> Hive是一个基于Hadoop的数据仓库系统，它提供了SQL-like查询语言来操作Hadoop上的大数据。在使用Hive时，我们通常需要做一些参数调优来提高其性能。本文将介绍一些常用的Hive参数以及如何进行性能调优。

一、Hive参数

压缩参数

压缩是Hive性能优化中的一个重要参数。使用Hadoop支持的压缩格式，可以优化Hive查询的执行速度，减少磁盘空间的占用。以下是一些常见的Hive压缩参数：

```
hive.exec.compress.output：是否对输出进行压缩，默认为false。
hive.exec.compress.intermediate：是否对中间结果进行压缩，默认为false。
hive.exec.compress.intermediate.codec：中间结果压缩编解码器，默认为org.apache.hadoop.io.compress.DefaultCodec。
hive.output.compression.type：输出压缩类型，默认值是NONE，支持的压缩类型有：DEFLATE、GZIP、BZIP2、LZO、SNAPPY、ZSTD等。
hive.mapred.output.compression.codec：输出压缩编解码器，默认为org.apache.hadoop.io.compress.DefaultCodec。
hive.exec.orc.default.compress：ORC文件压缩格式，默认是SNAPPY，可选的有：ZLIB、NONE、LZO、SNAPPY、ZSTD等。
```

容错参数

Hive在执行查询过程中可能会遇到错误，以下是一些常用的容错参数：

```
hive.exec.failure.hooks：在任务失败时执行的hook，默认为org.apache.hadoop.hive.ql.hooks.FailOnErrorsHook。
hive.execution.engine.strict.mode：如果设置为true，则在遇到任务失败时，Hive会立即停止执行查询并返回错误，默认为false。
hive.mapred.local.mem：如果为非0，则表示Mapper和Reducer的内存都会超过该阈值时，会试图使用本地磁盘进行排序，而不是在内存中进行
```

并行度参数

并行度是Hive性能调优中的一个重要参数。以下是一些常见的Hive并行度参数：

hive.exec.parallel：是否开启并行执行，默认为true。

hive.exec.parallel.thread.number：并行执行的线程数，默认为8。
hive.exec.reducers.bytes.per.reducer：每个Reducer处理数据的大小，默认为256MB。
hive.tez.container.size：Tez执行引擎使用的容器大小，单位为MB，默认为1024MB。

IO参数

IO是Hive性能调优中的另一个重要参数。以下是一些常见的Hive IO参数：

hive.fetch.task.conversion：是否开启FETCH任务转换模式，默认为false。

hive.fetch.task.aggr：是否开启FETCH任务的聚合模式，默认为false。

hive.fetch.task.conversion.threshold：开启FETCH任务转换模式时的阈值，单位为字节，默认为268435456（256MB）。
hive.fetch.task.aggr.bytes.per.reducer：开启FETCH任务聚合模式时每个Reducer处理数据的大小，默认为256MB。

hive.optimize.skewjoin：是否开启Skew Join优化，默认为false。

hive.optimize.skewjoin.compiletime：是否在编译阶段使用Skew Join优化，默认为true。

数据存储参数

Hive数据存储也是性能调优的重要方面。以下是一些常用的存储参数：

hive.optimize.index.filter：是否开启索引过滤器优化，默认为false。

hive.index.compact.file.max.size：压缩索引时，每个文件的最大大小，默认为250MB。

hive.optimize.sort.dynamic.partition：是否对动态分区进行排序，默认为false。

hive.exec.dynamic.partition.mode：是否开启动态分区模式，默认为nonstrict。

性能调优

除了设置参数外，还可以通过以下方法进行Hive性能调优：

数据分区

数据分区是Hive性能优化的重要手段。通过将数据分成多个分区，可以减少查询所需要 扫描的数据量，提高查询效率。同时，分区也方便数据管理和维护。在创建表时，可以使用PARTITIONED BY关键字指定分区字段。分区字段可以是时间、地理位置等等，根据具体业务需求选择合适的字段进行分区。

数据压缩

如前文所述，数据压缩能有效地减少磁盘空间的占用和提高查询速度。在创建表时，可以指定压缩格式和压缩编解码器，如：

```
CREATE TABLE table_name (column1 data_type, column2 data_type)
STORED AS ORC
TBLPROPERTIES("orc.compress"="SNAPPY");
```

合理使用聚合函数

在查询中使用聚合函数时，应该尽量减少聚合函数的使用次数。例如，先将数据按照分组字段排序，然后再调用聚合函数。这样可以避免重复计算聚合函数，提高查询效率。

合理使用索引

在Hive中使用索引可以提高查询效率，但是索引也会占用一定的磁盘空间。因此，在创建表时，应该根据实际情况选择建立索引。同时，对于查询中经常使用的字段，可以考虑建立索引以提高查询效率。

避免全表扫描

在查询中应该尽量避免全表扫描，因为全表扫描会消耗大量的CPU和IO资源。可以通过数据分区、使用索引等方法避免全表扫描。

总结

在使用Hive时，合理的参数设置和性能调优能够提高查询效率和减少资源消耗。同时，数据分区、数据压缩、合理使用聚合函数、索引和避免全表扫描等方法也是优化Hive性能的重要手段。在实际使用中，应该根据具体业务需求和数据规模选择合适的参数和优化方法，以达到最佳的性能表现