# KV设计文档

## 基本的数据结构

### 1)采用Btree树来存储Key，以及Value所在文件的偏移量



其中白色框表示Btree的节点，对应磁盘中的data page页，page页分为两种，一种为meta page页，如下图2所示，每当启动加载数据时，总是从meta page加载，从中解析出Root 节点，这样通过root 节点可以递归的构建整颗Btree树，data page 页的数据结构如下图3所示。Pagesize 的大小对应于系统的page页大小，或者是它的整数倍，这样就可以通过memmap的方式直接对应到高速缓冲区，而不需要拷贝到用户态。



其中Key所对应的Value是存储在数据文件中，每一个value，只通过append方式添加的文件的末尾，删除的，只要把删除标志位置为删除就行了。



### 2) Value存储内存池

对于新写入或者的删除的数据，并不需要立即刷入到磁盘中，而是先写到内存块MemeryTail中，MemeryTail通过链表的方式组织，每一个MemeryTail的大小可以自行设置，默认为2M，当MemeyTail 写满的时候，把满的MemeryTail append到Fullqueue中，再从emptyQueue中获取空的MemeryTail，当Fullqueue 中MemeryTail，大于一定阈值的时候，起一个goroutine把所有满的MemeryTail，刷入到磁盘中。把空的MemeryTail重新放到EmptyQueue中，待再次使用。



## 基本的算法策略

### 1) Btree节点的LRU策略

由于的内存的容量是有限制的，所以当数据量增大，Btree的节点增加到一定数量的时候，需要把最不常用的节点换出到磁盘的page上面去。

### 2) 数据文件的GC策略

由于数据的更改或者删除，都会在数据文件上面留下空洞，因此需要在空洞率超过一定阈值的时候，对磁盘中存在空洞，进行回收。GC数据是通过单独起一个线程去拷贝文件的数据。首先对数据文件进行扫描，遇到没有被删的数据重新插入到新的Bree树当中，当遇到数据文件eof的时，对old btree加上写锁， 然后gc线程把old btre内存中的数据flush到文件当中，刷完之后，把剩余的数据插入到new btree中去。



## 基本的测试用例

### 1.put/ get测试

1) 随机插入正常的10000000条数据后，退出程序，再次运行，查看数据是否都存在。

2) 插入异常的数据，插入空的key，看是否能插入成功。插入空的value，看是否能插入

成功

### 2. delete测试

1) 先插入一条数据，再删掉，再查看是否能删除成功

### 3.iterate测试

1) 随机插入10000000条数据，测试是否能把数据range出来，有两种情况，一开始的数据是否在数据库中，另外一种是不存在

### 4. GC测试

1) 随机生成10000000条数据，其中每一个key随机重复N次。看是否能成功GC。

## 性能测试

此次测试的环境配置：

CPU: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz

内存：16g DDR3

磁盘：机械磁盘sata盘7200转

目前只支持单线程写入，因此只测试单线程的性能

### 1.PUT性能：

从空数据库开始插入1000万条数据，key为int64整数，value大小255byte

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 存储引擎 | 耗时 | QPS | 最大put耗时/平均耗时 | 超过10ms的次数 | 内存最高使用 | 磁盘数据 |
| DDJkv | 18.924s | 528541qps | 2.975s/1.89us | 8 | 1.37g | 2.5g |

从1000万再插入1000万不重复的数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 存储引擎 | 耗时 | QPS | 最大put耗时/平均耗时 | 内存最高使用 | 磁盘数据大小 |
| DDJkv | 18.312 | 546089 | 2.909s/1.83us | 2.3g | 5.3g |

### 2. GET性能测试

先随机插入1000万条数据，再GET出所有的数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 存储引擎 | 总计耗时 | QPS | 最大get耗时/平均耗时 | 超过10ms的次数 |
| DDJkv | 26.647 | 375276 | 3.562s/2.66us | 8 |

### 3. SCAN性能测试

先插入1000万条数据，scan 1000条数

|  |  |
| --- | --- |
| 存储引擎 | 平均耗时 |
| DDJkv | 210.21us |