Spark App自动化分析和故障诊断

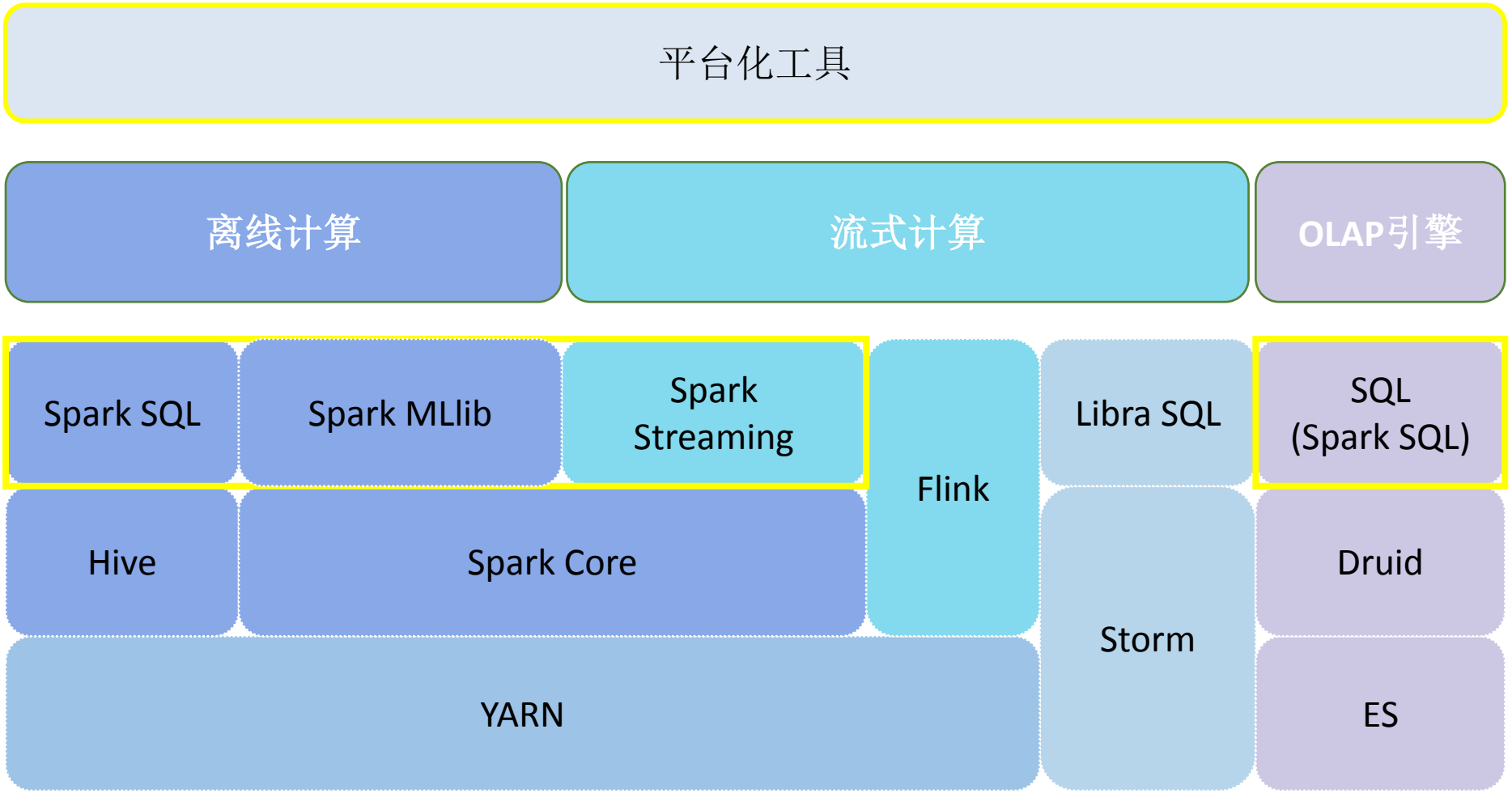
关于我们：

- ✓苏宁易购大数据平台研发中心
- ✓为集团提供大数据存储和计算能力

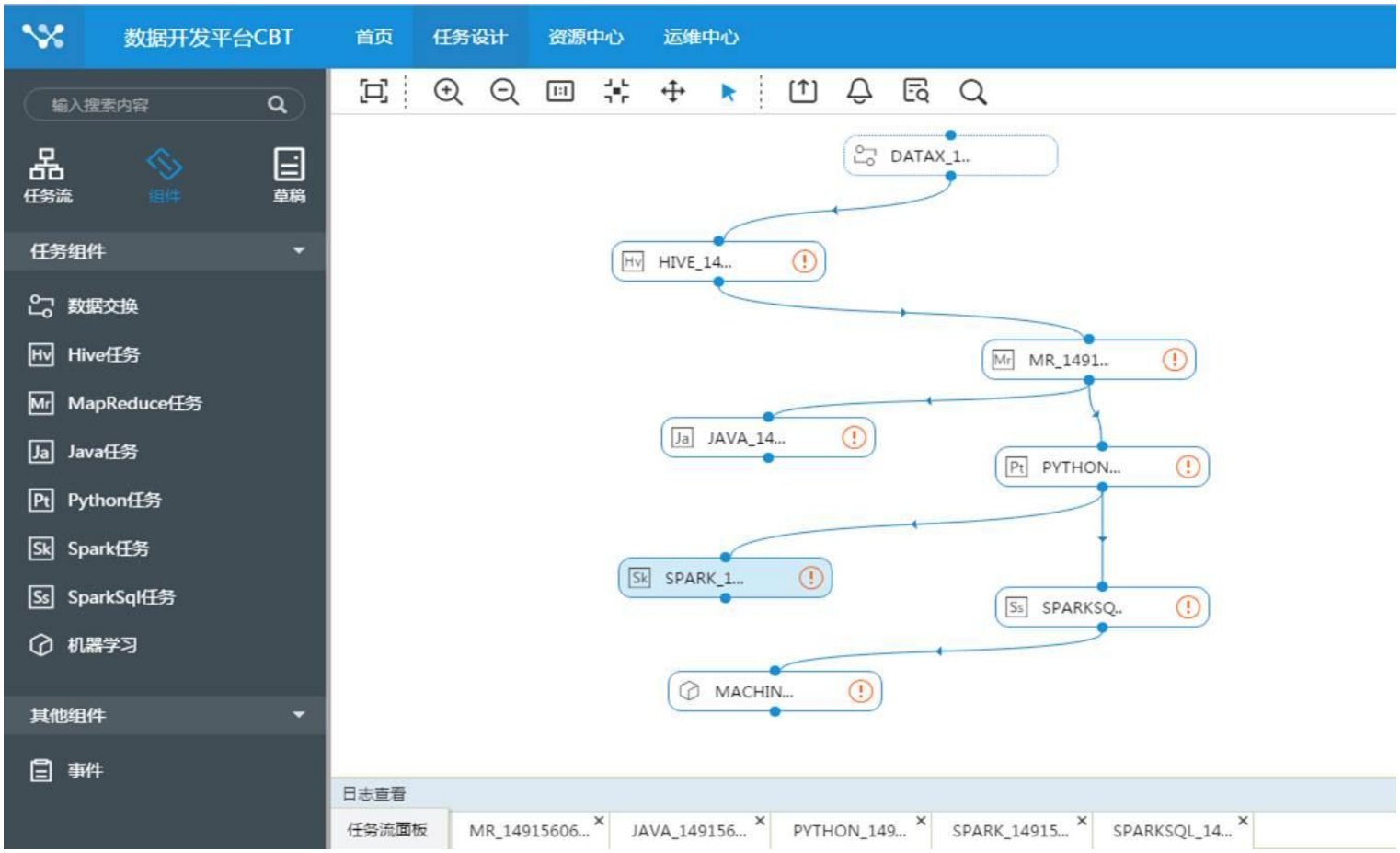
关于我：

- ✓ 陈泽 离线计算平台负责人
- ✓ 主要从事Spark，Flink，Druid，ES等计算组件研发工作。

- 苏宁大数据计算平台架构
- Spark平台化遇到的问题
- Spark自动化分析和故障诊断

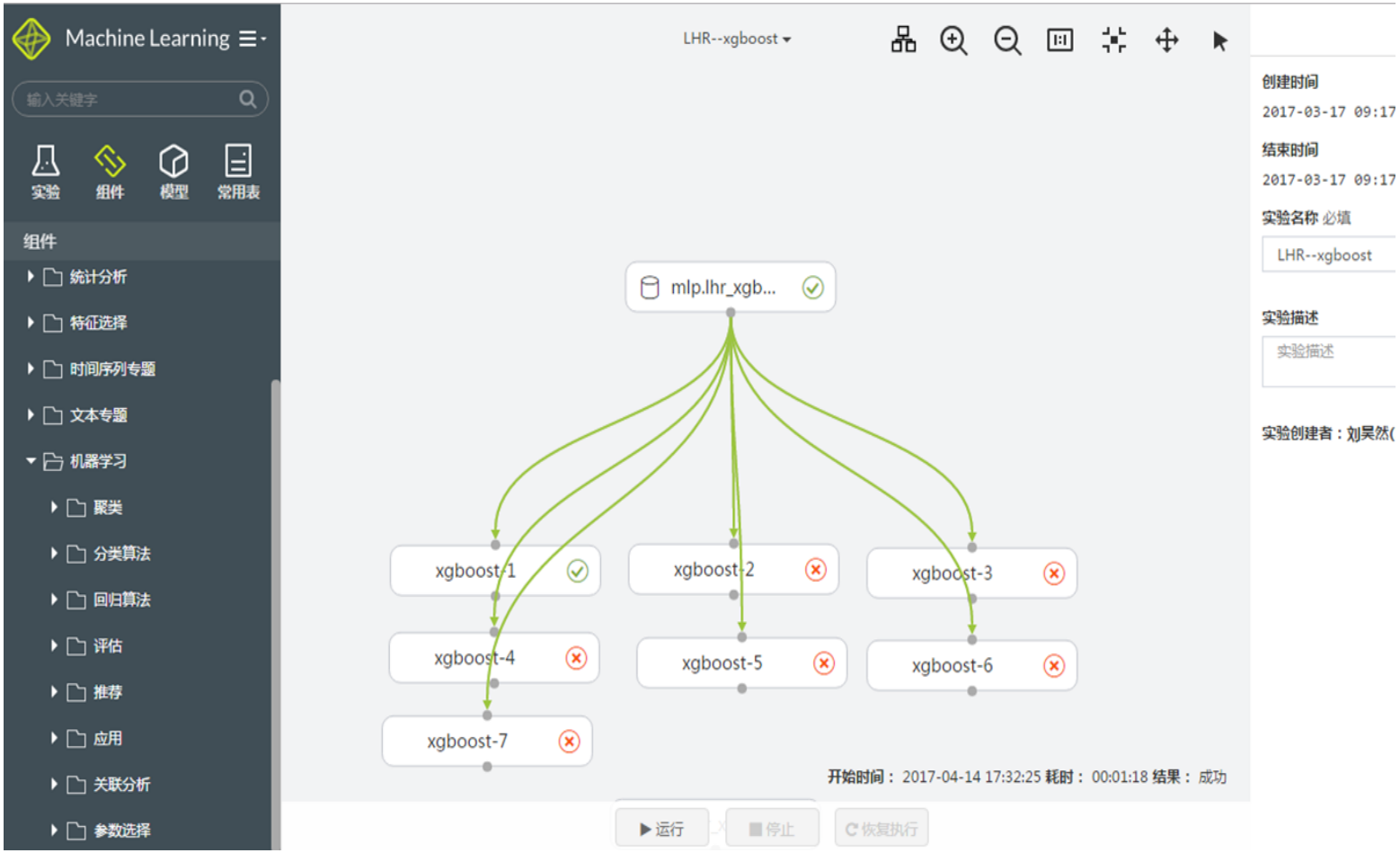

平台化工具
离线计算
流式计算
OLAP引擎
Spark SQL
Spark MLlib
Spark Streaming
Flink
Libra SQL
SQL (Spark SQL)
Hive
Spark Core
Storm
Druid
YARN
ES

CBT调度平台：为Spark、Spark SQL、Hive等离线任务提供任务流的配置、管理以及调度能力。



SSMP平台：针对Spark Streaming提供任务托管、资源托管以及监控报警的能力，为App 24小时Long Running提供稳定性保证。

机器学习平台：基于Spark MLlib、 GraphX和Streaming构建的在线机器学习平台，实现模型训练、调优、发布的统一。



集群规模：Yarn集群700节点；

任务规模：5W 任务/天；

处理的数据量：300TB/天；

任务类型	2016Q4任务数	2017年Q2任务数
Hive任务	10,682个	18,765个
Spark/Spark SQL任务	**250个**	**3000个**
Spark Streaming任务	**0个**	**29个**

现状：在苏宁，Spark应用越来越广泛

- 苏宁大数据计算平台架构
- Spark平台化遇到的问题
- Spark自动化分析和故障诊断

Spark平台化过程中，我们遇到很多问题：

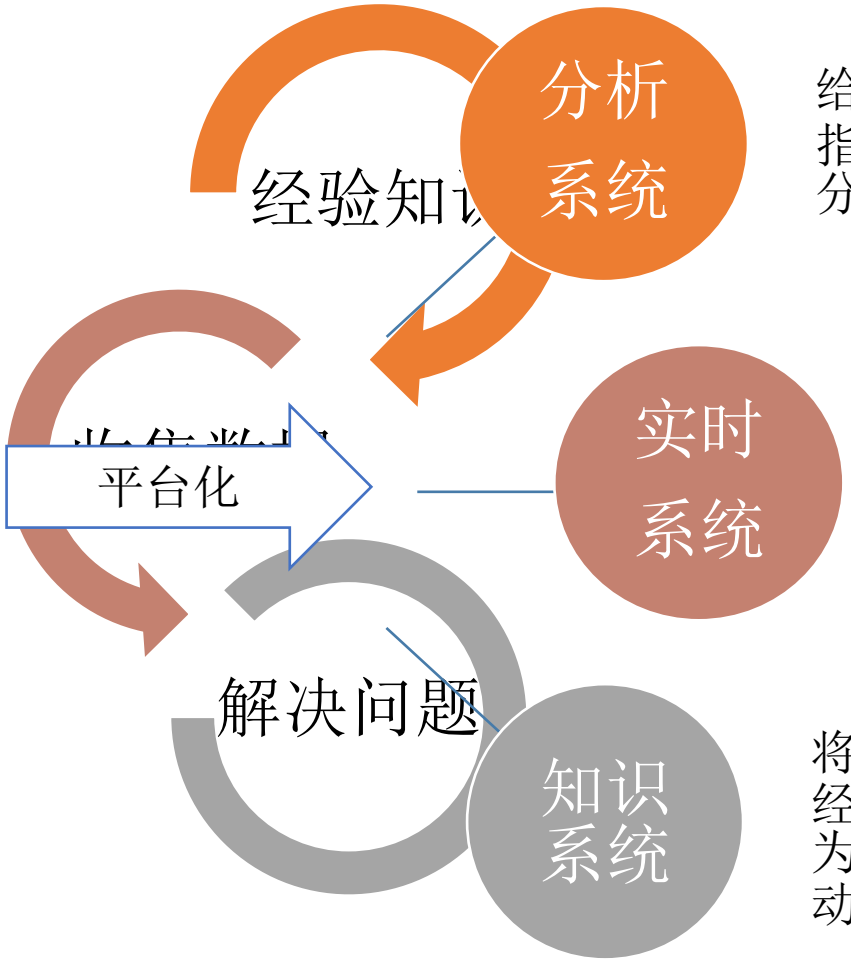
- 对Spark SQL的认识不够，导致过度的依赖Spark RDD层面的API。
- 对内存计算存在误解，不合理使用Cache机制，导致资源浪费。
- 不合理的Map、Reduce并发度设置，导致任务计算效率低，或导致任务调度Overhead过高，并产生过多小文件。
- 。。。。

业务方面

- 平台未提供数据来指导用户去优化Executor个数以及内存参数，导致平台资源利用率过低。
- 平台未提供数据来指导用户发现并解决任务执行过程中，存在的数据倾斜、HDFS Commit阻塞，及CPU，内存，网络资源瓶颈等各种问题。
- 。。。。

平台方面

Spark平台化遇到的问题



给业务App提供指标分析和事件分析能力。

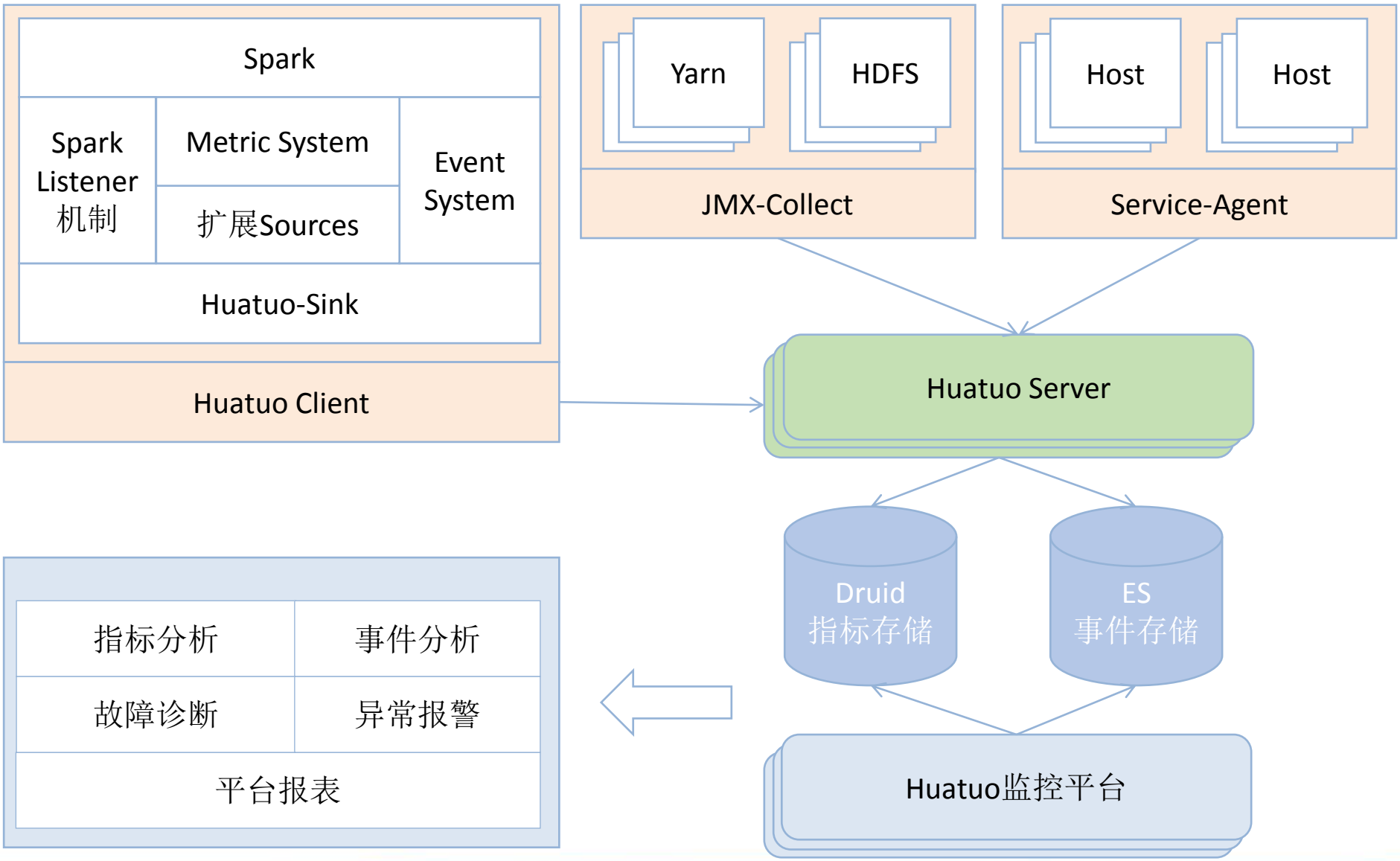
支持对正在运行的App进行数据采集、状态分析，以及实时故障诊断。

将平台积累下来的经验进行代码化，为后续App提供自动化分析工具

- 苏宁大数据计算平台架构
- Spark平台化遇到的问题
- Spark自动化分析和故障诊断

华佗—Spark App自动化分析和故障诊断系统

Spark自动化分析和故障诊断



- Druid是一种适用于时序化数据的OLAP分析引擎。
- 特别适合于**统计分析、系统监控**等业务场景。

timestamp	publisher	advertiser	gender	country	click	price
2011-01-01T01:01:35Z	bieberfever.com	google.com	Male	USA	0	0.65
2011-01-01T01:03:63Z	bieberfever.com	google.com	Male	USA	0	0.62
2011-01-01T01:04:51Z	bieberfever.com	google.com	Male	USA	1	0.45
2011-01-01T01:00:00Z	ultratrimfast.com	google.com	Female	UK	0	0.87
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK	0	0.99
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK	1	1.53

时间　　维度　　指标

- **支持：** TopN/GroupBy等聚合查询以及简单明细查询

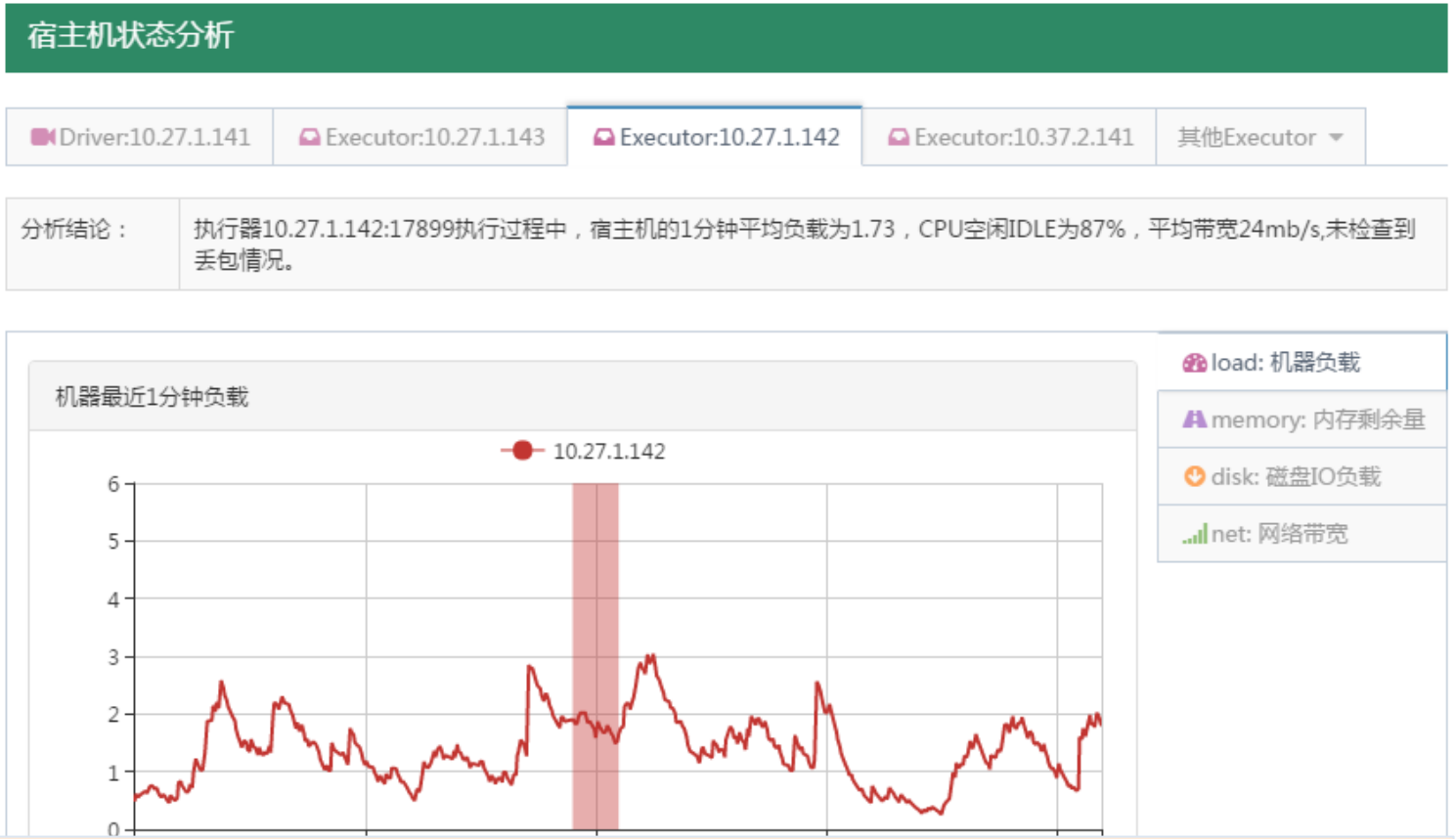
在苏宁：

目前Druid作为主要的OLAP引擎进行推广，支撑销售报表、金融自助分析、风控平台以及平台监控等十多个业务场景。

目前提供分析和诊断能力

资源角度	性能角度	故障角度
• 宿主机器状态分析 • HDFS资源使用分析	• Task耗时链分析 • 长尾Task分析	• Shuffle数据倾斜 • HDFS Commit阻塞 • Executor丢失 • Spill事件分析 • 高维Parquet写性能诊断 • RDD Size Estimator耗时诊断 • 任务事件流
• Driver/Executor进程状态分析 • 资源利用率分析	• 任务调度Overhead分析 • Reduce并发度分析 • JDBC并发度分析 Kafka读并发度分析	
• Cache 利用率分析 • Shuffle内存利用率分析		

资源角度：宿主机器状态分析

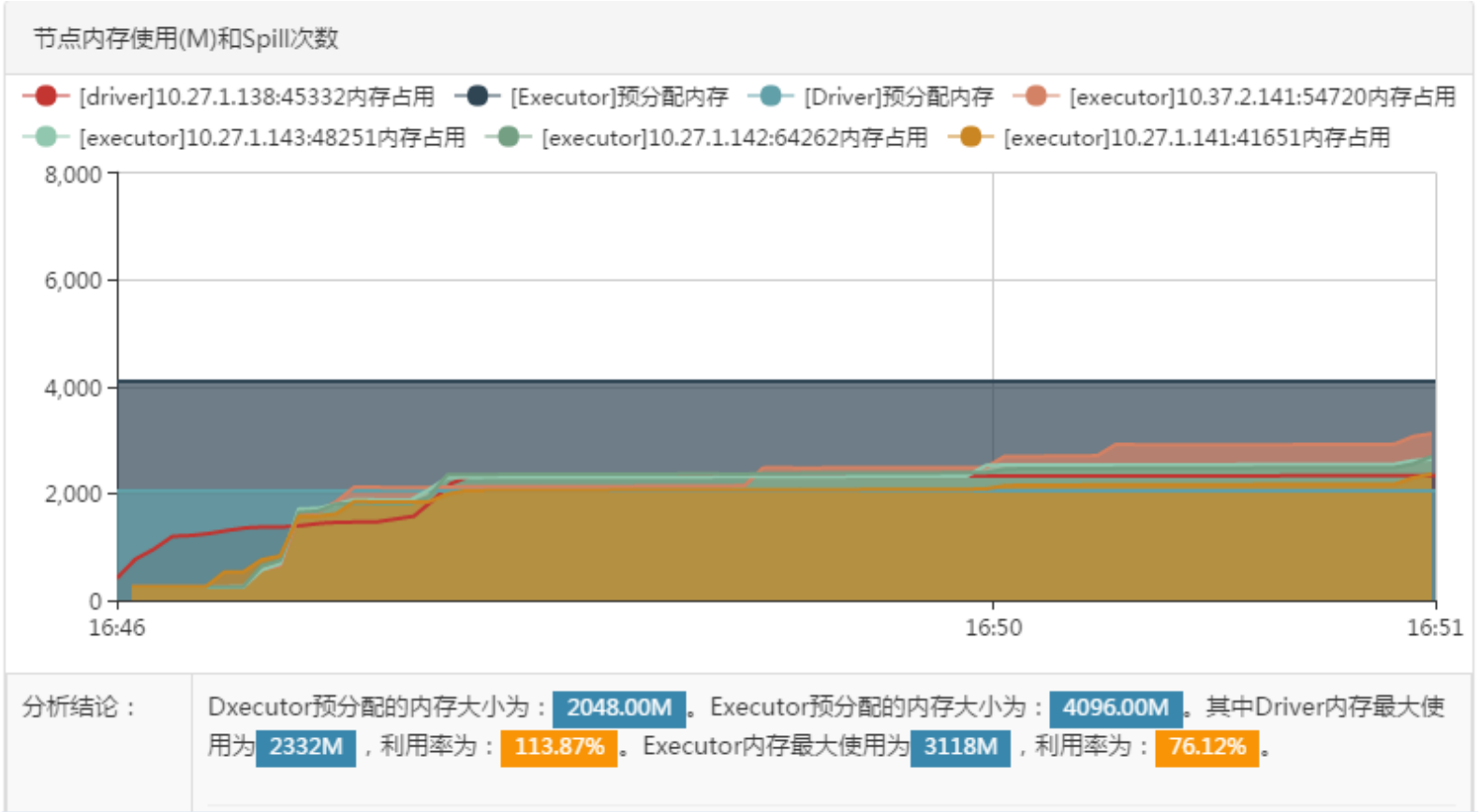


通过对任务运行过程中宿主机器状态进行分析，可以判断任务的性能瓶颈或故障是否与平台稳定性有关。

资源角度：Driver/Executor进程状态分析

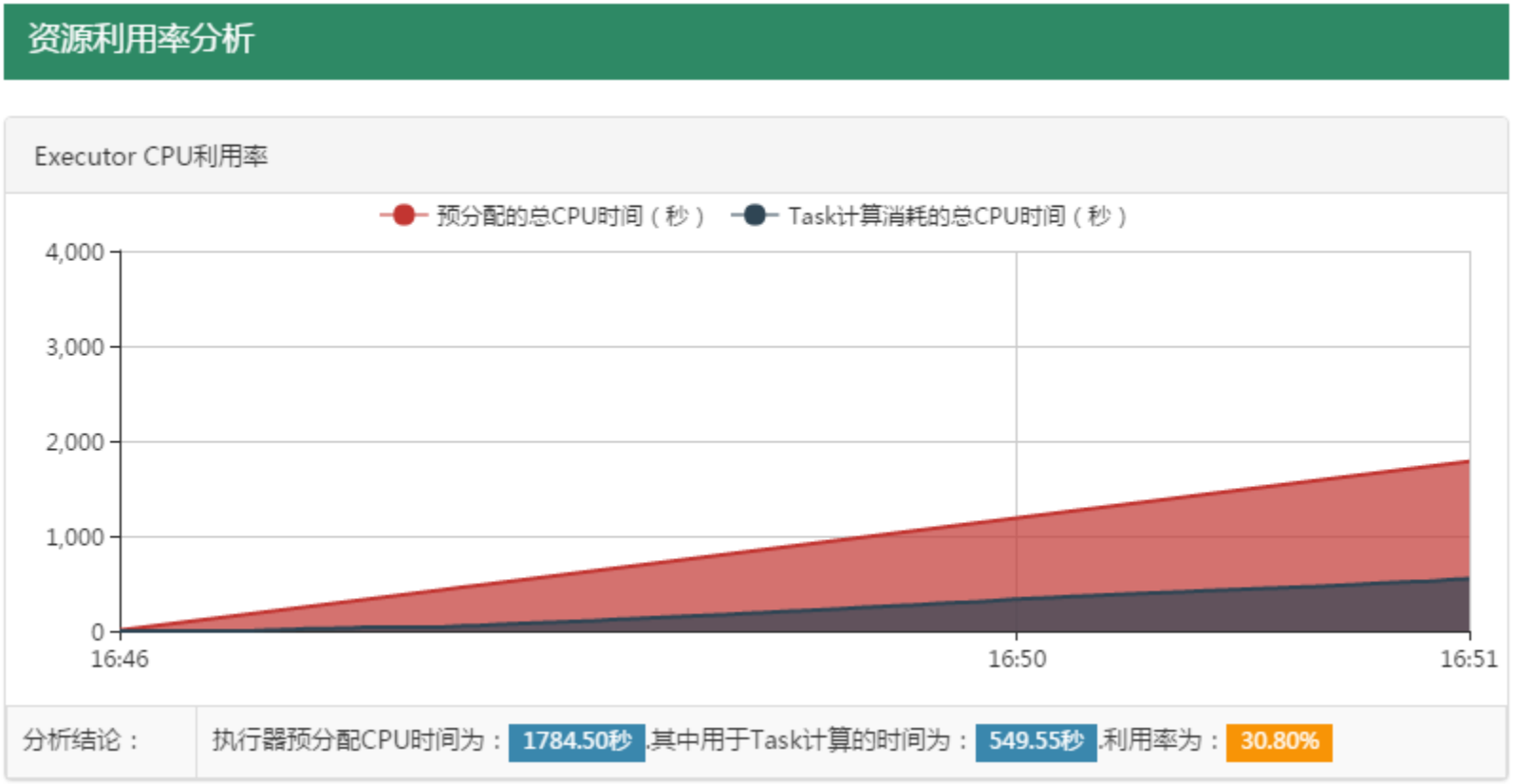
通过对任务运行过程中宿主进程状态进行分析，可以对进程的资源使用情况进行监控，并判断是否存在性能瓶颈、句柄泄露等异常。

资源角度：资源利用率分析



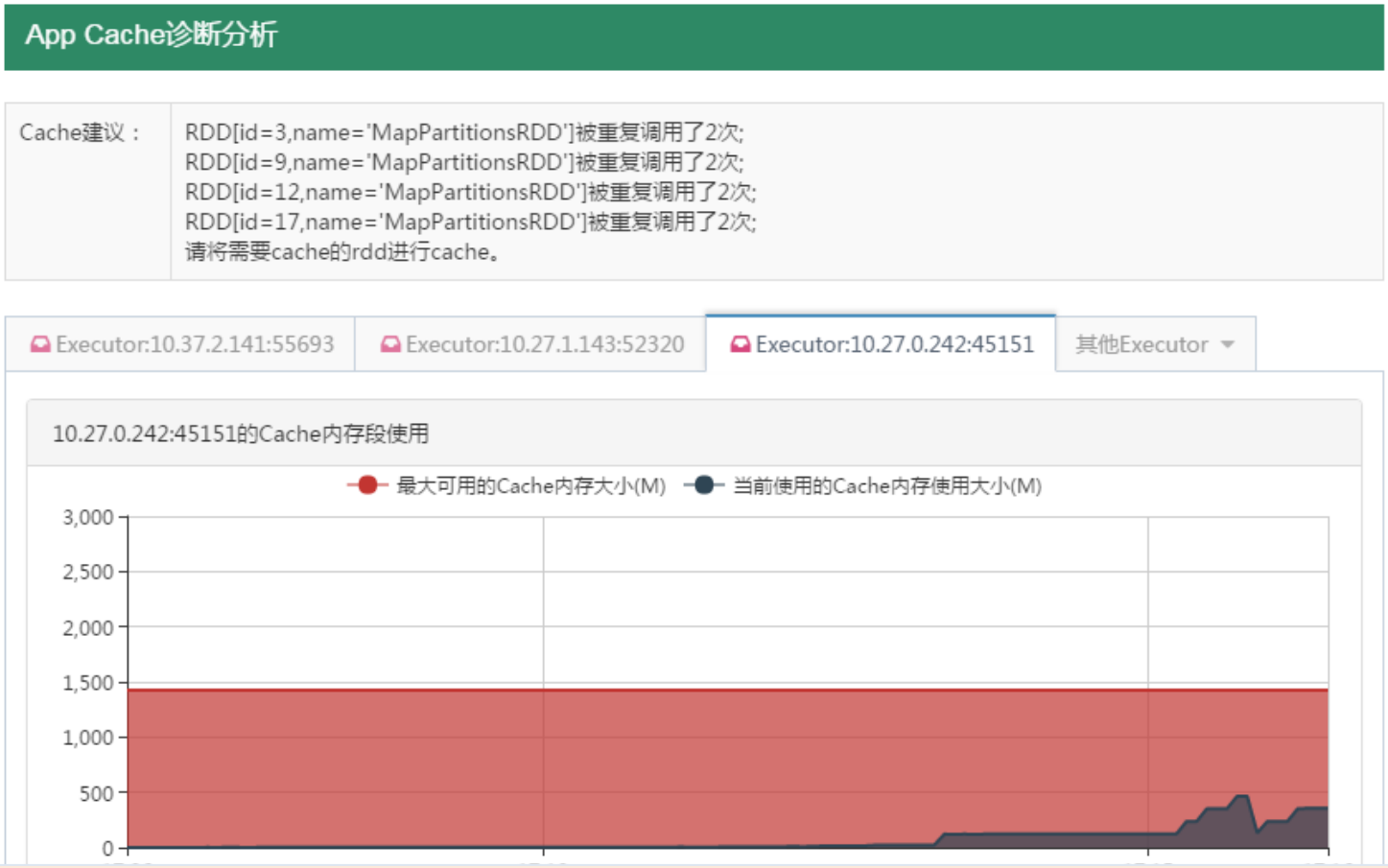
通过对任务运行过程中宿主进程的内存利用率进行分析，来指导业务优化任务的相关内存参数。

资源角度：资源利用率分析



通过对任务运行过程中Executor CPU时间片利用率进行分析，来指导业务优化任务的执行器个数。

资源角度：Cache利用率分析



对Executor Cache内存段的利用率进行分析；同时对任务的DAG图进行分析，对无用Cache进行诊断，以及针对调用2次以上的RDD给出Cache建议。

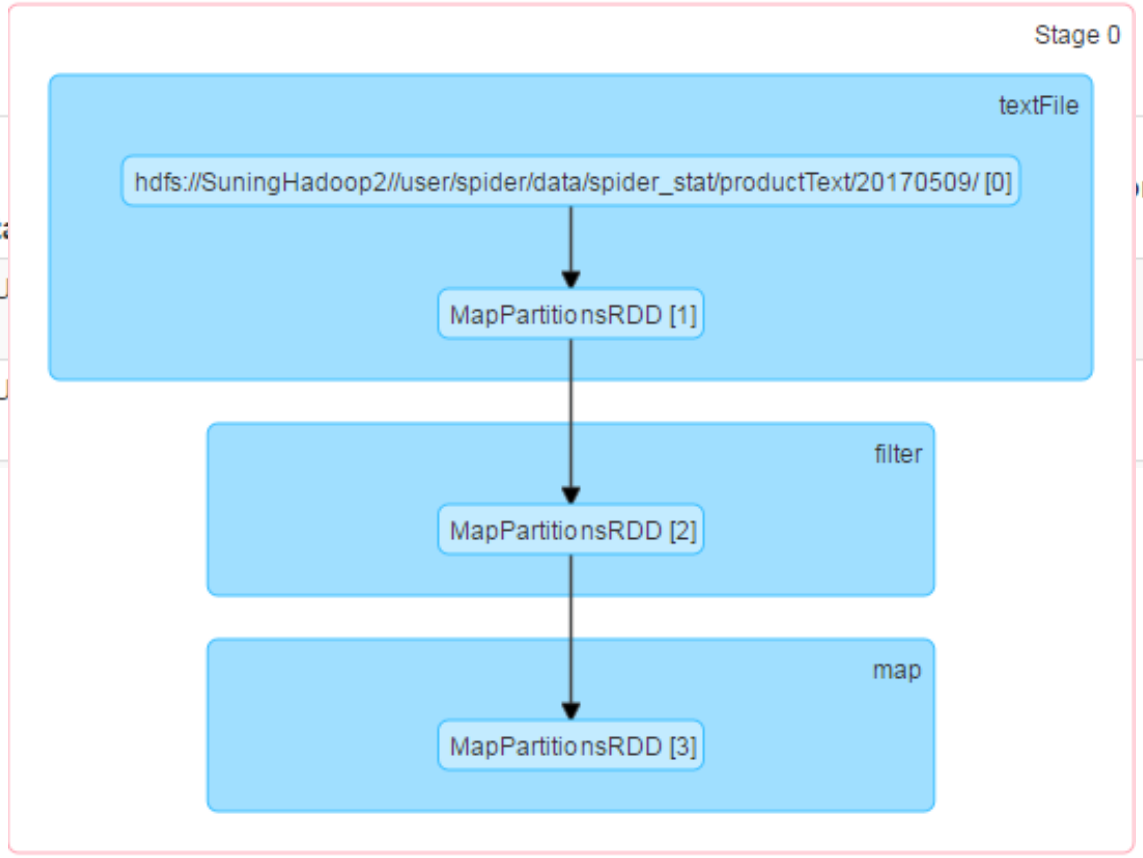
目前提供分析和诊断能力

资源角度	性能角度	故障角度
• 宿主机器状态分析 • HDFS资源使用分析	• Task耗时链分析 • 长尾Task分析	• Shuffle数据倾斜 • HDFS Commit阻塞 • Executor丢失 • Spill事件分析 • 高维Parquet写性能诊断 • RDD Size Estimator耗时诊断 • 任务事件流
• Driver/Executor进程状态分析 • 资源利用率分析	• 任务调度Overhead分析 • Reduce并发度分析 • JDBC并发度分析 Kafka读并发度分析	
• Cache 利用率分析 • Shuffle内存利用率分析		

性能角度：Task耗时链分析

- Task耗时分析

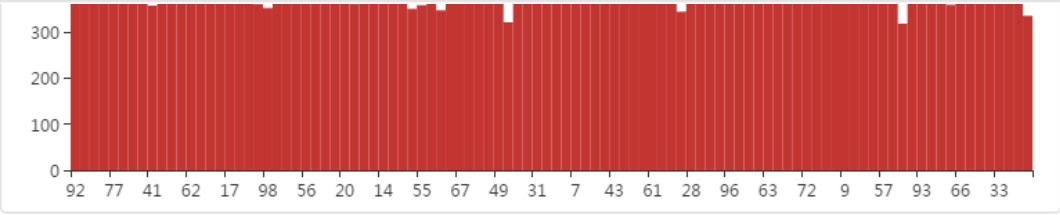
Index	ID	Attempt	Sta	...n	GC Time	Result Serialization Time
12	16	0	SU		0.4 s	0 ms
23	23	0	SU		0.4 s	0 ms



性能角度：Task耗时链分析

Task耗时链分析

TaskID	耗时链
0	HadoopRDD[0] at textFile at CommonStatRdd.java:73, 耗时： 16074ms. MapPartitionsRDD[3] at mapToPair at CommonStatRdd.java:189, 耗时： 2741ms. MapPartitionsRDD[2] at filter at CommonStatRdd.java:228, 耗时： 18181ms. MapPartitionsRDD[1] at textFile at CommonStatRdd.java:73, 耗时： 2012ms.
1	HadoopRDD[0] at textFile at CommonStatRdd.java:73, 耗时： 26744ms. MapPartitionsRDD[3] at mapToPair at CommonStatRdd.java:189, 耗时： 3858ms. MapPartitionsRDD[2] at filter at CommonStatRdd.java:228, 耗时： 29806ms. MapPartitionsRDD[1] at textFile at CommonStatRdd.java:73, 耗时： 2709ms.
10	HadoopRDD[0] at textFile at CommonStatRdd.java:73, 耗时： 16574ms. MapPartitionsRDD[3] at mapToPair at CommonStatRdd.java:189, 耗时： 2066ms. MapPartitionsRDD[2] at filter at CommonStatRdd.java:228, 耗时： 18634ms. MapPartitionsRDD[1] at textFile at CommonStatRdd.java:73, 耗时： 1355ms.



性能角度：Task耗时链分析

- 耗时链是基于RDD-Iterator来实现，WholeStageCodeGeneration目前还不支持。

```
//收集Iterator耗时
class Iterator[+T] extends Iterator[T] {
  def hasNext: Boolean; //collect Time
  def next(): T;//collect Time
}

//收集Iterator预先构造的耗时
data.build//collect Time
data.toIterator

//构建Iterator链
//比如data.build的耗时需要剔除parent的构建耗时
```

性能角度：长尾Task

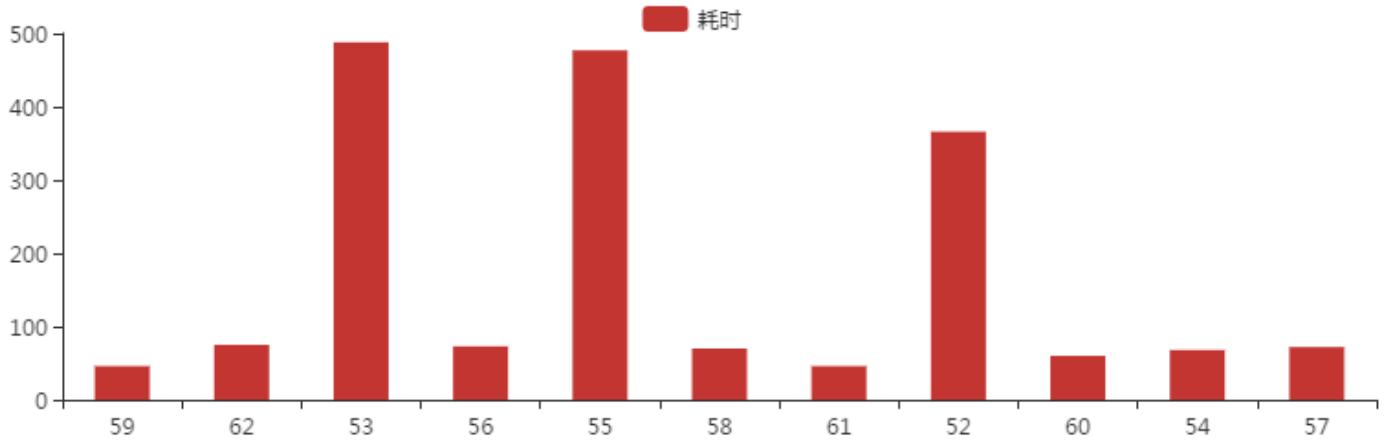
长尾Task运行诊断

JobId.StageID(3.15)检测到长尾

相关信息：这批Task当前运行时间，已经超过当前成功运行的Task的平均耗时的467%

Stage[jobid.stageid=3.15]执行结束。是否运行成功：**true（成功/失败:11/0）**。整体耗时：**678MS**。

每个Task耗时分布情况(MS)



性能角度：任务调度Overhead过大

任务调度OverHead诊断

任务调度OverHead 是指单Task计算耗时较短，但Task总数过多，导致消化过多时间在这些细小的任务调度上。 对于调度OverHead的任务，可以尝试进行 coalesce 操作，减小Task个数

JoibID.StageID	Stage耗时	Task个数	Task平均耗时	Task整体耗时	
1.2	1492ms	40	21.52ms	861ms	任务过碎
1.1	2444ms	1	764.00ms	764ms	正常
0.0	2688ms	1	961.00ms	961ms	正常

不管是Map操作还是Reduce操作，过于细碎Task会占用过多时间在调度以及等待调度上，导致任务调度Overhead过大。

性能角度：Reduce并发度分析

- 不合理的Reduce并发度，是导致任务调度overhead主要原因之一。
- Spark SQL 2.0+ 支持通过spark.sql.adaptive.enabled来设置Reduce大小自适应。

Reduce并发度诊断

Reduce并发度：在Spark 2.0版本之前，Reduce的个数都是通过 spark.default.parallelism 和 spark.sql.shuffle.partitions 两个参数进行配置。如果配置过大，将会导致下游产生很多碎片化的Task，或下游HDFS产生很多小文件；如果设置过小，将会导致单ReduceTask计算负载过大。建议配合 Shuffle数据倾斜诊断 和 任务调度Overhead诊断 进行组合分析。

任务ID	数据总量(KB)	并发度	平均数据量(KB)	最大数据量(KB)	最小数据量(KB)	评估
0.1	2158.88	40	53.97	60.08	45.05	建议优化
0.0	546867.16	40	13671.68	13868.37	13503.92	建议优化

目前提供分析和诊断能力

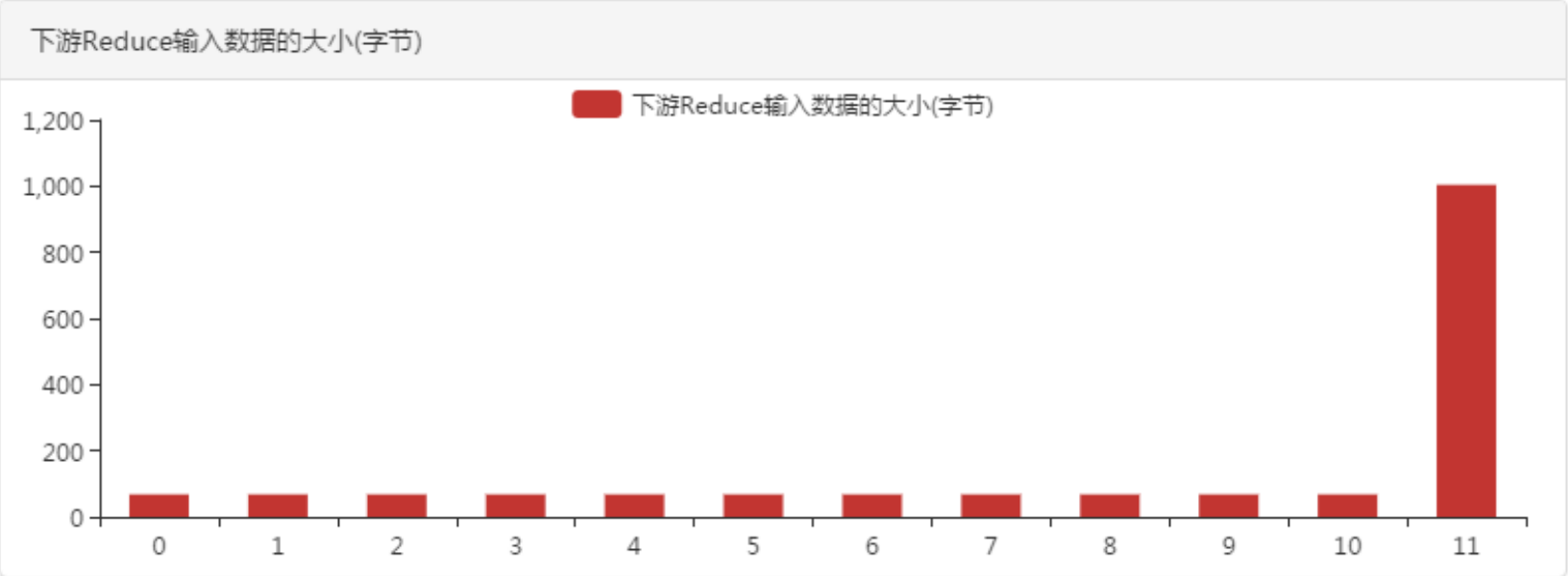
资源角度	性能角度	故障角度
• 宿主机器状态分析 • HDFS资源使用分析	• Task耗时链分析 • 长尾Task分析	• Shuffle数据倾斜 • HDFS Commit阻塞 • Executor丢失 • Spill事件分析 • 高维Parquet写性能诊断 • RDD Size Estimator耗时诊断 • 任务事件流
• Driver/Executor进程状态分析 • 资源利用率分析	• 任务调度Overhead分析 • Reduce并发度分析 • JDBC并发度分析 Kafka读并发度分析	
• Cache 利用率分析 • Shuffle内存利用率分析		

故障角度：Shuffle数据倾斜

- Shuffle数据倾斜是一个非常常见的故障Case。对于业务来说，应该主动去感知并处理可能出现的数据倾斜。

Shuffle数据倾斜诊断

事件： **[INFO]** ShuffleWriteStage[**9.17**]执行结束。任务耗时：**4918MS**。ShuffleWrite的Meta大小为：**228字节**。



故障角度：HDFS Commit阻塞

- 详见: MAPREDUCE-4815： If a job generates many files to commit then the commit Job method call at the end of the job can take minutes

HDFS Commit耗时及小文件诊断

Commit操作次数	Commit耗时最大值	Commit耗时最小值	Commit耗时平均值
1次	906290.85ms	906290.85ms	906290.85ms

分析结论：	Commit平均耗时为：906290.85ms，较为严重。请优化写文件个数，或将任务调度时间迁移到早上6点以后。

统计报表

资源报表

注意： 虚拟资源的利用率将确定是否可以对业务账户进行资源扩容。如果利用率较低，请先推进优化已有任务的逻辑实现或参数配置。

账户名	MR任务数	Spark任务数	预分配虚拟内存	预分配虚拟	虚拟内存峰值	虚拟VCore峰值	虚拟CPU利用率	虚拟内存利用率
aps	304	3	81920M	40	82944M	40	37%	7%
bi	230	6	184320M	50	187392M	22	39%	12%
erp	319	18	204800M	40	193536M	41	57%	54%
sousuo	974	148	429600M	40	144384M	41	43%	67%
spider	383	1	163840M	20	63488M	21	92%	10%
ztbd	121	144	409600M	60	423936M	61	64%	45%
bdapp	463	1	81920M	40	83968M	10	24%	3%
etl	259	2234	163840M	40	8192M	3	72%	38%
lbi	308	19	245760M	20	54272M	21	82%	76%

通过HDFS报表以及任务资源报表，与业务之间构成一个Feed-Back机制，推进业务主动对App的逻辑以及配置进行优化。

谢谢！