Machine Learning on Spark

He Yunlong

SSG/DRD/PRC

Agenda

Machine Learning in Today

Introduction to Spark

Machine Learning on Spark

Large Scale Neural Network

Optimization

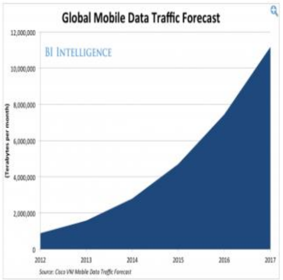
Q & A

Machine Learning in Today

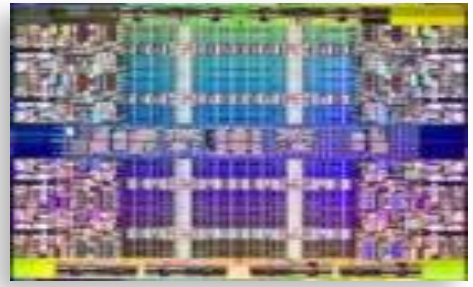
Old Time: Before Mobile Internet Booming Era: Data Scale is small :

- Shallow analysis is enough, no need complex algorithm
- Limited data set could not get precise training model.
- Computing Capability is limited by technology

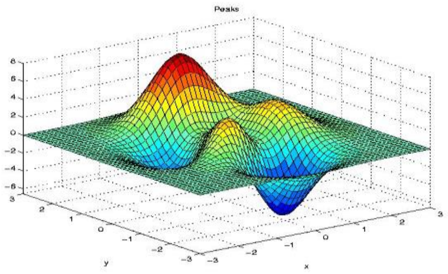
Today : Big Data laid a foundation of Machine Learning:

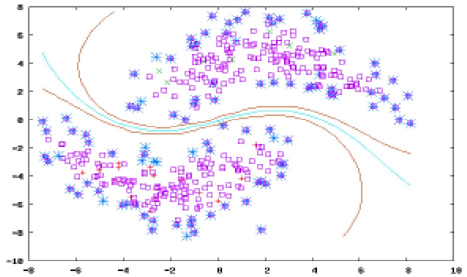
The widespread of smart phones and the development of IoT provides comprehensive data sources for **Big data**.



In the past couple of decades, **Computing Power** is growing exponentially by following the Moore's Law



With the rapid development of science and technology, more **Complex Models** are extracted, built-up and deployed in industry



More and more **Efficient Algorithms** of Machine Learning are researched and developed by scientists and domain experts

Data are the Greatest Strategic Resources for Internet Companies

ML: Big data + Computing Power + Complex Model + Efficient Algorithm

- **Create user experiences**
- **Create commercial values**

Machine learning Phase - 1

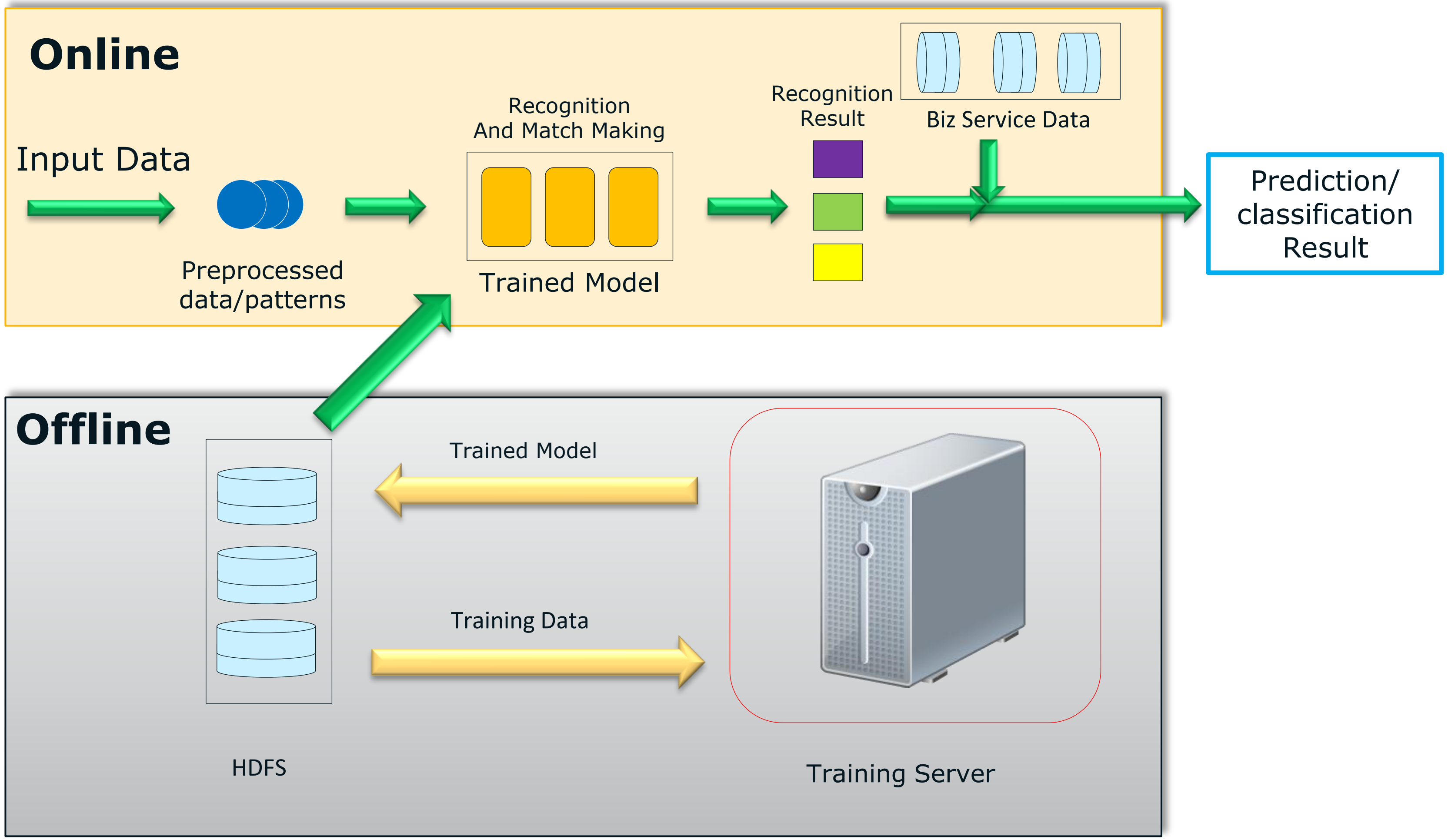
Model small

Dataset small

Training time minutes ~ hours

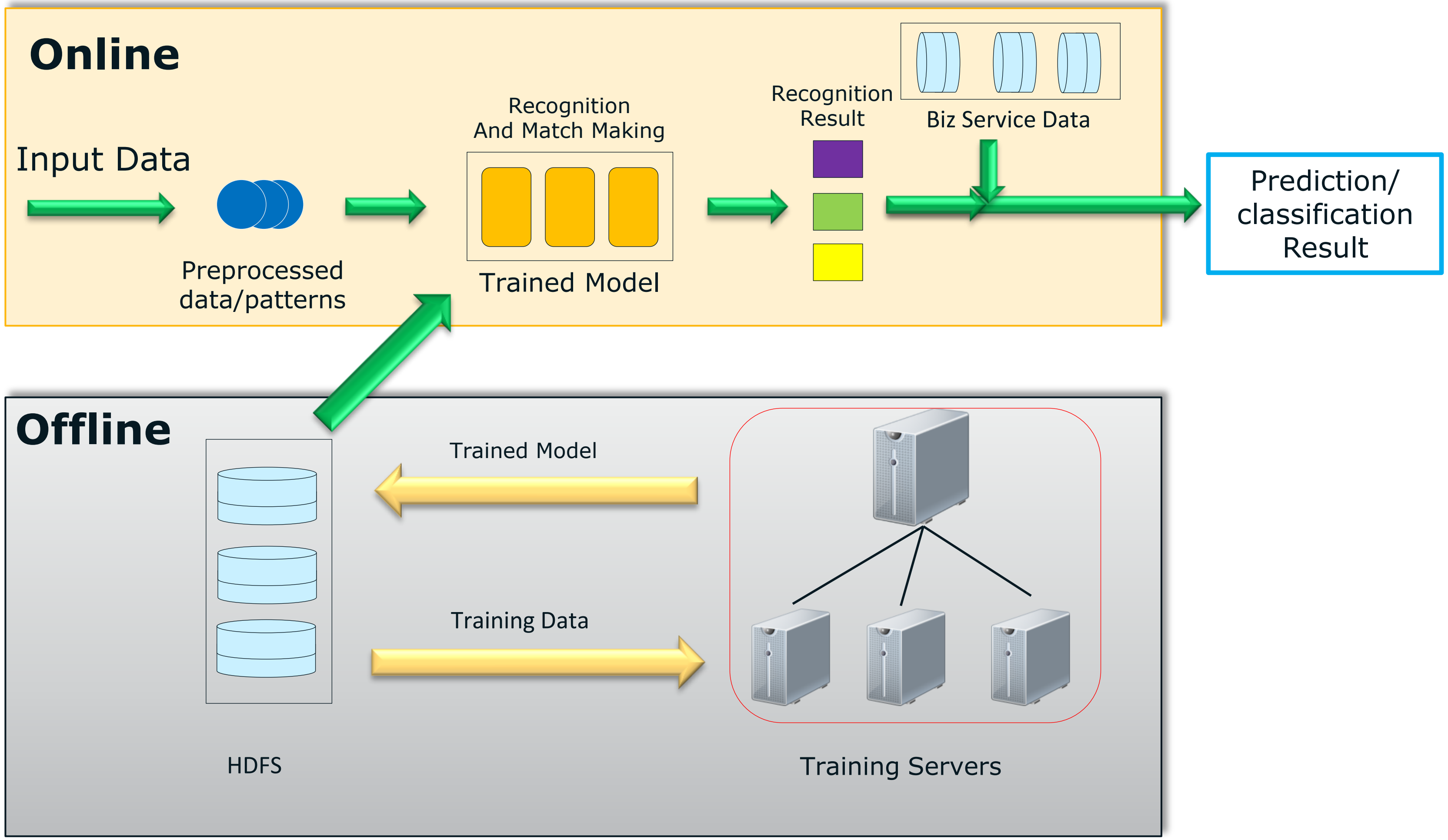
Tools Matlab, R, Python …

Services Junk Detect, Association



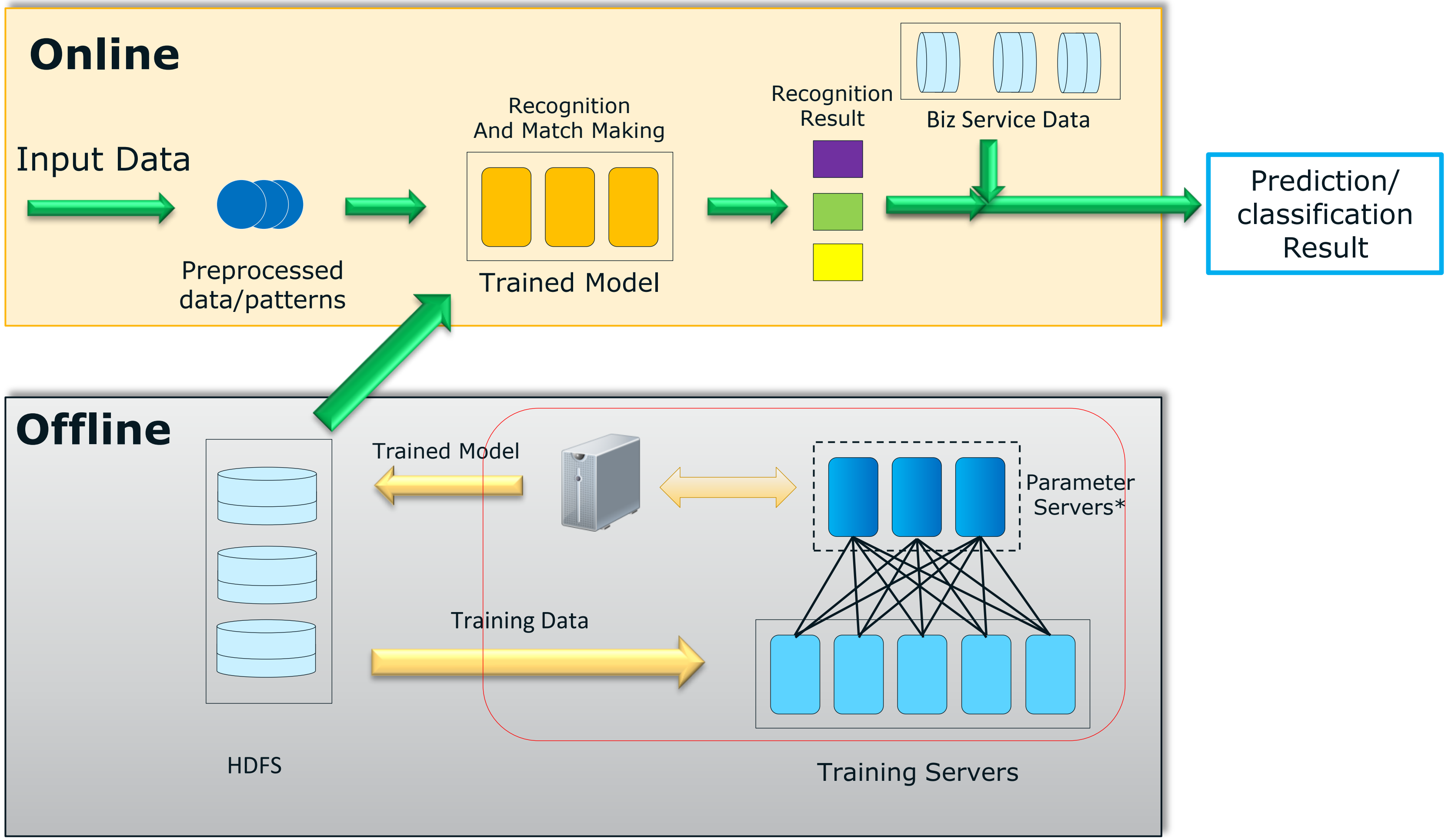
Machine learning Phase - 2

Model	Small
Dataset	big
Training time	minutes ~ days
Tools	Mahout, Mllib, …
Services	CTR, Doc Classify



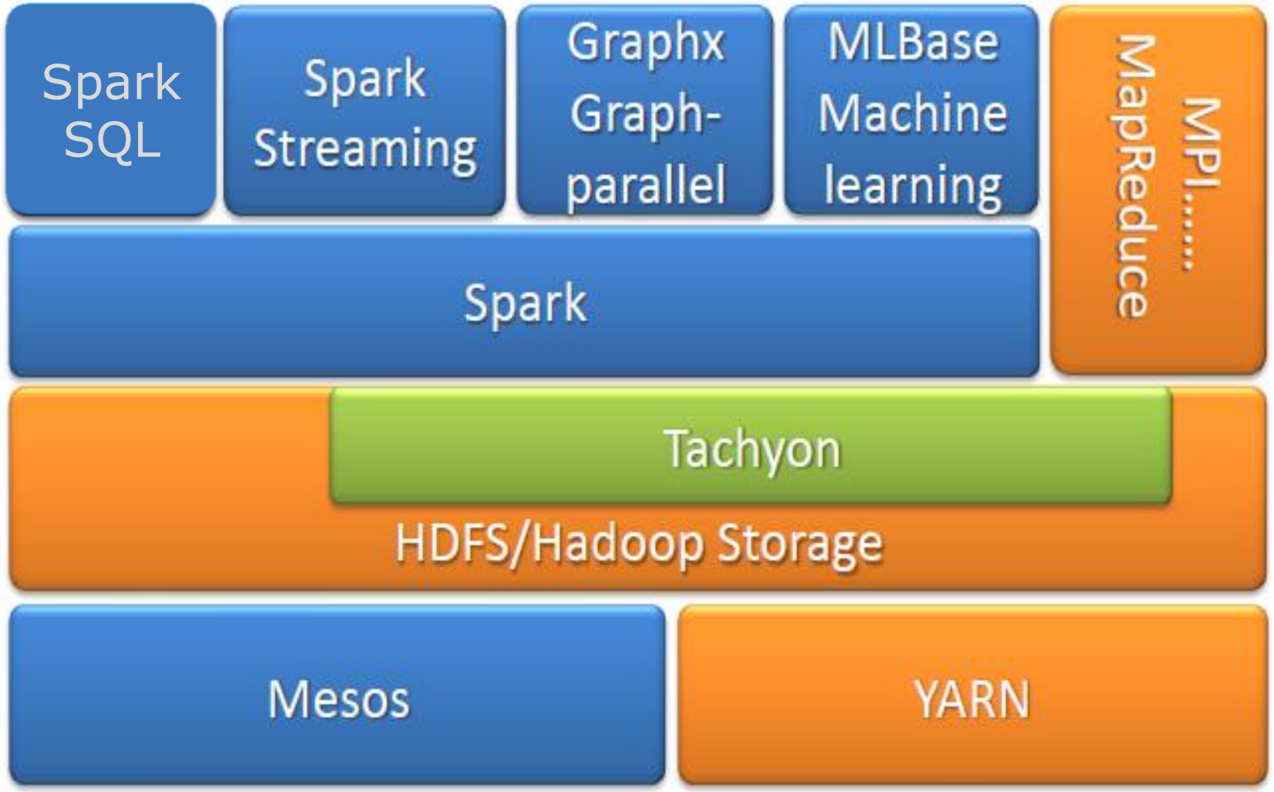
Machine learning Phase - 3

Model	big
Dataset	huge
Training time	minutes ~ days
Tools	ParamServer, DistBelief ...
Services	Speech Recog, Image Search



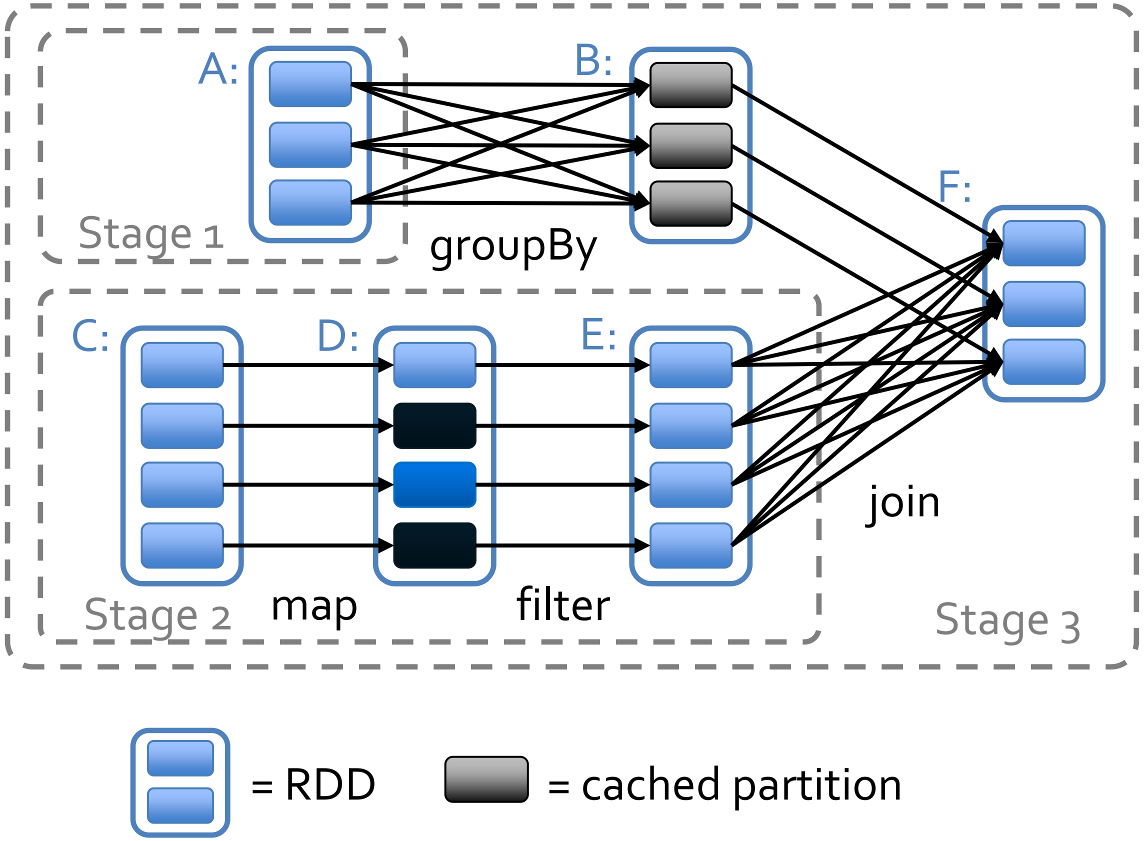
About Spark

- Fast In-Memory data analytics cluster computing framework
- Originally developed in the AMPLab, became an Apache Top-Level Project in February 2014
- Suitable for Iterative tasks
- Proven scalability to 2000 nodes in the research lab on EC2 and 1000 nodes in production.



Spark - Program Model： RDD

- General task graphs
- Automatically pipelines functions
- Data locality aware
- Partitioning aware to avoid shuffles



Spark - Program Model : RDD

Base RDD

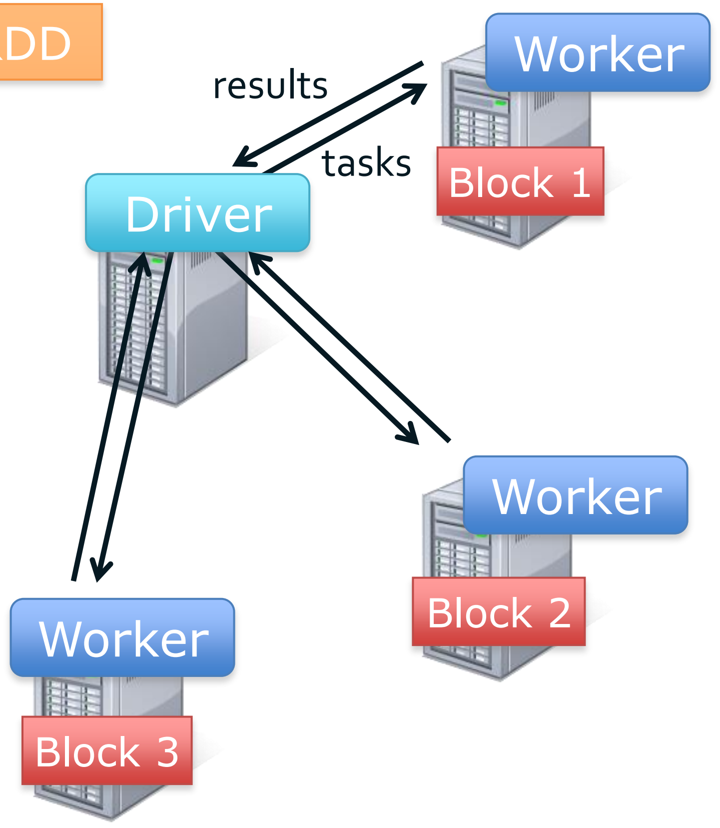
Transformed RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()
```

Cached RDD

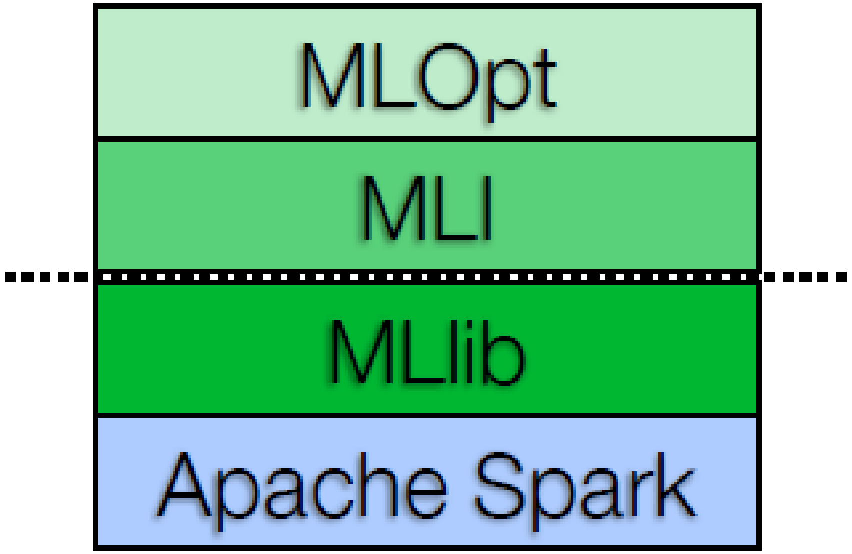
```
cachedMsgs.filter(_.contains("foo")).count
cachedMsgs.filter(_.contains("bar")).count
. . .
```

Parallel operation

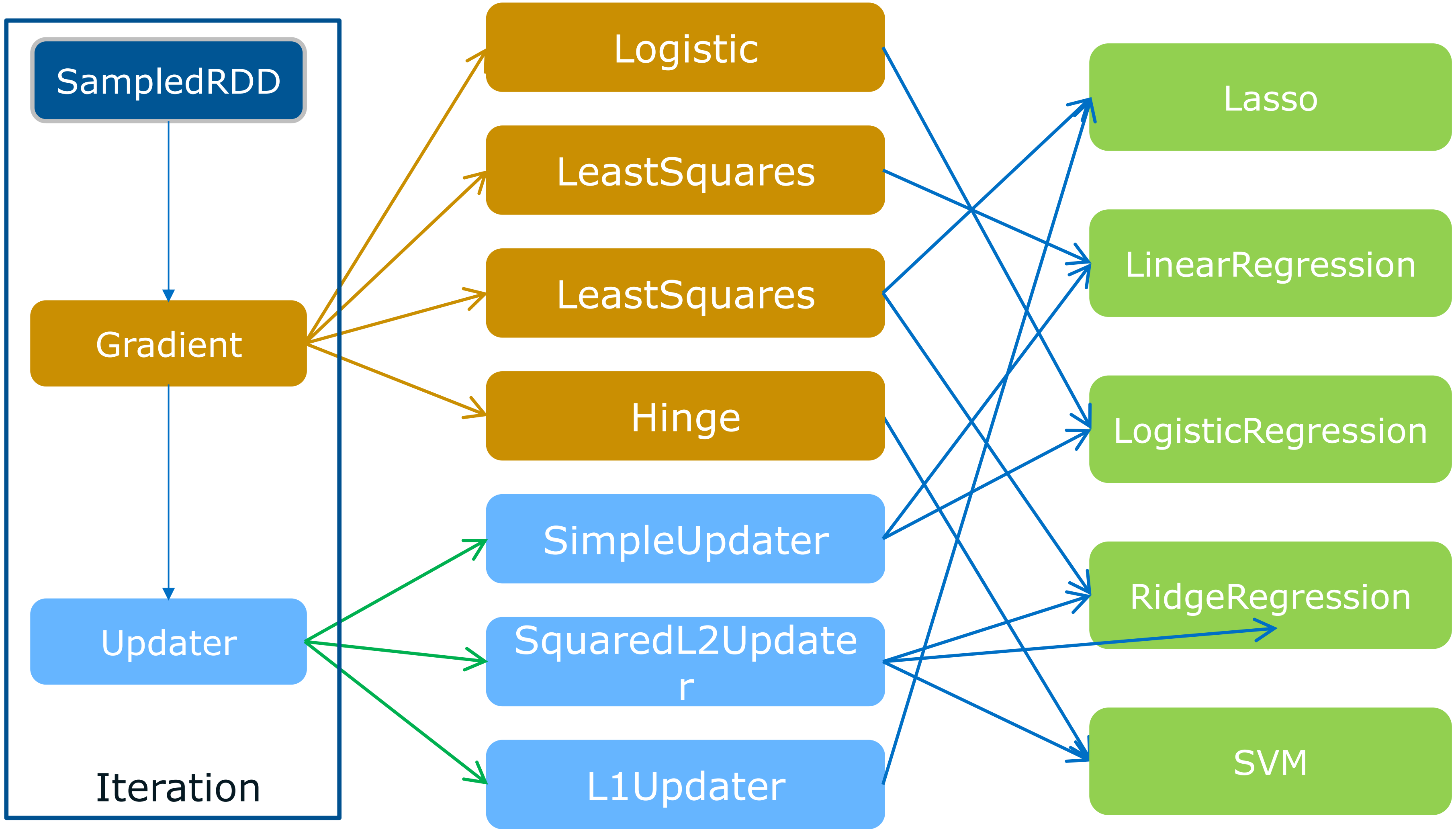


Mllib: Machine Learning on Spark

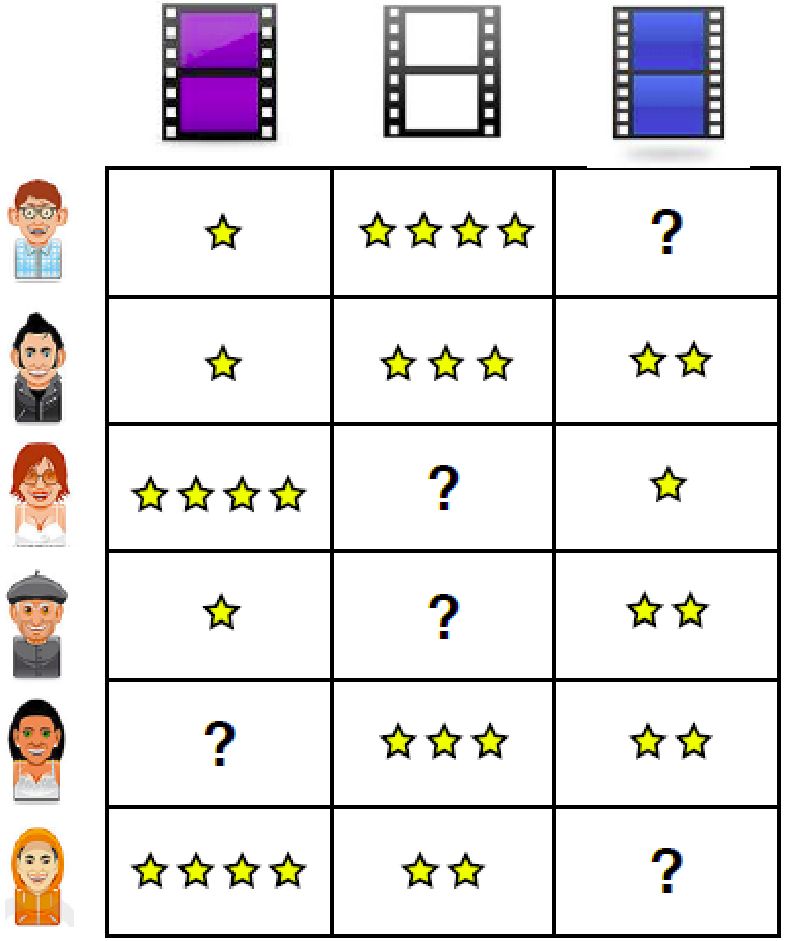
- **Classification**
  - logistic regression, linear support vector machine(SVM), naive Bayes, classification tree
- **Regression**
  - generalized linear models (GLMs), regression tree
- **Collaborative filtering**
  - alternating least squares (ALS)
- **Clustering**
  - k-means
- **Decomposition**
  - singular value decomposition (SVD), principal component analysis (PCA)



Mllib – Regression

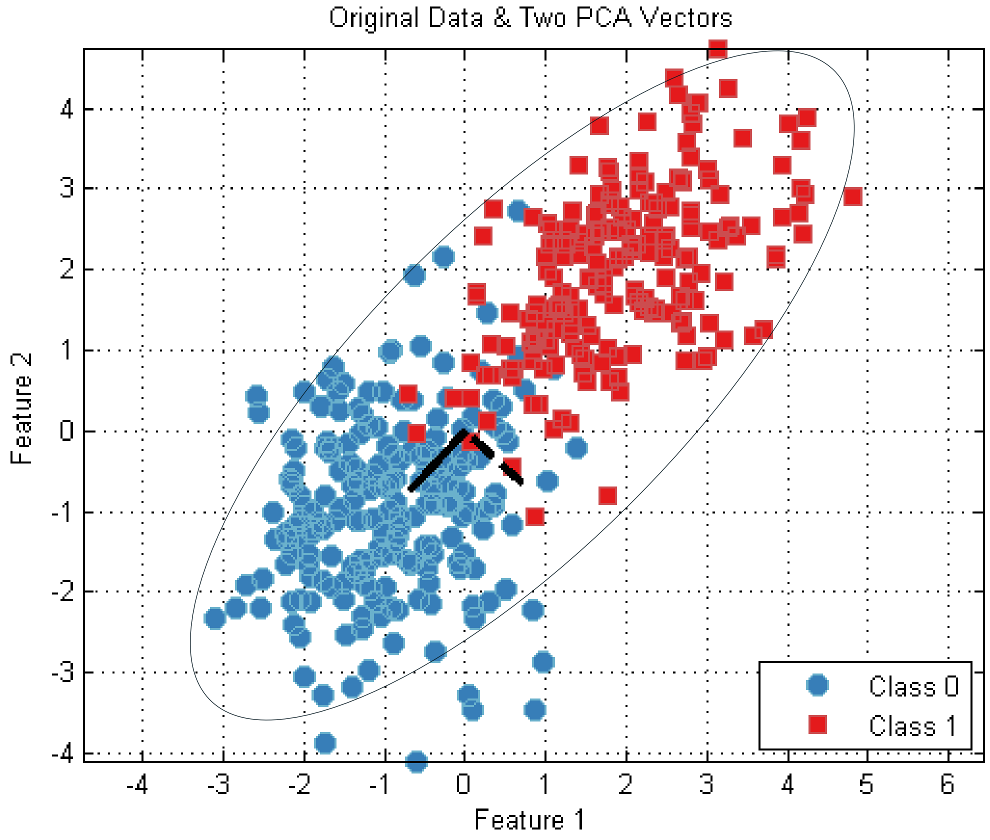


Mllib – Collaboration Filter



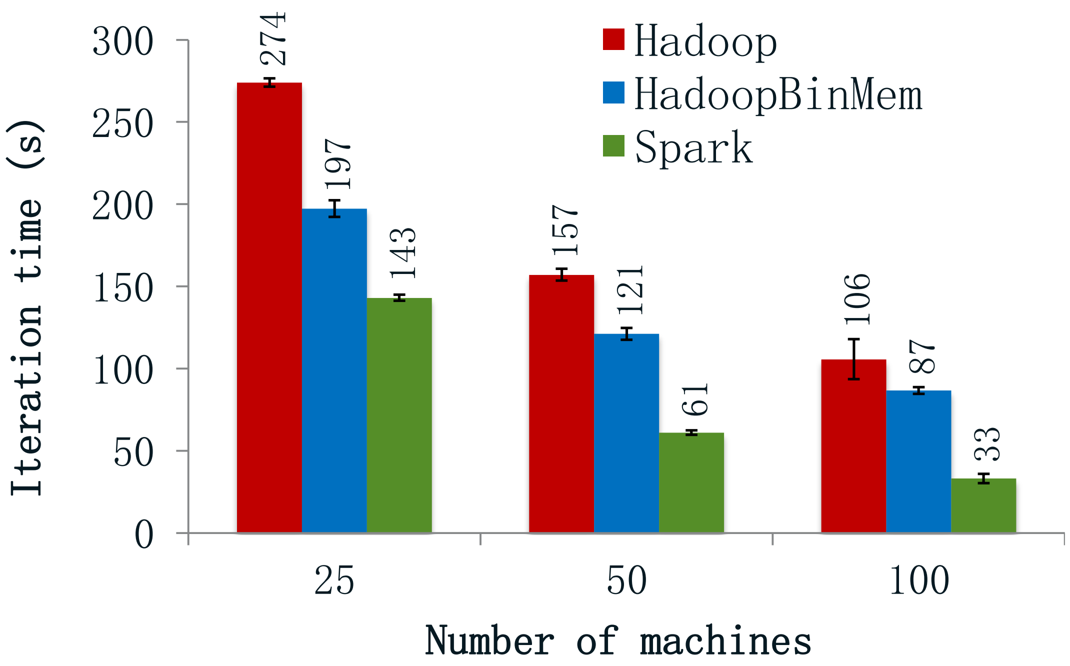

```scala
// Load and parse the data
val data = sc.textFile("mllib/data/als/test.data")
val ratings = data.map(_.split(',') match {
case Array(user, item, rate) =>
Rating(user.toInt, item.toInt, rate.toDouble)
})

// Build the recommendation model using ALS
val numIterations = 20
val model = ALS.train(ratings, 1, 20, 0.01)

// Evaluate the model on rating data
val usersProducts = ratings.map { case Rating(user, product,
rate) =>
(user, product)
}
val predictions = model.predict(usersProducts)
```

Mllib - Dimension reduction+ k-means

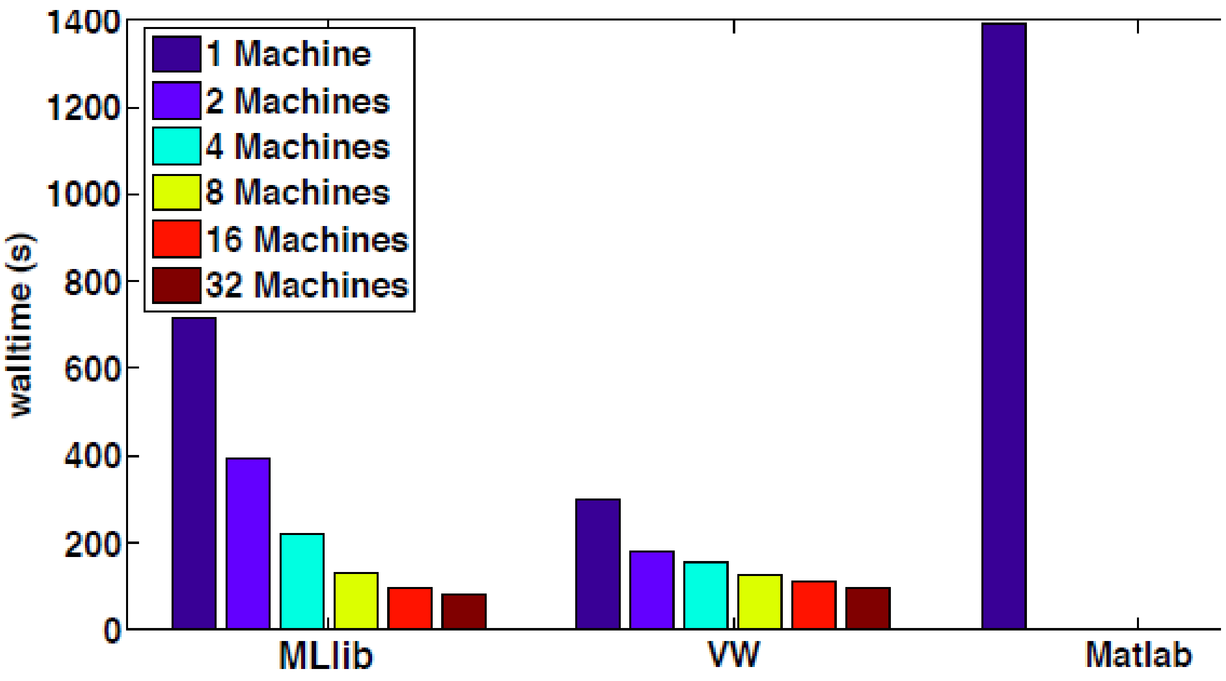



```
// compute principal components
val points: RDD[Vector] = ...
val mat = RowMatrix(points)
val pc = mat.computePrincipalComponents(20)

// project points to a low-dimensional space
val projected = mat.multiply(pc).rows

// train a k-means model on the projected data
val model = KMeans.train(projected, 10)
```

Mllib – Performance



Running times for iterations after the first in Hadoop, HadoopBinMem, and Spark. The jobs all processed 100 GB.



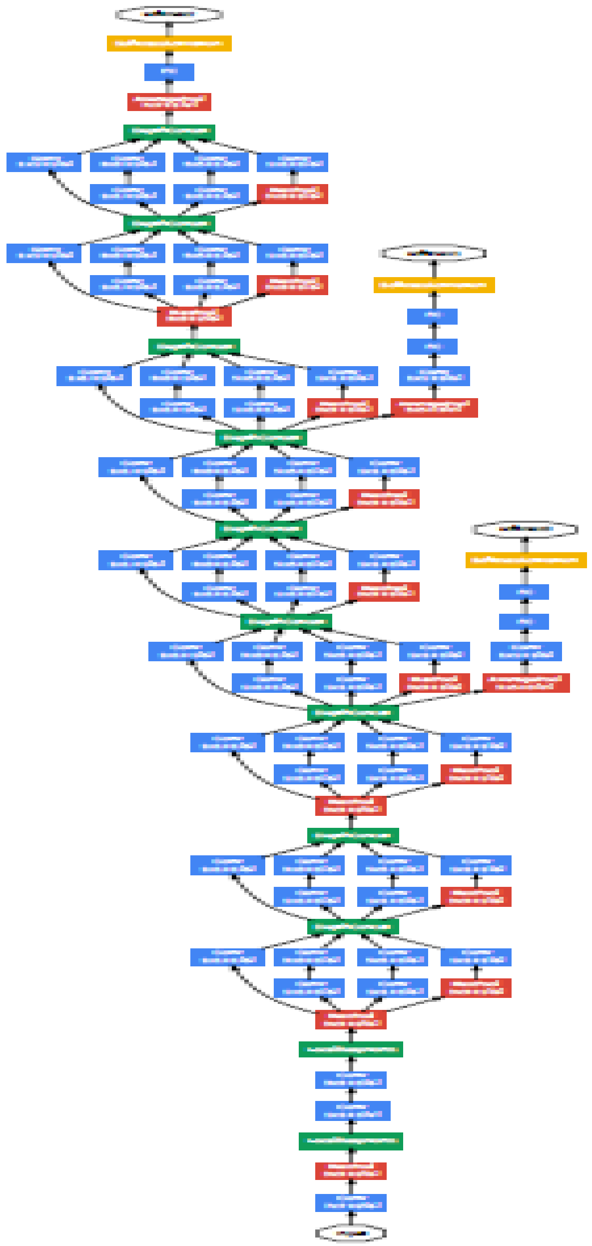
- Fixed Dataset: 50K images, 160K dense features.
- MLlib exhibits better scaling properties.
- MLlib is faster than VW with 16 and 32 machines.

Mllib 1.1?

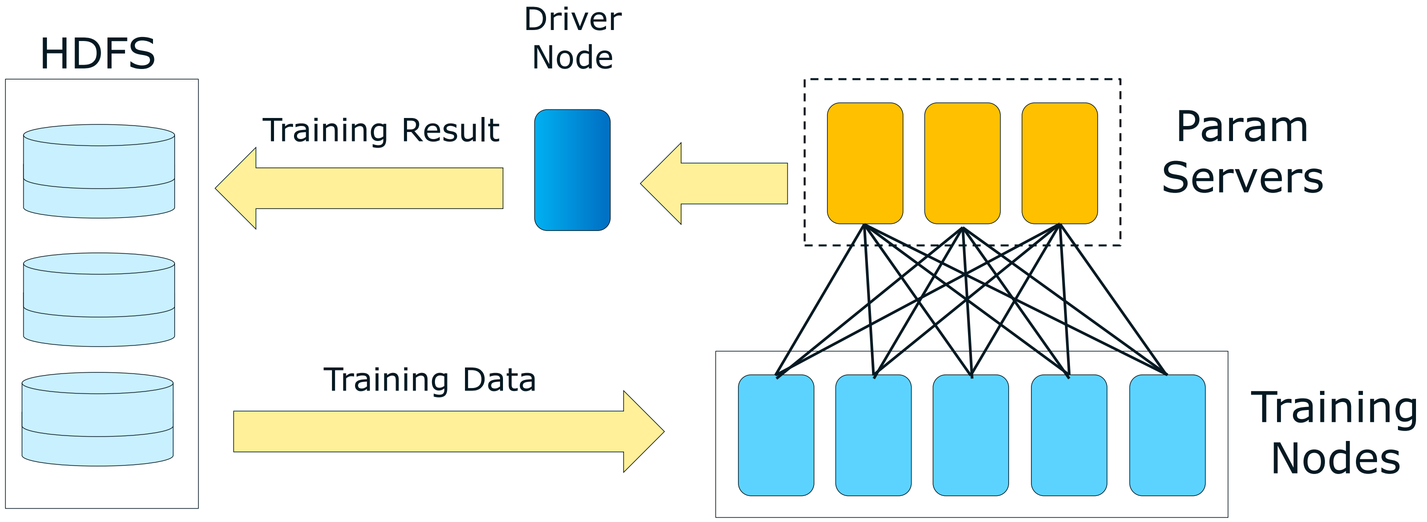
- Model selection!
  - training multiple models in parallel
  - separating problem/algorithm/parameters/model
- Learning algorithms!
  - Latent Dirichlet allocation (LDA)
  - Random Forests
  - Online updates with Spark Streaming
- Optimization algorithms!
  - Alternating direction method of multipliers (ADMM)
  - Accelerated gradient descent
- Neural Network?

New Challenge: Deep Learning

- Challenges
  - Very big model
  - Huge training data
- Spark Limitation
  - Only local model supported
  - RDD is read-only, expensive for neural network parameters
  - Broadcast is not feasible for big model
- We need
  - Distributed model
  - High-performance training process



Distributed Neural Network on Spark



- Distributed parameters
- Configurable server/worker nodes
- Multiple training workers
- Parallel fetching/training/pushing in each worker
- Adaptive learning rate

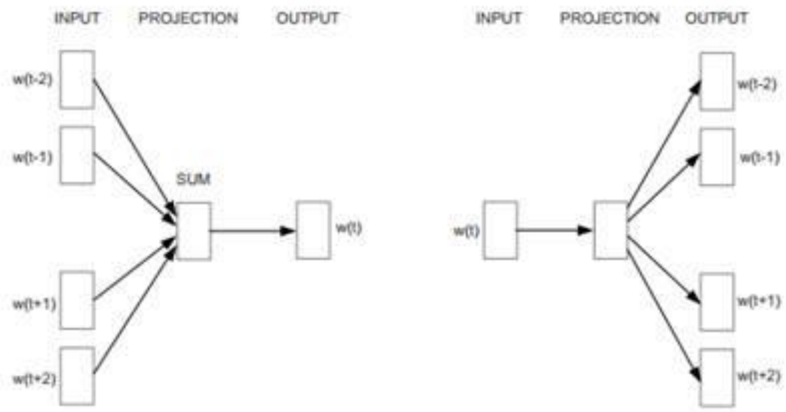
Example - Word2Vec

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research

- dog => [0.792 -0.177 0.98 -0.9 .....]
- cat => [0.76 0.12 -0.54 0.9 0.65 ....]

In some cases, word2vec can be used to modelling non-wording services, which makes its model very large

Distributed parameter servers helps to scaling the model size linearly

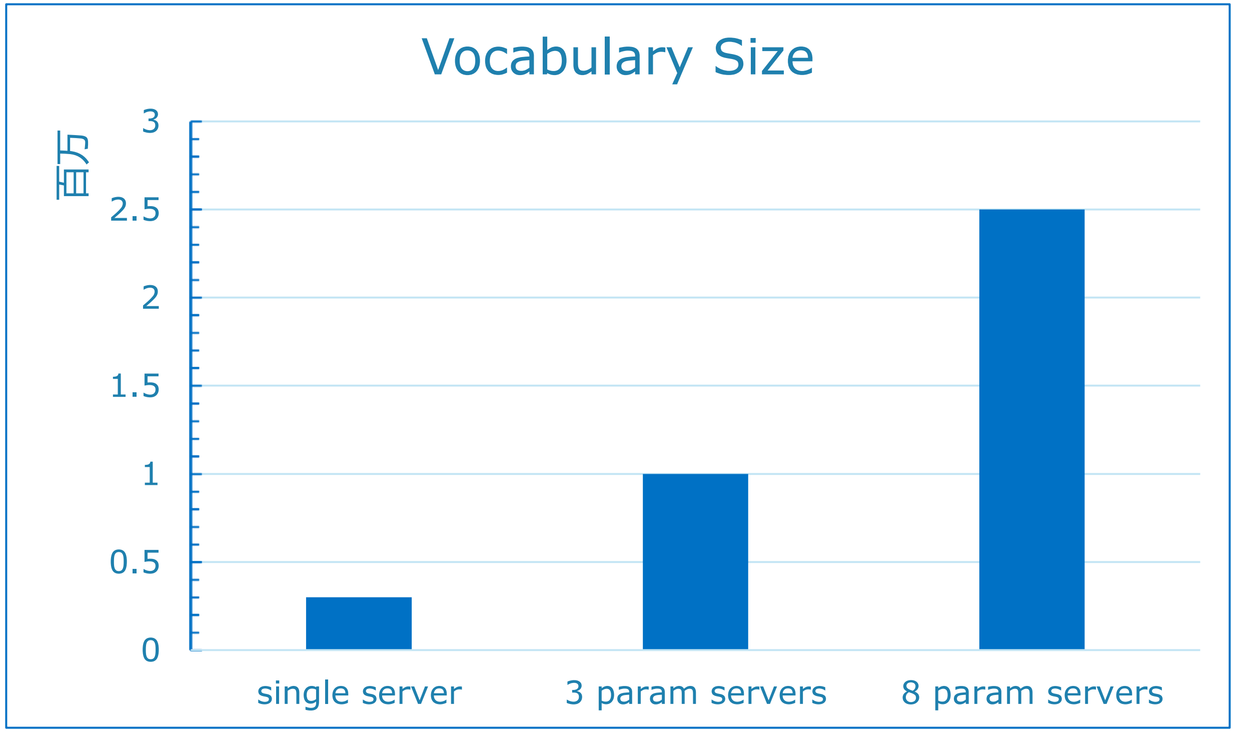


CBOW Skip-gram

Word	Cosine distance
France,	0.729900
Italy	0.720465
MORZINE,	0.681200
Germany	0.680331
Spain	0.673912
Russia	0.666366
Poland	0.652955
Spain,	0.648663
France.	0.646427
Germany,	0.642493

Test Result

- Extensibility
  - Linear extendable model size
  - Huge dataset supported
- Accuracy
  - Tradeoff between accuracy and performance, small batch size can raise accuracy, but hurt performance
  - Adaptive learning rate help to raise accuracy, but enlarge parameters too.
- Performance
  - Network is bottleneck, 10GbE is preferred
  - Multi-worker can obviously speed-up training
  - Optimization with MKL can speed up by nearly 50%



Q & A