BlueFS is a super-simple “file system”

* all metadata loaded in RAM on start/mount
* no need to store block free list
* coarse allocation unit (1 MB blocks)
* all metadata lives in written to a journal
* journal rewritten/compacted when it gets large

**metadata**

bluefs的元数据主要有superblock、dir、fnode。  
其中superblock是固定位置的，放在BDEV::DB设备的头部。  
superblock 中除了记录一些必要的标示外，最重要的就是journal文件的fnode，这个fnode比较特别，它的inode是1。  
journal文件不仅实现了bluefs的事务，同时其本身也是bluefs元数据的载体。

**how to**

journal一般是循环使用的顺序写文件，元数据如何存在journal中呢？ 原来bluefs的元数据平时全部存到内存中，重启的时候通过replay，遍历journal文件然后重建出bluefs的元数据。

这个文件是循环使用的，journal每次做checkpoint的时候会持久化下当前的dirmap和filemap，这样下次重启后replay可以不必管之前的那些journal了。bluefs mount的时候最重要的事件就是执行replay，构建dirmap和filemap，以及通过filemap得到空间管理的元数据。

**checkpoint**

既然是循环使用，顺序写，并且还要重启replay，那肯定要checkpoint来提到replay的效率。  
这里元数据有一个特殊处理，在做checkpoint的时候，会重写下所有元数据，就有就不用依赖以前的journal了。实现参考BlueFS::\_compact\_log\_dump\_metadata函数。

* 注意journal fnode里面的size并不是journal 的实际大小，而是当前写入位置。get\_allocated返回的才是journal file的真正size。

checkpoint是通过compact log来实现的。如果启动compact会释放以前的journal空间，重新分配空间。这里对于log\_seq的处理有点别扭，新设备上的journal第一条seq是1，然后又jump到当前使用的seq(因此我们在replay的时候会看到第一条journal entry的log seq必须为1)。

为什么不在super block中直接记录当前的seq呢？

这两种方式都可以实现。

为什么bluefs的log的切换，使用sync\_metadata触发？ 而不是写到多少数据自动触发?

这样实现意味着将bluefs的元数据作为rocksdb目录的元数据看待，这样也是合理的。

切换的时候故障了怎么处理？

有些操作顺序需要保证，比如必须log fnode持久化成功之后，才能释放之前的journal空间。其实这种顺序本身也是wal。fnode记录了空间分配操作，只有记录持久化成功了，才能去真正实施修改。否则中间挂了就完了。

**空间管理**

由于bluefs是在裸盘之上的，因而必须自己管理空间。  
bluefs没有使用freelist来管理空间，空间的使用情况记录在每个文件的fnode中。

bluefs默认使用bitmap allocator，最小分配空间是1MB。由于bluefs默认可支持多个设备，因而分配后的extent需要记录其空间来自哪个设备，bluefs\_extent\_t记录了这个信息。

**rocksdb结合**

在RocksDbStore中做了一些调整，比如wal路径，option.env等。 BlueRocksEnv是通过目录来区分BDEV的使用的。

如果想单独跑rocksdb + bluefs怎么搞呢？

Copy bluestoredirectory and several dependent source files from Cephsource code, and compiled BlueFSas a standalone library libbluefs.a.  
Replace db\_benchtest tool’s default POSIX environment with BlueRocksEnv

**bluefs的价值**

这个有待测试验证了，从性能角度来说个人觉得效果有限，原因有2点：  
一是因为rocksdb本身用到的文件并不多，bluefs在这种情况下不见得比xfs等优化多少。  
二是在ssd上，rocksdb本身加上dio支持后性能也优化了一些。

作者：jerry\_xr  
链接：https://www.jianshu.com/p/30948b467308  
來源：简书  
简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。