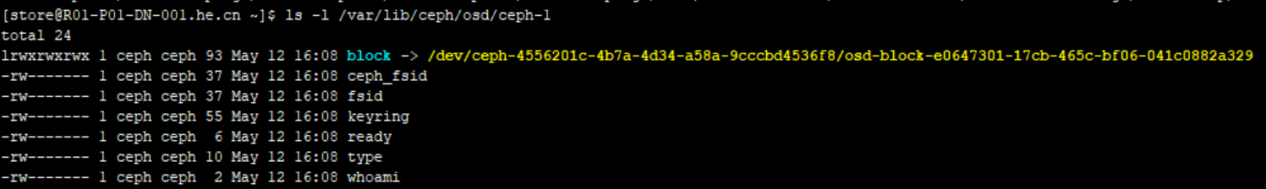
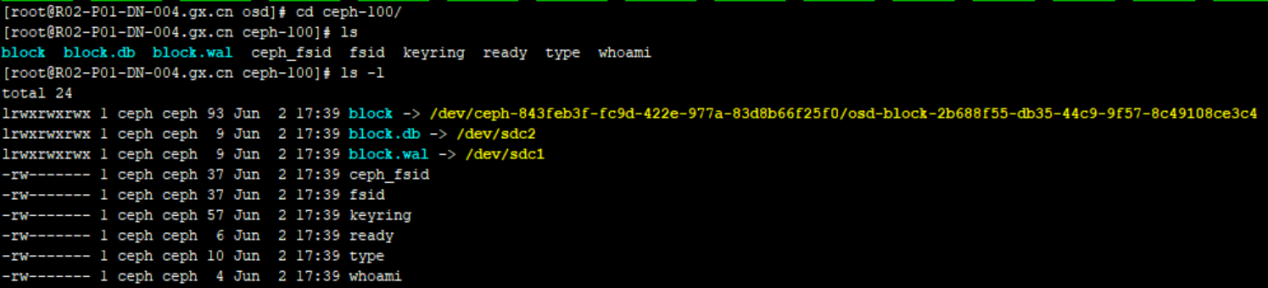
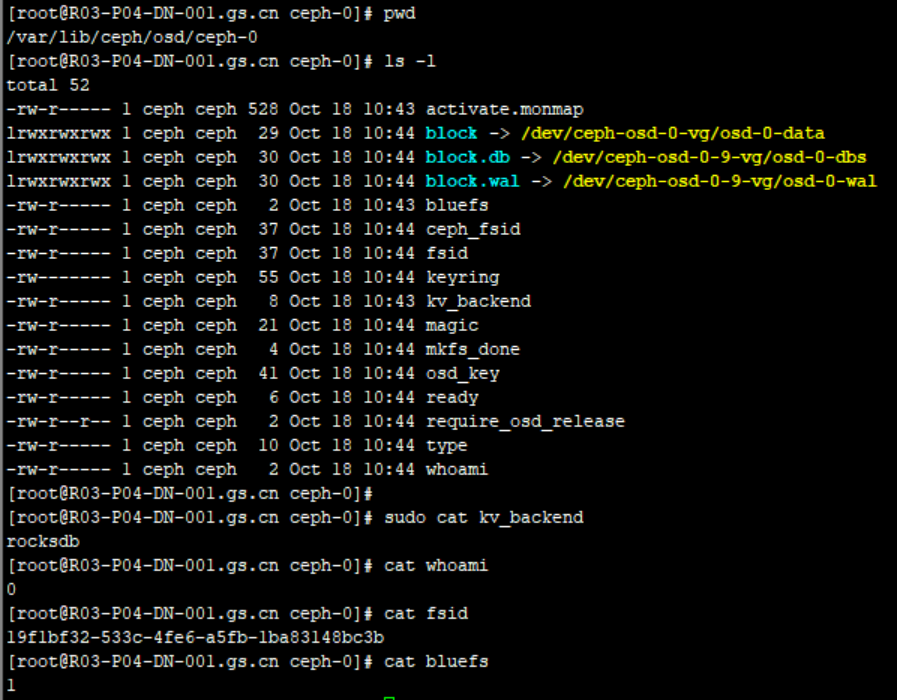
WAL、DB、DATA 共用



WAL、DB、DATA分开



Bluestore.mkfs



1. 读取mkfs\_done文件，检查是否已经进行过格式化;
2. 读取type文件,检查store类型
3. 固定使用freelist\_type=bitmap
4. 打开path路径（对应上图/var/lib/ceph/osd/ceph-0）
5. 打开fsid文件，得到fsid文件的句柄，保存在成员变量中;读取fsid文件内容，如果文件为空，生成新的fsid。读取文件时，对文件加锁。
6. 建立path/block文件到配置指定的数据存储设备的连接（对应上图/dev/ceph-osd-0-vg/osd-0-data）；
7. 如果是使用bluestore，建立path/block.wal到配置指定的日志存储设备的链接(/dev/ceph-osd-0-vg/osd-0-wal)；建立path/block.db到DB存储设备的链接(/dev/ceph-osd-0-vg/osd-0-dbs)。
8. 打开path/block设备（主设备）,如果不存在则创建。创建BlockDevice对象，并调用open方法，获得设备大小dev\_size，如果设备支持标签的话（NVME设备不支持），设置标签为main；初始化全局参数block\_size、block\_mask、block\_size\_order等。
9. 从配置项bluestore\_min\_alloc\_size里面获取最小配置单元min\_alloc\_size,如果没有配置，则判断设备是否HDD（通过设备类的is\_rotational方法来判断，在一般linux系统中，可通过读取/sys/block/sdX/queue/rotational 来判断是否为HDD盘，对应源码src/common/blkdev.cc block\_device\_is\_rotational），如果是HDD,则使用bluestore\_min\_alloc\_size\_hdd配置项作为min\_alloc\_size，否则使用bluestore\_min\_alloc\_size\_ssd。
10. \_open\_db(true),打开db如果不存在则创建,打开db时，默认不进行修复操作。如果是新建DB，则从配置里面读取bluestore\_kvbackend和bluestore\_bluefs，否则，从元数据中读取kv\_backend（对应上图kv\_backend文件、bluefs文件，bluefs文件的内容为1对应程序变量do\_bluefs=true）。Do\_bluefs为true时：创建一个BlueFS对象，判断是否有专门的path/block.db设备，如果没有，则跟path/block共享，需要将path/block设备作为BDEV\_DB;如果有专门的path/block.db设备，则先调用bluefs->add\_block\_device加入BDEV\_DB设备，如果设备支持标签，则将设备加上bluefs db标签，调用bluefs->add\_block\_extent（单独分析）将DB设备的extent加入的block\_all[BDEV\_DB]。接下来加载path/block设备，如前面所述，如果是共用情况，则将block的空间登记到BDEV\_DB下面，否则登记到BDEV\_SLOW下面。Block设备的中间部分留出一定的空间给bluefs使用，这部分空间的大小由（bluestore\_bluefs\_min\_ratio+bluestore\_bluefs\_gift\_ratio）决定，并按最小分配单元对齐，需要注意的是，这部分空间除了调用add\_block\_extent加入之外，还会加入bluefs\_extents（还有哪些空间会加入bluefs\_extents列表?Bluefs\_extents会持久化到super的元数据中）。

处理WAL空间:如果有独立的WAL设备，则调用add\_block\_device将改设备空间登记到BDEV\_WAL,如果设备支持标签，则设置bluefs wal标签。如果是新建DB，则

add\_block\_extent(BlueFS::BDEV\_WAL,BDEV\_LABEL\_BLOCK\_SIZE,bluefs->get\_block\_device\_size(BlueFS::BDEV\_WAL) -BDEV\_LABEL\_BLOCK\_SIZE)

注意WAL设备预留BDEV\_LABEL\_BLOCK\_SIZE

如果没有独立的WAL设备，将去除DB的separate\_wal\_dir选项。

调用bluefs->mkfs()

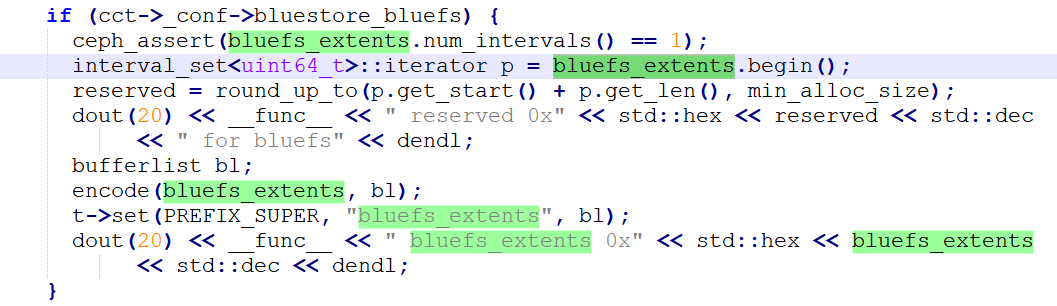
Bluefs->mount()

创建Env 和设置option

Do\_bluefs为false时：创建path/db目录，如果指定了separate\_wal\_dir选项，则再创建path/db.wal目录

创建DB（KeyValueDB::create）得到db，并db->init(options)初始化配置项，然后调用db->create\_and\_open打开db

1. Open\_fm。如果是新建文件系统，那么将freelist\_type（固定值bitmap）记入super元数据;调用fm->allocate(0, reserved, t)预留空间（SUPER\_RESERVED）；



注意到ceph\_assert(bluefs\_extents.num\_intervals() == 1); 初始情况下，这个bluefs\_extents这有一个区间，即path/block设备中预留给bluefs的空间。Reserved 是区间的结束位置按照最小分配单元对齐的结果。这个值正常情况下没有用到，打开bluestore\_debug\_prefill开关才用到。

BlueFs.Mkfs：

1. 初始化分配器。容器Block\_all保存了各个设备的extent,初始化时按照配置的创建指定类型的分配器，并调用init\_add\_free将extent加入分配器管理。（extent何时初始化？Opendb）
2. \_init\_logger初始化PerfCounters
3. 初始化super结构，并持久化到设备中。Super在DB分区位置Offset=4096，length=4096。包含version、block\_size（BDEV\_DB的大小）、logfile.ino（1）