Jaeyeol Kim

Look into source code

1. Where it start?
   we can find the initialize function in pblk-init.c. The initialize function is pblk_init

```
/* physical block device target */
static struct nvm_tgt_type tt_pblk = {
        .name           = "pblk",
        .version        = {1, 0, 0},

        .make_rq        = pblk_make_rq,
        .capacity       = pblk_capacity,

        .init           = pblk_init,
        .exit           = pblk_exit,

        .sysfs_init     = pblk_sysfs_init,
        .sysfs_exit     = pblk_sysfs_exit,
        .owner          = THIS_MODULE,
};

static int __init pblk_module_init(void)
{
        int ret;

        pblk_bio_set = bioset_create(BIO_POOL_SIZE, 0, 0);
        if (!pblk_bio_set)
                return -ENOMEM;
        ret = nvm_register_tgt_type(&tt_pblk);
        if (ret)
                bioset_free(pblk_bio_set);
        return ret;
}

static void pblk_module_exit(void)
{
        bioset_free(pblk_bio_set);
        nvm_unregister_tgt_type(&tt_pblk);
}

module_init(pblk_module_init);
module_exit(pblk_module_exit);
MODULE_AUTHOR("Javier Gonzalez <javier@cnexlabs.com>");
MODULE_AUTHOR("Matias Bjorling <matias@cnexlabs.com>");
MODULE_LICENSE("GPL v2");
MODULE_DESCRIPTION("Physical Block-Device for Open-Channel SSDs");
```

2. When does gc start?
   a. It starts at pblk-init.

```
        goto fail_free_rwb;
    }

    ret = pblk_writer_init(pblk);
    if (ret) {
        if (ret != -EINTR)
            pr_err("pblk: could not initialize write thread\n");
        goto fail_free_l2p;
    }

    ret = pblk_gc_init(pblk);
    if (ret) {
        pr_err("pblk: could not initialize gc\n");
        goto fail_stop_writer;
    }
```

b. at the pblk_gc_ init it starts several thread for garbage collection

```
int pblk_gc_init(struct pblk *pblk)
{
        struct pblk_gc *gc = &pblk->gc;
        int ret;

        gc->gc_ts = kthread_create(pblk_gc_ts, pblk, "pblk-gc-ts");
        if (IS_ERR(gc->gc_ts)) {
                pr_err("pblk: could not allocate GC main kthread\n");
                return PTR_ERR(gc->gc_ts);
        }

        gc->gc_writer_ts = kthread_create(pblk_gc_writer_ts, pblk,
                                                        "pblk-gc-writer-ts");
        if (IS_ERR(gc->gc_writer_ts)) {
                pr_err("pblk: could not allocate GC writer kthread\n");
                ret = PTR_ERR(gc->gc_writer_ts);
                goto fail_free_main_kthread;
        }

        gc->gc_reader_ts = kthread_create(pblk_gc_reader_ts, pblk,
                                                        "pblk-gc-reader-ts");
        if (IS_ERR(gc->gc_reader_ts)) {
                pr_err("pblk: could not allocate GC reader kthread\n");
                ret = PTR_ERR(gc->gc_reader_ts);
                goto fail_free_writer_kthread;
        }

        timer_setup(&gc->gc_timer, pblk_gc_timer, 0);
        mod_timer(&gc->gc_timer, jiffies + msecs_to_jiffies(GC_TIME_MSECS));

        gc->gc_active = 0;
        gc->gc_forced = 0;
        gc->gc_enabled = 1;
```

3. what is pblk_gc_ts
    a. main thread for garbage collection
    b. when gc operate
        i. gc is activate & free block in under the threshold
        ii. read_inflight_gc is lower than PBLK_GC_L_QD
            1. inflight_gc seems to be a parallel unit.
4. Overall Sequence of GC
    a. Get victim line
    b. Change line state(to PBLK_LINESTATE_GC)
    c. delete from linelist
    d. add the line into r_list - maybe ready list
    e. increase read_inflight_gc
    f. pblk_gc_reader_kick
        i. gc_reader_ts
        ii. pblk_gc_reader_ts → it is a thread
        iii. pblk_gc_read
    g. pblk_gc_should_run
        …..

there is some more sequence, I think that it move the victim line into wright buffer.
after write the data just decrement the reference count of the target line.

5. what is BIO
    - https://lwn.net/Articles/26404/