**1 Profiling使用说明**

* 1. **Profiling设计目标**

**理解当前Hive执行SQL查询时性能瓶颈是我们第一阶段的工作目标。我们把Hive运行的时间消耗归为三方面:**

1. **访问磁盘数据造成的时间消耗, 比如从HDFS中对查询数据的读取过程消耗。**
2. **SQL操作计算所造成的时间消耗, 比如select, join, group by操作的计算过程消耗。**
3. **网络传输数据所造成的时间消耗, 比如shuffle阶段从集群各节点获取中间数据的过程消耗。**

**根据集群环境，查询负载的不同，以上三方面都有可能成为性能瓶颈。理解Hive执行过程在各个部分的详细时间占比可以指导下一步利用GPU的优化方案设计。**

**我们首先调研了当前Hive平台下的开源profiling工具。Tez-UI是一款Apache开源的网页UI插件，可提供在Tez执行引擎下每个map/reduce任务的起始终止的timeline信息。BaBar是一款支持mapreduce/spark的资源利用分析工具，可提供一次SQL查询执行过程中I/O和CPU资源消耗等情况。然而, 以上两者无法提供较为详细且能够反映磁盘访问/网络传输/SQL计算消耗的分析结果。因此我们自主设计出一个基于探针的profiling工具，以到达分析性能瓶颈的目的。**

* 1. **Profiling环境配置**

**当前实验集群环境是基于hive-3.1.2和tez-0.10.1。Profiling工具中涉及到的对于源码的部分改动，也是基于以上版本中的开源代码。**

* 1. **Profiling运行说明**

**文件夹jar, extract中提供了相关profiling工具。现在说明运行profiling的步骤:**

1. **修改源码, 编译获得jar包(可跳过)。**

* **文件夹 jar/中提供编译好的jar包, 可直接用于步骤2。**
* **提供的jar包有: 1) hive-exec-3.1.2.jar; 2) tez-mapreduce-0.10.1-SNAPSHOT.jar; 3) tez-runtime-internals-0.10.1-SNAPSHOT.jar; 4) tez-runtime-library-0.10.1-SNAPSHOT.jar。**
* **源码修改细节请阅读章节2，修改方案可扩展到Hive和Tez的相近版本。**

1. **在已编译的Hive和Tez工程中代替jar包。**

* **Hive工程下lib/路径中, 可找到原生hive-exec-3.1.2.jar。 用jar/hive-exec-3.1.2.jar替代此原生jar:**

**cp jar/hive-exec-3.1.2.jar hive-3.1.2/lib/**

* **Tez工程下tez-mapreduce/路径中, 可找到原生tez-mapreduce-0.10.1-SNAPSHOT.jar。用jar/hive-exec-3.1.2.jar替代此原生jar:**

**cp jar/tez-mapreduce-0.10.1-SNAPSHOT.jar tez-0.10.1/tez-mapreduce/target/**

* **Tez工程下tez-runtime-internals/路径中, 可找到原生tez-runtime-internals-0.10.1-SNAPSHOT.jar。用jar/tez-runtime-internals-0.10.1-SNAPSHOT.jar替代此原生jar:**

**cp jar/tez-runtime-internals-0.10.1-SNAPSHOT.jar tez-0.10.1/tez-runtime-internals/target/**

* **Tez工程下tez-runtime-library/路径中, 可找到原生tez-runtime-library-0.10.1-SNAPSHOT.jar。用tez-runtime-library-0.10.1-SNAPSHOT.jar替代此原生jar:**

**cp jar/tez-runtime-library-0.10.1-SNAPSHOT.jar tez-0.10.1/tez-runtime-library/target/**

1. **启动Hive, 运行SQL query。**

* **启动Hive， 运行SQL query，并记录Application ID.**

1. **保存运行日志记录到本地。**

* **保存运行日志记录到本地文件run.log:**

**yarn logs -applicationId application\_1606874353529\_0402 > run.log**

* **筛选有效信息到本地文件info.log:**

**egrep 'Profiling: Tez|Container: container\_' > info.log**

1. **利用可运行程序extract,获取分析结果。**

* **文件夹extract中有可执行文件extract。将运行日志info.log放置同一路径下，运行extract可得到性能分析结Profile.csv:**

**./extract**

* 1. **Profiling结果说明**

**经过章节1.3我们可获得性能分析数据，其保存在CSV文件Profile.csv Profile.csv中一种有11列数据，每种数据所代表的意义:**

* **TaskName: Map/reduce task所对应的名称，在Hive生成执行计划DAG时默认生成。**
* **TaskID: 每个Map/reduce task被分解成若干子任务。子任务在Yarn分配的一个container上运行。TaskID是任务分配过程中默认生成。**
* **Init. (Initailization): 每个Map/reduce任务启动过程中的初始化时间，其包括获取当前任务的执行计划，解析执行计划，初始化算子等过程。**
* **Input: Map task的数据获取时间，即从HDFS读取数据的过程。**
* **Shuffle. (Shuffle&Merge): Reduce task的数据获取时间，即从集群中通过网络传输读取数据的过程。**
* **SQL Op. (SQL Operator): Map/reduce task的SQL算子计算时间，对应着可以GPU加速计算优化的过程。**
* **Sink: Map/reduce task的SQL计算结果sink过程。**
* **Spill: Map task把计算结果排序准备传输到下一个task的时间。**
* **Output: Map task把计算结果排序准备传输到下一个task, 或直接写入HDFS作为query结果的时间。**
* **Start. (Starting time): Map/reduce task开始运行的时间点。**
* **End. (Ending time): Map/reduce task结束运行的时间点。**
* **Cost. (Cost time): Map/reduce task运行的时间消耗。**
* **Locate (Node locate): Map/reduce task运行所在的节点名称。**

**Table 1. Map Task的Profiling结果例子 (单位: ms)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TaskName | TaskID | Init. | Input | SQL op. | Sink | Spill | Start. | End. | Cost | Locate |
| Map 10 | attempt\_1 | 199.324 | 1665.4 | 1701.46 | 169.753 | 929.996 | 2927 | 7601 | 4674 | 20 |
| Map 1 | attempt\_14 | 54.4029 | 182.263 | 491.676 | 60.1895 | 355.901 | 7730 | 8877 | 1147 | 20 |

**Table 2. Reduce Task的Profiling结果例子 (单位: ms)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TaskName | TaskID | Init. | Shuffle. | SQL op. | Sink | Output | Start. | End. | Cost | Locate |
| Reduce11 | attempt\_4 | 23.5157 | 106.404 | 9956.71 | 231.764 | 232.844 | 14609 | 25161 | 10552 | 20 |
| Reduce 7 | attempt\_2 | 16.4252 | 13.7799 | 10259.9 | 200.705 | 211.877 | 25169 | 35872 | 10703 | 20 |

**2 Profiling设计说明**

**2.1 Tez运行流程**

**根据Apace Tez官方文档，Tez的Map/reduce任务的逻辑流程可分为:**

* **Input: 对于map task, 指从HDFS存储模型InputFormat中读取并解析为key/value的过程；对于reduce task, 指shuffle过程, 从其他task上复制一片数据，进行排序合并作为本task的输入数据。**
* **Processor: 指SQL算子或用户定义的计算逻辑处理的过程。基于Tez的源码实现, 我们进一步将其分为SQL operator和sink两部分。**
* **Output: 对于Map task, 指将计结果数据排序准备输出到一下任务的过程；对于Reduce task, 指将计结果数据排序准备输出到一下任务，或者直接写入文件系统作为最终查询结果的过程。**

**当计划利用GPU去加速Tez运行时，我们倾向于关注Processor过程是否成为运行过程的性能瓶颈。**

**2.2 源码修改说明**

**为实现Profiling，我们对以下文件进行了修改：**

1. **Hive源码**

* **ql/src/java/org/apache/hadoop/hive/ql/exec/tez/MapRecordProcessor.java**
* **ql/src/java/org/apache/hadoop/hive/ql/exec/tez/ReduceRecordProcessor.java**
* **ql/src/java/org/apache/hadoop/hive/ql/exec/vector/reducesink/VectorReduceSinkCommonOperator.java**
* **ql/src/java/org/apache/hadoop/hive/ql/exec/FileSinkOperator.java**
* **ql/src/java/org/apache/hadoop/hive/ql/exec/ReduceSinkOperator.java**

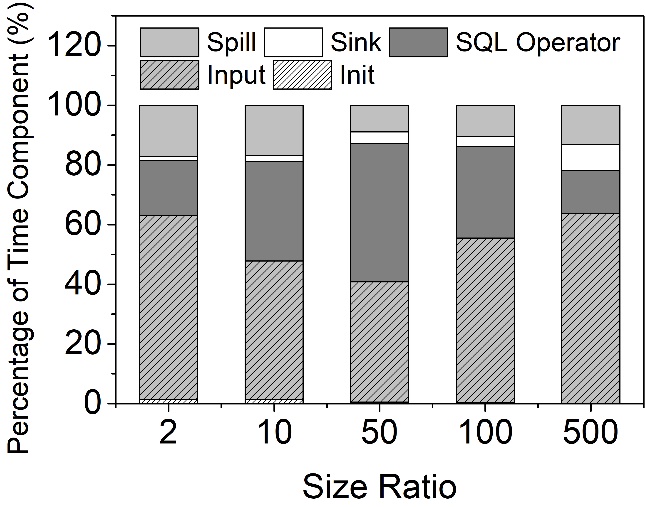
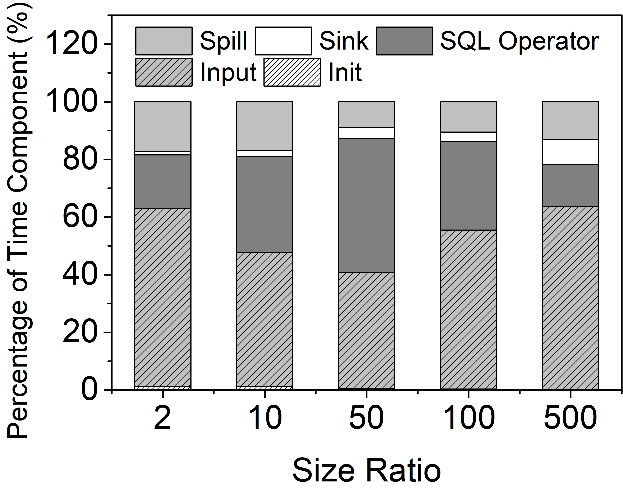
1. **Tez源码**

* **tez-runtime-internals/src/main/java/org/apache/tez/runtime/LogicalIOProcessorRuntimeTask.java**
* **tez-mapreduce/src/main/java/org/apache/tez/mapreduce/lib/MRReaderMapred.java**
* **tez-runtime-library/src/main/java/org/apache/tez/runtime/library/common/sort/impl/PipelinedSorter.java**

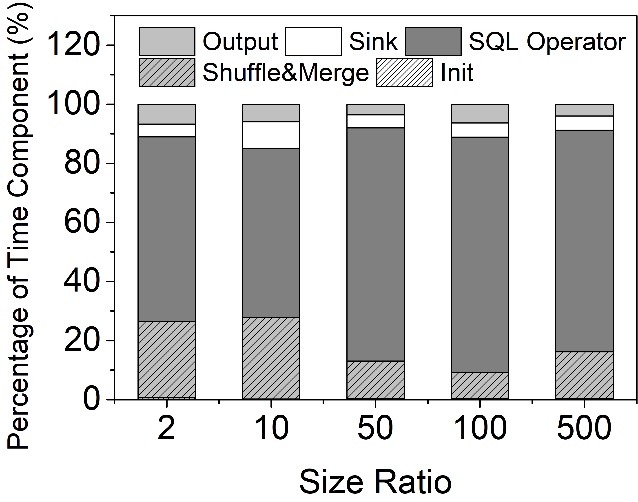
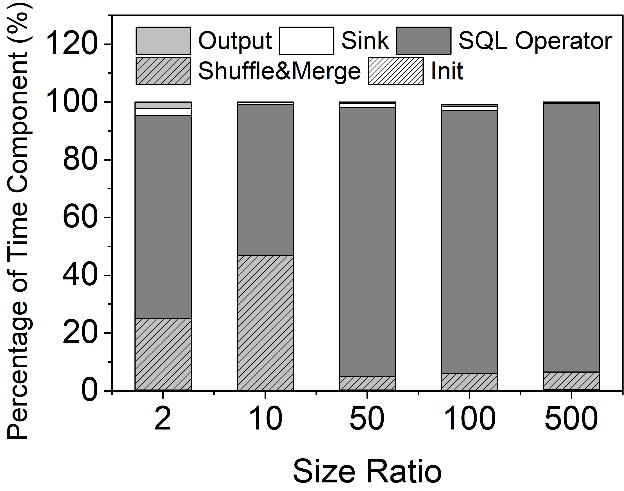
**具体源码修改请参考src/hive-3.1.2和src/tez-0.10.1中各对应文件。**

1. **Profiling效果样例**
   1. **TPC-DS效果样例**

**按照章节1.4介绍的profiling的流程，我们对TPC-DS query4进行了分析。对query4，Hive生成的DAG执行计划中有5个map tasks和21个reduce tasks. 图1和2分别展示map1，map17，和redcue7，reduce22, 图中X轴表示TPC-DS的不同scale factors。**

** **

**图1：在不同的数据大小下，Map1 (左)和Map17 (右)的Profiling分析**

** **

**图2：在不同的数据大小下，redcue7 (左)和redcue22 (右)的Profiling分析**