解析WiredTiger实现：数据组织

袁荣喜

2016-06-12

在《WiredTiger实现：事务》一文中提到了WiredTiger（简称为WT）是为了适应现代CPU、内存和磁盘的特性而设计的存储引擎，它的特点就是充分利用CPU的速度和内存的容量来弥补磁盘访问速度不足。在介绍WT的组织方式前，先来看看传统数据库引擎的数据组织方式，传统数据库引擎大都是设计一个磁盘和内存完全一样的数据组织方式—page,这个结构是固定的空间大小（innodb的page是16KB）,访问它必须遵守严格的The FIX Rules规则：

 修改一个page需要获得该页的x-latch lock

访问一个page是需要获得该页的s-latch lock或者x-latch lock

持有该page的latch直到修改或者访问该页的操作完成,latch unlock

WT没有像传统的数据库引擎那样设计一套内存和磁盘page完全一致的数据组织方式，而是针对磁盘和CPU、内存三者之间特点设计了一套独特的数据组织方式，这种数据组织结构分为两部分：

In-memory page: 内存中的数据页(page)

Block extent: 基于磁盘文件的偏移量的范围存储

WT内存中的page是一个松散自由的数据结构，而磁盘上的extent只是一个变长的序列化后的数据块，这样做的目的有以下几点：

1.内存中的page松散结构可以不受磁盘存储方式的限制和The FIX Rules规则的影响，可以自由的构建page的无锁多核并发结构，充分发挥CPU多核的能力。

2.可以自由的在内存page和磁盘extent之间实现数据的压缩，提高磁盘的存储效率和减少I/O访问时间。

3.extent的数据进行压缩后比原始数据小很多，在写入文件时，可以充分利用操作系统文件的高速缓冲区(page cache)来提高读写速度。

关于WT的压缩效率见下图

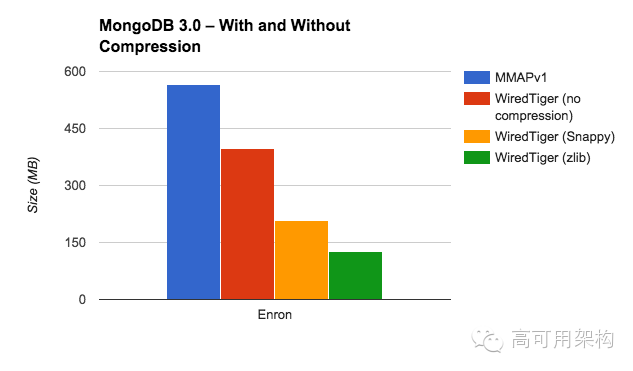


图1

如果需要了解整个WT引擎的原理，首先应该分析和理解它的数据组织方式。在本文中我们将会重点围绕WT的设计目标来分析和梳理WT的数据组织方式和实现。

# WT测试样例

在分析WT数据组织之前，先来看看WT这种数据组织方式产生的效果，我们在一个普通的开发机器上对WT做了原生的测试。测试环境如下：

CPU: i7-4710MQ CPU @ 2.50GHz，8核

内存：4G

硬盘：1TB SATA,5400转

WT数据库配置：

Cache size: 1GB, page max size:64KB, OS page cache:1GB

Key:一个从零开始自增长的整数

Value:一个长度介于100 ~ 200的随机字符串。

测试流程：

测试程序先新建一张表，用8条线程并发向表中插入指定数量（以百万为单位）的K/V对，在插入完成后做一次checkpoint让插入的数据写入磁盘，统计这个表在磁盘上的大小和这个过程的耗时。再用单线程在表中用随机查询1000个不同K/V的查询并统计查询的耗时,通过耗时可以计算出insert和query的TPS。统计的这三个参数分别就是磁盘空间占用、写性能和读性能。我们分别进行不压缩extent的测试和进行ZIP压缩extent的测试。

磁盘空间统计对比：

图2

从上图可以看出，2亿条记录，没有压缩的磁盘空间30GB，而压缩后的磁盘空间2GB左右。

写性能对比:

图3

从上图可以看出，写入的数据在1亿条（14GB）压缩(518K TPS)和没压缩(631k TPS)的写性能差距不大，但随着数据量增大，大量的数据在内存和磁盘间swap，选择extent压缩的写性能(790K TPS)要好于没压缩的写性能（351K TPS）。

读性能对比：

图4

从上图可以在数据磁盘上的数据没有超过WT的OS\_page\_cache限制时，**无压缩的读性能**要好于**有压缩的读性能**，这是因为extent缓冲在操作系统高速缓冲区中，当发生访问时从高速缓冲区直接获取无压缩extent转换成内存中的page，只要一次内存中的结构重建即可，而压缩的extent需要进行解压缩后再重建，所以没有压缩的读性能更好。 如果超出OS\_page\_cache限制后，extent数据要从磁盘上读取，而有压缩的extent占用的空间比较小，从磁盘上读取的I/O访问时间是小于无压缩读取的时间的，这时有压缩的读性能要好于无压缩的读性能。（后面单独用一个篇幅来分析WT的磁盘I/O访问及调度管理）

除了压缩优化了数据的读写，WT内存中的无锁page结构也使得读写操作具有更好的并发性，才使得WT在非常普通机器上有如此好的表现。

# 数据页(in-memory page)

## 从磁盘上读取的row对象

### K/V cell结构

### Cell与row的关系

## 内存中的修改

### K/V跳表(skiplist)

### 内存中的K/V

### 内存中value的MVCC list

## Overflow page

## 页内检索

# Disk extent结构

## Extent address结构

## Page header

## Block header

## 数据区

# page磁盘读写

## 压缩

### Page整体压缩

### Key的前缀压缩

### 页内值字典引用存储

## 读过程

## 写过程

# 后记