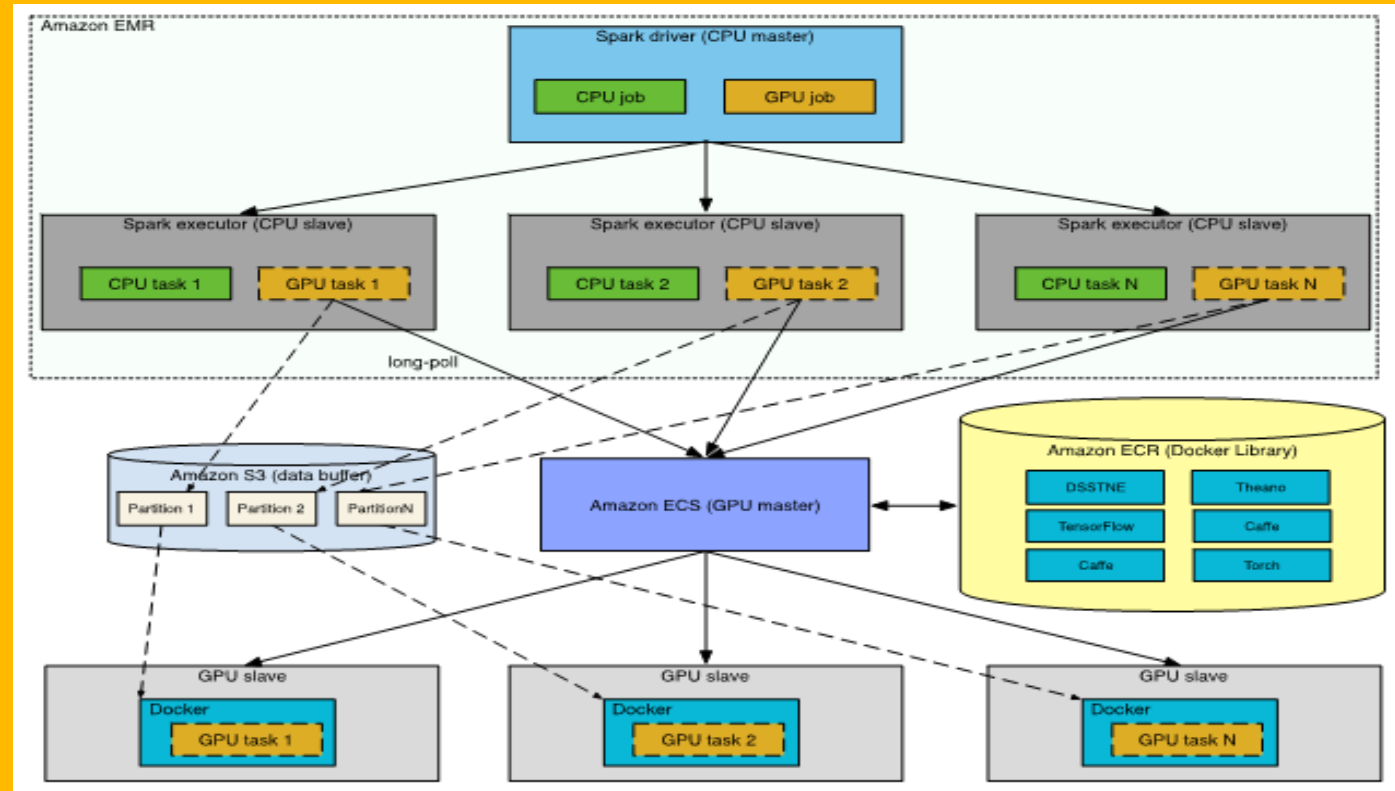
### 2. 국외 대표 업체들의 접근 방식

# Amazon 따라하기! + 알파

## Amazon의 사례 (for Personalized Deep Learning Approach)

### Large Scale Parallel Deep Learning Example

- <https://aws.amazon.com/ko/blogs/big-data/generating-recommendations-at-amazon-scale-with-apache-spark-and-amazon-dsstne/>
- <https://github.com/amzn/amazon-dsstne>



# BigData 진영 대표주자 Spark의 진화 => BigData 에서 BigData + AI 로...

MELTINGCON  
COMMUNITY  
ALLIANCE

## Spark Summit is Becoming the Spark + AI Summit



by Matei Zaharia

Posted in COMPANY BLOG | December 6, 2017

We're excited to announce that Spark Summit is expanding its coverage in 2018 to include in-depth content on artificial intelligence. We are also renaming the conference **Spark + AI Summit**. AI has always been one of the most exciting applications of big data and Apache Spark, so with this change, we are planning to bring in keynotes, talks and tutorials about

## Spark Summit 2017 에서도 이미...



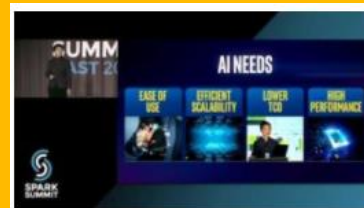
Deep Learning to Big Data Analytics on Apache Spark Using BigDL  
Xianyan Jia (Intel)



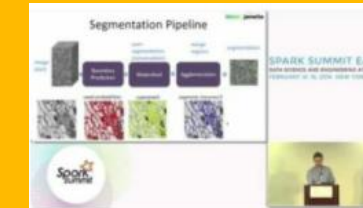
Very Large Data Files, Object Stores, and Deep Learning—Lessons Learned While Looking for Signs o...



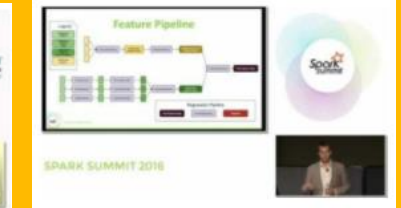
Building Deep Learning Powered Big Data



Accelerating Machine Learning and Deep Learning At Scale...With Apache Spark



Distributed Tensor Flow on Spark: Scaling Google's Deep Learning Library



Scalable Deep Learning Platform On Spark In Baidu  
Kyle Tsai (Baidu)



TensorFlow On Spark: Scalable TensorFlow Learning on Spark Clusters

Andy Feng (Yahoo)  
Lee Yang (Yahoo)



Netflix - Productionizing Spark On Yarn For ETL At Petabyte Scale  
Ashwin Shankar (Netflix)



Netflix's Recommendation ML Pipeline Using Apache Spark  
DB Tsai (Netflix)



Going Real-Time: Creating Frequently-Updating Datasets for Personalization  
Shriya Arora (Netflix)

# BigData Scale Deep Learning!

## Drill down~~

### 3. 기존에 시도해본 방식

# Intent Classifier 예시

- |    |  |        |
|----|--|--------|
| 1. | Word2Vec + CNN (Batch Normalize + Augmentation)  | 72.30% |
| 2. | Word2Vec + LSTM                                  | 73.94% |
| 3. | Word2Vec + CNN + LSTM                            | 72.97% |
| 4. | Word2Vec + Bidirectional GRU                     | 74.36% |
| 5. | Word2Vec + Bidirectional GRU + Attention Network | 73.15% |
| 6. | FastText   | 72.50% |
| 7. | Glove + LSTM (BigDL on Spark Cluster)            | 75.25% |

450여개 Multiple Class , Top 1 문제.

8. 다양한 Approach 의 조합 최종.



Data도 달라짐.  
정제, 클리닝,  
Argumentation,  
Data 원본 품질 재  
정비



89.6%



# Deep Learning 은 고행인가?

- Word2Vec + Bidirectional GRU
- Tesla M40 GPU
- Training Data 165만 건
- Learning Rate 0.0005

```
Epoch 1/5  
1649415/1649415 [=====] - 10174s - loss: 1.0124 - acc: 0.6992 - val_loss: 0.9239 -  
Epoch 2/5  
1649415/1649415 [=====] - 9949s - loss: 0.9461 - acc: 0.7157 - val_loss: 0.9076 -  
Epoch 3/5  
1649415/1649415 [=====] - 9951s - loss: 0.9318 - acc: 0.7199 - val_loss: 0.9068 -  
Epoch 4/5  
1649415/1649415 [=====] - 9953s - loss: 0.9234 - acc: 0.7220 - val_loss: 0.9039 -  
Epoch 5/5  
1649415/1649415 [=====] - 9962s - loss: 0.9176 - acc: 0.7241 - val_loss: 0.9079 - val_acc  
: 0.7436  
tail: /keras_output.txt: file truncated
```

- 5 Epoch 에 50000초 = 833분 = 약 14시간

# Deep Learning 은 고행인가?

- seq2seq
- Tesla K80 GPU \* 2
- Training Data 180만 건
- Training 시간 약 24시간

```
dsrm-gpu05.eastus.cloudapp.azure.com - PuTTY
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID ace4:00:00.0
Total memory: 11.17GiB
Free memory: 466.44MiB
I tensorflow/core/common_runtime/gpu/gpu_device.cc:777] Peer access not supported between device ordinals 0 and 1
I tensorflow/core/common_runtime/gpu/gpu_device.cc:777] Peer access not supported between device ordinals 1 and 0
I tensorflow/core/common_runtime/gpu/gpu_device.cc:906] DMA: 0 1
I tensorflow/core/common_runtime/gpu/gpu_device.cc:916] 0: Y N
I tensorflow/core/common_runtime/gpu/gpu_device.cc:916] 1: N Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:975] Creating TensorFlow device (/gpu:0) -> (device: 0, name: Tesla K80, memory-limit: 466.44MiB)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:975] Creating TensorFlow device (/gpu:1) -> (device: 1, name: Tesla K80, memory-limit: 466.44MiB)
Initialize variables...
WARNING: Restoring previous model from /datadrive/hoondong/deep_qa/DeepQA/save/model/model.ckpt
Testing: Launch interactive mode:

Welcome to the interactive mode, here you can ask to Deep Q&A the sentence you want. Don't have high expectation. Try
Q: hi
A: 안녕하세요

Q: 안녕. 영어도 좀 하네.
A: 안녕.

Q: 영어는 언제 배웠어?
A: 내가 뭘?

Q: 뭘 소리야
A: 말씀드릴 수 없습니다

Q: 막 지껄이는 구나.
A: 난 그냥 ...

Q: 너 근데 말이 좀 짧다.
A: 오, 이런.

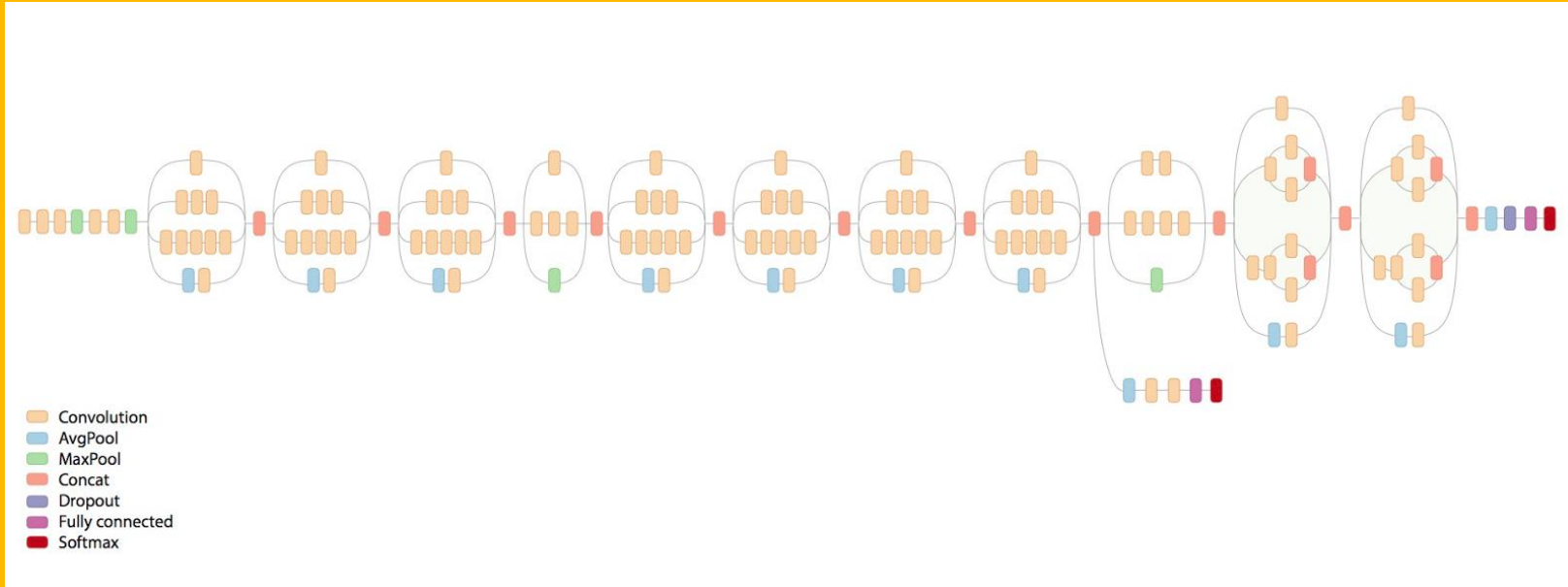
Q: 너의 의견을 말 해봐.
A: 왜?

Q: 너의 생각이 궁금하니까.
A: ?

Q: 난 너가 좀 길게 말 했으면 좋겠어.
A: 난 널 사랑해.
```



# Training & Serving 속도에 대하여



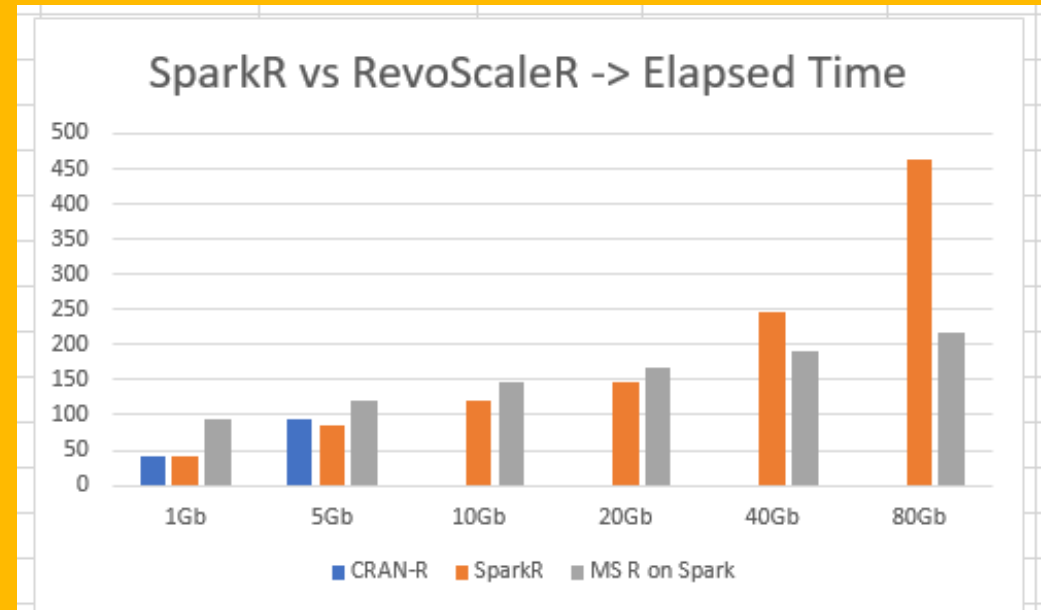
- 이미지와 달리 Text NLP 등은 오픈 된 Pre training Set 이 거의 없음.
- 이미지의 경우에도 실무에서는 정확도를 올리기 위해 Fine Tuning 하는 경우가 많음.
- 깊은 층의 모델로 학습시키는 경우 어마어마한 시간이 걸림.
- 게다가 다양한 Model 의 Mesh Up 필요.
- Live 상품갯수 500만건, 이미지 4000만 건?? 게다가 매일 매일 유입되는 수천건의 이미지?? 매일 매일 바뀌는 수천건의 상품들....
- Production 의 Serving 은 또 다른 문제....

# R on Spark

- SparkR
- Sparklyr
- RevoScaleR(MS R)

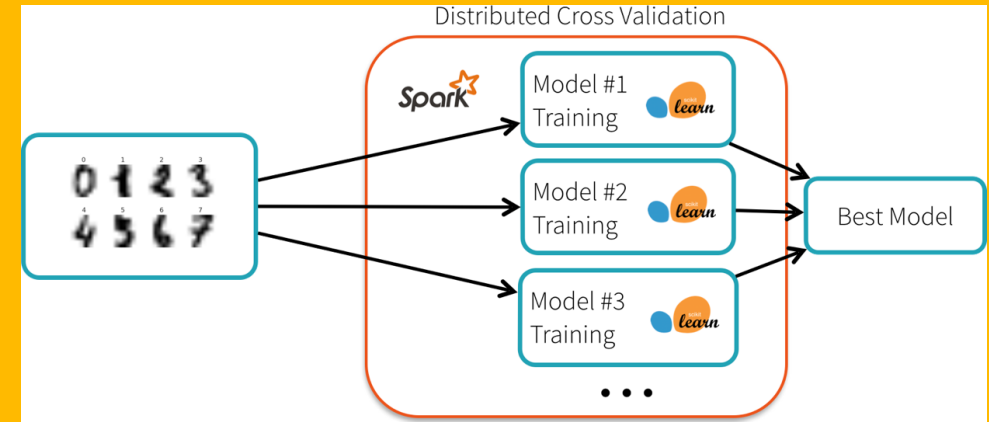
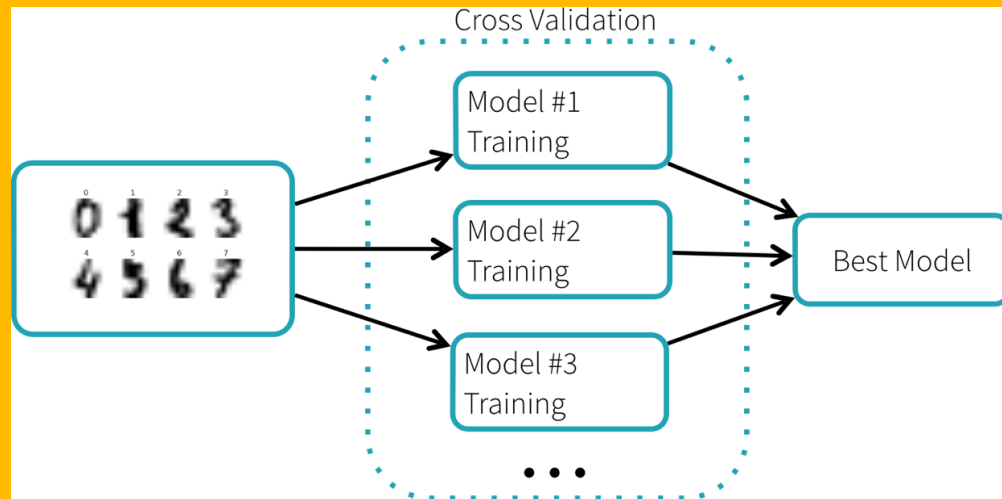
7 Machine on Spark Cluster

8Core – 65GB Memory. 7 Machine.

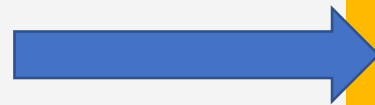


Y축 : Elapsed Time 낮을 수록 성능 좋음.

# Python Machine Learning on Spark



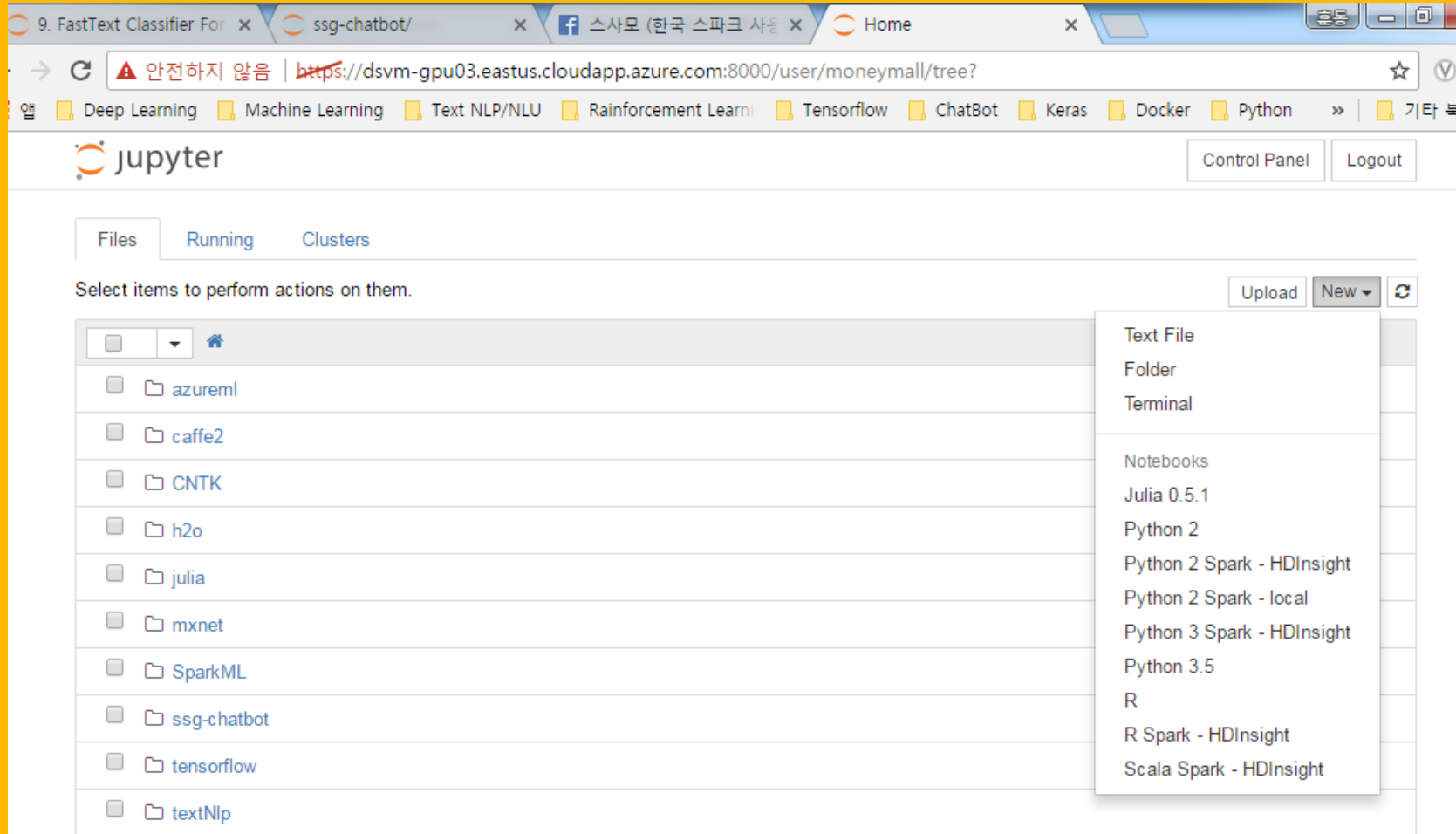
```
from sklearn import grid_search, datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.grid_search import GridSearchCV
digits = datasets.load_digits()
X, y = digits.data, digits.target
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [1, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"],
              "n_estimators": [10, 20, 40, 80]}
gs = grid_search.GridSearchCV(RandomForestClassifier(), param_grid=param_grid)
gs.fit(X, y)
```



```
from sklearn import grid_search, datasets
from sklearn.ensemble import RandomForestClassifier
# Use spark_sklearn's grid search instead:
from spark_sklearn import GridSearchCV
digits = datasets.load_digits()
X, y = digits.data, digits.target
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [1, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"],
              "n_estimators": [10, 20, 40, 80]}
gs = grid_search.GridSearchCV(RandomForestClassifier(), param_grid=param_grid)
gs.fit(X, y)
```

# Jupyter Notebook 도 BigData Scale 로

Zupyter on PySpark, Zupyter on Hadoop



# SSG.COM BigDL

GPU Memory Resource 가 문제될 때는 BigData Scale Large Cluster Parallel Deep Learning Approach with BigDL

- BigDL Deep Learning Job on Hadoop Yarn Manager (by Spark Job)

The screenshot shows the Hadoop Yarn Manager interface for a cluster named 'bdnodeb201.prod.moneymall.ssgbi.com:8088/cluster/apps/RUNNING'. The page is titled 'RUNNING Applications' and shows various metrics and application details.

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
20	0	1	19	33	197 GB	29.46 GB	0 B	65	104	0	13	0	0	0	0

**User Metrics for dr.who**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Fair Scheduler	[MEMORY, CPU]	<memory:5120, vCores:1>	<memory:26120, vCores:8>

**Application List**

ID	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1493106244744_0020	moneymall Text classification	SPARK	root_moneymall	Fri Jun 23 18:11:58 +0900 2017	N/A	RUNNING	UNDEFINED		ApplicationMaster

# SSG.COM BigDL

MELTINGCON

COMMUNITY

ALLIANCE

GPU Memory Resource 가 문제될 때는 BigData Scale Large Cluster Parallel Deep Learning Approach with BigDL

- BigDL Deep Learning Text Classification

```
moneymall@bdnodeb203:/data01/src/moneymall-scala-BigDL/poc
17/06/23 19:52:51 INFO optim.DistriOptimizer$: [Epoch 19 14336/16012][Iteration 152][Wall Clock 320.05617011s] Epoch finished. Wall clock time is 328355.00156ms
17/06/23 19:52:51 INFO optim.DistriOptimizer$: [Wall Clock 328.35500156s] Validate model...
17/06/23 19:52:58 INFO optim.DistriOptimizer$: Top1Accuracy is Accuracy(correct: 3757, count: 3985, accuracy: 0.9427854454203263)
17/06/23 19:53:00 INFO optim.DistriOptimizer$: [Epoch 20 0/16012][Iteration 153][Wall Clock 328.35500156s] Train 2048 in 1.185880 seconds. Throughput is 1726.9873 records/second. Loss is 0.1897074.
17/06/23 19:53:01 INFO optim.DistriOptimizer$: [Epoch 20 2048/16012][Iteration 154][Wall Clock 329.540881721s] Train 2048 in 1.102029seconds. Throughput is 1846.1228 records/second. Loss is 0.19171959.
17/06/23 19:53:02 INFO optim.DistriOptimizer$: [Epoch 20 4096/16012][Iteration 155][Wall Clock 330.65023375s] Train 2048 in 1.212983seconds. Throughput is 1688.6016 records/second. Loss is 0.17787887.
17/06/23 19:53:03 INFO optim.DistriOptimizer$: [Epoch 20 6144/16012][Iteration 156][Wall Clock 331.863071733s] Train 2048 in 1.387857seconds. Throughput is 1475.6342 records/second. Loss is 0.18688582.
17/06/23 19:53:05 INFO optim.DistriOptimizer$: [Epoch 20 8192/16012][Iteration 157][Wall Clock 333.25094959s] Train 2048 in 1.312171seconds. Throughput is 1560.0063 records/second. Loss is 0.18679756.
17/06/23 19:53:06 INFO optim.DistriOptimizer$: [Epoch 20 10240/16012][Iteration 158][Wall Clock 334.563764761s] Train 2048 in 1.240623seconds. Throughput is 1638.3467 records/second. Loss is 0.23115912.
17/06/23 19:53:07 INFO optim.DistriOptimizer$: [Epoch 20 12288/16012][Iteration 159][Wall Clock 335.813805384s] Train 2048 in 1.287065seconds. Throughput is 1635.7996 records/second. Loss is 0.21648079.
17/06/23 19:53:08 INFO optim.DistriOptimizer$: [Epoch 20 14336/16012][Iteration 160][Wall Clock 337.065792449s] Train 2048 in 1.045743seconds. Throughput is 1922.9221 records/second. Loss is 0.17337748.
17/06/23 19:53:08 INFO optim.DistriOptimizer$: [Epoch 20 14336/16012][Iteration 160][Wall Clock 337.065792449s] Epoch finished. Wall clock time is 345860.270441ms
17/06/23 19:53:08 INFO optim.DistriOptimizer$: [Wall Clock 345.860270441s] Validate model...
17/06/23 19:53:15 INFO optim.DistriOptimizer$: Top1Accuracy is Accuracy(correct: 3743, count: 3985, accuracy: 0.939272273163112)
[moneymall@bdnodeb203 poc]$
```

# SSG.COM BigDL

GPU Memory Resource 가 문제될 때는 BigData Scale Large Cluster Parallel Deep Learning Approach with BigDL

- BigDL Deep Learning Job on Hadoop Yarn Manager (by Spark Job)

The screenshot shows the Hadoop Yarn Manager interface for a cluster named 'bdnodeb201.prod.moneymall.ssgbi.com:8088/cluster/apps/RUNNING'. The user is logged in as 'dr.who'. The main section displays 'RUNNING Applications' with a table of cluster metrics. A blue circle highlights the 'Memory Used' column, showing 311 GB. Another blue circle highlights the application details for 'application\_1493106244744\_0024', which is a 'Text classification' job using 'SPARK' on the 'root.moneymall' queue, running since 'Fri Jun 23 19:34:39 +0900 2017'.

Cluster Metrics																
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	
24	0	1	23	52	311 GB	829 GB	78 GB	103	104	26	13	0	0	0	0	

User Metrics for dr.who														
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved		
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0		

Scheduler Type		Scheduling Resource Type		Minimum Allocation		Maximum Allocation	
Fair Scheduler		[MEMORY, CPU]		<memory:5120, vCores:1>		<memory:26120, vCores:8>	

ID	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1493106244744_0024	moneymall Text classification	SPARK	root.moneymall	Fri Jun 23 19:34:39 +0900 2017	N/A	RUNNING	UNDEFINED		ApplicationMaster

# SSG.COM BigDL

MELTINGCON  
COMMUNITY

ALLIANCE

GPU Memory Resource 가 문제될 때는 BigData Scale Large Cluster Parallel Deep Learning Approach with BigDL

- BigDL Deep Learning Text Classification

```
moneymall@bdnodeb203:/data01/src/moneymall-scala-BigDL/poc
all cLock time is 423454.711522ms
17/06/23 19:42:29 INFO optim.DistriOptimizer$: [Wall Clock 423.454711522s] Validate model...
17/06/23 19:42:33 INFO optim.DistriOptimizer$: Top1Accuracy is Accuracy(correct: 3787, count: 3961, accuracy: 0.9560716990658924)
17/06/23 19:42:34 INFO optim.DistriOptimizer$: [Epoch 20 0/16036][Iteration 305][Wall Clock 423.454711522s] Train 1024 in 0.97112
7371seconds. Throughput is 1054.4446 records/second. Loss is 0.0725722.
17/06/23 19:42:35 INFO optim.DistriOptimizer$: [Epoch 20 1024/16036][Iteration 306][Wall Clock 424.425838893s] Train 1024 in 1.11
0467798seconds. Throughput is 922.1339 records/second. Loss is 0.06961141.
17/06/23 19:42:36 INFO optim.DistriOptimizer$: [Epoch 20 2048/16036][Iteration 307][Wall Clock 425.536306691s] Train 1024 in 1.09
0115128seconds. Throughput is 939.3504 records/second. Loss is 0.0707581.
17/06/23 19:42:37 INFO optim.DistriOptimizer$: [Epoch 20 3072/16036][Iteration 308][Wall Clock 426.626421819s] Train 1024 in 0.99
5664449seconds. Throughput is 1028.4589 records/second. Loss is 0.089765705.
17/06/23 19:42:38 INFO optim.DistriOptimizer$: [Epoch 20 4096/16036][Iteration 309][Wall Clock 427.622086268s] Train 1024 in 1.02
5258522seconds. Throughput is 998.77246 records/second. Loss is 0.07824515.
17/06/23 19:42:39 INFO optim.DistriOptimizer$: [Epoch 20 5120/16036][Iteration 310][Wall Clock 428.64734479s] Train 1024 in 1.137
579537seconds. Throughput is 900.1568 records/second. Loss is 0.0916085.
17/06/23 19:42:40 INFO optim.DistriOptimizer$: [Epoch 20 6144/16036][Iteration 311][Wall Clock 429.784924327s] Train 1024 in 1.09
8894292seconds. Throughput is 931.8458 records/second. Loss is 0.06505428.
17/06/23 19:42:42 INFO optim.DistriOptimizer$: [Epoch 20 7168/16036][Iteration 312][Wall Clock 430.883818619s] Train 1024 in 1.02
8518317seconds. Throughput is 995.60693 records/second. Loss is 0.06360096.
17/06/23 19:42:43 INFO optim.DistriOptimizer$: [Epoch 20 8192/16036][Iteration 313][Wall Clock 431.912336936s] Train 1024 in 1.09
4126942seconds. Throughput is 935.906 records/second. Loss is 0.06956351.
17/06/23 19:42:44 INFO optim.DistriOptimizer$: [Epoch 20 9216/16036][Iteration 314][Wall Clock 433.006463878s] Train 1024 in 1.13
4310924seconds. Throughput is 902.7507 records/second. Loss is 0.057195123.
17/06/23 19:42:45 INFO optim.DistriOptimizer$: [Epoch 20 10240/16036][Iteration 315][Wall Clock 434.140774802s] Train 1024 in 1.0
55282562seconds. Throughput is 970.3562 records/second. Loss is 0.063872576.
17/06/23 19:42:46 INFO optim.DistriOptimizer$: [Epoch 20 11264/16036][Iteration 316][Wall Clock 435.196057364s] Train 1024 in 0.9
83703331seconds. Throughput is 1040.9642 records/second. Loss is 0.06324861.
17/06/23 19:42:47 INFO optim.DistriOptimizer$: [Epoch 20 12288/16036][Iteration 317][Wall Clock 436.179760695s] Train 1024 in 0.9
94621057seconds. Throughput is 1029.5378 records/second. Loss is 0.07590675.
17/06/23 19:42:48 INFO optim.DistriOptimizer$: [Epoch 20 13312/16036][Iteration 318][Wall Clock 437.174381752s] Train 1024 in 1.0
53322129seconds. Throughput is 972.1623 records/second. Loss is 0.05631814.
17/06/23 19:42:49 INFO optim.DistriOptimizer$: [Epoch 20 14336/16036][Iteration 319][Wall Clock 438.227703881s] Train 1024 in 1.0
4527194seconds. Throughput is 979.64935 records/second. Loss is 0.084993005.
17/06/23 19:42:50 INFO optim.DistriOptimizer$: [Epoch 20 15360/16036][Iteration 320][Wall Clock 439.272975821s] Train 1024 in 0.9
82663203seconds. Throughput is 1042.066 records/second. Loss is 0.06305286.
17/06/23 19:42:50 INFO optim.DistriOptimizer$: [Epoch 20 15360/16036][Iteration 320][Wall Clock 439.272975821s] Epoch finished. W
all cLock time is 444659.473167ms
17/06/23 19:42:50 INFO optim.DistriOptimizer$: [Wall Clock 444.659473167s] Validate model...
17/06/23 19:42:54 INFO optim.DistriOptimizer$: Top1Accuracy is Accuracy(correct: 3763, count: 3961, accuracy: 0.950012623074981)
[moneymall@bdnodeb203 poc]$ ls
```



# SSG.COM TensorflowOnSpark

## Tenosrflow 의 Legacy 를 활용한 BigData Scale Deep Learning 은?

- TensorflowOnSpark 수행 결과

```
money@mail@bdnodeb203:/data01/src/tensorflowOnSpark-src/mnist
17/08/31 21:20:53 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:20:54 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:20:55 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:20:56 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:20:57 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:20:58 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:20:59 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:21:00 INFO yarn.Client: Application report for application_1493106244744_0048 (state: RUNNING)
17/08/31 21:21:01 INFO yarn.Client: Application report for application_1493106244744_0048 (state: FINISHED)
17/08/31 21:21:01 INFO yarn.Client:
  client token: N/A
  diagnostics: N/A
  ApplicationMaster host: 10.203.5.191
  ApplicationMaster RPC port: 0
  queue: root.money@mail
  start time: 1504181998298
  final status: SUCCEEDED
  tracking URL: http://bdnodeb201.prod.money@mail.ssgbi.com:8088/proxy/application_1493106244744_0048/
  user: money@mail
17/08/31 21:21:01 INFO util.ShutdownHookManager: Shutdown hook called
17/08/31 21:21:01 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-bf4a7862-d2c5-4093-ae5-93afd90522ab
[money@mail@bdnodeb203 ~]$
```

# BigDL vs TensorflowOnSpark

## What is BigDL

BigDL is a distributed deep learning library for Apache Spark; with BigDL, users can write their deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters.

- **Rich deep learning support.** Modeled after [Torch](#), BigDL provides comprehensive support for deep learning, including numeric computing (via [Tensor](#)) and high level [neural networks](#); in addition, users can load pre-trained [Caffe](#) or [Torch](#) or [Keras](#) models into Spark programs using BigDL.
- **Extremely high performance.** To achieve high performance, BigDL uses [Intel MKL](#) and multi-threaded programming in each Spark task. Consequently, it is orders of magnitude faster than out-of-box open source [Caffe](#), [Torch](#) or [TensorFlow](#) on a single-node Xeon (i.e., comparable with mainstream GPU).
- **Efficiently scale-out.** BigDL can efficiently scale out to perform data analytics at "Big Data scale", by leveraging [Apache Spark](#) (a lightning fast distributed data processing framework), as well as efficient implementations of synchronous SGD and all-reduce communications on Spark.

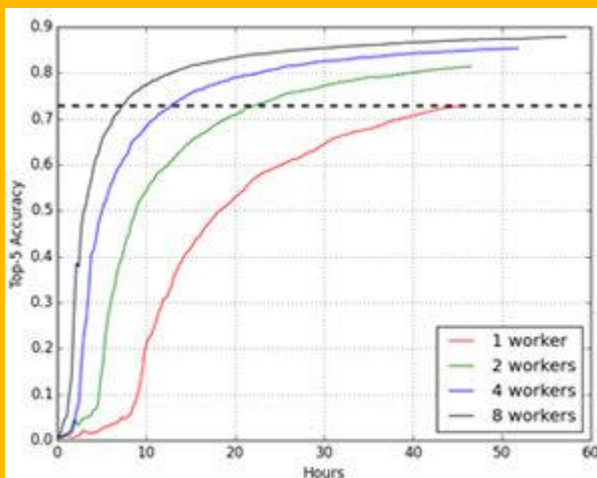
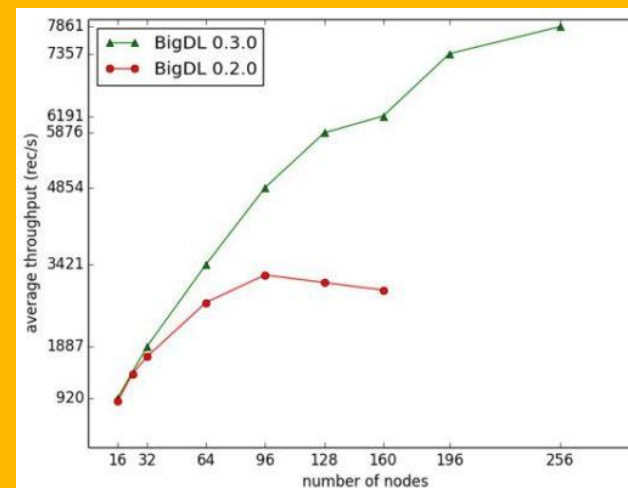


Figure 4: TFoS training of inception networks

## Why TensorFlowOnSpark?

TensorFlowOnSpark provides some important benefits (see [our blog](#)) over alternative deep learning solutions.

- Easily migrate all existing TensorFlow programs with <10 lines of code change;
- Support all TensorFlow functionalities: synchronous/asynchronous training, model/data parallelism, inferencing and TensorBoard;
- Server-to-server direct communication achieves faster learning when available;
- Allow datasets on HDFS and other sources pushed by Spark or pulled by TensorFlow;
- Easily integrate with your existing data processing pipelines and machine learning algorithms (ex. MLLib, CaffeOnSpark);
- Easily deployed on cloud or on-premise: CPU & GPU, Ethernet and Infiniband.

# Batch Size에 대하여 (무조건 mini Batch가 좋은가???)

## Single Host GPU vs Multi Host GPU vs Super Multi Host CPU

```
Epoch 18/200
34447/34447 [=====] - 71s 2ms/step - loss: 0.1010 - acc: 0.9873 - val_loss: 0.0797 - val_acc: 0.9762
Epoch 19/200
34447/34447 [=====] - 71s 2ms/step - loss: 0.0999 - acc: 0.9873 - val_loss: 0.0859 - val_acc: 0.9760
Epoch 20/200
34447/34447 [=====] - 70s 2ms/step - loss: 0.1017 - acc: 0.9877 - val_loss: 0.0855 - val_acc: 0.9767
Epoch 21/200
34447/34447 [=====] - 70s 2ms/step - loss: 0.0989 - acc: 0.9877 - val_loss: 0.0842 - val_acc: 0.9765
Epoch 22/200
34447/34447 [=====] - 71s 2ms/step - loss: 0.0972 - acc: 0.9882 - val_loss: 0.0841 - val_acc: 0.9774
```

- Single Host GPU : 64 batch size. 71초 \* 22 epoch(Auto Early Stopping) = 1562 초

- Multi Host GPU : 1024 batch size. 6초 \* 58 epoch(Auto Early Stopping) = 348 초

그렇다면,  
Super Multi  
Host CPU 클  
러스터는?

```
Epoch 53/200
34447/34447 [=====] - 6s 166us/step - loss: 0.0899 - acc: 0.9882 - val_loss: 0.0725 - val_acc: 0.9779
Epoch 54/200
34447/34447 [=====] - 6s 166us/step - loss: 0.0902 - acc: 0.9878 - val_loss: 0.0730 - val_acc: 0.9776
Epoch 55/200
34447/34447 [=====] - 6s 166us/step - loss: 0.0911 - acc: 0.9878 - val_loss: 0.0730 - val_acc: 0.9774
Epoch 56/200
34447/34447 [=====] - 6s 168us/step - loss: 0.0899 - acc: 0.9869 - val_loss: 0.0738 - val_acc: 0.9781
Epoch 57/200
34447/34447 [=====] - 6s 166us/step - loss: 0.0909 - acc: 0.9880 - val_loss: 0.0734 - val_acc: 0.9766
Epoch 58/200
34447/34447 [=====] - 6s 166us/step - loss: 0.0875 - acc: 0.9880 - val_loss: 0.0731 - val_acc: 0.9782
```

그런데...

BigData Scale  
Deep Learning Training

TensorflowOnSpark 는 Parallel Inference 도 제공하지만...  
Batch System 으로는 적당할 것 같은데...

그런데...

Parallel Inference 의 Production 에 대한 고민...

개인화하는? (Not batch, On the fly Model)

Model 에 즉각적인 최신성 반영은?

운영 중 즉각적인 무 중지 배포는?

Auto Scale Out / Auto Scale Down 은?

# BigData Scale Deep Learning!

## Drill down~~

4. 최근에 시도한 방식(serverless public cloud)

# 국내 상위 쇼핑몰 정도 트래픽, 요즘 트렌디한 개인화 추천 정도 하려고 해도...

- 더 이상 Analytics 자체는 경쟁력이 아니다.
- 더 이상 Machine Learning 자체는 진입 장벽이 아니다.
- Deep Learning 은 아직까지는 경쟁력이나, 점점 툴이 막강해지고, 쉬워지고 있다.
- BigData Scale Machine Learning 도 경쟁력이다. (최신성 + Big Scale + 개인화)
- 더 더 어려운 문제는 **Advanced Analytics** 의 결과물을 **개인화** 하여 노출하고, 그 **반응**을 다시 **실시간**으로 모델에 리턴 하여, 그로부터 즉각적인 **선순환 시스템**을 만든 것이다. (예, 고객 실시간 검색 히스토리, 클릭 스트림 에 반응하는 Dynamic 한 개인화 추천. 개인화 된 Dynamic Pricing 모델, Enterprise Full Stack Chatbot 등)
- **Machine Learning / Deep Learning 시스템에 있어서 Serving Layer 의 중요성**

BigData + MicroService + DevOps + Auto Scale Out Training + Auto Scale Out Serving 은  
목표 Motive 를 달성하기 위해서는, 선택이 아닌 필수

국내 상위 쇼핑몰 정도 트래픽,  
요즘 트렌디한 개인화 추천 정도 하려고 해도...

- RDBMS -> **현존 최고 HW 로 Oracle 5 Node 로도...**



- MariaDB 샤딩 시스템. 수천대. Select 부하분산.
- **큰 테이블 Data 는? Insert & Update 는?**



- NoSQL + RDBMS 샤딩(R) + RDBMS Master for (CUD)  
+ Kafka (Message Queue for Data Sync)
- **복잡한 RDBMS 로직 SQL => NoSQL 로 마이그레이션 불가???**



- NoSQL + RDBMS 샤딩(R) + RDBMS Master for (CUD)  
+ Kafka (Message Queue for Data Sync) + Redis(Memory Cache)  
+ MicroService (Scale Out WAS for **복잡한 Business Logic** 의 Application layer 처리 )  
+ Machine Learning/Deep Learning Training/Serving Layer Auto Scale Out Infra 구성  
(예, Tensorflow Serving PaaS 혹은 flask Serverless Microservice) + 기타(Docker, DevOps...)

Training Layer  
와,  
Serving Layer  
는 아키텍처가  
또 다름.



우리가 추가로 했던 시도들...

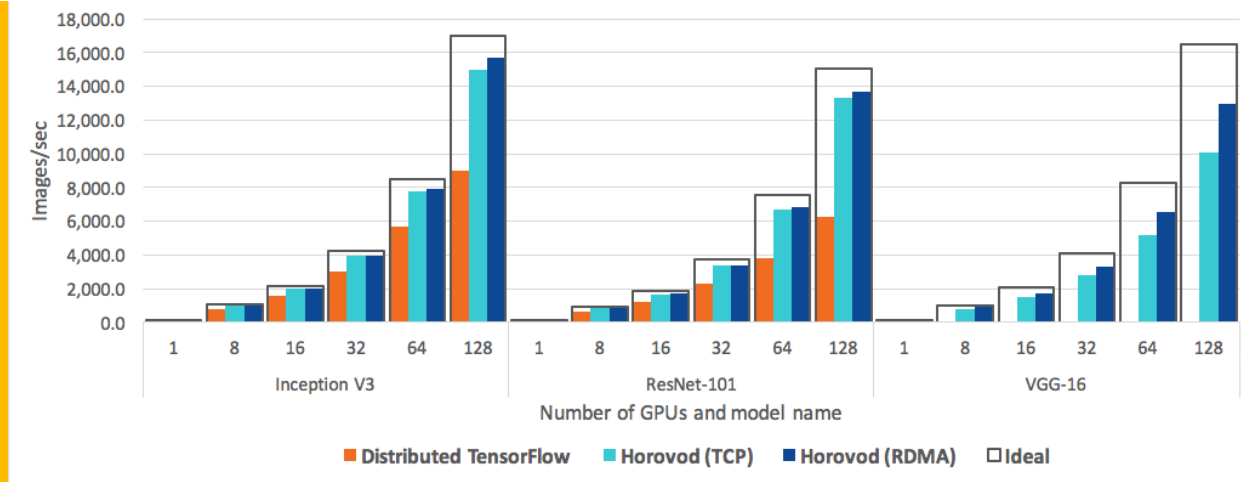
# Horovod(by uber) with Tensorflow, Keras

## Why not traditional Distributed TensorFlow?

The primary motivation for this project is to make it easy to take a single-GPU TensorFlow program and successfully train it on many GPUs faster. This has two aspects:

1. How much modifications does one have to make to a program to make it distributed, and how easy is it to run it.
2. How much faster would it run in distributed mode?

Internally at Uber we found the MPI model to be much more straightforward and require far less code changes than the Distributed TensorFlow with parameter servers. See the [Usage](#) section for more details.



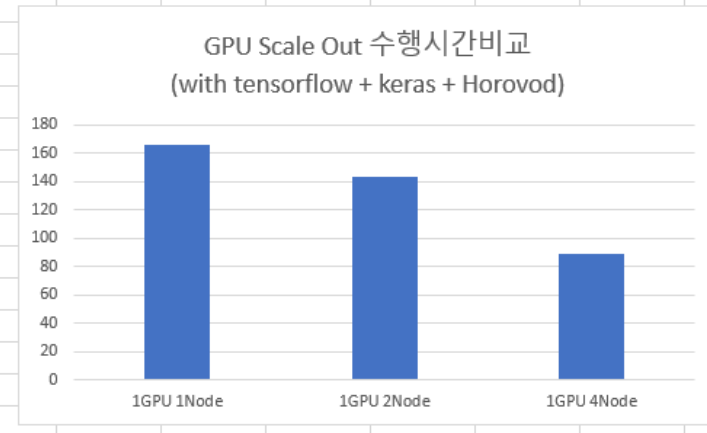
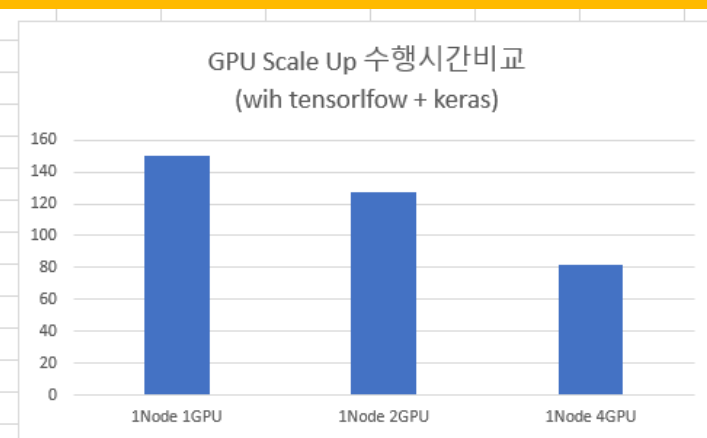
# Keras Scale Up vs. Horovod Scale Out

[http://hoondongkim.blogspot.kr/2018/01/deep-learning-multi-host-multi-gpu\\_11.html](http://hoondongkim.blogspot.kr/2018/01/deep-learning-multi-host-multi-gpu_11.html)

GPU Scale Up (with tensorflow + Keras)	
	수행시간
1Node 1GPU	150
1Node 2GPU	127
1Node 4GPU	82

GPU Scale Out (with tensorflow + keras + Horovod)	
	수행시간
1GPU 1Node	166
1GPU 2Node	143
1GPU 4Node	89



# Deep Learning Online Inference CPU vs GPU (1 time Inference. BatchSize=1)

<http://hoondongkim.blogspot.kr/2017/12/deep-learning-inference-serving.html>

## 동접 테스트

CPU Serving : 1 Machine 265 TPS

GPU Serving : 1 Machine 16 TPS

### CPU 1000 Times Serial Execution

```
In [40]: %%time
for i in range(0,1000):
    model.predict_classes(predictTokenizer('비밀 번호를 잃어버렸어요 그래서 로그인이 안되요'),verbose=0)
executed in 21.6s, finished 05:49:06 2017-12-16
CPU times: user 1min 7s, sys: 20.6 s, total: 1min 28s
Wall time: 21.6 s
```

```
In [41]: %%time
token = predictTokenizer('비밀 번호를 잃어버렸어요 그래서 로그인이 안되요')
for i in range(0,1000):
    model.predict_classes(token,verbose=0)
executed in 20.0s, finished 05:49:32 2017-12-16
CPU times: user 1min 4s, sys: 21 s, total: 1min 25s
Wall time: 20 s
```

### GPU 1000 Times Serial Execution

```
In [46]: %%time
for i in range(0,1000):
    model.predict_classes(predictTokenizer('비밀 번호를 잃어버렸어요 그래서 로그인이 안되요'),verbose=0)
CPU times: user 58.3 s, sys: 2.06 s, total: 1min
Wall time: 59 s
```

```
In [45]: %%time
token = predictTokenizer('비밀 번호를 잃어버렸어요 그래서 로그인이 안되요')
for i in range(0,1000):
    model.predict_classes(token,verbose=0)
CPU times: user 57.8 s, sys: 1.96 s, total: 59.8 s
Wall time: 58.2 s
```

# Deep Learning Online Inference CPU vs GPU (1 time Inference. BatchSize=1)

<http://hoondongkim.blogspot.kr/2017/12/deep-learning-inference-serving.html>

**1. Async + Auto Scale Out  
을 추가!**

**2. 속도가 받쳐주면, Model 을 Fix 하지 않아도 된다.**

**3. [궁극에는] Online Training + Realtime Inference 를  
한번에**

# Engineering Art!

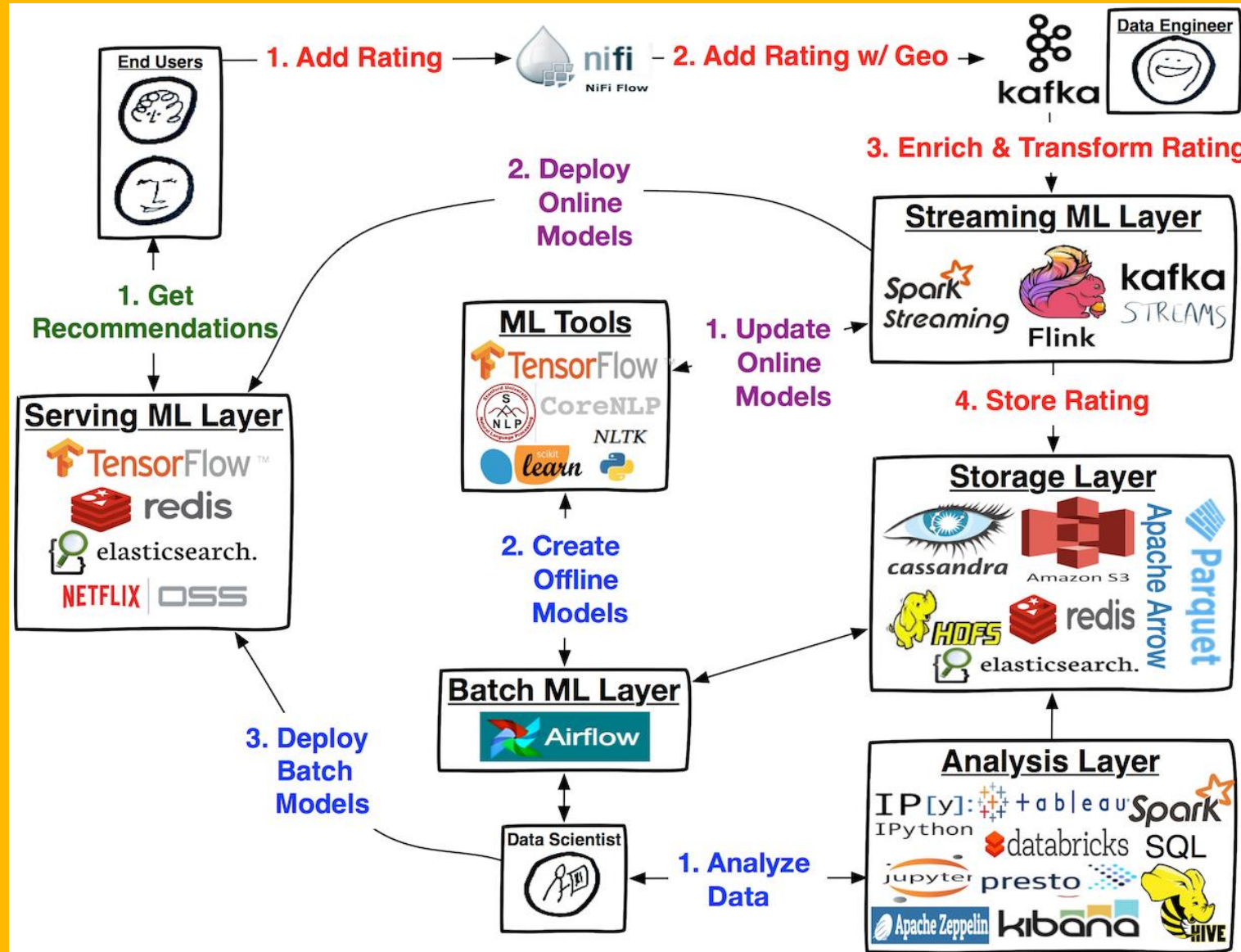
Async & Auto Scale Out

for

양방향 Inference Layer!

Inference Layer 가 사용자 세션 타임 내에 원순환 구조로 Online Training???

# Engineering Art!



# Auto Scale Out Deep Learning Training On Cloud PaaS

<http://hoondongkim.blogspot.kr/2018/01/deep-learning-multi-host-multi-gpu.html>

Tensorflow + Keras + Horovod + Azure Batch AI

The image is a collage illustrating a deep learning training setup on a cloud PaaS. It features three main components:

- Jupyter Notebook (Left):** A screenshot of a Jupyter notebook titled "SSG.COM Deep Learning-Tensorflow-Keras-Horovod Model-on-Batch AI Training-Multi Host & GPU". The notebook shows the execution of a training script with various parameters and the resulting output, including training progress, loss, and accuracy metrics. The output shows a successful fit of the SSG.COM Deep Learning Model.
- Distributed Training by Cluster (Center):** A diagram titled "Distributed Training by Cluster" showing two "computing node01" and "computing node02" connected to "Script and Input Data" and "Output Model" stored in "NFS Shared Storage". The nodes are labeled as "mounted", indicating they have access to the shared storage.
- Azure Batch AI Console (Right):** A screenshot of the Azure Batch AI console interface. It displays a list of jobs, including "horovod\_keras\_01\_26\_2018\_071121". The console shows the job's execution state as "succeeded", with a duration of 4h 5m 57.28s. It also provides cluster statistics, including the number of nodes (1) and the number of jobs (55).



# Serverless Architecture AI Service 구성

Serverless Architecture ( For BigData Scale Deep Learning AI Production On Azure )

- **BaaS (Backend as a Service)**
  - **Web App** : Web 및 Admin 관리
  - **Web App for Linux Container** : Container 를 커스터마이징 할 수 있으나, Container 관리는 위임.
  - **Azure Batch AI**
  - **Azure Batch**
- **FaaS (Function as a Service)**
  - **Azure Function**
  - **Logic App**
- **기타, Serverless PaaS 및 SaaS**
  - **Cosmos DB**
  - **Application Insight**
  - **Container Registry**
  - **Conatiner Web Hook**
- **CI/CD PaaS**
  - **Team Service**
  - **Github Enterprise**

# Why Not Tensorflow Serving?

## 1. Over the GRPC

1. 이를 Web Service 화 하기 위해서는 별도의 Tier 가 필요하다.
2. 이에 대한 방법론은 뒤에서 다시 언급...

## 2. Over the Tensorflow. ( Keras , CNTK , Spark ML , BigDL, Deep Learning 4j, etc ...)

1. 물론 Keras 를 Serialize 및 Tensorflow Graph 로 Model Export 하고, 이를 로딩하는 Client 를 만드는 경우 Keras 를 Tensorflow Serving 하는 것이 불가능 하진 않음.

## 3. Tensorflow Serving is only published for Python2

1. 물론 수동 빌드로 Python3에서 구동하는 것이 불가능하진 않지만...

## 4. Training 과 Serving 을 동시에??

1. 물론 이 시나리오는 Serving 최적화를 포기해야만 가능.
2. Scale Out 모델에서, 성능이 어느 정도 나온다면...

On-premise Docker 가 아닌 Serverless PaaS 상의 자동 관리 Docker 라고 한다면, 구성이 무겁거나 복잡한 것보다, 가볍고 단순하며, Debug가 유리한 것이 더 현실적임.

## SSG닷컴, MS 애저 기반 AI 챗봇 서비스 구축

한편 이번 챗봇 서비스는 신세계에서 오픈소스 기술로 자체 개발한 AI 모델을 적용해 개발했다. 이는 MS 클라우드 서비스 '애저'에서 제공되는 다양한 플랫폼 기술을 활용해 가동된다. 고객 접점에서 사용자 요청을 1차 처리하는 부분에는 서버리스 기술인 애저 펄션, AI 트레이닝에 필요한 GPU 자원과 작업을 관리하는 기술은 애저 배치 AI 서비스, 사용자 질문의 의도를 분석하고 처리하는 작업은 애저 PaaS와 도커 기술 상에서 구현됐다.

## "사진만 찍었는데 잇템 찾아준다"... 신세계몰, 이미지 검색 서비스 '쓱렌즈' 시작

달리닝 이미지 검색, 1:1챗봇 등 유통업계에서도 인공지능 바람

진범용 기자 프로필보기 | 최종편집 2018.05.08 09:21:19



▲쓱렌즈 화면 캡처, ©신세계백화점

고객상담
✕

궁금한건  
**24시간**<sup>Ⓞ</sup>  
언제든지  
쓱 -  
물어보세요!

**Beta** 고객상담 챗봇

배송, 교환, 환불  
문의까지 챗봇으로  
빠르고 간편하게~

고객상담사

전문상담사의  
친절하고 꼼꼼한  
답변이 필요하다면!

고객상담 문의하기 >

친구
대화
선물
셀러
고객상담

# 성능에 대하여 ...

http://hoondongkim.blogspot.kr/2017/12/deep-learning-inference-serving.html

Azure Data Science VM	12 vCore CPU + 112gb Memory * 1 VM	Tesla K80 + 2 GPU + 112gb Memory * 1 VM	
Flask + tensorflow	270 TPS	16 TPS	
Flask + ***** + tensorflow	883 TPS	17 TPS	
**** + Flask + **** + ***** + tensorflow	2633 TPS	15 TPS	
Azure Web App for Linux Docker	4 vCore * 1 Docker	4 vCore * 5 Docker	4 vCore * 10 Docker
Flask + tensorflow	132 TPS	634 TPS	1281 TPS
Flask + ***** + tensorflow	485 TPS	2260 TPS	4627 TPS
**** + Flask + **** + ***** + tensorflow	1207 TPS	5538 TPS	11529 TPS

g 9

받은편지함으로 이동

한국어 ▼ 메일 번역

Microsoft  
Azure

**i** Autoscale successfully started scale operation for resource 'ssg-bigdata-ai-serving-plan-linux-prod' from capacity '10' to capacity '9'

You may view more details in the [Microsoft Azure Management Portal](#).

RESOURCE ID:  
/subscriptions/bd9717a4-bfa5-4695-99a5-9a34fa78038e/resourceGroups/

[Gmail]/INBOX/계...

라벨 더보기 ▼

- 훈동 ▼ +
- Seungjin Kim 나: ○ ○
- 승혜 한승혜 안녕하세요, 단국대학.
- 현수님, Seungjin님 나: 아니 그냥 꺾꺾이.

# Conclusion!

결론!

# [1]~[3] 각 레이어의 관계는?

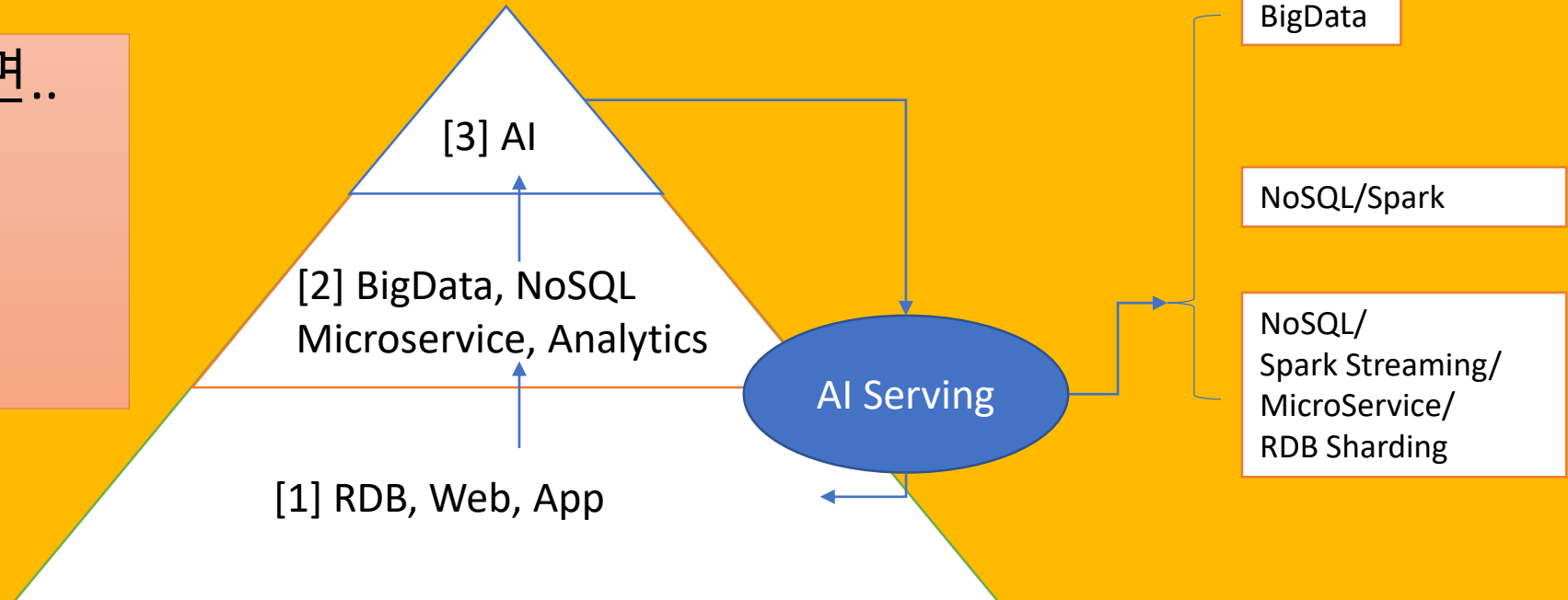
- [1] RDB Sharding, Web, App, Front Tech... 는 기본 중에 기본.
- [2] BigData, NoSQL, Microservice, Analytics 는?
- [3] AI(Machine Learning, Deep Learning) 는?

각각 경쟁관계가 아님.  
상호 보완 관계이고,  
장단점이 극명 함.

Production 으로 가면..

종합 예술이다!

Engineering Art 다!



# 다 직접 개발하는 것이 능사는 아니다!

집단 지성의 힘!  
검증된 것들을 Mesh Up!  
핵심경쟁력이 아닌 것은 빠르게 하는 것이 더 좋음....  
(Over Engineering 방지)

## (Open Source + Cloud PaaS)

요즘 가장 Hot 한 Develop 방식이 추구하는 것들...

- No-Ops (or Dev Ops)
- Scale
- Low Cost
- Performance

# AI 시대에 임하는 자세!

- Think Big.
  - 길게 보고 Plan을 세우자.
  - 당장의 효과에 연연하지 말자.
- Bottom Up! ( Not Top Down! )
  - 유행에 편승해서, 혹은 윗 분들의 지시에 의해서가 아닌,
  - 실무자들에 의해 기획하고,
  - 내부 개발자 내재화를 고려하며 개발하도록 하자.
- 작게 그리고 빠르게 시작하자.
  - 금년에 시작한다면, 10살 수준이고, 2~3년이 지나야 15살 수준이 될 수도 있지만, 1~2년 뒤에 시작하면, 평생 경쟁사보다 동생일 수 있음.
- Mesh Up 하고, 결합 Merge 하고, 반복 개선 시키자.

Serverless PaaS , Cloud 의 많은 기술을 응용하면, AI도 BigData 도, BigData가 결합된 AI도, INFRA, Framework 의 많은 부분을 추상화 하여, 빠른 진입이 가능하다.



# Thank You

- 기타 문의는...
- <http://hoondongkim.blogspot.kr>
- <https://www.facebook.com/kim.hoondong>