FPGA-Accelerated Compactions for LSM-based Key-Value Store

Teng Zhang, Alibaba Group

本文被收录在第18届USENIX文件和存储技术会议论文集(FAST ' 20)

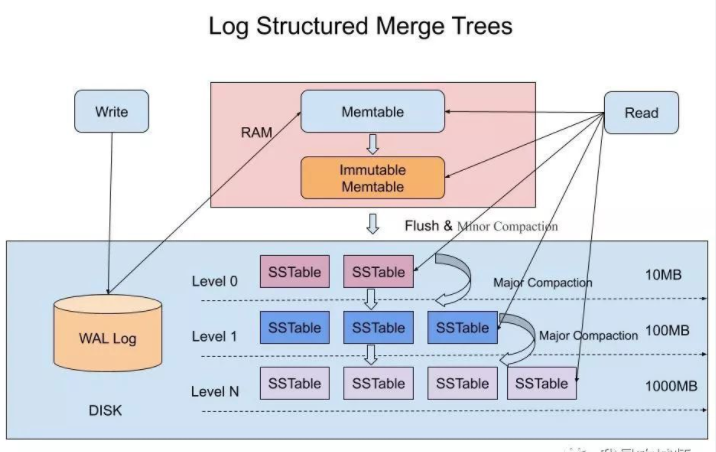
课外：

**什么是LSM-Tree？**

<https://cloud.tencent.com/developer/article/1441835>

LSM-Tree的设计思路是，将数据拆分为几百M大小的Segments，并是顺序写入基于LSM-Tree分层存储能够做到写的高吞吐，带来的副作用是整个系统必须频繁的进行compaction，写入量越大，Compaction的过程越频繁。而compaction是一个compare & merge的过程，非常消耗CPU和存储IO，在高吞吐的写入情形下，大量的compaction操作占用大量系统资源，必然带来整个系统性能断崖式下跌，对应用系统产生巨大影响，当然我们可以禁用自动Major Compaction，在每天系统低峰期定期触发合并，来避免这个问题。

阿里为了优化这个问题，在X-DB引入了异构硬件设备FPGA来代替CPU完成compaction操作，使系统整体性能维持在高水位并避免抖动，是存储引擎得以服务业务苛刻要求的关键。



**总结：**

由于lsm树中未检查的超大级别以及CPU和I/O的资源争用，缓慢的压缩会危害系统性能。

在合并短kv时，最新磁盘存储不断提高的I/O能力已经迫使压缩受到cpu的限制。这会导致查询/事务处理和后台压缩在lsm树KV存储中大量争夺瓶颈CPU资源，

解决方法：本文就提出了将compaction的功能分配给FPGA缓解CPU本身带来的瓶颈。

在FPGA上设计实现多级压实管道(multistaged compaction pipeline)，支持merge和delete操作。引入了异步调度器(驱动程序)来协调CPU和FPGA来转移compaction。对FPGA压缩吞吐量进行了建模分析。使用该模型，能够预测不同compaction卸载的方案性能。

**1.Instruction：**

LSM-tree在磁盘中保存包含键范围的多层记录，在关键路径中用于查询和事务处理的资源和用于后台压缩的资源之间存在困难，后台压缩会消耗大量的计算和磁盘I/O资源，特别是对于同时包含读和写的WPI工作负载。如果将更多的软件线程分配给压缩，我们就会限制存储的后台维护，这样做的风险是剥夺cpu实际处理查询和事务的能力，从而损害整体性能。

将压缩从cpu转移到fpga以加速执行。

**2.Motivation：**

合并操作有两种类型的策略:

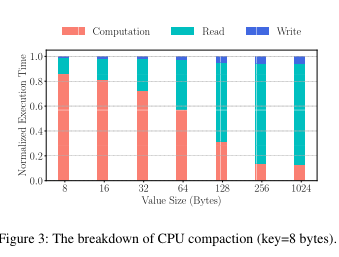
1. leveling policy:对于要合并到下一个级别(或组件)的每个批处理，一个策略是将其与目标级别中的现有数据合并，称为调平策略。这种方法以牺牲压缩速度为代价，将数据保持在一个有序的级别中。
2. tiering policy:简单地将数据追加到下一层而不合并,压缩本身是快速的，但要牺牲一个级别内的排序顺序.

problem 1:碎片化的L0

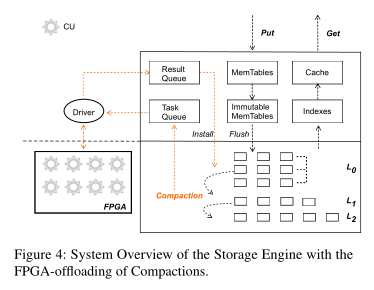
  L0的SST是直接从memtable上dump下来的。所以不可避免的出现key range重叠的情况，查找L0的时候就需要大范围地查找SST。同时当compaction操作较慢的时候，又会导师L0中SST的堆积，更加剧查找L0时候的延迟。加上L0是较新的数据，往往热度较高，从而更加拖慢了LSM-tree的速度。

 problem 2：瓶颈的转移

  compaction的步骤主要为：decode，merge，encode，在不同大小的KV对下，其消耗的资源是不同。



1. **Design&implement：**



这里使用X-engine维护一个全局索引以加速查找。

为了顺利将compaction顺利offload到FPGA，CPU端会创建三种线程：

**builder thread**

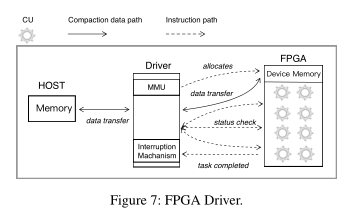
对于每个compaction，builder thread会将所需的范围分割成大小近似的不同组(group);每个group最终会形成单独的compaction任务。

**dispatcher thread**

dispatcher消耗task队列，并将任务发送以round-robin的方式发送给FPGA中的CUs。由于compaction tasks是通过分割成相近大小的形式得到的，不同的CU之间会得到相对平衡的负载。

**driver thread**

driver thread将每个compaction task相关的数据传输到FPGA的片上内存，并通知对应的compaction unit开始工作。当有compaction task完成的时候，driver thread就会被中断来执行：将获得的compacted data block传输回host内存中；将completed tasks压入result queue；



为了使用FPGA进行compaction，通过PCIe将指令和数据进行传输。

Instruction Path-用于传输小的，经常使用的数据(指令)

Compaction Data Path-用来传输用于compaction的数据，使用的是DMA(direct memory access)， 无需CPU的参与。

Memory Management Unit(MMU)-用来申请从device申请空间用来存储到来的数据。

**EVALUATION**

除了验证CU的分析模型外，我们首先使用微基准测试来评估压缩单元(CU)在FPGA上的表现。然后，我们评估了采用不同工作负载的fpga -卸载压缩对X-Engine的贡献。X-Engine是最先进的lsm树KV存储.