01 从零实现 KV 存储—初识 KV 数据库

既然我们这个课程是从零实现一个完整的 KV 存储项目，那么我们首先来看看什么是 KV 存储呢？

KV 存储，即键值数据存储，是一种基于键值对的存储方式，它将数据存储为一个由键和值组成的二元组，其中键（key）是一个唯一的标识符，值（value）是与这个 key 相关联的数据，KV 存储的应用场景很多，比如用作数据库的缓存层、分布式系统中元数据的存储、分布式锁等，当然一些数据库的底层也会有存储的模块，例如 Postgres 或者 MySQL。

对大多数开发者来说，最熟悉的 KV 存储产品应该是 Redis 了，但我们这里说的 KV 存储和 Redis 不太一样，因为 Redis 是一种基于内存的 KV 数据库，虽然它也有一些持久化的策略例如 AOF 和 RDB，但从本质上来说它还是面向内存进行设计的，数据的持久性并不能完全得到保证。

我们这里说的 KV 存储一般来说是面向磁盘的，数据的持久性能够得到很好的保证，并且配合适当的设计，虽然在性能上无法与 Redis 匹敌，但也基本能够做到和 Redis 一个数量级。

当然，最关键的是 Redis 的数据量受限于内存，而基于磁盘的 KV 存储，可以处理远超内存容量的数据，并且在性能上依然可以很强悍，这就是为什么 KV 存储价值巨大，也非常的受欢迎。

一般来说 KV 数据库的数据组织存储模型大致分为了两种，一个是 B+ 树，一个是 LSM 树，基于 B+ 树的项目比较著名的有 BoltDB，而充分利用顺序 IO、写性能更优的 LSM Tree 存储模型在近些年更加的受欢迎，其中最具有代表性的项目有 LevelDB、RocksDB。

> https://github.com/etcd-io/bbolt
> https://github.com/google/leveldb
> https://github.com/facebook/rocksdb

LevelDB 应该是最早实践 LSM 存储模型的存储项目了，由 Google 的传奇大神 Jeff Dean 开发，后来 Facebook 基于 LevelDB，对其进行了一系列的优化，总体还是保留了 LevelDB 的存储模型，只是增加了更多的特性，例如支持更完善的事务处理、并发 compaction、高级压缩算法等，并且拥有更高的性能，在处理较大数据量时表现优异。

所以越来越多的产品采用 RocksDB 作为它们的存储引擎，例如 MyRocks 使用 RocksDB 给 MySQL 作为存储引擎，意在取代 InnoDB，CockroachDB 也使用 rocksdb 作为它的存储引擎，也有很多基于 RocksDB 构建分布式 KV 的项目，例如 PingCAP 的 TiKV。

前面说了很多，回到我们自身，学习 KV 存储都有哪些好处呢？

从我自身的经历来说，我从毕业开始是使用 Java 做业务开发的，后来一个偶然的机会，让我接触到了 KV 存储项目，并且自己实践，使用 Go 写了一个简单的 KV 数据库，然后自己慢慢折腾，从业务开发转到基础架构，开始做分布式 KV 存储相关的工作，后来再转到数据库内核开发。

有的时候，凭借着一些偶然的因素，然后我们硬着头皮去学习了一些之前想接触但是没能力接触，或者根本没接触过的东西，我们职业生涯的发展或许就会有更多的出路。

从更加功利的角度来说，从零去学习并实现一个 KV 存储引擎，这样的项目写到简历上也是很加分的，如果你是一个在校学生的话，比一些电商系统、学生管理系统等烂大街的项目要更加吸引眼球。我之前面试的时候，基本上就是和面试官再聊我的两个开源项目，甚至连工作中的内容都很少谈到，这也从侧面反映出拥有一个硬核项目的作用。

其实回过头来看，最开始的那个 KV 存储项目算是我的启蒙了，并且如果你想做分布式存储的话，那么单机 KV 存储是一定需要掌握的，如果你想要数据库相关的内容，存储也是一个非常适合入门的项目，因为基本上所有的数据库最后都会和存储打交道。

并且 KV 存储从设计上来说是数据库的各个组件当中比较简单的，从这里入手，可以显著的降低入门数据库内核开发的门槛，提升我们的学习热情。并且可以让我们对数据库的基础设计有一些了解，例如如何在磁盘上组织数据，如何拥有更加高效的读写性能，这些都是数据库中比较核心的一些设计要点。

所以我相信，跟着本课程，学习完一个完整的 KV 存储引擎的设计和代码实现，你会对操作系统、文件等基础知识的概念理解更加清晰，并且入门数据库也是没有太大问题的。

全文评论

飞书用户3615 3月6日 11:52
点赞