实验心得

1. 19335258 余世龍

在本次实验中，我们的实验目的是并行化bulkloading来更高效地构建B+树，在修改前的源代码中，首先串行构建了叶子结点，然后再串行自底向上、从左往右构建了索引结点，所以我们也分别并行化这两个过程，难点在于，在并行构建的过程中，block需要按照顺序，不能因为并行化而打乱了block的顺序，经过源代码的阅读，对两种结点以及该B+树整体结构有清晰的认识，便会容易许多，我们组经过讨论和思考，拟定了三种实现方案，写入了小组报告中。

在阅读源代码的过程中，我体会到了一些代码中对B+树的设计和构建的巧妙之处，例如结点首先被构建，在处理构建下一个结点的过程中，对上一个结点进行析构，就存入了文件中。其次，理解到了自底向上构建结点相比于自顶向下的许多优势，当last_end_block == last_start_block时意味着当前层只剩一个索引结点，就代表索引结点构建完毕了，且唯一的结点就是根结点。在结点的数据结构上，虽然两种结点内部的结构区别很大，但是都是略小于一个块的大小（512），既充分利用了空间，同时方便定位结点（块）在文件中的位置。报告中的第一部分是我写的，所以更多的理解在这就不再赘述了。

在分工中，我首先阅读了《并行程序设计导论》一书的几个主要、重要的章节，编书写了阅读笔记和多线程函数的使用两个文档，供组员快速学习、了解和使用。其次，我还参与了组内对于实现并行化的讨论以及参与编写了叶子结点和索引结点的并行化。除此我编写了检查函数来检查并行化结果的正确性，并对代码进行了性能优化和调整参数，参与书写了部分小组报告。

在做这次大作业之前，我对B+树的数据结构的理解比较薄弱，对文件读写以及多线程编程都练习的很少，所以在这次大作业中，我最大的收获就是加深了数据结构的理解以及阅读了《并行程序设计导论》一书中几个重要章节的内容，练习到了文件读写和多线程编程。

最后想说，某些位组员态度不积极，真的导致积极参与大作业的组员工作量翻倍，并且少了集思广益和多人协作的好处。还有的组员，讨论和代码编写全都不参与，分配了一点报告的编写任务，直接上网随便搜了些错误的东西就复制上去，差点害了整组，让我有些失望，本人本身基础也一般，心有余而力不足，已经尽力了，实现方面的许多不足和缺陷还望见谅。

最后，感谢老师和Ta们贡献的这次巧妙并且有挑战性的课程设计作业，让我获益良多。

2. 19335118 梁冠轩

在写实验心得前和TA说明一下组员马超的情况：一行代码不写，不与组员讨论代码实现方法，编写实验报告第一部分bulkload的理解，上百度一搜就找到了原文，而且是毫不相干的东西，导致我们要临时补救修改实验报告。

bulkloading的理解

每一个block的大小都是固定的，为每一个叶子结点分配block号，block号在叶子结点链中是顺序的，在该block中存储该叶子结点的key数组和id数组。通过block号，可以重新加载叶子结点和索引结点，这是为了实现索引结点的构建，每一个索引结点也会分配一个block值，也是顺序的。当索引结点的层数为第一层时，该层的儿子结点为叶子结点，重加载得到的叶子结点，可以根据他的block号找到他的key值，往索引结点中写入key值和block值。当索引结点层数大于一时，该层的儿子结点为索引结点，也可以通过他的block号重加载索引结点，得到他的key值。逐层往上构建索引节点，直到根节点构建完成。

通过bulkloading，可以大大节省构建B+树的资源和时间，若能通过并行实现bulkloading，速率可能会更快。

线程池的实现

本次实验要求使用到多线程并行操作，为了实现实验的要求，去学习掌握了pthread并且个人独立实现了线程池的全部结构和操作。

线程池简单来说就是有一堆已经创建好的线程，最大数目一定，初始时他们都处于空闲状态，当有新的任务进来，从线程池中取出一个空闲的线程处理任务，然后当任务处理完成之后，该线程被重新放回到线程池中，供其他的任务使用，当线程池中的线程都在处理任务时，就没有空闲线程供使用，此时，若有新的任务产生，只能等待线程池中有线程结束任务空闲才能执行。因为线程的创建、和清理都是需要耗费系统资源的。假设某个线程的创建、运行和销毁的时间分别为T1、T2、T3，当T1+T3的时间相对于T2不可忽略时，线程池的就有必要引入了，尤其是处理数百万级的高并发处理时。线程池提升了多线程程序的性能，因为线程池里面的线程都是现成的而且能够重复使用，我们不需要临时创建大量线程，然后在任务结束时又销毁大量线程。一个理想的线程池能够合理地动态调节池内线程数量，既不会因为线程过少而导致大量任务堆积，也不会因为线程过多了而增加额外的系统开销。

要实现线程池，需要使用到pthread中的pthread_mutex_t和pthread_cond_t数据结构。pthread_mutex_t用于实现线程的互斥加锁，pthread_cond_t用于控制线程的状态。

线程池的基础结构，针对执行需要的线程任务，可对线程结构进行修改，添加参数（本次实验就使用到了四个参数）。其他线程池的操作在实验报告中有所讲述，在此不再多说。

```
typedef struct task{
    void *(*run)(void *args);   //需要执行的任务
    void *arg;                  //参数
    struct task *next;
}task;

typedef struct condition{
    pthread_mutex_t pmutex;
    pthread_cond_t pcond;
}condition;

typedef struct threadpool{
    condition condition_;   //状态量
    task *first;            //任务队列中第一个任务
    task *last;             //任务队列中最后一个任务
    int run_thread;         //运行线程数
    int space_thread;       //空闲线程数
    int max_thread;         //最大线程数
    int quit;               //是否退出标志
}threadpool;
```

并行实现bulkload

bulkload主要分为两部分，一部分是通过读入table中的id和key值，实现叶子结点的构建，另一部分是通过叶子结点逐层构建索引结点。

为了实现并行操作，个人独立设计并且完成了两个并行函数void* task_leaf_node和void* task_index_node，分别负责叶子结点的并行构建和索引结点的并行构建，两个方法的设计思路相似。

通过之前不断地尝试debug得知，即使叶子结点的block是从小到大顺序的，但是如果叶子结点的文件写入不是顺序的话，在构建索引结点时，通过block值重加载得到的叶子结点，该叶子结点得到的key值有可能是错误的。

所以为了减少错误的发生，选择了一个较为简单的方法：通过阅读代码，可以得知一个叶子结点需要读入115个id值，创造一个叶子结点数组，通过计算可以得知一共需要多少个叶子结点，然后将这些叶子结点先初始化，最后多个线程并行往叶子节点中写入数据的方法。

小组成员谢忠清曾提出过双层同时构建索引结点的方法，但由于多次实验也无法正确实现，最后选择了叶子节点相似较为简单的并行构建方法：由于索引结点数量较少，可以使用两个线程并行构建一层索引结点，方法与叶子节点相同，逐层往上，最后根节点只需要一个线程进行构建。

3. 19335228 谢忠清

本次实验要求我们以小组形式将串行实现的B树Bulkload改写成并行，ta提供的样例代码虽然对我们来说还是挺复杂的，但是主体的Bulkload方法并不难懂，通过查看研究那些底层的api，我也学到了许多通过代码与硬盘交互的方式、帮助我巩固了指针和链表的使用等等。

读完代码后，我的第一想法是，叶子结点的并行操作空间不大，应该也就是双线程从两边向中间建立，反向的线程需要一个倒着add_leaf_node的方法，到中间后合并+平衡最中间的两个结点，或者可以提前计算好所需的结点数目，给两个线程预先分配好工作量，也能规避最后相遇处的结点key数目小于容量的一半的问题。

关于索引结点的建立，我一开始的设想同样是两个线程交替建立索引结点，如下所示：

pth2

pth1 (1,3) (5)　　//进行到第三步时，将5加入后需要新建结点

　　//确认pth1层不为root，传信号给pth2建立结点

叶子 (1,2) (3,4) (5,6) (7,8).....

————————————

pth2 (1,5)

pth1 (1,3) (5,7)　　//此时正常将pth1，2两层建立完成后，变成如下所示：

叶子 (1,2) (3,4) (5,6) (7,8).....

————————————

pth2

pth1

“叶子”（pth2先前建立）(1,5)(9,13)..... //重复上述情况

后经小组其他同学启发，设计出新的算法，可以用线程池，使得若干层索引结点同时建立，同时完成，效率更高，核心思想大概为：

设索引共需要有n层，每一层由一个线程建立，从第1层开始建立，若第k层结点数量超过1，则说明k<n，索引未完成，需建立k+1层。第k层增加新结点时，应发信号给第k+1层的线程，将新结点插入，若因此需要新建结点，则同样上传信号到k+2层的线程，以此类推。当索引全部建立完成后应进行平衡，因为某层的最后一个结点容量可能不足一半，只需将最右边的两个结点中的key平均分即可，因为B(容量)+x(x>=1)>B/2。

实例与上面双线程的类似，只是pth2会进行同pth1类似的操作，即结点超过1个后会立刻上报，申请线程建立新一层。

可惜由于能力有限，上述算法均未能实现。。。。

总的来说，本次大作业还是让我受益匪浅，让我学到了很多东西，进行了很多思考，就连ta给的例程也是一笔宝贵的财富www，同时感谢小组组员们的辛勤付出。

4. 19335086 黄杰豪

這次的課程設計終於結束了, 十分感謝其他組員的付出, 因為這次的作業對代碼能力很弱的我來說實在是太困難了, 所以如果只有我的話甚麼都做不了, 我在小組中的貢獻很小, 但其他同學真的很好, 他們不僅不嫌棄我, 還帶領我學習去閱讀實驗代碼, 讓我明白如何去做這次程序的設計, 使我在這次課程設計學到了很多。特別是B+樹bulkloading過程, 讀入Table建立葉子結點和索引結點, 以及在學習并行方法時也有回顧到以往操作系統及數據結構的知識, 再連繫到數據庫。但同時也讓我意識到自己的不足, 實驗結束之後, 我也會針對我不熟識的地方去重新學習, 感謝老師佈置的課程, 更感謝我的組員, 特別是組長真的有很好地帶領我們去做這次實驗。

5. 19335152 马超

本次实验理解了B+树的结构，对B+树的构造以及作用有了更深入的了解，能够做到运用B+树来完成对应任务，同时对批量载入有了基本认知，能够做到B+树的批量载入，此外，进行了并行程序设计的实践，对并行算法的设计有了更清晰的认知