如何在 Flink 中规划 RocksDB 内存容量?

作者：Stefan Richter
翻译：毛家琦
校对：胡争

本文描述了一些配置选项，这些选项将帮助您有效地管理规划 Apache Flink 中 RocksDB state backend 的内存大小。在前面的文章[1]中，我们描述了 Flink 中支持的可选 state backend 选项，本文将介绍跟 Flink 相关的一些 RocksDB 操作，并讨论一些提高资源利用率的重要配置。

Tips：从 Flink 1.10 开始，Flink 自动管理 RocksDB 的内存，详细介绍如下：

https://ci.apache.org/projects/flink/flink-docs-release-1.10/ops/state/state_backends.html#memory-management

RocksDB 的状态后端

在深入了解配置参数之前，先回顾一下在 Apache Flink 中如何使用 RocksDB 来进行状态管理。当选择 RocksDB 作为状态后端时，状态将作为序列化字节串存在于堆外内存（off-heap）存储或本地磁盘中。

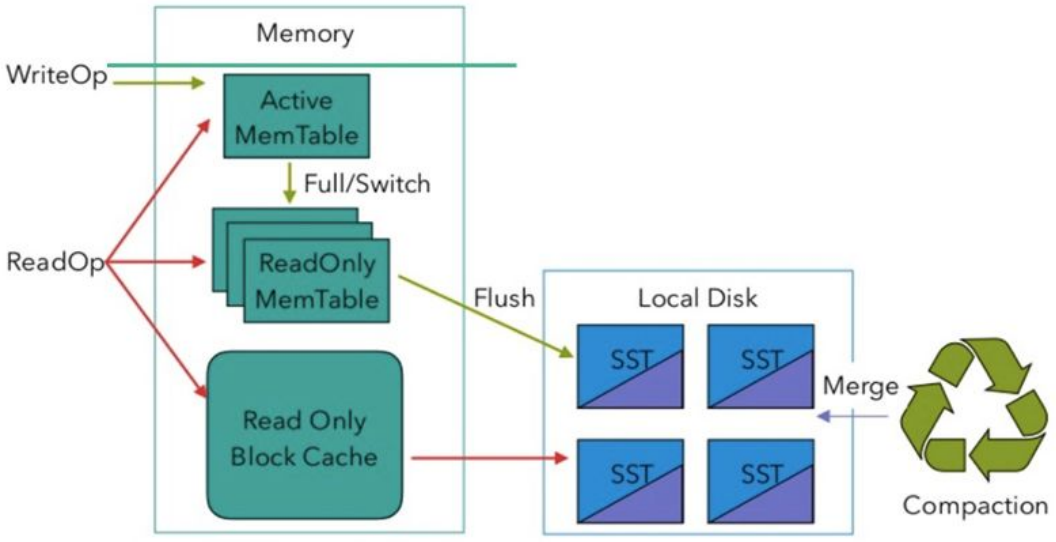
RocksDB 是一个以日志合并树(LSM 树）作为索引结构的 KV 存储引擎。当用于在 Flink 中存储 kv 状态时，键由 的序列化字节串组成，而值由状态的序列化字节组成。每次注册 kv 状态时，它都会映射到列族（column-family）（类似于传统数据库中的表），并将键值对以字节串存储在 RocksDB 中。这意味着每次读写（READ or WRITE）操作都必须对数据进行反序列化或者序列化，与 Flink 内置的 in-memory 状态后端相比，会有一些性能开销。

使用 RocksDB 作为状态后端有许多优点：

- 不受 Java 垃圾回收的影响，与 heap 对象相比，它的内存开销更低，并且是目前唯一支持增量检查点（incremental checkpointing）的选项。
- 使用 RocksDB，状态大小仅受限于本地可用的磁盘空间大小，这很适合 state 特别大的 Flink 作业。

下面的图表将进一步阐明 RocksDB 的基本读写操作。

RocksDB 的一次写入操作将把数据写入到内存的 MemTable 中。当 MemTable 写满时，它将成为 READ ONLY MemTable，并被一个新申请的 MemTable 替换。只读 MemTable 被后台线程周期性地刷新到磁盘中，生成按键排序的只读文件，这便是所谓的 SSTables。这些 SSTable 是不可变的，通过后台的多路归并实现进一步的整合。如前所述，对于 RocksDB，每个注册状态都是一个列族，这意味着每个状态都包含自己的 MemTables 和 SSTables 集。



RocksDB 中的读取操作首先访问活动内存表（Active Memory Table）来反馈查询。如果找到待查询的 key，则读取操作将由新到旧依次访问，直到找到待查询的 key 为止。如果在任何 MemTable 中都找不到目标 key，那么 READ 操作将访问 SSTables，再次从最新的开始。SSTables 文件可以：

1. 优先去 RocksDB 的 BlockCache 读取；
2. 如果 BlockCache 没有的话，就去读操作系统的文件，这些文件块又可能被操作系统缓存了；
3. 最差的情况就是去本地磁盘读取；
4. SST 级别的 bloom filter 策略可以避免大量的磁盘访问。

管理 RocksDB 内存的 3 种配置

现在，我们理解了 Flink 和 Rocksdb 的协作机制，接下来看看可以更有效地管理 RocksDB 内存大小的配置选项有哪些？请注意，下面的选项并不详尽，因为您可以使用 Apache Flink 1.6 中引入的 state TTL（Time To Live）功能来规划 Flink 应用程序的状态大小。

以下三种配置可以有效帮助您管理 Rocksdb 的内存开销：

1.block_cache_size 的配置

此配置最终将控制内存中缓存的最大未压缩块数。随着块数的不断增加，内存大小也会增加。因此，通过预先配置，您可以保持固定的内存消耗水平。

2.write_buffer_size 的配置

这种配置控制着 RocksDB 中 MemTable 的最大值。活跃 MemTables 和只读的 MemTables 最终会影响 RocksDB 中的内存大小，所以提前调整可能会在以后为您避免一些麻烦。

3.max_write_buffer_number 的配置

在 RocksDB 将 MemTables 导出到磁盘上的 SSTable 之前，此配置决定并控制着内存中保留的 MemTables 的最大数量。这实际上是内存中“只读内存表“的最大数量。

除了上面提到的资源之外，您还可以选择配置索引和 bloom 过滤器，它们将消耗额外的内存空间， Table 级别的 Cache 也是一样。

在这里，Table 缓存不仅会额外占用 RocksDB 的内存，还会占用 SST 文件的打开文件描述符，(在默认情况下设置的大小是不受限制的)，如果配置不正确，可能会影响操作系统的设置。

我们刚刚给您指导了一些使用配置选项，这些配置有助于高效管理 RocksDB 作为 Flink statebackend 的内存大小。有关更多配置选项，我们建议查看 RocksDB 优化指南[2]或 Apache Flink 文档。

参考资料：

[1] https://www.ververica.com/blog/stateful-stream-processing-apache-flink-state-backends
[2] https://github.com/facebook/rocksdb/wiki/RocksDB-Tuning-Guide

原文链接：https://www.ververica.com/blog/manage-rocksdb-memory-size-apache-flink