Level DB 调研报告

赵岳

14231027

摘要

本文从传统的数据库谈起，结合当下的时代发展和人们对网络更高的要求，引出了新型数据库的产生背景，继而又 Key-Value 存储数据库谈到了 Level DB 的兴起和发展。文章的第二部分主要介绍了 Level DB 的特点、实现的关键技术和基本原理，并于传统数据库进行对比，较为全面地阐述了 Level DB 的优缺点以及未来的发展方向。除此以外，本文还结合相关例子以及笔者个人在工程中对 Level DB 的应用，总结了 Level DB 的技术特点以及其应用场景，并展望了未来新型数据库的发展。

产生背景

随着人们对网络的依赖日渐增长,接入互联网的终端设备数呈现了指数式的增长。更多的使用者和群体就意味着更大的数据量。为了应对这一现状和趋势，从功能到使用,甚至到用户的习惯,各大网络公司全方位兼顾,从产品界面甚至体验角度考虑的服务已接近完美。如亚马逊可以为我们推荐想看的书,Google 可以推荐相关网站,淘宝知道我们喜欢的产品,腾讯 QQ 可以猜出我们认识谁,网易云音乐可以精准地猜测出我们喜欢的那些歌曲并把它播放出来，而几乎人人都用的社交平台微信更是可以将通讯录和 QQ 好友推荐加为朋友。然而这些都需要大量数据分析和存储才能实现,数据从最初的 MB、GB 到 TB 甚至到现在的 PB,面对如此大量级数据就不得不解决一系列问题。传统的数据库已然不能胜任如此大量级、多样化、分散性的数据存储[1]。国外的 Google、Microsoft、Amazon,国内的 BTA(Baidu、Tencent、阿里巴巴)，甚至新兴起的滴滴研究院，作为国内外大数据和互联网的行业巨头,也把海量数据处理的研究作为后端核心技术之一。如何才能提供更高稳定性、更大可用性的服务成为各企业面临的瓶颈;如何解决数据丢失、破坏和数据延时问题,已静迫在眉睫。

在这种背景下，各种新型的数据库层出不穷，各种数据库的概念也让我们眼花缭乱。比如 spark、Hadoop 等。其中，Key-value 数据库则是一颗升起的新星。具备高可靠性及可扩展性的海量数据存储对互联网公司来说是一个巨大的挑战，传统的数据库往往很难满足该需求，并且很多时候对于特定的系统绝大部分的检索都是基于主键的的查询，在这种情况下使用关系型数据库将使得效率低下，并且扩展也将成为未来很

大的难题。在这样的情况下，使用 Key-value 存储将会是一个很好的选择，它被广泛应用于缓存，搜索引擎等等领域。

提起 LevelDB，就不得不提两个人：Jeff Dean 和 Sanjay Ghemawat。这两位是 Google 公司重量级的工程师，为数甚少的 Google Fellow 之二。Jeff Dean， Google 大规模分布式平台 Bigtable 和 MapReduce 主要设计和实现者。Sanjay Ghemawat，Google 大规模分布式平台 GFS，Bigtable 和 MapReduce 主要设计和实现工程师。LevelDB 就是这两位大神级别的工程师发起的开源项目，简而言之，LevelDB 是能够处理十亿级别规模 Key-Value 型数据持久性存储的 C++ 程序库，简而言之，LevelDB 本身只是一个 lib 库，在源码目录 make 编译即可，然后在我们的应用程序里面可以直接 include LevelDB/include/db.h 头文件即可。正像上面介绍的，这二位是 Bigtable 的设计和实现者，如果了解 Bigtable 的话，应该知道在这个影响深远的分布式存储系统中有两个核心的部分：Master Server 和 Tablet Server。其中 Master Server 做一些管理数据的存储以及分布式调度工作，实际的分布式数据存储以及读写操作是由 Tablet Server 完成的，而 LevelDB 则可以理解为一个简化版的 Tablet Server。

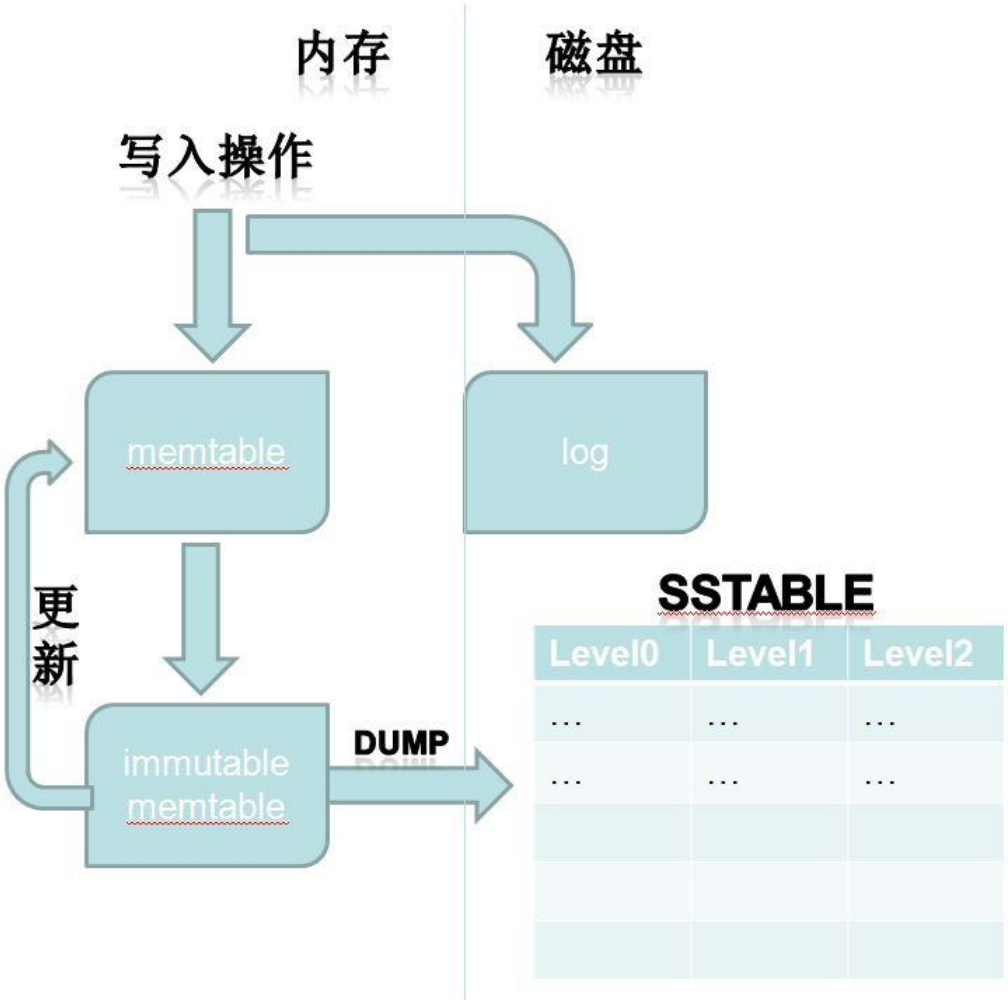
总而言之，LevelDB 就是在海量数据的时代背景下，在顶尖的工程师手下诞生的开源 lib 库， 采用了 Key-value 技术的存储库。

基本原理

谈到 LevelDB 的基本原理，就必须先要介绍 LevelDB 的核心特点。首先，LevelDB 是一个持久化存储的 Key-value 系统，和 Redis 这种内存型的 KV 系统不同，LevelDB 不会像 Redis 一样吞掉内存，而是将大部分数据存储到磁盘上（这点很重要，尤其在处理超大量数据时，我们不能将所有数据都存放在有限的内存上）。其次，LevelDB 在存储数据时，是根据记录的 key 值有序存储的，就是说相邻的 key 值在存储文件中是依次顺序存储的，而应用可以自定义 key 大小比较函数，LevelDB 会按照用户定义的比较函数依序存储这些记录，这就给了用户很大的操作空间，让用户可以自由地安排数据结构和存储组织形式。再次，像大多数 Key-value 系统一样，LevelDB 的操作接口很简单，基本操作包括写记录，读记录以及删除记录。也支持针对多条操作的原子批量操作。另外，LevelDB 支持数据快照（snapshot）功能，使得读取操作不受写操作影响，可以在读操作过程中始终看到一致的数据。LevelDB 性能非常突出，官方网站报道其随机写性能达到 40 万条记录每秒，而随机读性能达到 6 万条记录每秒[2]。总体来说，LevelDB 的写操作要大大快于读操作，而顺序读写操作则大大快于随机读写操作，也就是说，LevelDB 很适合应用在查询较少，而写很多的场景。具体的原因也会在后文做相应介绍。当然，LevelDB 的这些特点也决定了它是一个非关系型数据模型（NoSQL），不支持 SQL 语句，这与我们数据库课程上学习的数据库大相径庭。并且一次只允许一个进程访问一个特定的数据库。

至此，我们对 LevelDB 有一个粗略的印象了。LevelDB 是 Key-value 存储库中较为轻量级的一员。LevelDB 库提供了一种永久性的 key-value 存储。Key 和 value 都是任意的字节序列。在这个 key-value 存储系统中，key 按照用户声明的比较函数有序排列。这样的组织形式，有些类似于数据结构中的 Hash 散列表。Hash 表所建立的索引理论上可以将搜索的时间复杂度由 O(n)降低到 O(1)，那么举一反三便可知道 LevelDB 也有类似的功能。实际上，LevelDB 本质上是一套存储系统以及在这套存储系统上提供的一些操作接口。为了便于理解个系统及其处理流程，我们可以从两个不同的角度来看待 LevelDB：静态角度和动态角度。从静态角度，可以假想整个系统正在运行过程中（不断插入删除读取数据），此时我们给 LevelDB 照相，从照片可以看到之前系统的数据在内存和磁盘中是如何分布的，处于什么状态等；从动态的角度，主要是了解系统是如何写入一条记录，读出一条记录，删除一条记录的，同时也包括除了这些接口操作外的内部操作比如 compaction，系统运行时崩溃后如何恢复系统等等方面的信息。

LevelDB 的整体设计思路与“生产者-消费者问题”有些相似，因为都要解决读写互斥的矛盾。在这里，我们以数据的写入为例对 LevelDB 的运行模式进行介绍，其流程大致如下：当用户往 db 插入一条 key-value 数据的时候，会先写 log 文件，当写 log 成功之后再把当前记录写到 memtable 中。新插入的数据会保存在内存中，防止了由于系统崩溃导致数据丢失，因此要先写 log 文件保证落地之后，再写内存。这样即使系统崩溃了，也能够从 log 中恢复出来。memtable 中的数据是可读可写，当 memtable 的数据量达到一个数据量之后。当前的 memtable 变成了 immutable memtable，只读不可修改。重新生成新的 memtable 和 log 文件，新来的数据写到新的 log 和 memtable 中。immutable memtable 中的数据会被 dump 到磁盘中的 sstable 文件，磁盘中的 sstable 文件是有层级的，第一层 level0 到第 n 层 leveln...，每个 level 都有很多 sstable 文件，每个文件都是按照 key 排好序。注意了 level0 和其它 level 不一样，level0 中的 sstable 文件的 key 有可能重复，其它 level 的 sstable 文件的 key 保证不会有重复。由于每个 level 中有许多 sstable 文件，每个 sstable 文件都有 key range。所以需要一个文件来保存当前所有 level 中 sstabel 的 key range。manifest 文件主要就是用来存储每个 level 中的 sstable 的信息（类似磁盘中的 super block）。随着系统不断的运行，每个 level 中的 sstable 文件可能会越来越多，这个时候 DB 会自动把同一个 level 或者不同 level 中的 sstable 文件会进行 merge。这个时候 manifest 就会发生变化，因此我们需要一个 Current 文件（也可理解为指针）来记录当前最新的 manifest 文件。在这里，笔者以一张幻灯片展示其大致流程：



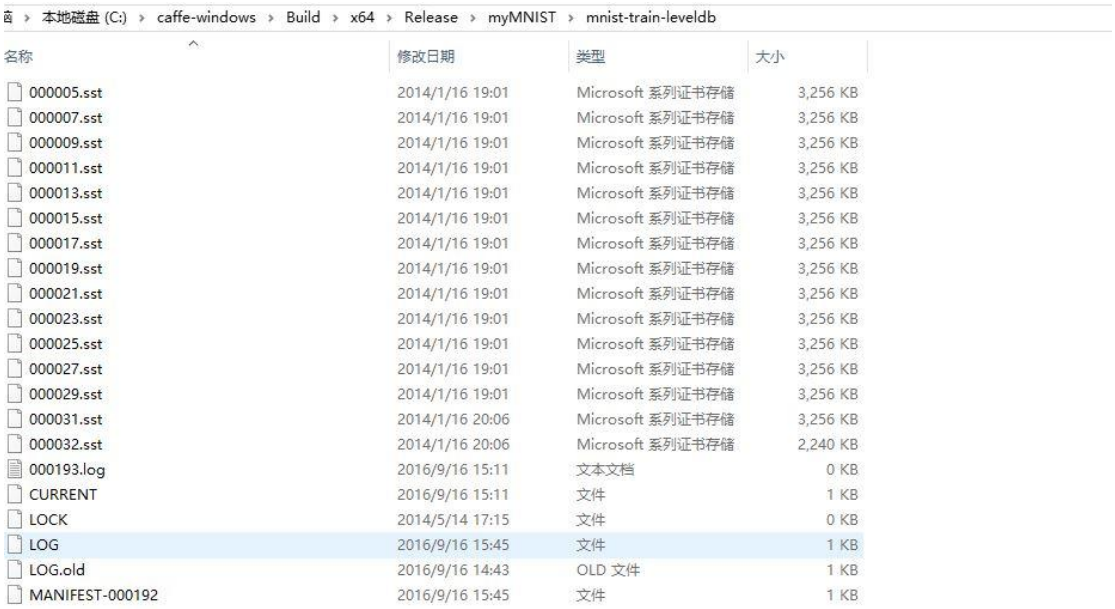
log 文件的详解与 sstable 的组织方式由于笔者研究不深，还不甚了解。可以参阅 Google 开源在 GitHub 上的代码和 Readme 文档进行了解：https://github.com/google/leveldb。

应用场景

LevelDB 可应用于很多场景，如用于网页浏览器存储最近存取网页的缓存，或用于操作系统存储安装包列表，或用于应用存储用户的设置参数。新版本的 Chrome 浏览器里部署的 IndexedDB HTML5 API 就是基于 LevelDB 打造的，Google 的数据库 Bigtable 掌管着数百万数据表也是用 LevelDB 的，其作为存储引擎被 Riak 和 Kyoto Tycoon 所支持。在国内，淘宝的 Tair 开源 Key-Value 存储也已经将 LevelDB 作为其持久化存储引

擎，并部署在线上使用[3]。2011 年 7 月，Google 宣布 LevelDB 项目开源，使用的开源授权协议为 BSD。

由于 LevelDB 不仅是 IO 速度远高于其他 SQL 数据库，还有一个特点是数据存储在磁盘上，不占用内存，也受到了大数据处理领域的青睐。例如在深度学习领域，尤其是在内存资源不足的设备上进行模型训练时，面对大量的数据和参数内存是完全不够的。笔者在个人电脑上进行基于 caffe 框架的深度学习模型训练时，就曾遇到过这种问题。对于 MNIST 的 demo（一个 5 万张手写图片库，用于手写数字识别网络的训练集），如果使用 matlab 的接口将其全部导入工作空间（内存）中，便会严重吞掉内存。后来笔者采用了 levelDB 格式的数据，将内存中的数据通过磁盘中的 sstable 导入，便节省了内存。



当然，LevelDB 的应用场景远不仅于此。随着未来数据量的进一步爆炸和扩增，相信不仅 LevelDB，其他 NoSQL 类型数据库必将兴起，成为大数据分析、深度学习、人工智能、互联网等各个科研、工程领域的利器。

参考文献

[1] 秦峥惠. 基于 Leveldb 的企业级大数据集群化存储设计与实现. 辽宁：辽宁科技大学,2015.

[2] doc_sgl. LevelDB 实现原理[DB/OL]. http://blog.csdn.net/doc_sgl/article/details/52725813, 2016-10-03.

[3] 李冯筱, 罗高松. NoSQL 理论体系及应用,电信科学 , 2012 年 12 期.