数领先机 智赢未来

DTCC 第九届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2018

2018.05.10 – 12 北京国际会议中心

IT168.com ChinaUnix ITPUB

SparkSQL在携程的实践

自我介绍

张翼

超过10年的互联网从业经验，超过7年的数据系统相关的经验

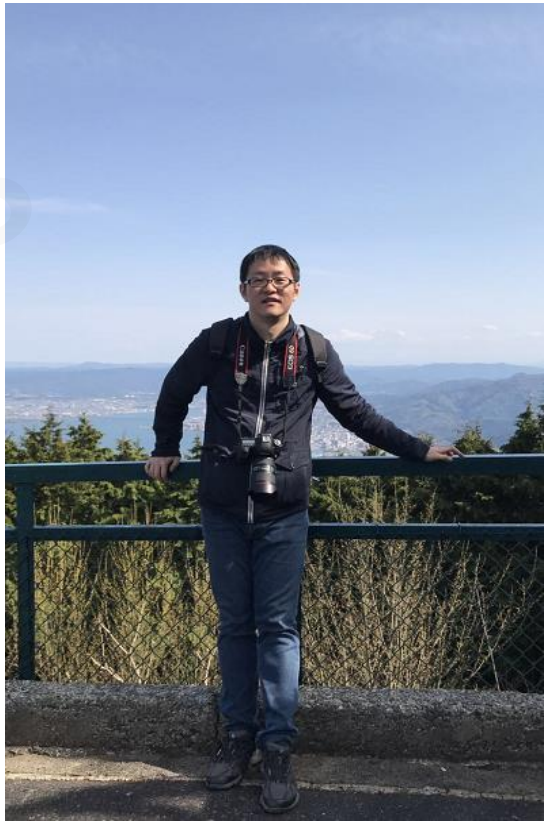
目前负责携程的大数据平台

浙江大学：本科 & 研究生

Ebay：软件开发工程师

大众点评：资深软件开发工程师 -> 技术经理

携程：技术总监



提纲

- 平台简介
- 总体方案和效果
- 经验分享
- 未来展望

携程大数据平台简介

携程大数据平台- 总体架构

数据开发系统 简介

集群规模：1300+

调度系统的Active任务数：75000+

每天运行调度任务实例数：130000+

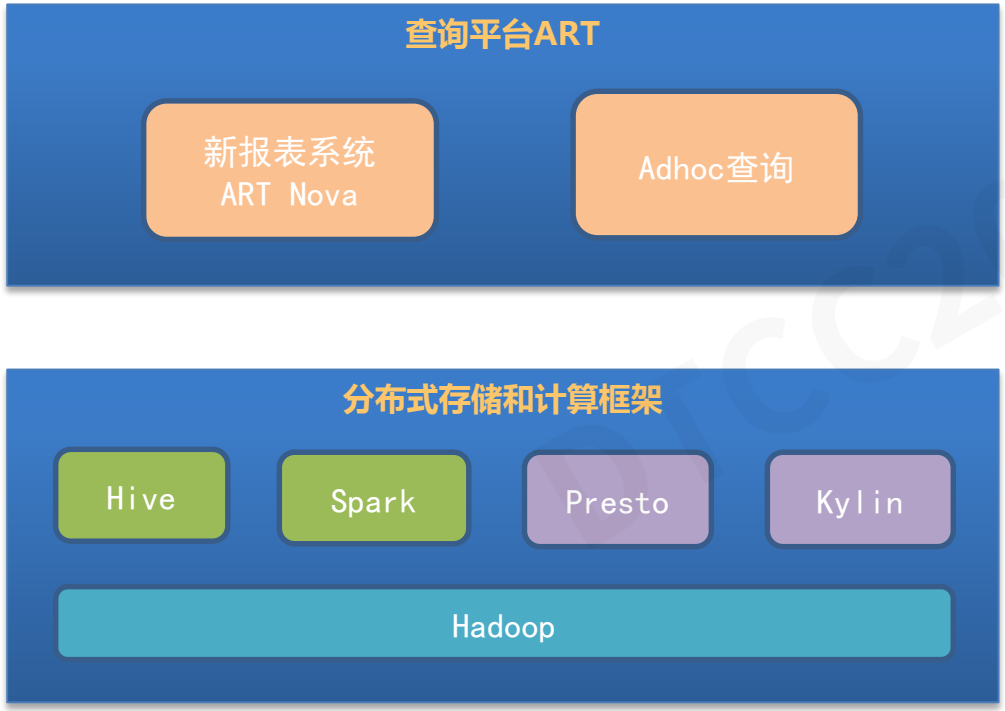
平均每天的MapReduce任务数：300000+

传输任务占比：~ 50%

ETL / 计算任务：~50%

2017 Q4绝大多数使用HQL（Hive）

数据查询系统 简介



1. Adhoc查询：10000+ / 天

   支持Hive / Presto

   2017Q4，85%使用Hive

2. 新建报表系统 ART Nova

   2017年12月正式上线

   报表数：

   设计最初考虑摒弃Hive

Hive的优点和问题

优点：

- “历史悠久”，被用户广泛接受
- 稳定性有保证

问题：

- 计算效率不高（慢！）
  - 相比spark / presto等新兴的计算引擎
  - HQL会转化为多个MR Job，MR Job之间需要数据落地
  - 代码架构混乱，增加优化方式的代价大

慢 行

候选方案

1. 给Hive换执行引擎：Tez / Spark
2. 换一个计算引擎
    - 能够兼容Hive Table（读，写，权限等等）
    - 能够保持HQL最大的兼容性

很长一段时间，对于开发平台的计算效率的提升，我们寄希望于Hive on Spark

SparkSQL / Presto则被用来作为“即席”查询的计算引擎

选择SparkSQL的原因

2017年9月左右，我们决定全面拥抱SparkSQL，将它作为开发和查询平台最主要的计算引擎

- 2017下半年，和携程规模相当，甚至比携程规模小的互联网公司完成或已经开始往SparkSQL的迁移
- SparkSQL的成熟，特别是2.2之后它的兼容性，稳定性，性能有很大的提升
- Hive on Spark除了Uber外很少有其他的用例
- Hive社区的衰落和Spark社区的繁荣
- Hadoop集群增长带来的稳定性问题基本解决

总体方案和效果

迁移SparkSQL的挑战

技术：

- 对于已经在运行的大量作业，如何将迁移的影响降到最小
  - 最重要：需要Enable灰度升级
  - SQL的语法上，尽量兼容Hive原有的语法
  - 权限控制需要兼容Hive原有的方式
- 大量的周边系统需要配合进行改造
  - 日志，Metrics的收集，监控系统
  - 对Dr Elephant的适配

团队：

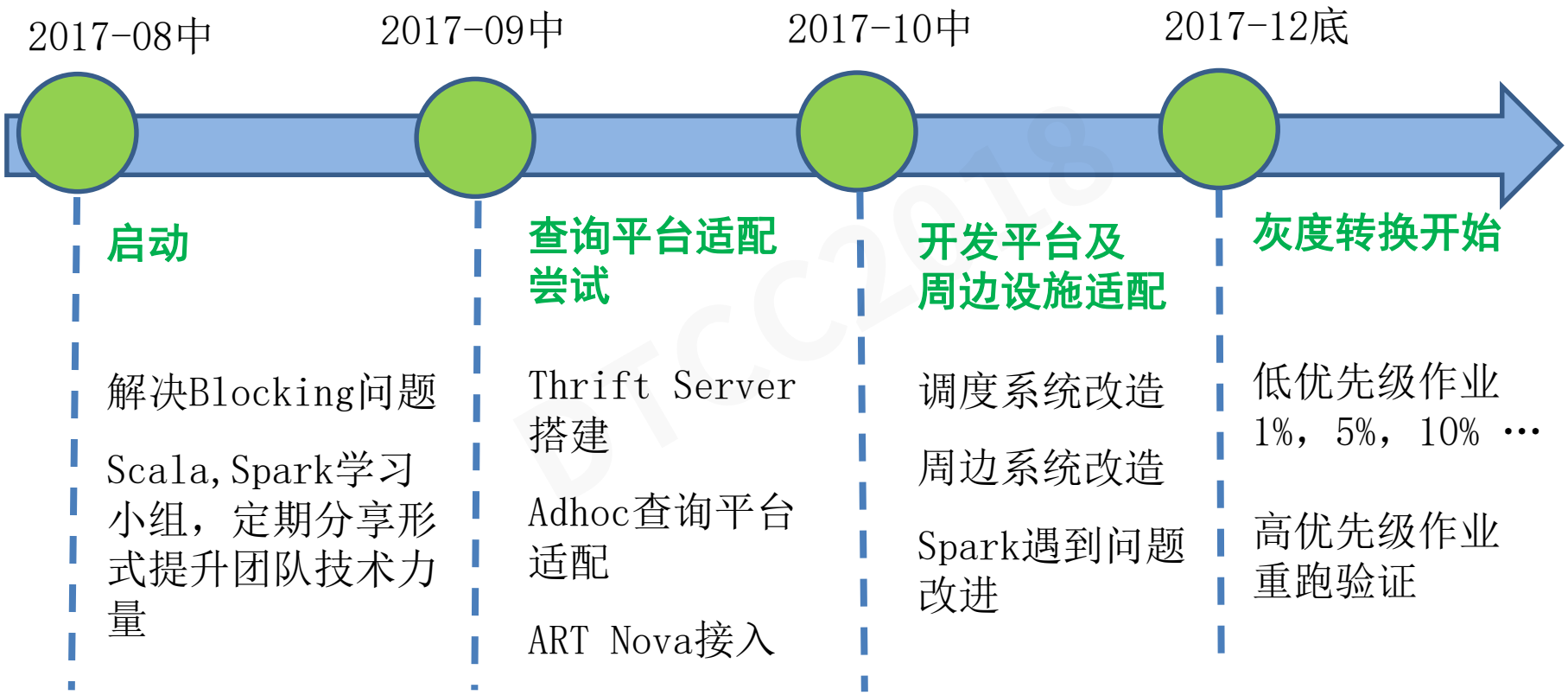
- 对于Spark SQL源码不熟悉，没有较大改动的经验；Scala也不太熟悉

总体迁移规划

我们将整个迁移过程，分为四个阶段：

1. 集中力量解决遇到的Blocking的技术问题，熟悉Spark和Scala，积累技术能力
    - 移植Hive的权限控制机制
    - 实现Thrift Server的impersonation
2. 在查询平台上开始初步尝试（Thrift Server）
    - Adhoc查询平台适配，采用Beeline方式
    - 新报表平台Art Nova使用JDBC直连的方式
3. 改造开发平台，并开始灰度推送
    - Spark cmd方式，1个作业脚本使用1个SparkContext
4. 全面完成升级，优化性能，处理长尾问题

迁移的时间线


2017-08中
2017-09中
2017-10中
2017-12底
启动
解决Blocking问题
Scala,Spark学习小组，定期分享形式提升团队技术力量
查询平台适配尝试
Thrift Server搭建
Adhoc查询平台适配
ART Nova接入
开发平台及周边设施适配
调度系统改造
周边系统改造
Spark遇到问题改进
灰度转换开始
低优先级作业1%，5%，10% …
高优先级作业重跑验证

进展和成果

使用占比：

- Adhoc查询工具，默认查询引擎，总查询量占比？%
- Art Nova中最主要的查询引擎，占比？%
- 开发平台的非数据传输作业？，转换为SparkSQL为？ 占比？%

性能提升（开发平台数据）：

- 计算效率提升**6-7倍**

报表：spark-hive运行总时长对比(最近七次平均运行对比) 数据		
hive-avg总计用时(小时)	spark-avg总计用时(小时)	节省用时(小时)
14609.77	2094.55	12515.22

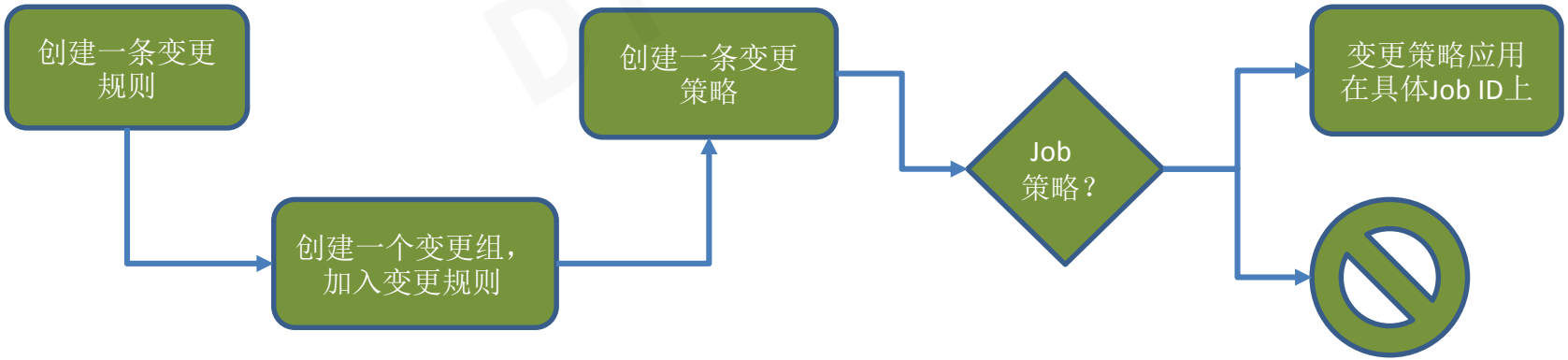
经验分享

开发平台对灰度变更的支持

开发了一套较为通用的可配置的灰度变更系统

- 支持环境变量，Hive/Spark conf，执行engine和shell cmd变更规则的创建
- 支持变更组，可以包括多条变更规则
- 支持多种类型的变更策略：全量，按照优先级的百分比，单个作业（优先级最高）
- 支持作业失败后，fallback到默认配置的功能

用户典型的操作流程：



实例 – SparkSQL灰度升级

1. 规则配置：

8	engine	SQL	spark-sql	更新 删除

2. 规则组配置：

7	8	更新 删除

3. 策略配置：

10	PRIORITY_RATE	7	T	low	1	hive	T	低优先级灰度推送hive类型作业使用spark-sql引擎执行策略	更新 删除
15	PRIORITY_RATE	7	T	high	0.7	hive	T	高优先级灰度推送hive类型作业使用sparksql引擎执行策略	更新 删除
4	BLACK_LIST	1	F				T	策略黑名单	更新 删除

4. 作业与关联配置：

4	2690	更新 删除

Spark上遇到的问题及优化

在整个灰度切换的过程中，我们遇到了很多问题，需有问题社区已经有了相关的解决方案（Apply社区Jira的修复超过30），还有很多问题需要我们自己解决

我们遇到的主要的问题：

1. 权限相关
    - Hive权限落地
    - Thrift Server Impersonation
2. 小文件合并
3. 资源利用率优化

1-1 – Hive权限的落地

Hive的权限控制（Authorization）模式：
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Authorization

1. Default Authorization（before Hive 0.13） 简称v1
2. SQL Standards Based Hive Authorization（Hive 0.13 or later） 简称v2
3. Storage Based Authorization in the Metastore Server
4. Authorization using Apache Ranger & Sentry

由于携程Hive使用的历史较长，所以我们目前主要使用的还是基于Default Authorization的权限控制体系，也修复了它的一些问题，比如说对于grant没有权限控制等问题

考虑到未来的需求，SparkSQL需要支持前两种的权限控制的方式

1-1 – Hive权限的落地

sql/core/src/main/scala/org/apache/spark/sql/execution/QueryExecution.scala

```
@@ -48,6 +48,7 @@ class QueryExecution(val sparkSession: SparkSession, val logical: LogicalPlan) {
    // Analyzer is invoked outside the try block to avoid calling it again from within the
    // catch block below.
    analyzed
+   authorized
    try {
      sparkSession.sessionState.analyzer.checkAnalysis(analyzed)
    } catch {
@@ -69,6 +70,11 @@ class QueryExecution(val sparkSession: SparkSession, val logical: LogicalPlan) {
    sparkSession.sessionState.analyzer.execute(logical)
  }

+ lazy val authorized: LogicalPlan = {
+   sparkSession.sharedState.externalCatalog.doPriCheck(analyzed)
+   analyzed
+ }
+
```

对于v2的支持非常简单

```
+   if ( state.isAuthorizationModeV2 ) {
+     val authorizer = state.getAuthorizerV2
+     val hiveOp = HiveOperationType.valueOf(getHiveOperation(logicalPlan).name())
+     val (inputsHObjs, outputsHObjs) = getInputOutputHObjs(logicalPlan)
+     val hiveAuthzContext = getHiveAuthzContext(logicalPlan, logicalPlan.toString)
+     authorizer.checkPrivileges(hiveOp, inputsHObjs, outputsHObjs, hiveAuthzContext)
+     return
+   }
```

ExternalCatalog ->

HiveExternalCatalog ->

HiveClient -> HiveClientImpl

+ 权限检查的方法

对于v1的支持相对复杂些

需要把hive中的相关逻辑移植到过来

400+ 行代码

1-2 – Thrift Server Impersonation

1. 我们在查询平台上主要使用Thrift Server的方式，目前社区版本的Thrift Server在启动时，同时使用超级用户的身份去启动executor

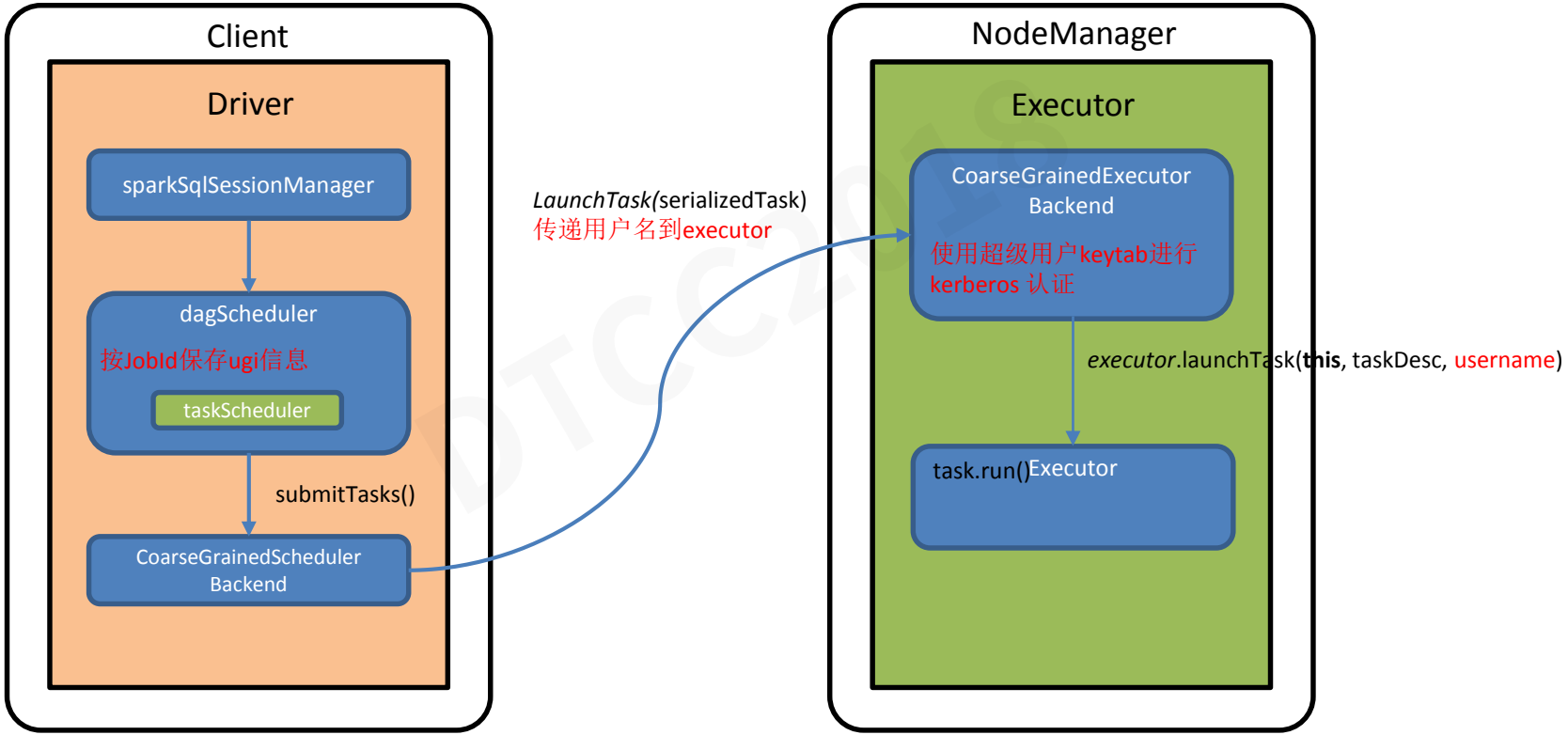
问题：数据写入都是以超级用户身份进行，造成数据权限的紊乱

2. Hortonworks的解决方案：Thrift Server只作为Proxy Server，在用户作业提交时再以其身份去启动AM和executor，以用户+connection id维度重用资源

问题：账号较多的情况下，executor的启停带来额外的开销

我们的方式：在Thrift Server启动时，预先启动AM和executor；预先把keytab分发到所有nodemanager；在executor端真正执行Task时，使用超级账户把用户impersonate成实际的用户

1-2 – Thrift Server Impersonation



2 – 小文件问题解决

Spark写数据时会生成很多小文件，这个会对NN产生巨大的压力：

- 在灰度30%的情况下（6000 Job / day），在不到3周的时间内就使NN的文件 + Block数飙升了近1亿，这个还是在每天有程序合并小文件的情况下
- 文件变小带来压缩率的降低，同样的数据会膨胀3-4倍

改动的方式也很简单：

- 在Insert Into Table或是Create Table as的情况下，如果本身没有RepartitionByExpression的话，就增加一个RepartitionByExpression的stage

入口：

sql/hive/src/main/scala/org/apache/spark/sql/hive/HiveSessionStateBuilder.scala

```
...  ...  @@ -75,6 +75,7 @@ class HiveSessionStateBuilder(session: SparkSession, parentState: Option[Session
75   75
76   76       override val postHocResolutionRules: Seq[Rule[LogicalPlan]] =
77   77         new DetermineTableStats(session) +:
     78 +       MergeSmallFiles(session) +:
78   79         RelationConversions(conf, catalog) +:
79   80         PreprocessTableCreation(session) +:
80   81         PreprocessTableInsertion(conf) +:
```

4 – 资源利用效率优化

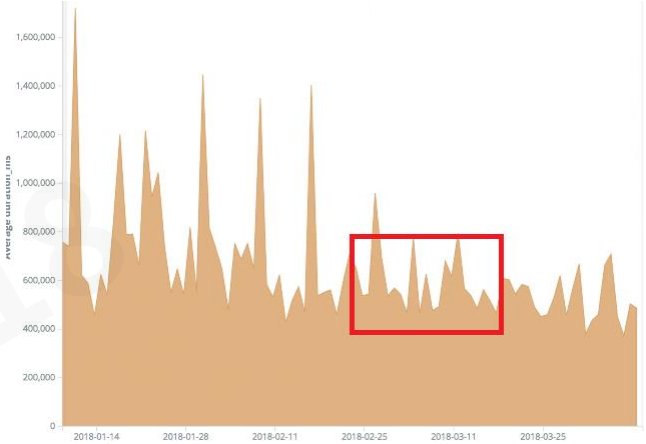
https://issues.apache.org/jira/browse/SPARK-22683

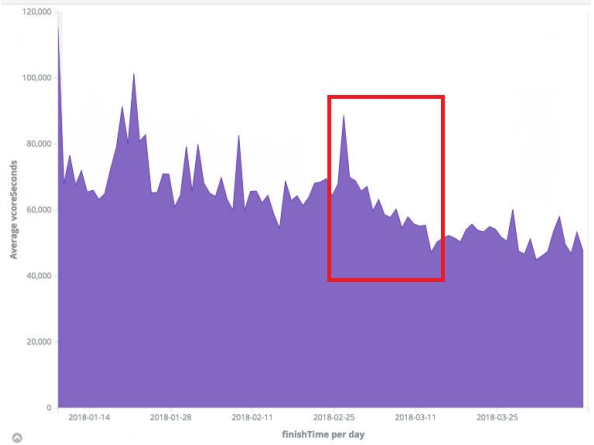
问题：Dynamic allocation的机制会根据task的数量去申请足够多的executor以达到最好的并行效果，最小化作业的latency

但是对于大量的小作业来说，executor allocation的消耗很大，而且很多的executor可能甚至没有做什么

解决方法：一个executor的slot能够支持更多的tasks

我们目前实际配置的值是 2 Tasks / Executors，从图上来看有30%左右





后续展望

后续展望

1. 优化作业内存的使用
作业转到SparkSQL之后，对内存的使用量也急剧上升，在某些事件点出现了应用内存分配满而无法分配更多作业的情况

解决思路：
- 根据作业历史的内存使用情况，在调度系统端自动设置合适的内存
- https://issues.apache.org/jira/browse/YARN-1011

2. 长尾Spark作业的性能提升
转换为Spark之后，有2.5%左右的作业会出现失败（400~），6%左右的作业运行效率不如Hive（1000~）

集中处理共性的问题，进一步提升Spark的计算性能

Q & A

Thanks ☺

讲师申请
联系电话（微信号）：18612470168

关注“ITPUB”更多
技术干货等你来拿~

与百度外卖、京东、魅族等先后合作系列分享活动

微学堂

让学习更简单

微学堂是以ChinaUnix、ITPUB所组建的微信群为载体，定期邀请嘉宾对热点话题、技术难题、新产品发布等进行移动端的在线直播活动。

截至目前，累计举办活动期数60+，参与人次40000+。

ITPUB学院

ITPUB学院是盛拓传媒IT168企业事业部（ITPUB）旗下
企业级在线学习咨询平台
历经18年技术社区平台发展
汇聚5000万技术用户
紧随企业一线IT技术需求
打造全方式技术培训与技术咨询服务
提供包括企业应用方案培训咨询（包括企业内训）
个人实战技能培训（包括认证培训）
在内的全方位IT技术培训咨询服务

ITPUB学院讲师均来自于企业
一些工程师、架构师、技术经理和CTO
大会演讲专家1800+
社区版主和博客专家500+

培训特色

无限次免费播放
随时随地在线观看
碎片化时间集中学习
聚焦知识点详细解读
讲师在线答疑
强大的技术人脉圈

八大课程体系

基础架构设计与建设
大数据平台
应用架构设计与开发
系统运维与数据库
传统企业数字化转型
人工智能
区块链
移动开发与SEO

联系我们

联系人：黄老师
电　话：010-59127187
邮　箱：edu@itpub.net
网　址：edu.itpub.net
培训微信号：18500940168