nvmlab 实验报告

------2020200982 闫世杰

使用 PMDK 库提供 api 的实现持久化的 KV 存储

实现思路:
设计合适的数据结构存储 KV,并对 KV 进行查询,个人采用二叉树进行存储
先写出 c++的版本,然后将 c++的部分接口翻译成 PMDK api 即可

数据结构:含孩子及父亲(方便查找后继)节点的二叉树

```
struct btree_node{
    char key[17]; char value[129];
    TOID(struct btree_node) left;
    TOID(struct btree_node) right;
    TOID(struct btree_node) parent;
}
```

功能的实现:
SET: 二叉树的插入操作
GET: 二叉树的查询操作
NEXT:
先向下到最底层,确定 key 是否在树中,存在则返回节点,不存在则返回最后访问到的节点
1)给出的 Key 值存在于二叉树,二叉树的后继查询操作
2)给出的 Key 值不再二叉树中
i 如果该节点的 key 值比 key 大,则该节点就是 NEXT,直接返回即可
ii 如果该节点的 key 值比 key 小,则该节点的 NEXT 就是给出 key 的 NEXT(画图很容易证明)

功能实现的思路比较简单,只有 NEXT 需要考虑两种情况,比较繁琐一点

实验主要的任务在于把实现功能的 c++代码转换为使用 PMDK api 来实现
虽然官方给出的文档说明较为详细,但整体实现起来还是存在不少细节问题,比如,内存池 pool 大小的分配(由于测试点数据量不同,pool 较小时,POBJ_ALLOC()类的分配接口无法正确执行,会 return-1,最开始没意识到这个问题,导致后续节点无内存储存,一直无法进行正常的访问)

在官方仓库中,给出了 btree 的示例,参照示例进行实现
参照的官方仓库:
https://github.com/pmem/pmdk/blob/master/src/examples/libpmemobj/btree.c
仓库中给出了建树的实现,参照代码根据需求进行修改即可实现 SET 操作
SET 操作完成后,GET 和 NEXT 操作较为简单,只要使用合适的变量类型对构造出的树形结构进行访问即可

缺点:没有使用事务进行操作,持久化实现都是手动实现,代码量可能稍多一点