GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

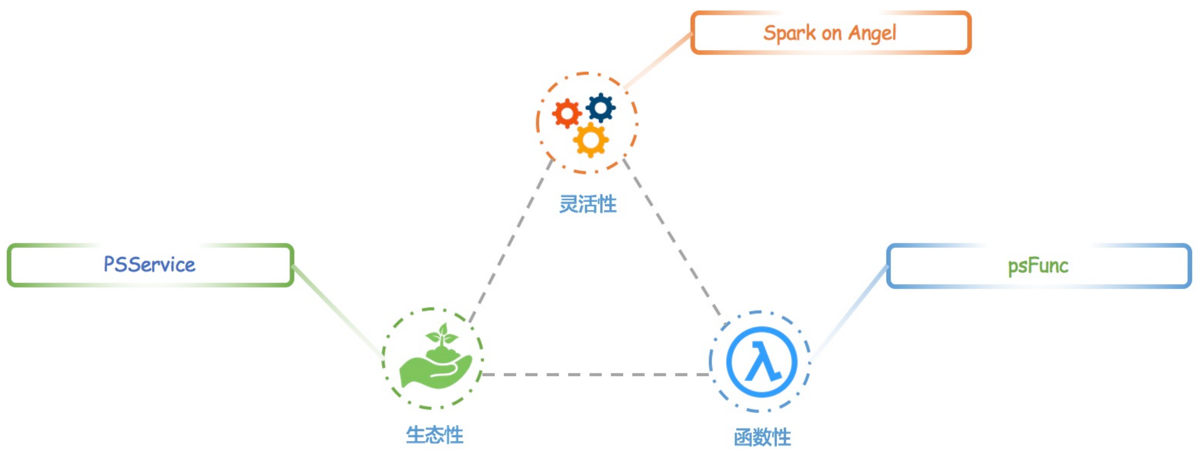
进击的巨人

基于Angel和Spark Streaming的高维度Online Learning

Andymhuang（黄明）

腾讯——数据平台部

Angel



一个基于参数服务器（Parameter Server）理念的高性能分布式机器学习平台

https://github.com/tencent/angel

整体介绍

Angel的架构

Spark on Angel

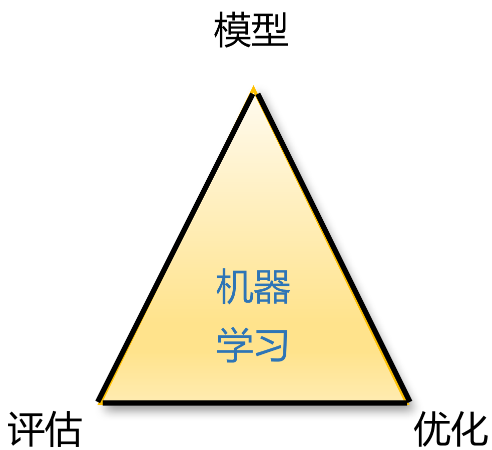
Online Learning with Angel

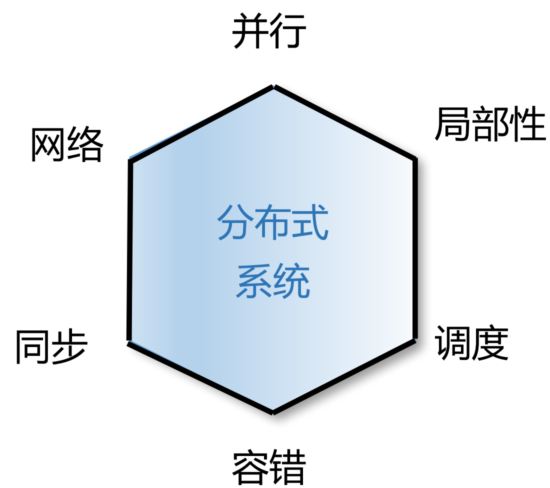
性能和比较

开源与展望

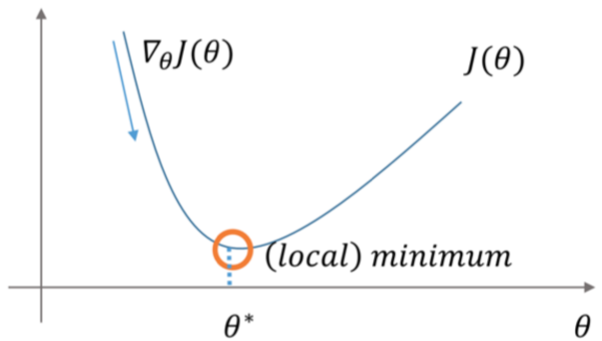
源起

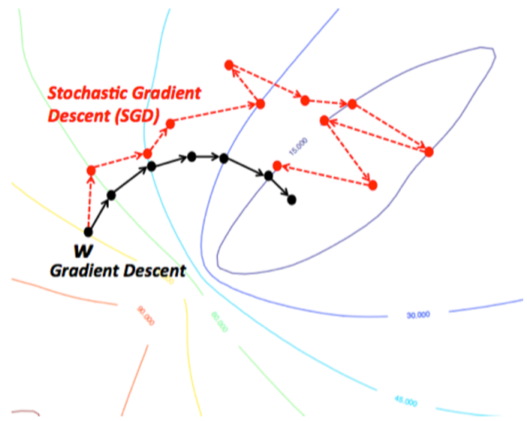
分布式机器学习





机器学习的目标

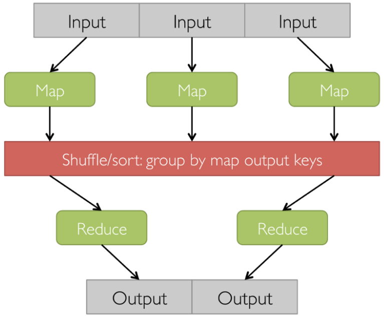




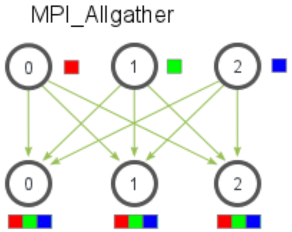
$$arg\min_{\vec{\theta}} f\left(\vec{x}, y, \vec{\theta}\right) = J(\{\overrightarrow{x_i}, y_i\}_{i=1}^{N}; \vec{\theta})$$

模型 **数据** **参数**

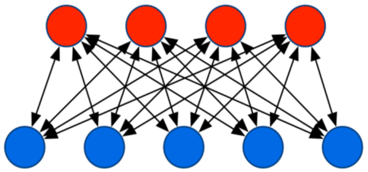
分布式的三种经典范式



MapReduce

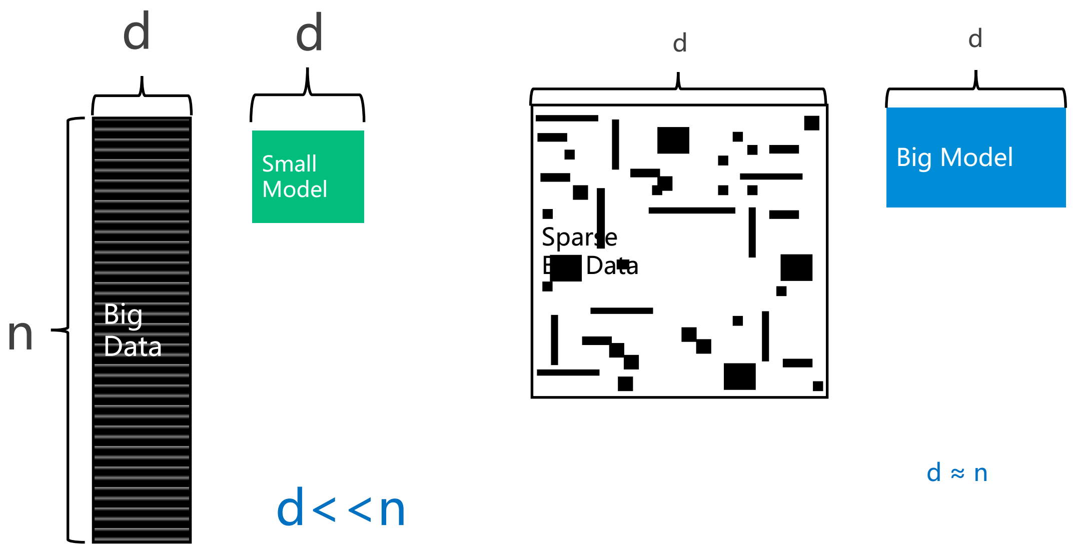


MPI



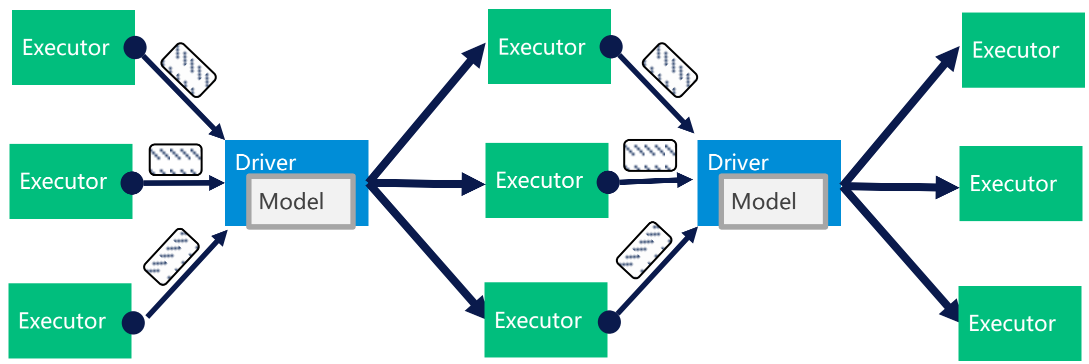
Parameter Server

腾讯的现网需求



寻找满足十亿级维度的工业级的分布式机器学习平台

Spark机器学习的瓶颈



- Driver成为参数汇总的单点瓶颈，难以支撑大规模模型及数据
- 十亿级维度的模型训练，实际应用中降维处理
- Executor之间相互等待，整体效率不高

其它机器学习平台

GraphLab

苹果收购了

PETUUM

实验室级别，而且融资后不开源了

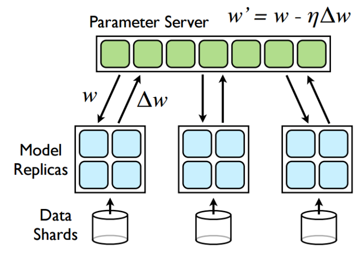
DMTK

转深度学习CNTK

dmlc XGBoost

针对性太强

DistBelief



转深度学习TensorFlow

Angel

开始研发

• 2015

投入生产

• 2016

正式开源 V1.0.0

• 2017

工业级别可用的参数服务器

- 能支持十亿级别维度的模型训练
- 基于Matrix/Vector的模型自动切分和管理，兼顾稀疏和稠密两种格式
- 提供多种同步控制机制（BSP/SSP/ASP）

丰富的机器学习及数学计算库

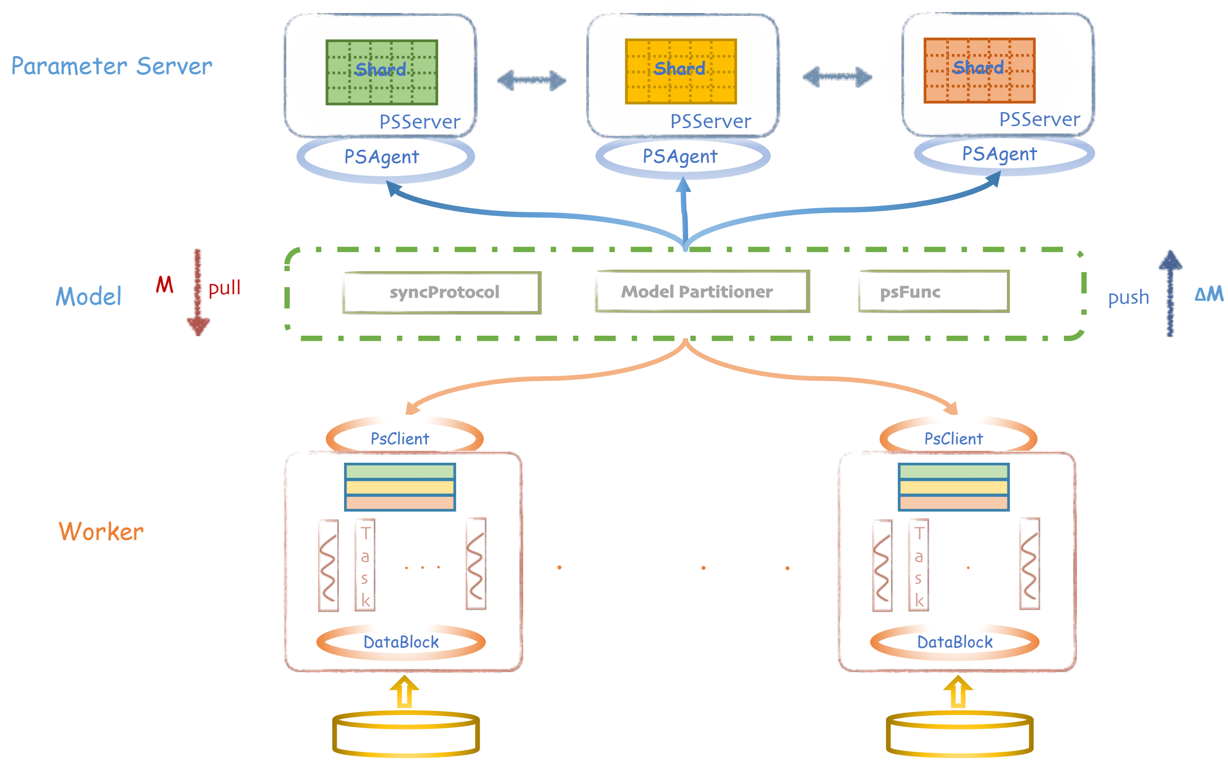
- 集成LR（ADMM-LR），SVM，KMeans，LDA，MF，GBDT等机器学习算法
- 多种优化方法，包括ADMM，OWLQN，LBFGS和GD
- 支持多种损失函数、评估指标，包含L1、L2正则项算法

友好的用户编程接口

- 基于PSModel的机器学习友好接口，以Model为核心编程
- 支持Spark on Angel，Spark代码小量改动就可以运行Angel之上
- 灵活的psFunc，便于复杂算法的开发，实现模型并行

Angel的架构

Angel的系统框架



Parameter Server
Shard
PSServer
PSAgent
Shard
PSServer
PSAgent
Shard
PSServer
PSAgent
Model
M
pull
syncProtocol
Model Partitioner
psFunc
push
ΔM
Worker
PsClient
Task
DataBlock
PsClient
Task
DataBlock

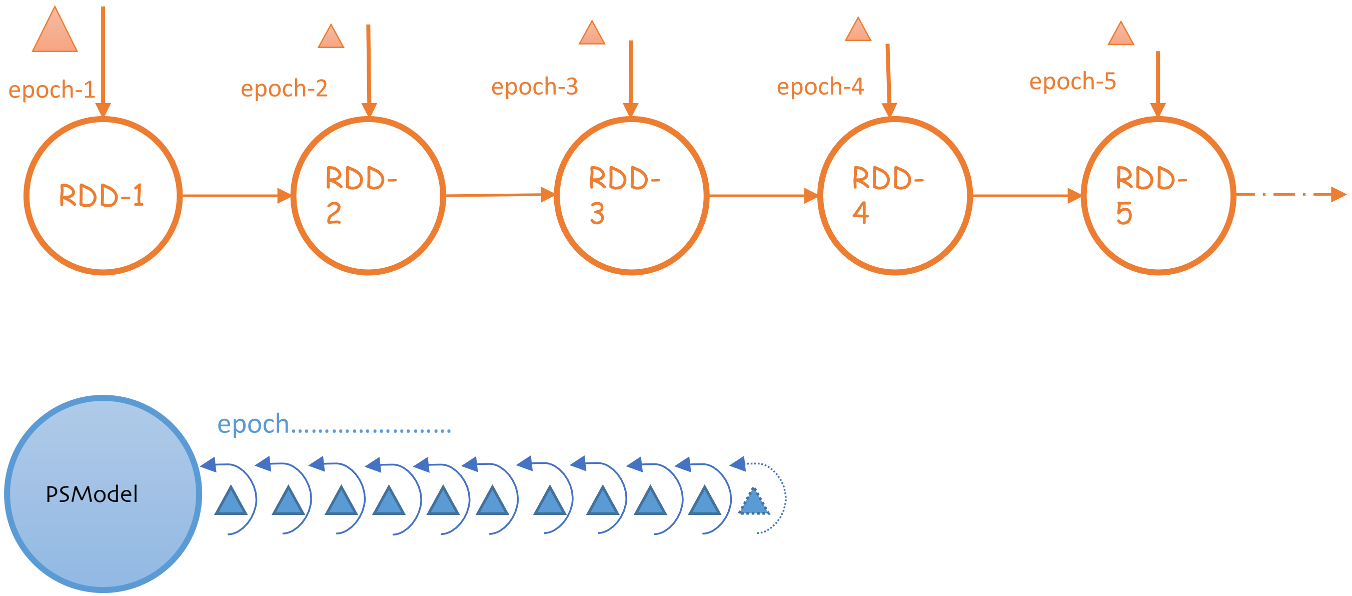
核心抽象

Mapper
Reducer

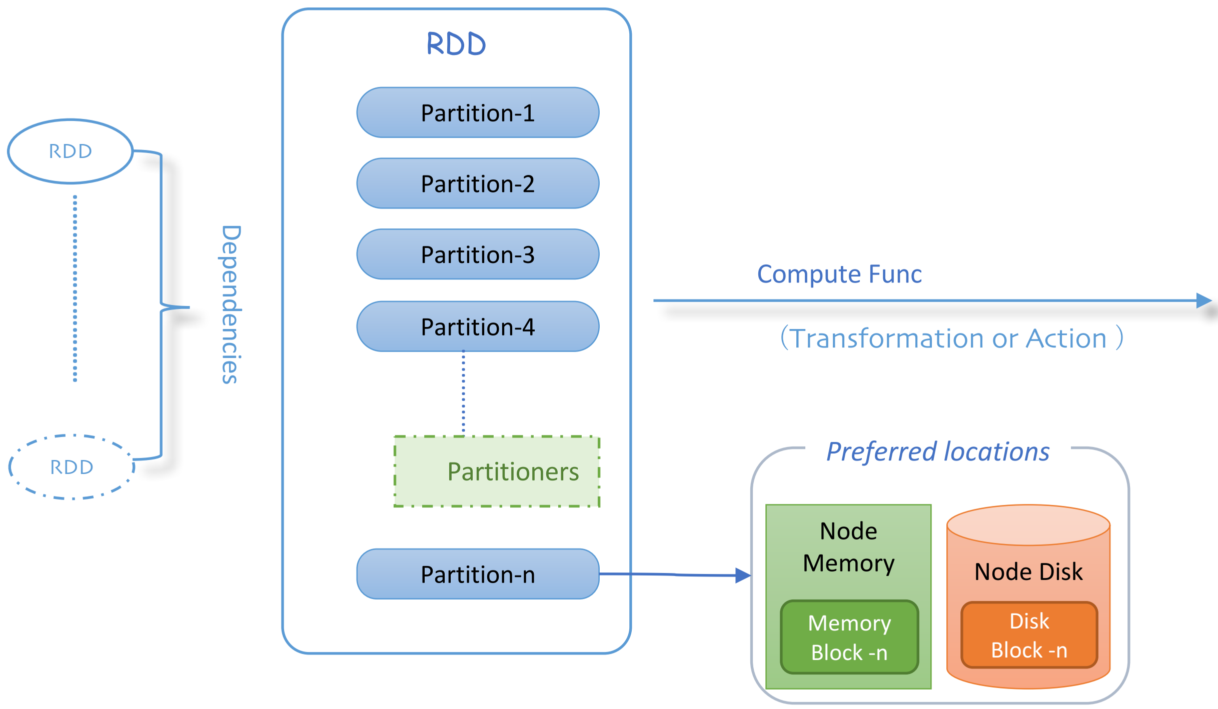
RDD

PSModel

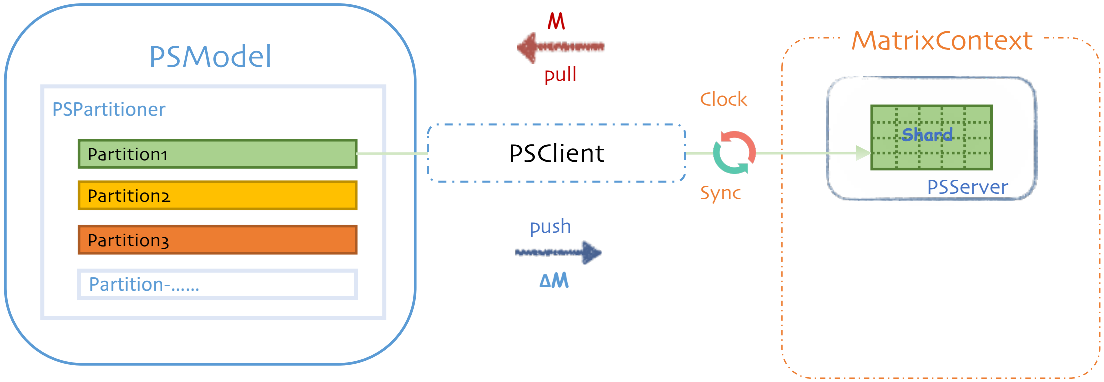
RDD vs PSModel



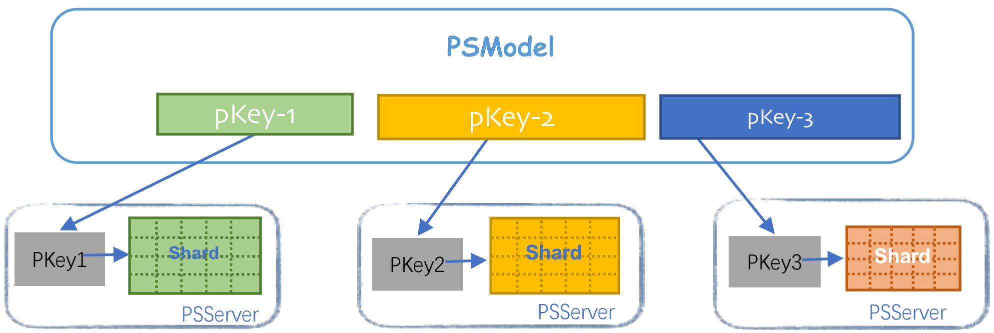
RDD和核心抽象


RDD
RDD
Dependencies
RDD
RDD
Partition-1
Partition-2
Partition-3
Partition-4
Partitioners
Partition-n
Compute Func
（Transformation or Action ）
Preferred locations
Node Memory
Memory Block -n
Node Disk
Disk Block -n

PSModel


PSModel
PSPartitioner
Partition1
Partition2
Partition3
Partition-......
M
pull
PSClient
push
ΔM
Clock
Sync
MatrixContext
Shard
PSServer

模型分区（Model Partitioner）



将模型切分成多个分区，
存储在不同PS节点上

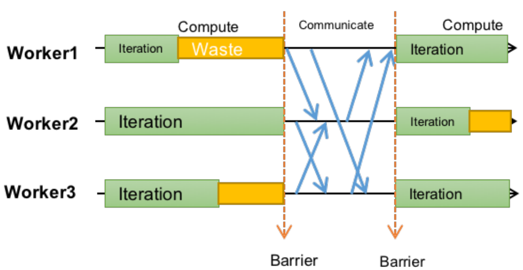
优点

- 保证Server负载均衡
- 避免Server单点性能瓶颈
- 支撑10亿～100亿级别维度参数

Angel的模型分区

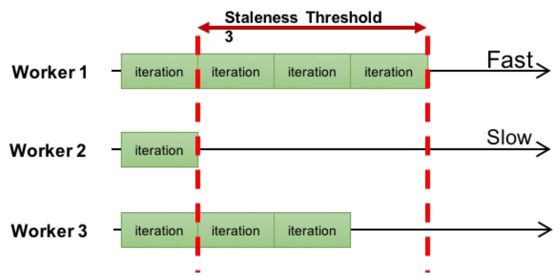
- 默认将模型分成大小相等的块
- 可以指定分区块大小
- 支持横切和纵切
- 自定义矩阵分区，量身定制区块分布方式

同步控制（Sync Controller）



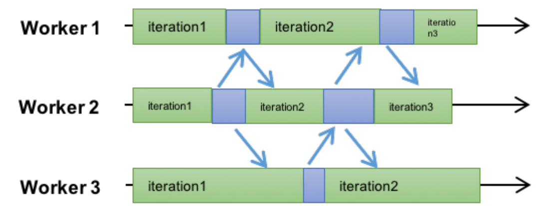
BSP

适用范围广，但等待时间长
angel.staleness = 0



SSP

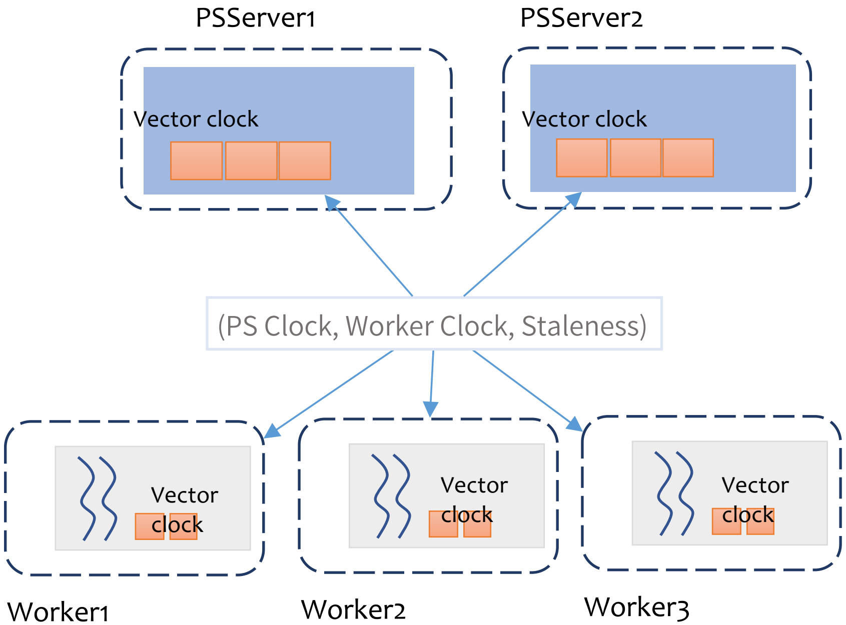
等待时间较短，但需要更多迭代
angel.staleness=N



ASP

无等待时间较短，收敛无保证
angel.staleness=-1

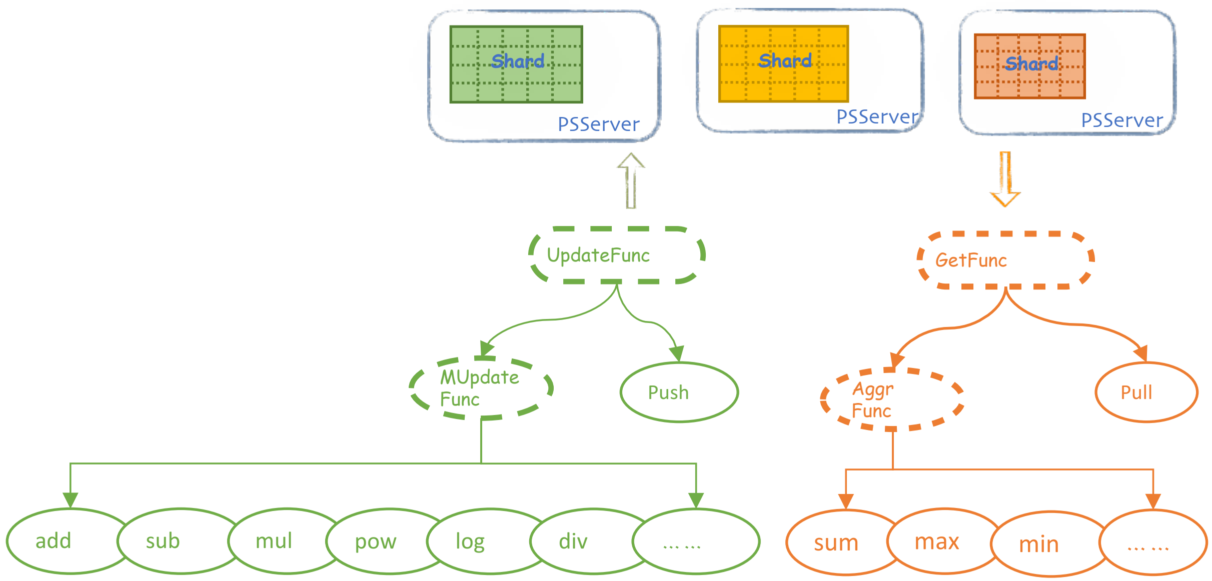
同步控制（Sync Controller）



向量时钟

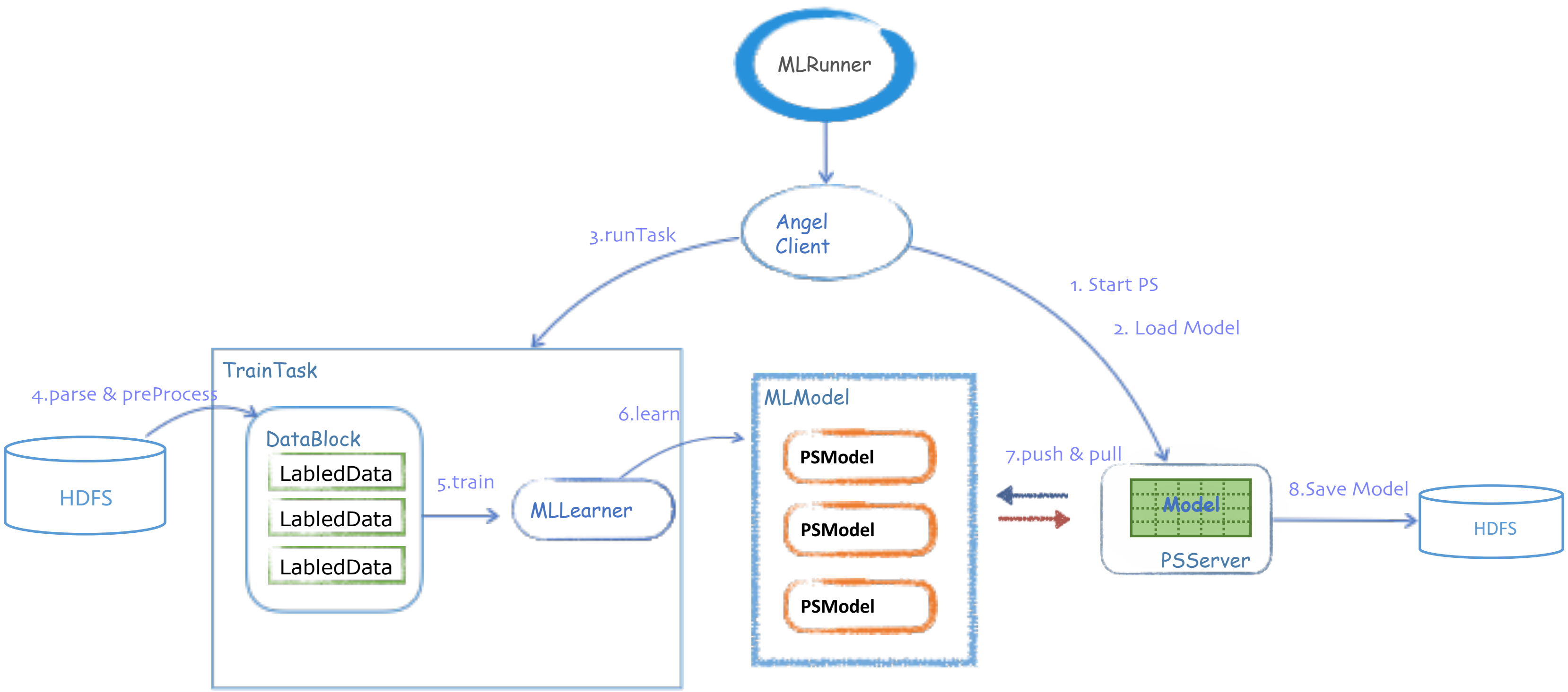
- 在Server端为每个分区维护一个向量时钟，记录每个worker在该分区的时钟信息
- 在Worker端维护一个后台同步线程，用于同步所有分区的时钟信息
- Task在对PSModel进行Get或其他读取操作时，根据本地时钟信息和staleness进行判断，选择是否进行等待操作
- 每次迭代完，调用Clock方法，更新向量时钟

ps函数（psFunc）



PSServer不仅仅是存储，也具备计算功能，可以实现模型并行

核心流程和类



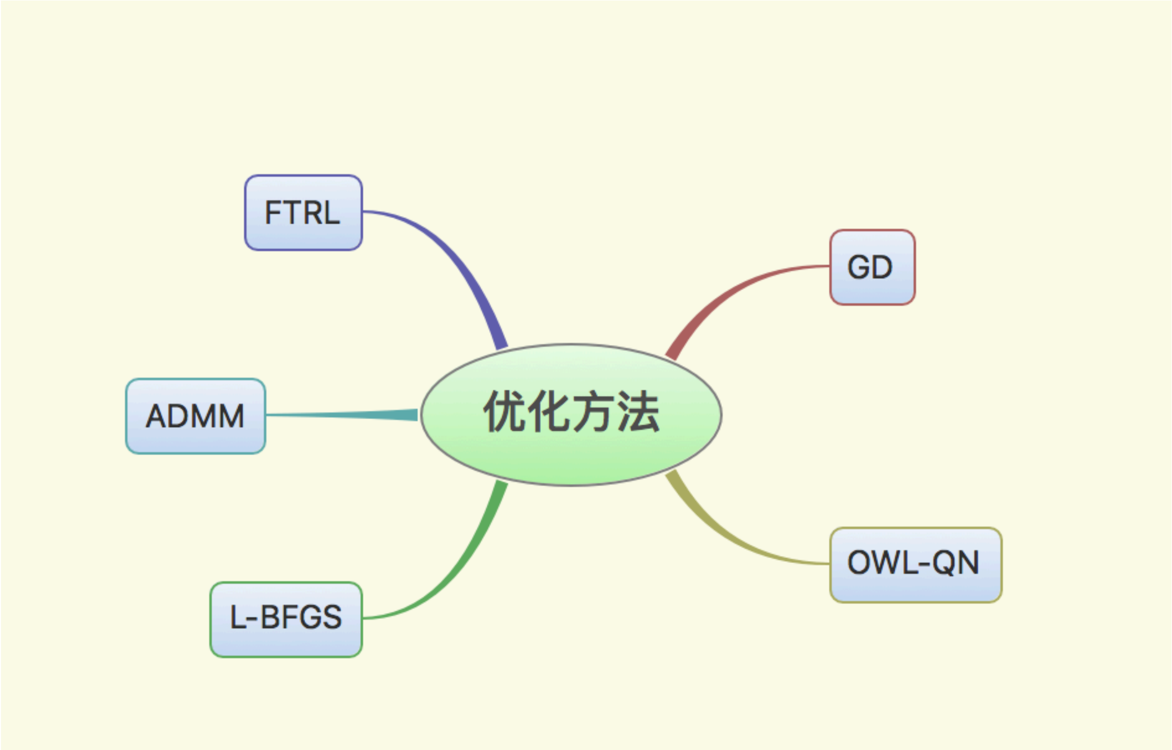
MLRunner
Angel Client
3.runTask
1. Start PS
2. Load Model
TrainTask
4.parse & preProcess
HDFS
DataBlock
LabledData
LabledData
LabledData
5.train
MLLearner
6.learn
MLModel
PSModel
PSModel
PSModel
7.push & pull
Model
PSServer
8.Save Model
HDFS

Algorithms on Angel

Angel的算法

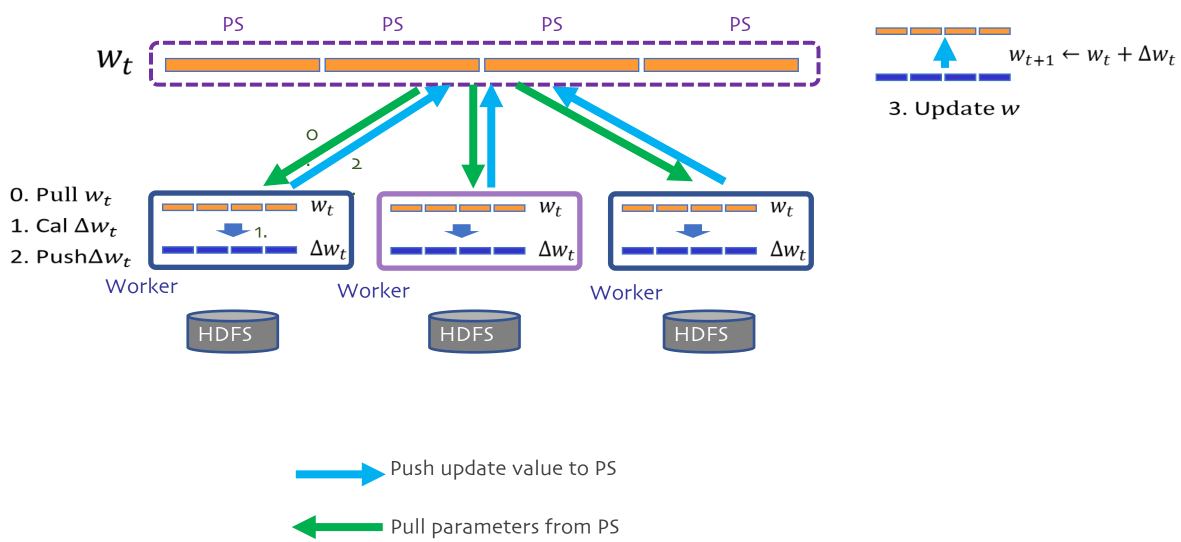
Algo on Angel
推荐
Matrix Factorization
Factorization Machines
主题模型
LDA*
树
GBDT（Gradient Descent Boosting Tree）
分类
Logistic Regression
Support Vector Machine
回归
Linear Regression
聚类
KMeans

优化方法



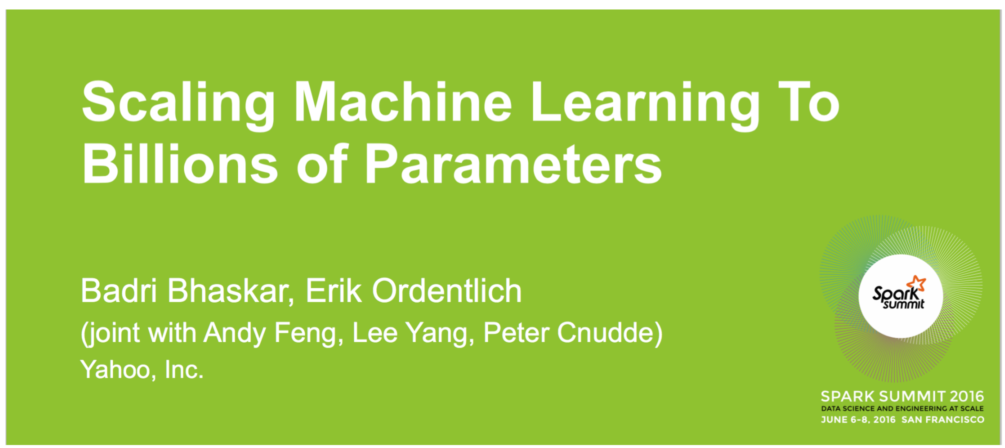
LR on Angel

Step 0: Worker从PS获得参数W_t
Step 1: Worker计算参数的更新值ΔW_t
Step 2: Worker把ΔW_t推送给PS
Step 3: PS更新参数（$W_{t+1} \leftarrow W_t + \Delta W_t$）



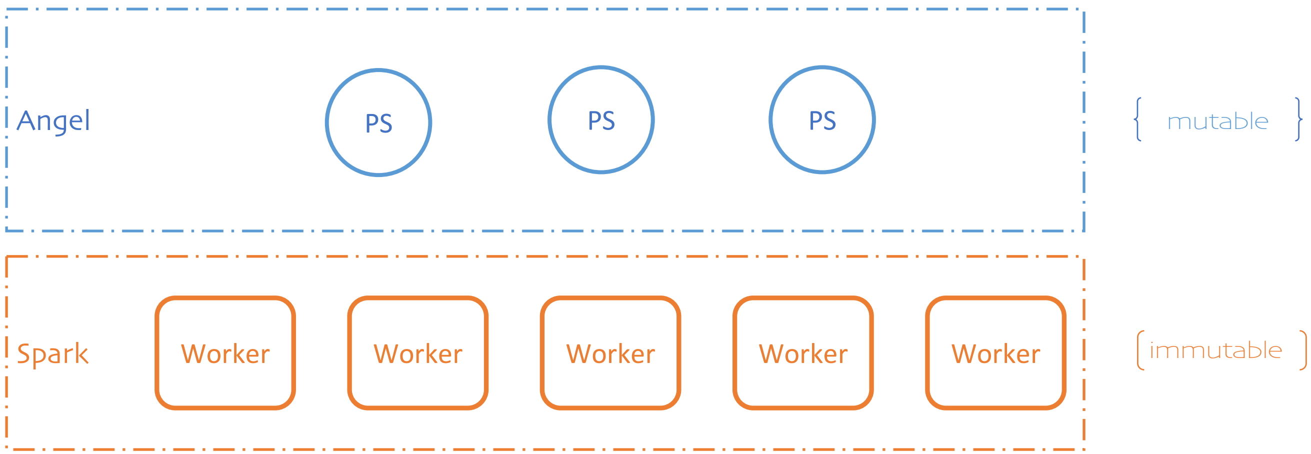
Spark on Angel

Spark on PS的回顾



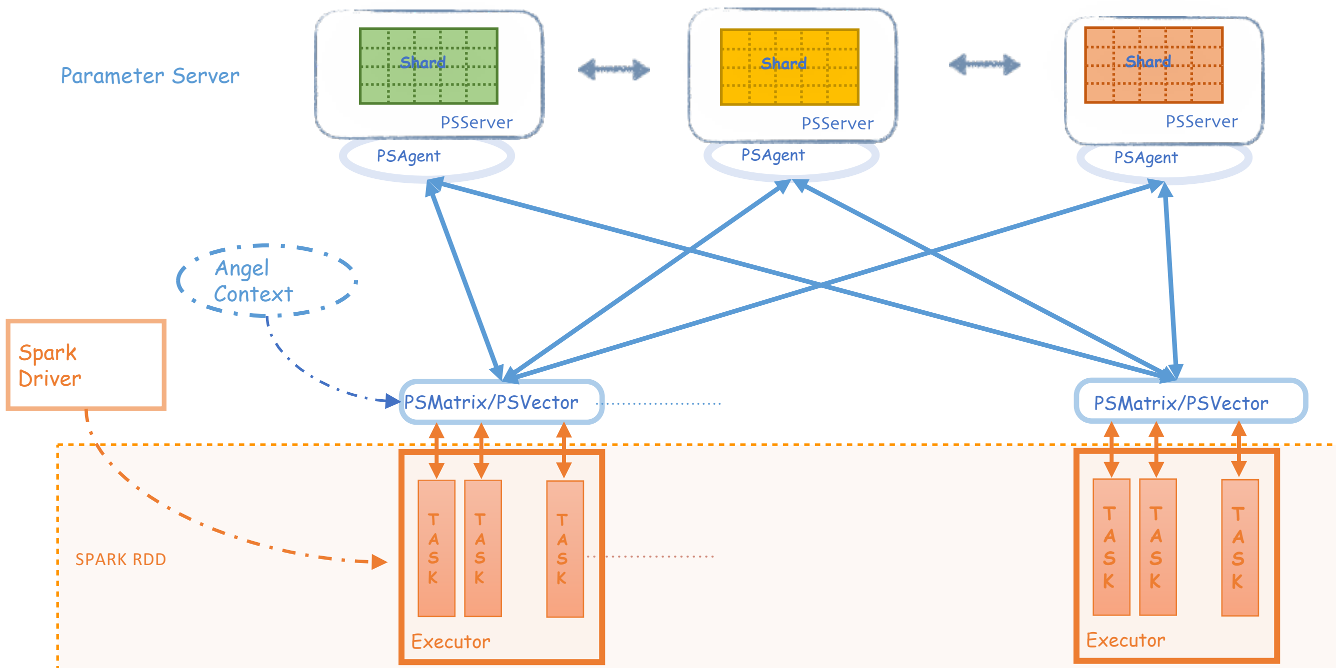
https://issues.apache.org/jira/browse/SPARK-6932

Spark on PS的基本理念



1. 分离变和不变
2. 以少博多
3. 降低侵入性

Spark on Angel的架构



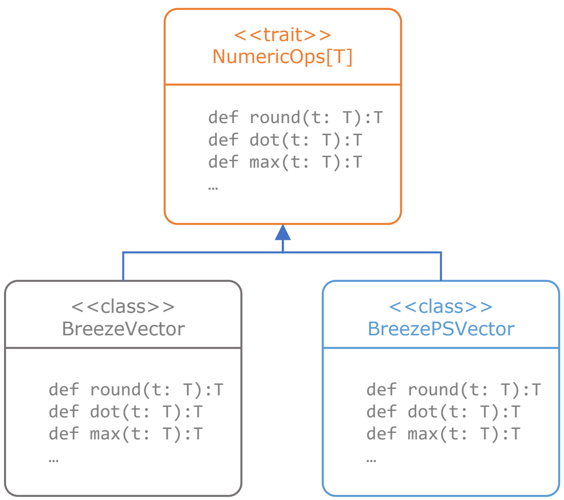
Spark on Angel的基础写法

```
PSContext.getOrCreate(spark.sparkContext)
val psVector = PSVector.dense(dim, capacity)
rdd.map { case (label , feature) =>
        psVector.increment(feature)
        ...
}
println("feature sum:" + psVector.pull.mkString(" "))
```

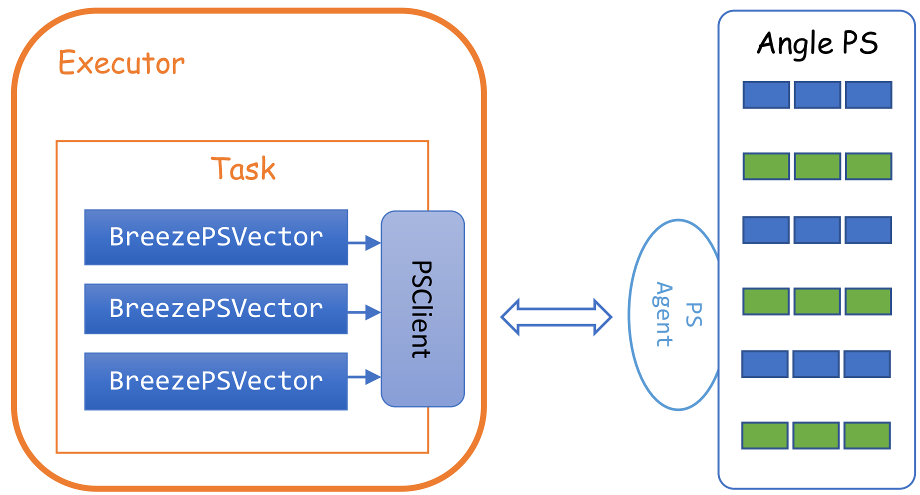
- 启动SparkSession
    - 初始化PSContext，启动Angel的PSServer
    - 通过PSContext, 创建PSVector
    - 在RDD的运算中，直接调用PSVector，进行模型更新
    - 终止PSContext
- 停止SparkSession

Vector的透明替换

混入相同特征



进行透明替换



- 将BreezeVector透明替换为PsVector
- 适用于MLLib大部分算法
- 替代成本非常低

Spark on Angel的进阶写法

- Spark

```
def runOWLQN(trainData: RDD[(Vector, Double)], dim: Int, m: Int, maxIter: Int): Unit = {

    val initWeight = new DenseVector[Double](dim)
    val l1reg = 0.0
    val owlqn = new BrzOWLQN[Int, DenseVector[Double]](maxIter, m, 0.0, 1e-5)

    val states = owlqn.iterations(CostFunc(trainData), initWeight)
    ……

}
```

- Spark on Angel

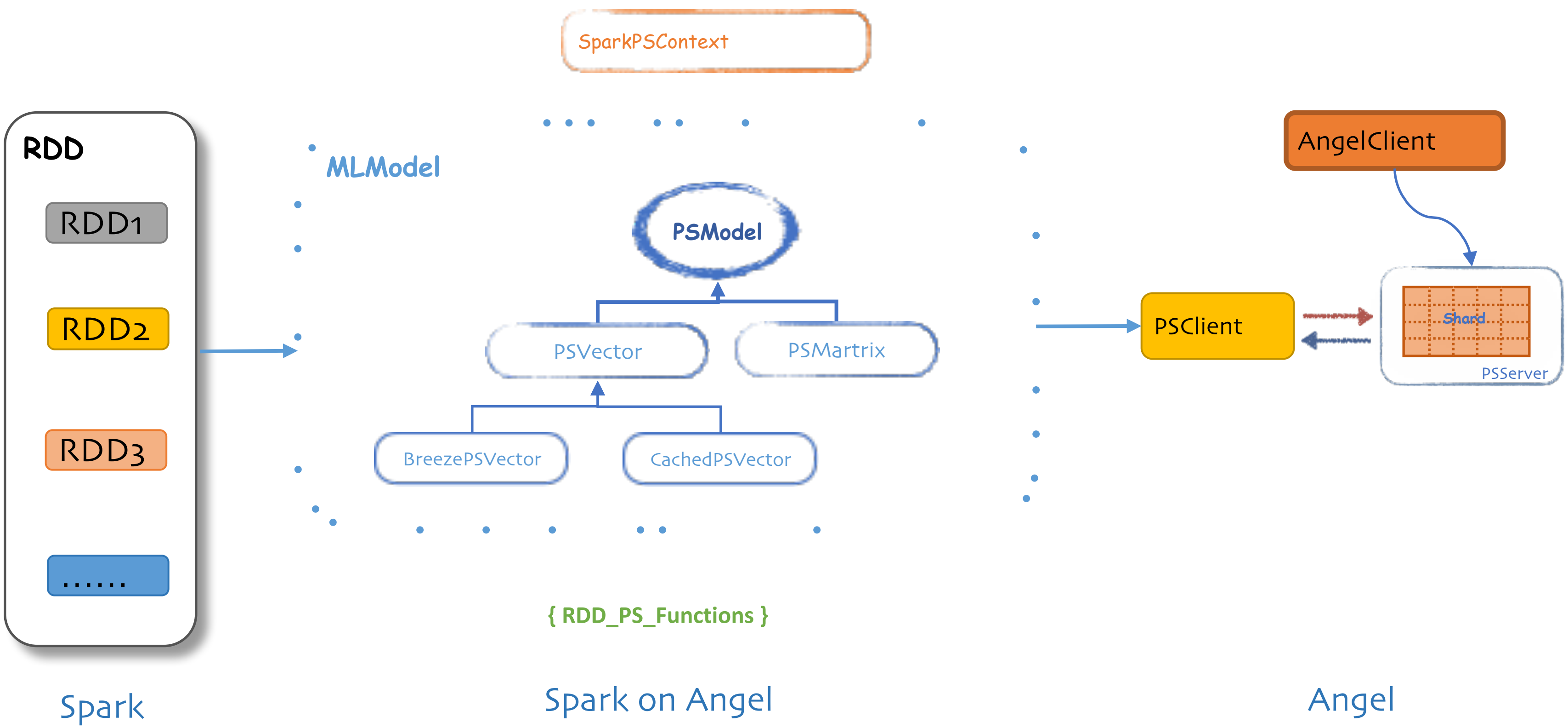
```
def runOWLQN(trainData: RDD[(Vector, Double)], dim: Int, m: Int, maxIter: Int): Unit = {

    val initWeightPS = PSVector.dense(dim, 20).toBreeze()
    val l1regPS = PSVector.duplicate(initWeightPS.component).zero().toBreeze

    val owlqn = new OWLQN(maxIter, m, l1regPS, tol)
    val states = owlqn.iterations(PSCostFunc(trainData), initWeightPS)
    ………

}
```

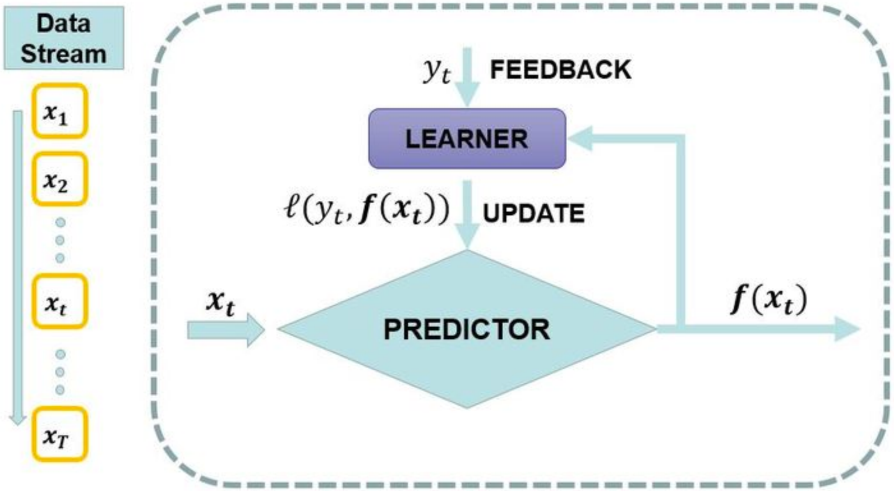
Spark on Angel的API设计



Spark Streaming on Angel

A Better Way of Online Learning

Online Learning



https://en.wikipedia.org/wiki/Online_machine_learning

在线随机优化 – 目标

- 目标：在online的场景下最小化后悔(regret)

For $t = 1, 2, \ldots, T$

•Learner receive an input $x_t \in X$
•Learner output prediction $p_t = f_t(x_t) \in Y$
•Nature looks at output p_t and send the learner the true label $y_t \in Y$
•Learner suffers loss $V(p_t, y_t)$ and updates its model

The learner is trying to minimize the regret

$$R_T(H) = \sum_{t=1}^{T} V(p_t, y_t) - \min_{f \in H} \sum_{t=1}^{T} V(f(x_t), y_t)$$

- 典型算法：FTRL(follow the regularized leader)

基于Logistic Regression的FTRL

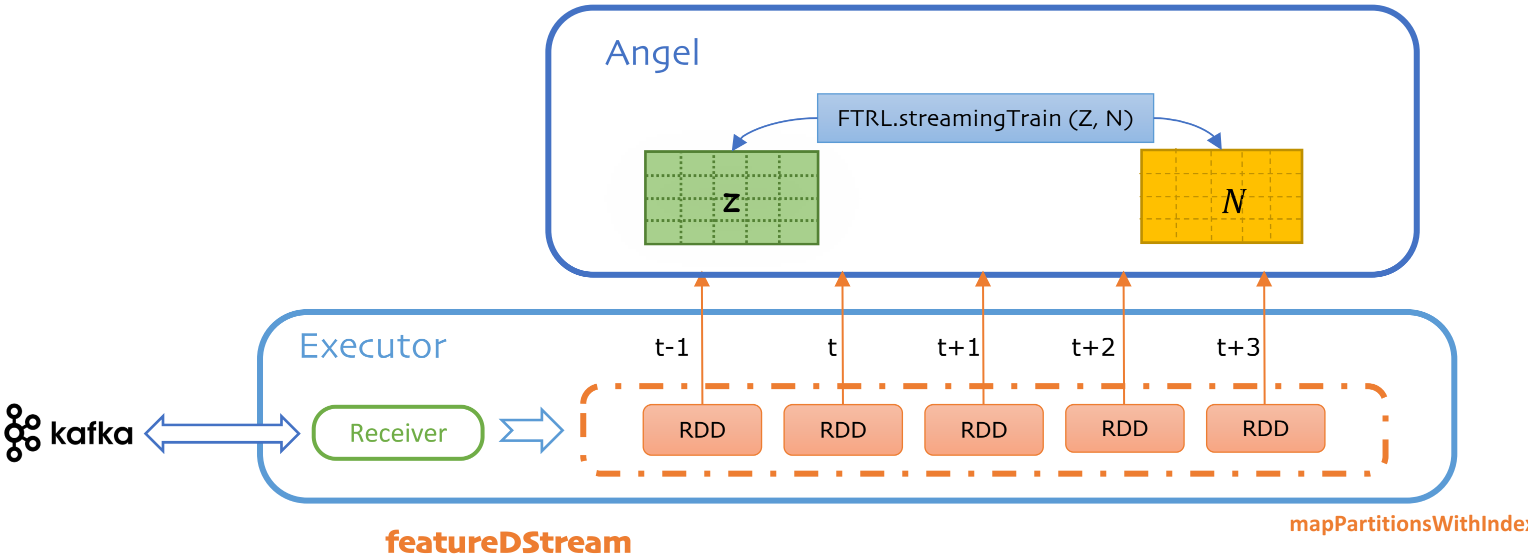
Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

#With per-coordinate learning rates of Eq. (2).
Input: parameters α, β, λ_1, λ_2
$(\forall i \in \{1, \ldots, d\})$, initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ **to** T **do**
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute

$$w_{t,i} = \begin{cases} 0 & \text{if } |z_i| \leq \lambda_1 \\ -\left(\frac{\beta+\sqrt{n_i}}{\alpha} + \lambda_2\right)^{-1}(z_i - \text{sgn}(z_i)\lambda_1) & \text{otherwise.} \end{cases}$$

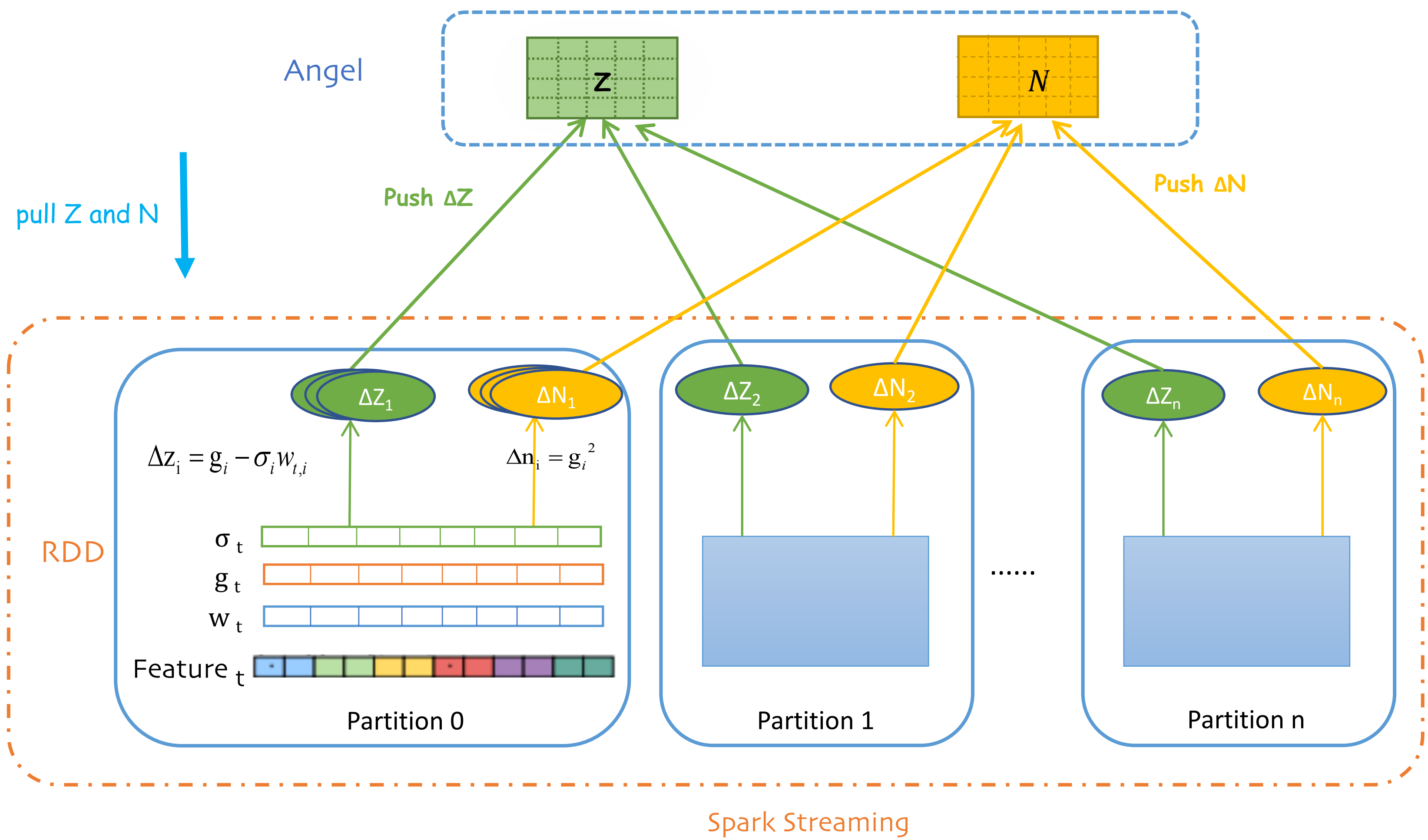
  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ **do**
    $g_i = (p_t - y_t)x_i$ *#gradient of loss w.r.t.* w_i
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$ *#equals* $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$
  end for
end for

[Spark Streaming on Angel] FTRL架构



[<SparseLRWithFTRL.scala>]()

基于Angel的实现



Angel
Z
N
pull Z and N
Push ΔZ
Push ΔN
RDD
ΔZ_1
ΔN_1
ΔZ_2
ΔN_2
ΔZ_n
ΔN_n
$\Delta z_i = g_i - \sigma_i w_{t,i}$
$\Delta n_i = g_i^2$
σ_t
g_t
w_t
Feature $_t$
Partition 0
Partition 1
......
Partition n
Spark Streaming

FTRL的整体代码框架

1. 初始化上下文（Spark Streaming & Angel）

```
val ssc = new StreamingContext(sparkConf, Seconds(streamingWindow))
val sc = ssc.sparkContext
PSContext.getOrCreate(sc)
```

2. 对接Kafka，创建DataStream （Receiver-base模式）

```
val topicMap: Map[String, Int] = Map(topic -> 1)
val featureDS = KafkaUtils.createStream(ssc, zkQuorum, group, topicMap).map(_._2)
```

3. 创建Angel的PSModel（PSVector）

```
val zPS: SparsePSVector = PSVector.sparse(dim)
val nPS: SparsePSVector = PSVector.sparse(dim)
```

4. 基于FeatureDS，训练zPS和nPS的算法流程

```
SparseLRWithFTRL.train(zPS, nPS, featureDS)
```

5. 启动SparkStreaming

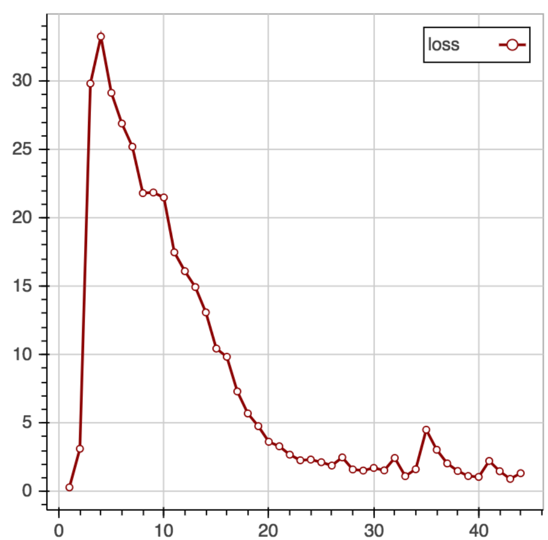
```
ssc.start()
ssc.awaitTermination()
```

资源和性能

资源参数	
* num-executors	100 分配计算节点数目
* driver-memory(g)	5 主节点内存大小，上限为30g
* executor-cores	3 每个子节点分配的cpu core数，推荐3~4
* executor-memory(g)	10 每个子节点分配的内存大小，上限为30g，推荐单个core配置2~3g

Angel资源参数	
* spark.ps.instances	20 Angel PS节点数
* spark.ps.cores	2 每个PS节点的Core数
* spark.ps.memory(g)	6 每个PS节点的Memory大小

alpha	0.1 FTRL的alpha参数
beta	1.0 FTRL的beta参数
* lambda1	0.1 FTRL的lambda1参数，L1正则项的系数
lambda2	0.1 FTRL的lambda2参数，L2正则项的系数
rho1	0.7 FTRL_VRG中权重W进行移动平均更新时采用的系数
rho2	0.85 FTRL_VRG中梯度进行移动平均更新时采用的系数

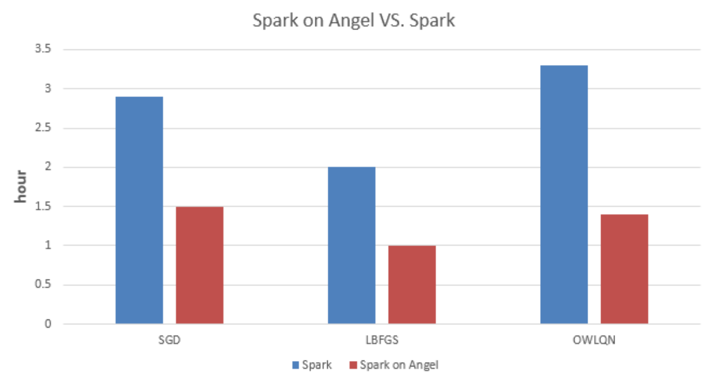


性能比对

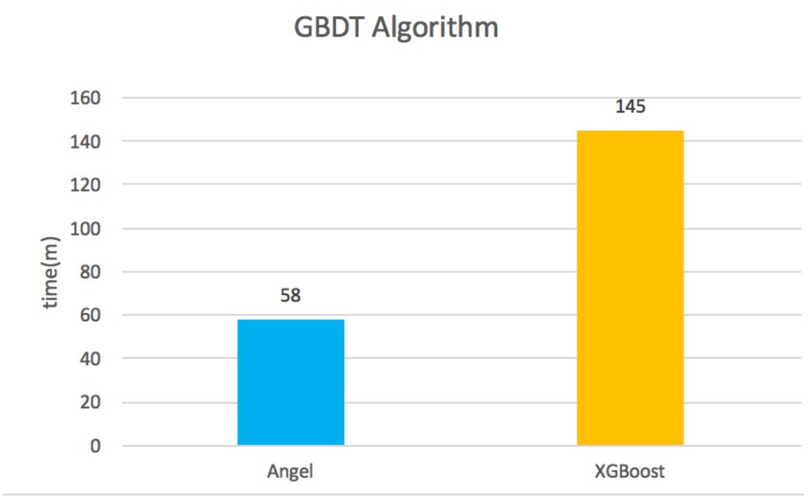
生产数据，现网环境，尽量公平

Spark on Angel vs Spark —— LR

	Spark	Spark on Angel	加速比例
SGD LR (stepSize=0.05,maxIter=100)	2.9 hour	1.5 hour	48.3%
L-BFGS LR (m=10, maxIter=50)	2 hour	1 hour	50.0%
OWL-QN LR (m=10, maxIter=50)	3.3 hour	1.4 hour	57.6%



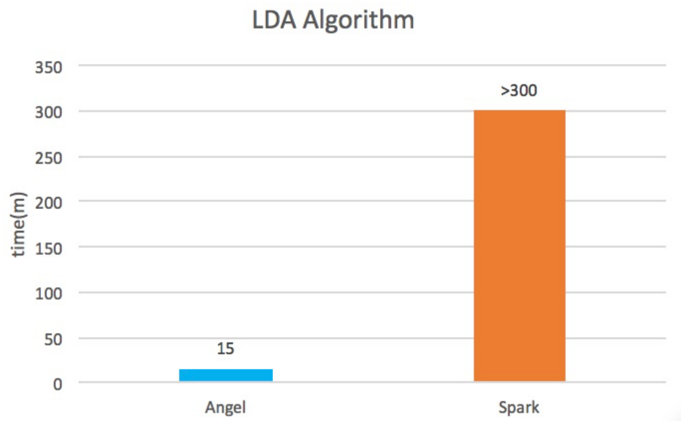
Angel vs XGBoost —— GBDT



框架	Worker	PS	建立20棵树时间
Angel	50 个(内存：10G / Worker)	10个 (内存：10G / PS)	58 min
XGBoost	50个 (内存：10G / Worker)	N/A	2h 25 min

数据： 腾讯内部某性别预测数据集，3.3×10^5 特征，1.2×10^8 样本

Angel vs Spark —— LDA



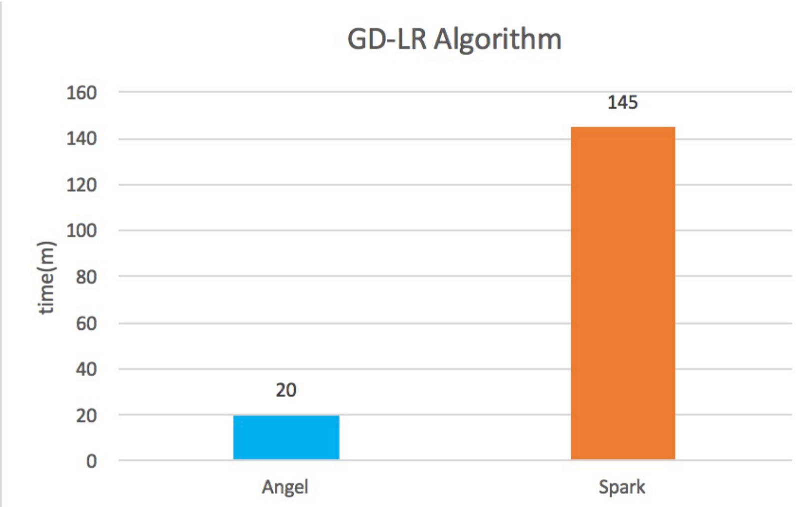
框架	Worker	PS	时间
Angel	20个(内存：8G/Worker)	20个(内存：4G/PS)	15min
Spark	20个(内存：20G/Worker)	N/A	>300min

数据：PubMED

框架	Worker	PS	时间
Angel	50个(内存：10G/Worker)	50个(内存：4G/PS)	1h 7min

DataSet：40G Token： 2 billion   Word：52w Topic： 1000

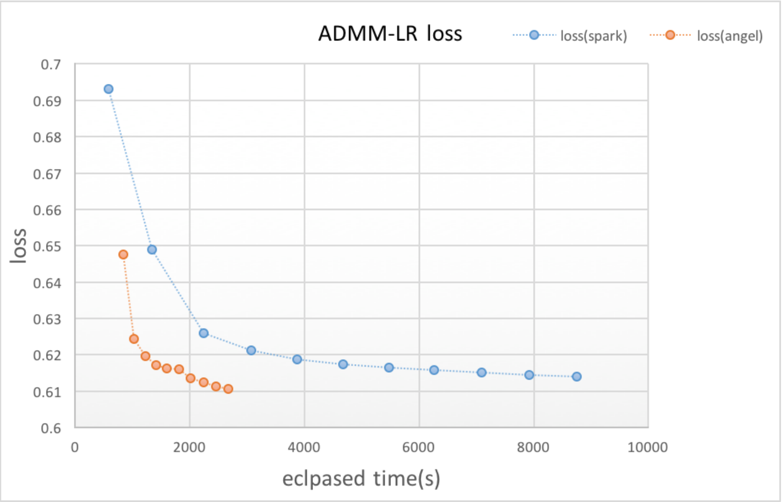
Angel vs Spark —— LR



框架	Worker	PS	迭代100次时间
Angel	50个(内存:10G/Worker)	20个(内存: 5G/PS)	20min
Spark	50个(内存:14G/Worker)	N/A	145min

数据：腾讯内部某推荐数据，5×10^7 特征，8×10^7 样本

Angel vs Spark —— ADMM-LR



框架	Worker	PS	收敛退出
Angel	100个(内存:10G/Worker)	50个(内存：5G/PS)	27 min
Spark	200个(内存:20G/Worker)	N/A	145 min

数据：腾讯内部某推荐数据，5千万特征，1亿样本

开源和展望

OpenSource & Perspective

Angel开源

Tencent / **angel**

Unwatch 321 | Unstar 2,889 | Fork 704

github:issues

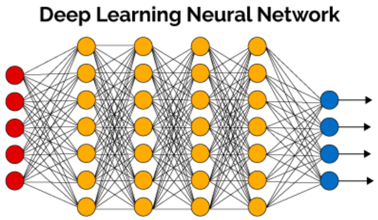
(PR 98)

- LightBGM作者：[GBDT] The purposes of using parameter server in GBDT #7
- 海外华人：English translation of documents #95
- 华为工程师：[WIP]Upgrade the netty version of RPC to 4.x #94
- 新浪微博：增强LR算法，加入y截距因子
- hbghhy：加入阿里巴巴用于CTR预估的MLR算法
- ……

学术创新

- **国际顶级会议Paper（CCF A类）**
    - LDA*: A Robust and Large-scale Topic Modeling System VLDB, 2017
    - Heterogeneity-aware Distributed Parameter Servers. SIGMOD, 2017
    - Angel: a new large-scale machine learning system. National Science Review (NSR), 2017
    - TencentBoost: A Gradient Boosting Tree System with Parameter Server. ICDE, 2017
    - ……

未来展望（What is Next）



Distributed Serving

V1.6

Deep Learning Framework Support

V2.0

Q & A

欢迎Star，Fork和PR

Angel技术交流群

扫一扫二维码，加入该群。

微博：@明风