# 目录

目录 1
write写入优化 2
客户端优化 2
批量写 2
Auto Flush 2
服务端优化 2
WAL Flag 2
增大memstore的内存 2
大量的HFile产生 2
read读取优化 2
客户端优化 2
get请求是否可以使用批量请求 2
大scan缓存是否设置合理 2
请求指定列族或者列名 2
离线计算访问Hbase建议禁止缓存 2
服务端优化 3
请求是否均衡 3
BlockCache是否合理 3
HFile文件数目 3
Compaction是否消耗较多的系统资源 3
Bloomfilter设置是否合理 3

# write写入优化

HBase基于LSM模式，写是写HLOG及Memory的，也就是基本没有随机的IO，所以在写链路上性能高效还比较平稳。很多时候，写都是用可靠性来换取性能。

## 客户端优化

### 批量写

批量写是为了减少rpc的次数。

```
1.HTable.put(List<Put>)
```

### Auto Flush

当autoflush=false时可以提升几倍的写入性能，但是还是要注意，如果数据超2M(hbase.client.write.buffer决定)或用户执行了hbase.flushcommits()时才向regionserver提交请求。需要注意：这里并不是写到了服务端。 通过HTable.setWriteBufferSize(writeBufferSize) 可以设置buffer的大小。

## 服务端优化

### WAL Flag

不写WAL可以成倍提升性能，因为不需要写HLog，减少3次IO。写MemStore是内存操作，是以数据可靠性为代价的。在数据导入时，可以关闭WAL。

### 增大memstore的内存

当前可以调高Memstore的数值，降低 BlockCache 的数，跟读优化的思路正好相反。

### 大量的HFile产生

如果写很快，很容易带来大量的HFile，因为此时HFile合并的速度还没有写入的速度快。需要在业务低峰期做majorcompaction，充分利用系统资源；如果HFile降低不下来，则需要添加节点。

# read读取优化

其实HBase还是比较灵活的，关键看你是否使用得当，以下主要列举一些读的优化。HBase在生产中往往会遇到Full GC、进程OOM、RIT问题、读取延迟较大等一些问题，使用更好的硬件往往可以解决一部分问题，但是还是需要使用的方式。 我们把优化分为： 客户端优化、服务端优化

## 客户端优化

### get请求是否可以使用批量请求

这样可以成倍减小客户端与服务端的rpc次数，显著提高吞吐量。

```
1.Result[] re= table.get(List<Get> gets)
```

### 大scan缓存是否设置合理

scan一次性需求从服务端返回大量的数据，客户端发起一次请求，客户端会分多批次返回客户端，这样的设计是避免一次性传输较多的数据给服务端及客户端有较大的压力。目前 数据会加载到本地的缓存中，默认100条数据大小。 一些大scan需要获取大量的数据，传输数百次甚至数万的rpc请求。我们建议可以 适当放开缓存的大小。

```
1.scan.setCaching(int caching) //大scan可以设置为1000
```

### 请求指定列族或者列名

HBase是列族数据库，同一个列族的数据存储在一块，不同列族是分开的，为了减小IO，建议指定列族或者列名。

### 离线计算访问Hbase建议禁止缓存

当离线访问HBase时，往往就是一次性的读取，此时读取的数据没有必要存放在blockcache中，建议在读取时禁止缓存。

1.scan.setBlockCache(false)

# 服务端优化

## 请求是否均衡

读取的压力是否都在一台或者几台之中，在业务高峰期间可以查看下，可以到 HBase管控平台查看HBase的ui。如果有明显的热点，一劳永逸是重新设计rowkey，短期是 把热点region尝试拆分

## BlockCache是否合理

BlockCache作为读缓存，对于读的性能比较重要。如果读比较多，可以调高 BlockCache 的数值，降低 Memstore的数值。

## HFile文件数目

因为读取需要频繁打开HFile，如果文件越多，IO次数就越多，读取的延迟就越高此需要主要定时做 major compaction，如果晚上的业务压力不大，可以在晚上做major compaction

## Compaction是否消耗较多的系统资源

compaction主要是将HFile的小文件合并成大文件，提高后续业务的读性能，但是也会带来较大的资源消耗。Minor Compaction一般情况下不会带来大量的系统资源消耗，除非因为配置不合理。 切勿在高峰期间做 major compaction。 建议在业务低峰期做major compaction。

## Bloomfilter设置是否合理

Bloomfilter主要用来在查询时，过滤HFile的，避免不需要的IO操作。Bloomfilter能提高读取的性能，一般情况下创建表，都会默认设置为：BLOOMFILTER => ‘ROW’。