实验报告

一、实验结果

本地测试所有测试点，结果均正确，提交到 ics.men.ci 后得分 71

编号	题目	状态	分数	总时间	内存	代码 / 答案文件	提交者	
#7973	#A. NVMLab	Partially Correct	71	14848 ms	337.96 M	C++ 17 (nvm) / 3.1 K	救命啊	2022

二、使用的相关 api

1. pmemobj_root(pop, sizeof(T));

获得存储在 pool 中数据结构的入口

2. pmemobj_persist(pop, mem_address, size_t);

持久化 size_t 大小的内存空间，其中 mem_address 为实际内存地址

3. pmemobj_memcpy_persist(pop, mem_address, void *, size_t);

持久化内存拷贝操作

4. pmemobj_direct(PMEMoid data)

获得 data 对象的实际内存地址

5.事务功能：

```
TX_BEGIN(pop) {
    /* TX_STAGE_WORK: 事务想要完成的功能区块*/
} TX_ONCOMMIT {
    /* TX_STAGE_ONCOMMIT: 事务提交时需要做的额外工作*/
} TX_ONABORT {
    /* TX_STAGE_ONABORT: 事务回滚时需要做的恢复工作*/
} TX_FINALLY {
    /* TX_STAGE_FINALLY: 事务 commit 或者 abort 都会执行的区块*/
} TX_END
```

三、实现思路及代码说明

设置结构体类型 kv 用于存储一对 key-value

```
struct kv{
    char k[key_length + 1];
    char v[value_length + 1];
};
```

修改持久化内存中的数据结构 my_root，其包含两个成员变量，my_state 是 kv 数组，用于存储 key-value 对。len 既表示 kv 数组中键值对的个数，又可以作为插入新的键值对时需要的偏移量

```
struct my_root
{
    int len;
    kv my_state[MAX_BUF_LEN];
};
```

修改纯读测试时的代码，使其将键值对从持久化内存读入到 state 中，以进行 GET 和 NEXT 操作。使用 pmemobj_root 和 pmemobj_direct 从持久化内存中得到 my_root 数据结构的指针，将 my_state 数组中的键值对依次读取后添加到 state 中即可。

```
if (file_exists(filename) != 0)
{
    //PMEMOBJ_MIN_POOL
    pop = pmemobj_create(filename, "QAQ", 150L<<20, 0666);
}
else
{
    pop = pmemobj_open(filename, "QAQ");

    PMEMoid r_root = pmemobj_root(pop, sizeof(struct my_root));
    struct my_root *r_rootp = (struct my_root *)pmemobj_direct(r_root);

    for (int i = 0; i < r_rootp->len; i++)
    {
        char k[key_length + 1] = {0}, v[value_length + 1] = {0};

        strcpy(k, r_rootp->my_state[i].k);
        strcpy(v, r_rootp->my_state[i].v);

        state[k] = v;
    }

    do_not_dump = true;
}
```

而在进行 SET 操作时，则需要将 q.key 和 q.value 写入持久化内存。先用 kv 类型的临时变量 qdata 存下 q 中的键值对，然后用 pmemobj_persist 更新 len（加 1），并使用 pmemobj_memcpy_persist 来实现 my_state 数组中新的键值对的添加

此外还用 TX_BEGIN 和 TX_END 实现 set 操作的原子性。

```
case Query::SET:
    TX_BEGIN(pop){
        kv qdata;
        strcpy(qdata.k, q.key.c_str());
        strcpy(qdata.v, q.value.c_str());
        rootp->len++;
        pmemobj_persist(pop, &rootp->len, sizeof(rootp->len));
        pmemobj_memcpy_persist(pop, rootp->my_state + (rootp->len - 1), &qdata, sizeof(qdata));
    }TX_END
```