hive数据倾斜原因和解决方法

在做Shuffle阶段的优化过程中，遇到了数据倾斜的问题，造成了对一些情况下优化效果不明显。主要是因为在Job完成后的所得到的Counters是整个Job的总和，优化是基于这些Counters得出的平均值，而由于数据倾斜的原因造成map处理数据量的差异过大，使得这些平均值能代表的价值降低。Hive的执行是分阶段的，map处理数据量的差异取决于上一个stage的reduce输出，所以如何将数据均匀的分配到各个reduce中，就是解决数据倾斜的根本所在。规避错误来更好的运行比解决错误更高效。在查看了一些资料后，总结如下。

1数据倾斜的原因

1.1操作：

关键词	情形	后果
Join	其中一个表较小， 但是key集中	分发到某一个或几个Reduce上的数据远高于平均值
	大表与大表，但是分桶的判断字段0值或空值过多	这些空值都由一个reduce处理，灰常慢
group by	group by 维度过小， 某值的数量过多	处理某值的reduce灰常耗时
Count Distinct	某特殊值过多	处理此特殊值的reduce耗时

1.2原因：

1)、key分布不均匀

2)、业务数据本身的特性

3)、建表时考虑不周

4)、某些SQL语句本身就有数据倾斜

1.3表现：

任务进度长时间维持在99%（或100%），查看任务监控页面，发现只有少量（1个或几个）reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。

单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多。 最长时长远大于平均时长。

2数据倾斜的解决方案

2.1参数调节：

hive.map.aggr = true

Map 端部分聚合，相当于Combiner

hive.groupby.skewindata=true

有数据倾斜的时候进行负载均衡，当选项设定为 true，生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

2.2 SQL语句调节：

如何Join：

关于驱动表的选取，选用join key分布最均匀的表作为驱动表

做好列裁剪和filter操作，以达到两表做join的时候，数据量相对变小的效果。

大小表Join：

使用map join让小的维度表（1000条以下的记录条数） 先进内存。在map端完成reduce.

hive数据倾斜原因和解决方法

在做Shuffle阶段的优化过程中，遇到了数据倾斜的问题，造成了对一些情况下优化效果不明显。主要是因为在Job完成后的所得到的Counters是整个Job的总和，优化是基于这些Counters得出的平均值，而由于数据倾斜的原因造成map处理数据量的差异过大，使得这些平均值能代表的价值降低。Hive的执行是分阶段的，map处理数据量的差异取决于上一个stage的reduce输出，所以如何将数据均匀的分配到各个reduce中，就是解决数据倾斜的根本所在。规避错误来更好的运行比解决错误更高效。在查看了一些资料后，总结如下。

1数据倾斜的原因

1.1操作：

关键词	情形	后果
Join	其中一个表较小， 但是key集中	分发到某一个或几个Reduce上的数据远高于平均值
	大表与大表，但是分桶的判断字段0值或空值过多	这些空值都由一个reduce处理，灰常慢
group by	group by 维度过小， 某值的数量过多	处理某值的reduce灰常耗时
Count Distinct	某特殊值过多	处理此特殊值的reduce耗时

1.2原因：

1)、key分布不均匀

2)、业务数据本身的特性

3)、建表时考虑不周

4)、某些SQL语句本身就有数据倾斜

1.3表现：

任务进度长时间维持在99%（或100%），查看任务监控页面，发现只有少量（1个或几个）reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。

单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多。 最长时长远大于平均时长。

2数据倾斜的解决方案

2.1参数调节：

hive.map.aggr = true

Map 端部分聚合，相当于Combiner

hive.groupby.skewindata=true

有数据倾斜的时候进行负载均衡，当选项设定为 true，生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

2.2 SQL语句调节：

如何Join：

关于驱动表的选取，选用join key分布最均匀的表作为驱动表

做好列裁剪和filter操作，以达到两表做join的时候，数据量相对变小的效果。

大小表Join：

使用map join让小的维度表（1000条以下的记录条数） 先进内存。在map端完成reduce.

hive数据倾斜原因和解决方法

在做Shuffle阶段的优化过程中，遇到了数据倾斜的问题，造成了对一些情况下优化效果不明显。主要是因为在Job完成后的所得到的Counters是整个Job的总和，优化是基于这些Counters得出的平均值，而由于数据倾斜的原因造成map处理数据量的差异过大，使得这些平均值能代表的价值降低。Hive的执行是分阶段的，map处理数据量的差异取决于上一个stage的reduce输出，所以如何将数据均匀的分配到各个reduce中，就是解决数据倾斜的根本所在。规避错误来更好的运行比解决错误更高效。在查看了一些资料后，总结如下。

1数据倾斜的原因

1.1操作：

关键词	情形	后果
Join	其中一个表较小， 但是key集中	分发到某一个或几个Reduce上的数据远高于平均值
	大表与大表，但是分桶的判断字段0值或空值过多	这些空值都由一个reduce处理，灰常慢
group by	group by 维度过小， 某值的数量过多	处理某值的reduce灰常耗时
Count Distinct	某特殊值过多	处理此特殊值的reduce耗时

1.2原因：

1)、key分布不均匀

2)、业务数据本身的特性

3)、建表时考虑不周

4)、某些SQL语句本身就有数据倾斜

1.3表现：

任务进度长时间维持在99%（或100%），查看任务监控页面，发现只有少量（1个或几个）reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。

单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多。 最长时长远大于平均时长。

2数据倾斜的解决方案

2.1参数调节：

hive.map.aggr = true

Map 端部分聚合，相当于Combiner

hive.groupby.skewindata=true

有数据倾斜的时候进行负载均衡，当选项设定为 true，生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

2.2 SQL语句调节：

如何Join：

关于驱动表的选取，选用join key分布最均匀的表作为驱动表

做好列裁剪和filter操作，以达到两表做join的时候，数据量相对变小的效果。

大小表Join：

使用map join让小的维度表（1000条以下的记录条数） 先进内存。在map端完成reduce.

hive数据倾斜原因和解决方法

在做Shuffle阶段的优化过程中，遇到了数据倾斜的问题，造成了对一些情况下优化效果不明显。主要是因为在Job完成后的所得到的Counters是整个Job的总和，优化是基于这些Counters得出的平均值，而由于数据倾斜的原因造成map处理数据量的差异过大，使得这些平均值能代表的价值降低。Hive的执行是分阶段的，map处理数据量的差异取决于上一个stage的reduce输出，所以如何将数据均匀的分配到各个reduce中，就是解决数据倾斜的根本所在。规避错误来更好的运行比解决错误更高效。在查看了一些资料后，总结如下。

1数据倾斜的原因

1.1操作：

关键词	情形	后果
Join	其中一个表较小， 但是key集中	分发到某一个或几个Reduce上的数据远高于平均值
	大表与大表，但是分桶的判断字段0值或空值过多	这些空值都由一个reduce处理，灰常慢
group by	group by 维度过小， 某值的数量过多	处理某值的reduce灰常耗时
Count Distinct	某特殊值过多	处理此特殊值的reduce耗时

1.2原因：

1)、key分布不均匀

2)、业务数据本身的特性

3)、建表时考虑不周

4)、某些SQL语句本身就有数据倾斜

1.3表现：

任务进度长时间维持在99%（或100%），查看任务监控页面，发现只有少量（1个或几个）reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。

单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多。 最长时长远大于平均时长。

2数据倾斜的解决方案

2.1参数调节：

hive.map.aggr = true

Map 端部分聚合，相当于Combiner

hive.groupby.skewindata=true

有数据倾斜的时候进行负载均衡，当选项设定为 true，生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

2.2 SQL语句调节：

如何Join：

关于驱动表的选取，选用join key分布最均匀的表作为驱动表

做好列裁剪和filter操作，以达到两表做join的时候，数据量相对变小的效果。

大小表Join：

使用map join让小的维度表（1000条以下的记录条数） 先进内存。在map端完成reduce.