Tencent Big Data

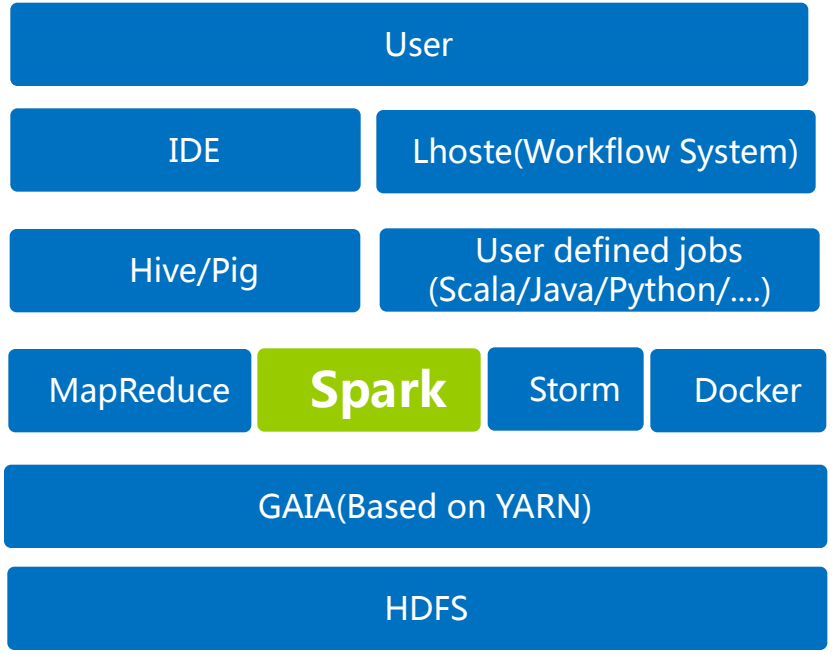
腾讯在Spark上的应用与实践优化

王联辉

Agenda

- Spark在腾讯的当前现状
- Spark在腾讯的典型应用及效果
- 腾讯在Spark上的实践和优化
- 未来工作和计划

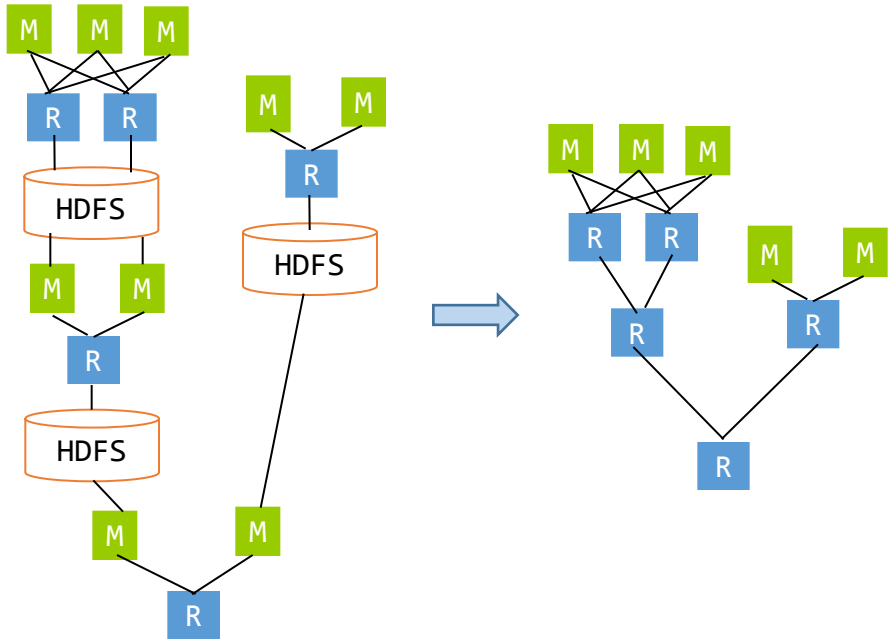
TDW(Tencent Distributed Data Warehouse) Overview



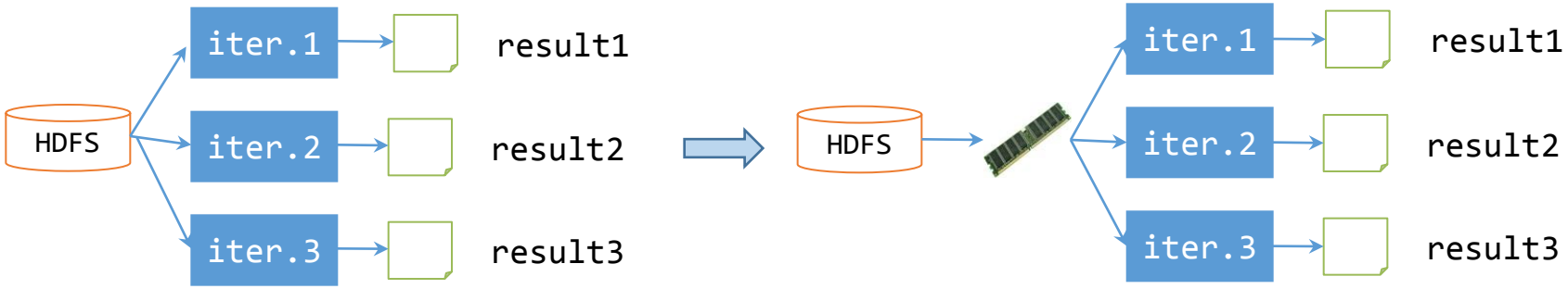
- Gaia集群结点数: **8000+**
- HDFS的存储空间: **150PB+**
- 每天新增数据: **1PB+**
- 每天任务数: **1M+**
- 每天计算量: **10PB+**

IDE: 用于提交SQL或脚本的Eclipse插件和Web界面
Lhoste: 各类作业的工作流调度系统，类似于Oozie
GAIA: 基于YARN进行定制和优化的资源管理系统

为什么我们引入Spark?

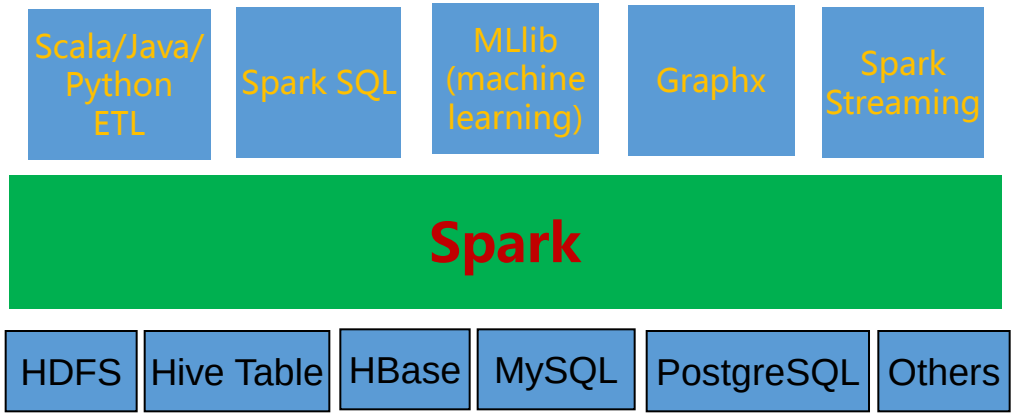


- DAG计算引擎
- Data Cache & Sharing
- 其他: task调度，数据广播等



Spark在腾讯的当前现状

- 作业类型：**ETL,SparkSQL,Machine Learning,Graph Compute,Streaming**
- 每天任务数：**10K+**
- 部署模式：**Gaia(8000+ Nodes,with 24 cores and 60G memory each)**
- 底层存储: **HDFS/Hive/HBase/MySQL/.....**
- 从2013年的Spark 0.6版本开始，目前的版本是Spark1.2



Spark在腾讯的典型应用及效果

- Case 1: 预测用户的广告点击概率
- Case 2: 计算二个好友间的共同好友数
- Case 3: 用于ETL的SparkSQL和DAG任务

Case 1: 预测用户的广告点击概率

Table-1

Gender	Age	Marital status	Location	Ad. one	Ad. two	Is click?
man	20	unmarried	NewYork	true	false	yes
woman	40	married	California	false	true	yes
man	60	married	California	false	true	No

Table-2

Gender	Age	Marital status	Location	Ad. one	Ad. two	Click's possibility
man	30	unmarried	NewYork	true	false	?
man	30	unmarried	NewYork	false	true	?

Model Train at Spark

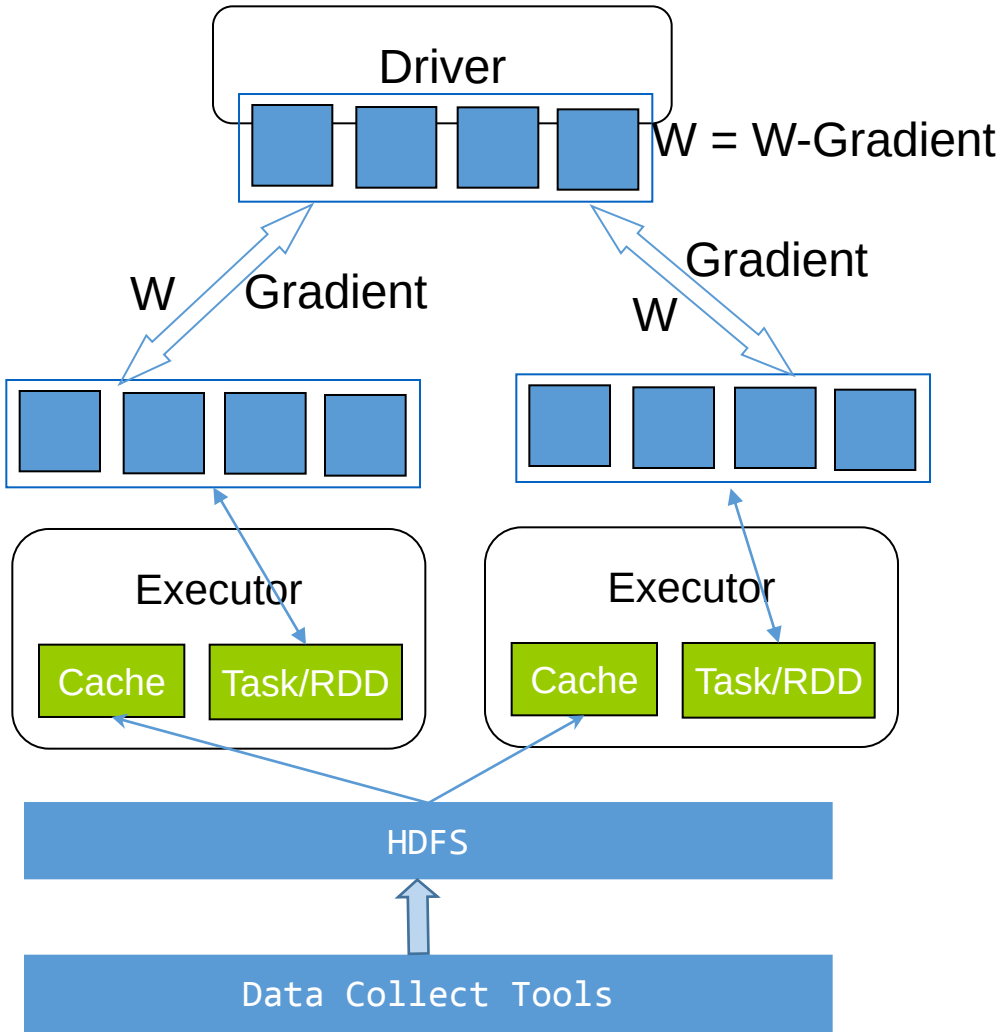
Table-3

w	man	woman	.....	Ad. 1	Ad.2
	-0.896	0.895	.....	-0.490	0.489

Table-4

Gender	Age	Marital status	Location	Ad. one	Ad. two	Click's possibility
man	30	unmarried	NewYork	true	false	30%
man	30	unmarried	NewYork	false	true	50%

Case 1: 预测用户的广告点击概率



- 实时预处理训练数据并推送到hdfs上
- 将训练数据(200G+) 导入至RDD & cache
- 初始化随机值给W并广播至各个executor
- 迭代训练60次左右
- for (i <- 1 to ITERATIONS) {

```
  val gradient = points.map { p =>
p.x * (1 / (1 + exp(-p.y * (w.dot(p.x)))) - 1) * p.y
    }.reduce(_ + _)
    w -= gradient
  }
```

- 将模型推送广告后台在线服务器
- **每个计算的时间在10~15分钟**

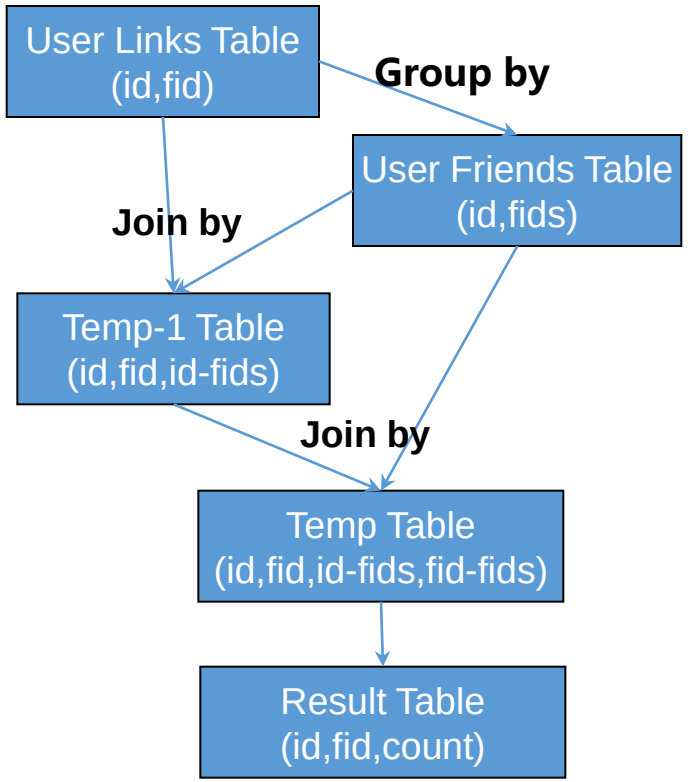
Case 2: 计算二个好友间的共同好友数

Table-1

user	friend
2	1
4	1
1	2
5	3
6	3
6	5
5	6
3	6

Table-2

user	friend	number of mutual friends
2	1	0
4	1	0
1	2	0
5	3	1
6	3	0
6	5	1
5	6	1
3	6	0



5亿用户数，500亿条边

10亿用户数，1000亿条边

Case 2: 计算二个好友间的共同好友数

Table-1

user	friend
2	1
4	1
1	2
5	3
6	3
6	5
5	6
3	6

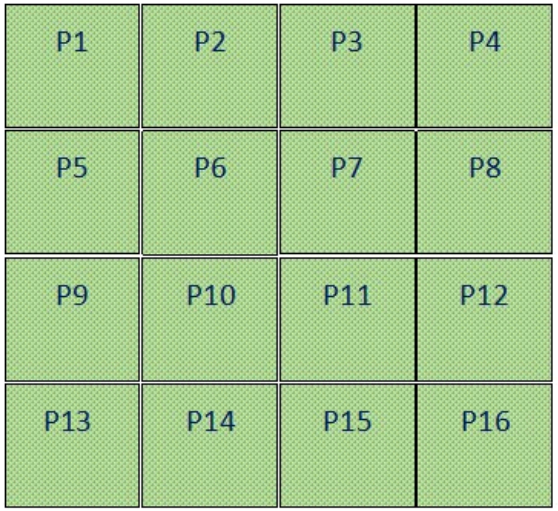


Table-2

Pid	id	fid
1	2	1
	1	2
2	3	6
3	4	1
	5	3
	6	3
4	6	5
	5	6

val partitionStrategy = PartitionStrategy.EdgePartition2D

Case 2: 计算二个好友间的共同好友数

Table-1

Pid	id	fid
1	2	1
	1	2
2	3	6
3	4	1
	5	3
	6	3
4	6	5
	5	6

Table-2

id	pid-List
1	1,3
2	1
3	2,3
4	3
5	3,4
6	2,3,4

Join by id

Table-3

id	friend-List
1	{2}
2	{1}
3	{6}
4	{1}
5	{3,6}
6	{3,5}

Table-4

id	pid-List	friend-List
1	1,3	{2}
2	1	{1}
3	2,3	{6}
4	3	{1}
5	3,4	{3,6}
6	2,3,4	{3,5}

Table-5

pid	uid-friendList
1	1->{2} 2 -> {1}
2	3->{6} 6 ->{3,5}
3	1->{2} 3 ->{6} 4 ->{1} 5 ->{3,6} 6->{3,5}
4	5 ->{3,6} 6 ->{3,5}

Join by Pid

Table-6

user	friend	total number of mutual friends
2	1	0
4	1	0
.....	.....	.....

Case 2: 计算二个好友间的共同好友数

- 根据shuffle的数据量来确定partition数
- 使用sort-based shuffle来提升性能和减少reduce的内存使用(Spark-1.2).
- 在大集群时当连接超时后选择重试来减少executor丢失的概率(Spark-1.2).
- 为避免executor被YARN给kill,通常需要设置 spark.yarn.executor.memoryoverhead.

edges/vertex	Nodes	Hive(Map Reduce)	Spark	Executor's memory,cores/partitions
50b/0.5b	200	12 h	2 h	15G/1/5000
50b/0.5b	1000	-	43m	15G/1/5000
100b/1b	200	24 h	6h	25G/1/10000
100b/1b	1000	-	2h	25G/1/10000

One Node, 55G memory,2*12T SATA disk,24*1.9GHz cpu,1Gbps network

Case 3: 用于ETL的SparkSQL和DAG任务

```
INSERT TABLE test_result
SELECT t3.d ,COUNT(*) FROM(
        SELECT DISTINCT a,b FROM join_1) t1
        JOIN (
           SELECT DISTINCT b ,c FROM join_2) t2
           ON (t1.a = t2.c)
        JOIN (
           SELECT DISTINCT c ,d FROM join_3) t3
           ON (t2.b = t3.d)
   GROUP BY t3.d
```

Compute	resources	time
Hive	200 Maps, 50 reduces(1 core, 3.2G memory)	**30 min**
SparkSQL	50 executors(1 core, 4G memory)	**5 min**

腾讯在Spark上的实践和优化

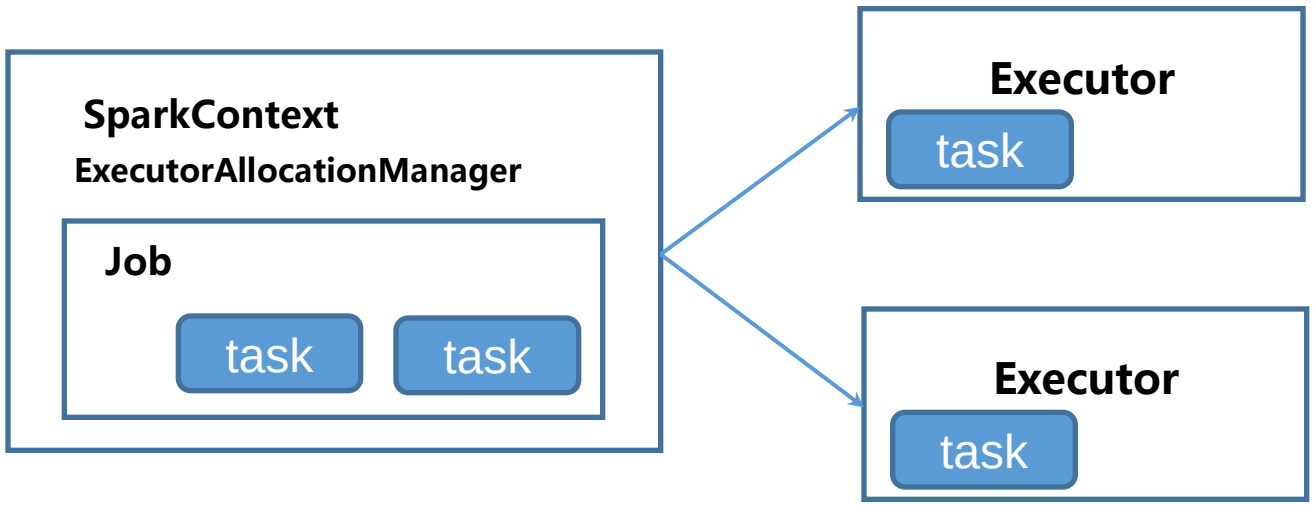
- 应用程序开发中的使用经验
- 对于ETL作业使用动态资源扩缩容特性
- Redcue阶段在Map阶段未全部完成前执行
- 基于数据的大小预测Stage的Partition数
- 为SparkSQL的每个Session分配一个Driver
- Count(distinct)的优化
- 基于排序的GroupBy/Join

应用程序开发中的使用经验

- 当有小表时使用broadcast Join代替Common Join
- 尽量使用ReduceByKey来代替GroupByKey
- 设置spark.serializer=org.apache.spark.serializer.KryoSerializer
- 根据Shuffle的数据量来设置Partition数，偏大些即可
- 大量的RDD在Union时使用new UnionRDD(sc, Seq(rdd))，防止StackOverflowError，而不是a.union(b).union(c).....
- 使用Yarn时设置spark.shuffle.service.enabled=true，减少shuffle数据重做的代价
- 将Akka的参数时间设长一些以及配置GC参数

对于ETL作业使用动态资源扩缩容特性

问题：在此这前,Spark任务在task执行前通过指定参数来启动固定数目的Executor,一旦达到用户指定值后保持住这个资源数,即使任务运行过程中的task数增加或减少,也不会去改变Executor的资源数目



spark.dynamicAllocation.enabled true

spark.dynamicAllocation.executorIdleTimeout 120

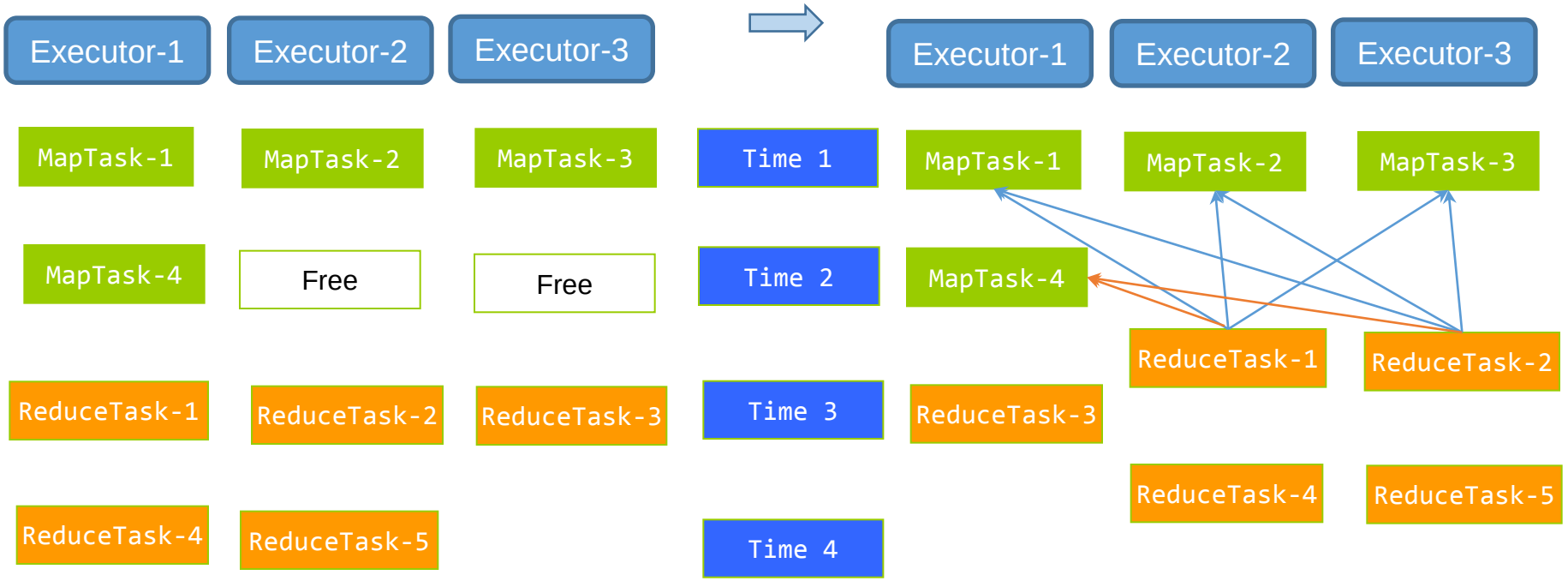
spark.dynamicAllocation.schedulerBacklogTimeout 10

spark.dynamicAllocation.minExecutors/maxExecutors

Redcue阶段在Map阶段未全部完成前执行

问题:当申请固定的executors时且task数大于executor数，这时存在资源的空闲状态

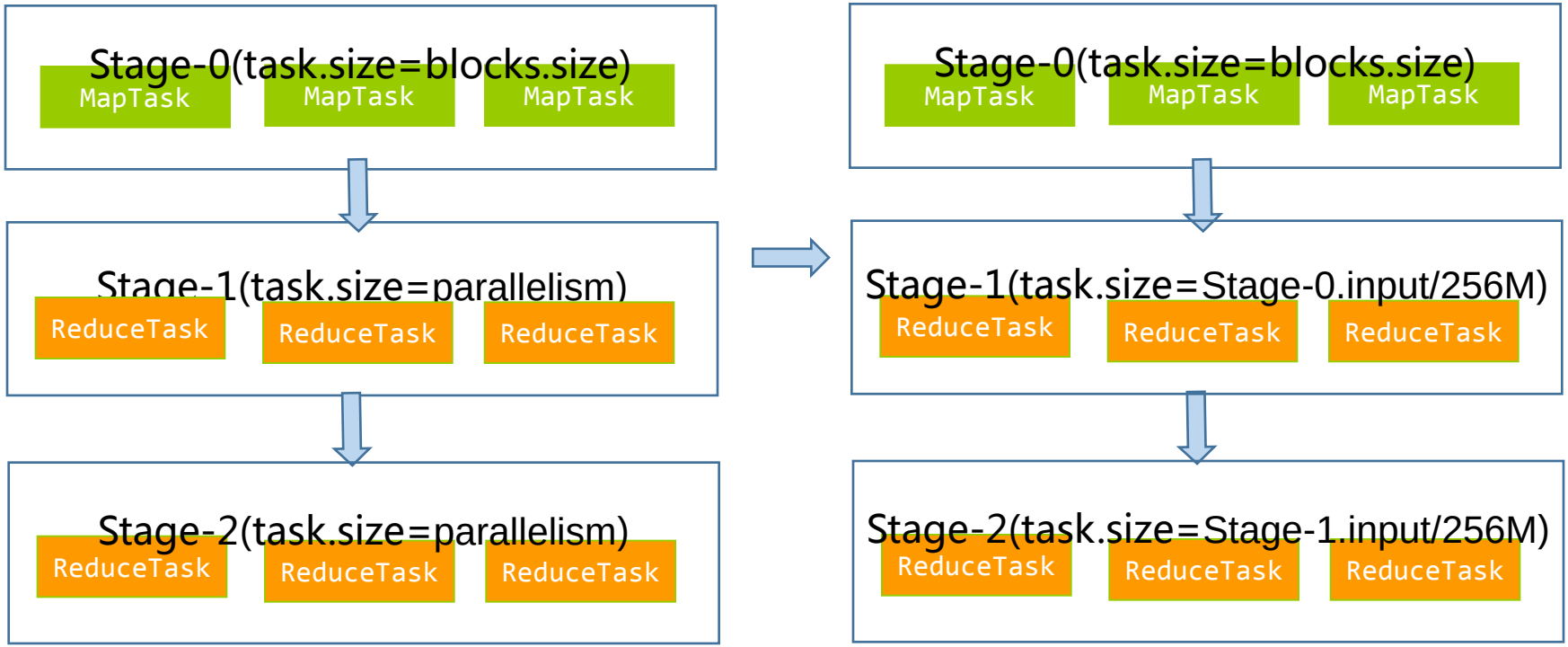
当executor没有map task需要执行时，即可在上面运行reduce task并提前去拉已完成的map数据。



基于数据的大小预测Stage的Partition数

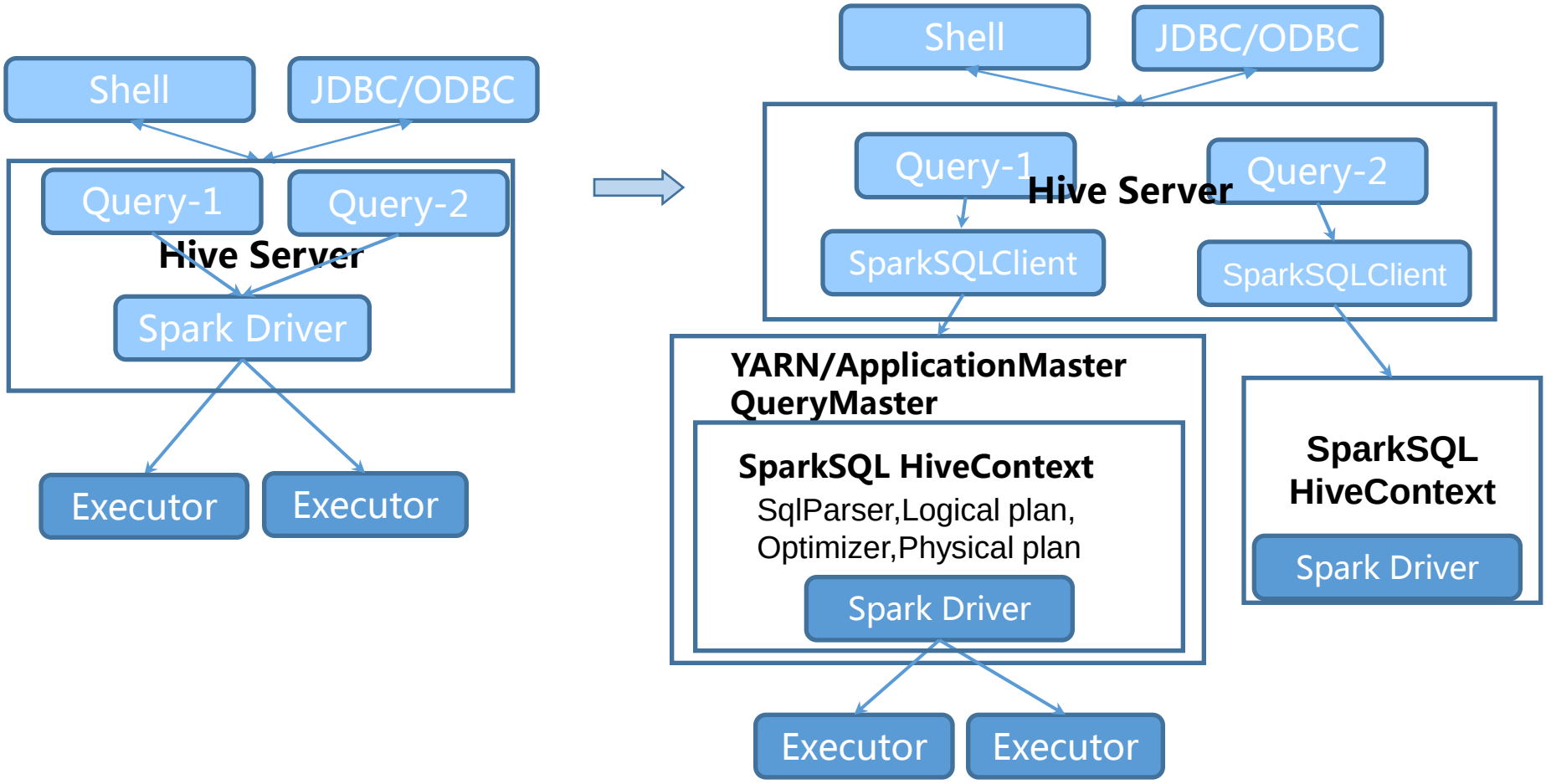
问题：

- 怎么去设置spark.default.parallelism/spark.sql.shuffle.partitions?
- 输入数据每天在变化且有大量的历史HiveSQL都没有设置partition数



为SparkSQL的每个Session分配一个Driver

问题: 一个HiveServer上所有的Query都使用一个Driver，导致query的内存和资源都会受到单个Driver的限制

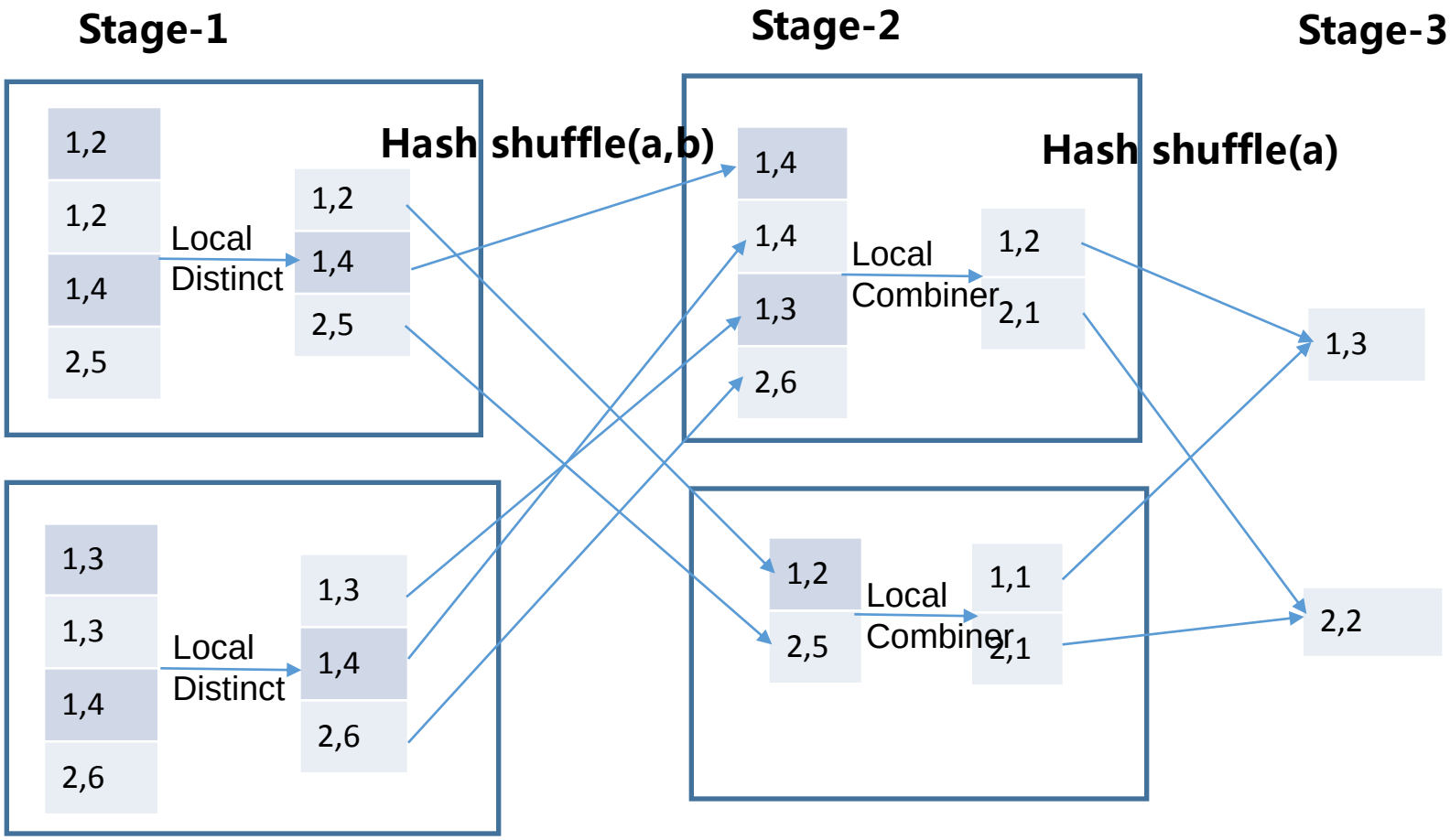


Count(distinct)的优化

语句: *select a, count(distinct b) from table1 group by a*

当b列的数据出现倾斜时，语句出现OOM

将该语句的执行计划改为如图所示



基于排序的GroupBy/Join

问题： 目前是基于Hash的GroupBy/Join，单个Key的所有Value必须能载入到内存中。往往在我们的应用中，有些Key存在倾斜，导致Value无法全部放到内存中

- sortMergeGroupByKey
- sortMergeJoin
- sortMergeLeftOuterJoin
- sortMergeRightOuterJoin

未来工作和计划

- 解决Driver的容灾与恢复
- 解决YARN上的memory reserved问题
- 持续优化SparkSQL
- 探索Spark Streaming上的应用案列
- 将我们的优化push到社区

Question

Thank you !