

说说PB级生产上重要的Spark 3.x性能优化方向

Flink 4月30日

以下文章来源于小猴学Java，作者小猴



小猴学Java

软件技术、职业生涯、学习方法

他们正在使用Apache Hudi(仅列举部分，排名不分先后)



本篇郑重承诺以下！

1. 本篇涉及的调优方向在PB级的集群上测试有效。
2. 本篇涉及的调优方向均为本人亲自调试。
3. 本篇涉及的调优方向基于数亿级别数据于数亿级别大表关联。
4. 本篇涉及的调优方向对应的调度任务规模数千个。
5. 本篇涉及的调优方向基于一个几十人的大数据开发团队。

- Background
- 血淋淋的战场
- 数据倾斜真的加加随机数就解决了吗？
 - 与分析师的对话
 - 与ETL的对话

image-20210428233623877

这么好用的广播join是不是到生产就无敌了吗？难道就是这么牛逼，参数一调然后就所有问题就解决了呢？

血淋淋的战场

我可以告诉大家，No！看似简单无比但到生产上却是寸步难行。因为你可能会不定时遇见：

- 一个应用提交到YARN，Spark Web UI卡死，连任务执行得什么样都不知道了。
- 大批量的作业在集群中调度，上千个作业作业还没有提交到YARN就已经全线瘫痪。
- 为了调错，增加了日志的输出详细程度，却不曾想巨大的作业数量，几千行、内外嵌套几十层的物理执行计划树分分钟数GB的日志文件，一眼物理执行计划望不到边。
- 决策部门分析人员一个SQL提交上来，弹性资源怎么也弹不动、公平调度再也不公平。
- 日入TB级的数据量，运维忙得焦头烂额，磁盘可以扩、经费可以批，但机房不能随便扩。数据你敢删吗？一旦出现问题，即使你是外包也可以让你在行业内脱几层皮。
- 当你想要把教程里面说的，加随机数，打散随机值，发现你根本连下手的地方都没有。因为数仓分层的上层，你会发现是一层噩梦，那里面的SQL会分分钟把你的脸打得稀烂。
- 他们说大数据都是内网隔离，不需要做安全控制，提交SQL的业务人员一个SQL把你的生产数据Overwrite得无影无踪，你想甩锅给业务部门？你会发现作为一个开发，自己居然连说话的机会都没有。
- 他们说公司有钱，资源无穷无尽。做核心业务分析的人员一个大SQL，数以百计的JSON字段解析，几十亿与几十亿的历史数据关联与回写，吃光吃尽所有集群资源，一个SQL卡掉整个集群。
- 他们说公司的内存足够，就算倾斜也没事，能跑下来，结果发现一到生产，发现一个Task里面正默默地shuffle着几十亿上TB的数据。难道你要去问问运维：兄弟，能给我把机房的机器内存升级到10TB吗？
- ...

我有太多血的经验，没有我的公众号里说过。你可能觉得，屁啦！这种场景只有国内最大的互联网公司里有吧。但你可能真的不知道，一些行业、一些公司对数据质量、数据可靠的要求性远超互联网公司。

所以，我想跟大家说说这样的环境，我们在Spark上能做什么，而不是把那些所谓的原则，未经真正超大规模数据测试然后就言辞凿凿。

数据倾斜真的加加随机数就解决了吗？

教科书总是教科书，理论也总是理论。其实，如果一个分析量大一点的平台，你会发现，加随机简直是天方夜谭。你想想就知道了，大家来看看这几个笑话。

与分析师的对话

开发：你提交这SQL不行啊，SQL里面有倾斜，你得处理啊。

分析师：好厉害哦！怎么处理呢？

开发：加随机数啊。就是通过加随机数打散，然后就快了。

分析师：不亏是开发！就是高端...可是，这个得你帮我哦，真的不懂....

开发：看我的。

- 开发打开那个SQL文件，傻眼了，40张表关联。随机数加在哪儿呢？
- 为什么不像大家说的那样，某几个key关联了大量数据？而是看起来差不多？
- 这写的什么SQL，完全看不懂。为什么不是group by和count/sum，而是这么多看起来像公式的东西？

开发挣扎了一会，很快就放弃了。因为他无从下手。对于真正的分析业务，他们根本不懂。

所谓地从业务上去解决，这种情况，根本就不要想了。

与ETL的对话

ETL：兄弟，咱们这个引擎升级后，以前只要几分钟跑完的任务，现在需要几个小时才能跑下来。咋回事？

开发：大哥！你这怎么没有开启广播啊，把广播打开，速度飞起！

ETL：这么牛吗！赶紧来帮我试试吧！

开发：看我的。

- 开发找到调度SQL的脚本，设置了开启广播。兴高采烈地等着任务的结束。
- 数小时过去了，等来的是一堆莫名其妙的异常。
- 广播似乎怎么都无法生效。

开发挣扎了一会，很快就放弃了。他不知道所错。

因为，即使改完了这个，还有好很多个...

学会分析倾斜

看Web UI中的SQL执行图。

重点关注每个Spark Plan的：min、medium、max。

如果发现某个节点时间、或者数据量跨度过大，恭喜你，杀手来了。

Analyze语句

有人会说：MapReduce比Spark慢的原因是因为MapReduce是要写入到磁盘，而Spark在内存。

有人会说：MapReduce要不断地向YARN申请资源，而不是像Spark这样可以固定分配资源。

真的是这样吗？其实，如果真的跑海量数据分析，很多时候，作业跑不下来的原因都不是磁盘的。而且，你真的确定Spark都是在内存而不会到磁盘吗？

真的是这样吗？你可以看看申请一千个容器的时间需要多久？它占了运行时间的百分比是真的高吗？

都不是。

真正影响性能是一个优秀的执行计划。而优秀的执行计划不是简单分析分析SQL就能够得出来的。只有拿到数据的分布情况、数据规模等统计信息，应用了大量的Spark优化策略，才能得到。而分析的环节尤其重要。不管是Spark引擎，或者是一些MPP计算引擎，例如：impala。

```
ANALYZE TABLE table_identifier [ partition_spec ]  
COMPUTE STATISTICS [ NOSCAN | FOR COLUMNS col [ , ... ] | FOR ALL COLUMNS ]
```

这个语句是会分析表中的数据，并将分析后的统计数据保存到Hive的MetaStore中（其实就是TableProperties）。我们可以执行NOSCAN，也可以针对所有列分析，也可以针对指定的列分析。NOSCAN是非常粗粒度的，可以在表或者分区级别。

大家可以试试，对比下ANALYZE和没有ANALYZE的执行计划。然后再检查下分析和不分析的执行时间，你会发现，差距会比较大。

这个是生产环境中，大规模数据处理所必须的。谨记！不然很多调优将不会生效。

SparkSQL自适应分区

自适应分区是Spark 3中推出的很靠谱的功能。因为它可以动态的根据数据量来适配任务。而不是固定分区数量，导致在生产环境中一些分区空转。还有一个非常重要的解决倾斜的配置。默认是不开启，强烈建议打开。

```
# 开启自适应分区
spark.sql.adaptive.enabled      true
# 开启自动处理倾斜
spark.sql.adaptive.skewJoin.enabled      true
# 倾斜因子
spark.sql.adaptive.skewJoin.skewedPartitionFactor 5
```

你会发现开启它会和不开启任务的执行会有很大的差别。你可能会惊恐，为什么很多的任务被skip掉了？

Spark SQL牛叉得可以动态地检测分区的数据量，超过阈值自动调度任务处理。在有倾斜的场景，性能数十倍、数百倍增长。

广播

广播优化是经常提及的。我们可以控制广播阈值来定义哪些表该广播、哪些不该广播。但需要考虑的是：广播怎么就知道表的大小呢？答案是：你需要做ANALYZE。不然，这将成为一个笑话。

```
spark.sql.autoBroadcastJoinThreshold # 默认10MB
```

注意：

spark.sql.autoBroadcastJoinThreshold	10MB	Configures the maximum size in bytes for a table that will be broadcast to all worker nodes when performing a join. By setting this value to -1 broadcasting can be disabled. Note that currently statistics are only supported for Hive Metastore tables where the command <code>ANALYZE TABLE <tableName> COMPUTE STATISTICS noscan</code> has been run, and file-based data source tables where the statistics are computed directly on the files of data.
--------------------------------------	------	---

image-20210429010047967

小猴告诉大家，千万不要把广播设得太大，不然，你会发现，跑大作业的时候，你的Web UI都会死掉。广播对Driver是有压力的。Driver出现问题，有人想过去换GC，其实，大部分这是徒劳的。即使你用G1。结果一样是一摊屎。

针对一些比较特殊的，我们可以强制让JOIN走SHUFFLE。所以，下面这些HINT，你得知道。

```
SELECT /*+ COALESCE(3) */ * FROM t
SELECT /*+ REPARTITION(3) */ * FROM t
SELECT /*+ REPARTITION(c) */ * FROM t
SELECT /*+ REPARTITION(3, c) */ * FROM t
SELECT /*+ REPARTITION_BY_RANGE(c) */ * FROM t
SELECT /*+ REPARTITION_BY_RANGE(3, c) */ * FROM t
```

CBO

CBO是基于代价的优化，对执行计划的优化也是可观的。它会根据数据的大小、分布以及算子的特点选择出来比较好的物理执行计划。打开CBO，一些大的作业的执行计划也可以看到明显的变化。

```
spark.sql.cbo.enabled true
spark.sql.cbo.joinReorder.enabled true
spark.sql.cbo.planStats.enabled true
```

spark.sql.cbo.enabled	false	Enables CBO for estimation of plan statistics when set true.
spark.sql.cbo.joinReorder.dp.star.filter	false	Applies star-join filter heuristics to cost based join enumeration.
spark.sql.cbo.joinReorder.dp.threshold	12	The maximum number of joined nodes allowed in the dynamic programming algorithm.
spark.sql.cbo.joinReorder.enabled	false	Enables join reorder in CBO.
spark.sql.cbo.planStats.enabled	false	When true, the logical plan will fetch row counts and column statistics from catalog.
spark.sql.cbo.starSchemaDetection	false	When true, it enables join reordering based on star schema detection.

image-20210429011534383

Resource Dynamic Allocation

集群的资源是宝贵的，买服务器、建机房的时候就知道了。所以，我们不要上来就分配特别大的资源。当跑完一些作业后，应该把资源让给其他应用。配合FAIR Shceulder，可以将资源尽可能地复用。

```
spark.dynamicAllocation.enabled true
...
```

注意哦！一个定在生产上把yarn spark shuffle service配置好。否则，你会发现，Container会不停地释放、申请、释放、申请。任务根本执行不下去。

参考：<https://spark.apache.org/docs/latest/configuration.html#dynamic-allocation>

优化项是要不断尝试，不断调试，然后才能得出来一个比较靠谱的参数。期间，你需要不断看官方配置文档、不断看Spark源码、跟踪调试，你才能从业务、成本、稳定、资源、性能之间找到平衡。注意：重点强调下。

业务、成本、稳定、资源、性能之间找到平衡。

不断提升自己的技术格局。

如果有人告诉你，我这版参数万能。请各位兄弟们，呵呵就可以了。

亲们，今天下班真的有点累了，今天在实验平台的新功能，因为文档大量缺失，基本靠分析源码解决。里面掺杂着Spar各种大数据组件、与JavaEE的代码，从前端到后端、到操作系统、再从前端到大数据组件，看得眼睛都有点花了。但没有感觉到精神上疲惫，因为我觉得我足够地爱技术，这与技术好坏无关，这与职位无关，就是喜欢。就是身体上有点疲惫了。

这会凌晨1点半了，有点迷糊了。就先给大家分享到这里。

以上。



喜欢此内容的人还喜欢

说说大数据要知道的Lambda架构

Flink

“千万别乱买网红纯欲裙！否则...” 啊啊啊买家秀真就刺激！

吐槽星君

这是咱们说好了的

bibi动物园