数字转型 架构演进

SACC

2019 中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2019

IT168.com ChinaUnix.net ITPUB

2019年10月31-11月2日 | 北京海淀永泰福朋喜来登酒店

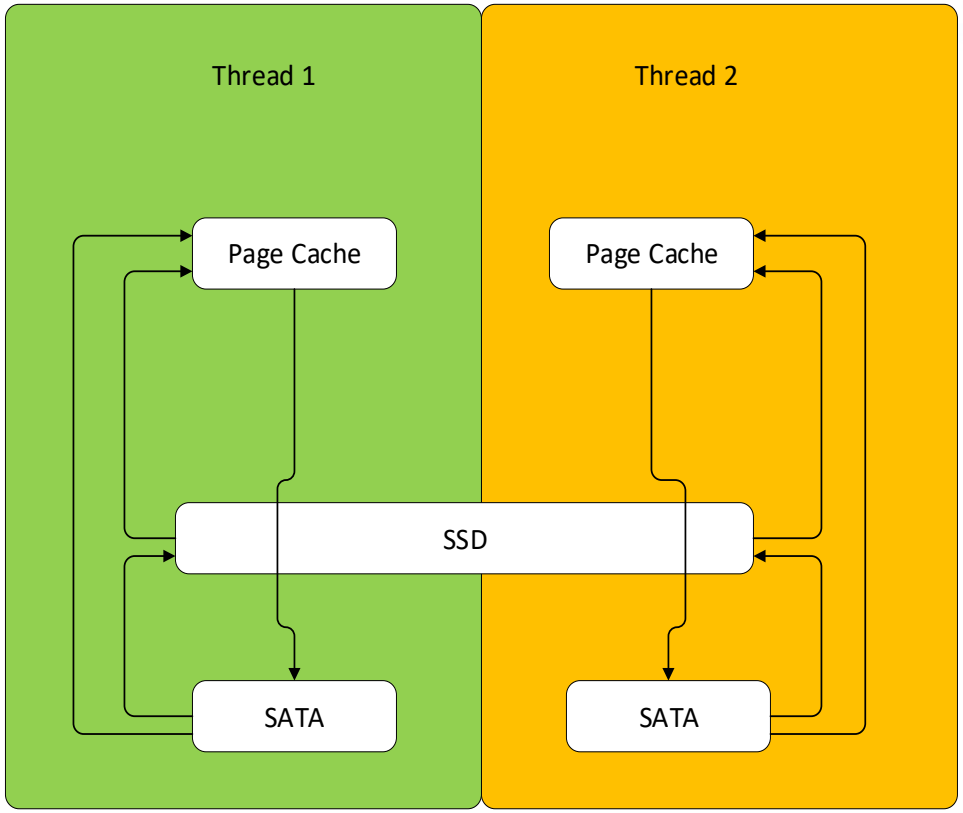
分级缓存DMA技术原理剖析

周超勇

金山云 架构师

zhouchaoyong@Kingsoft.com

SSD读缓存模式

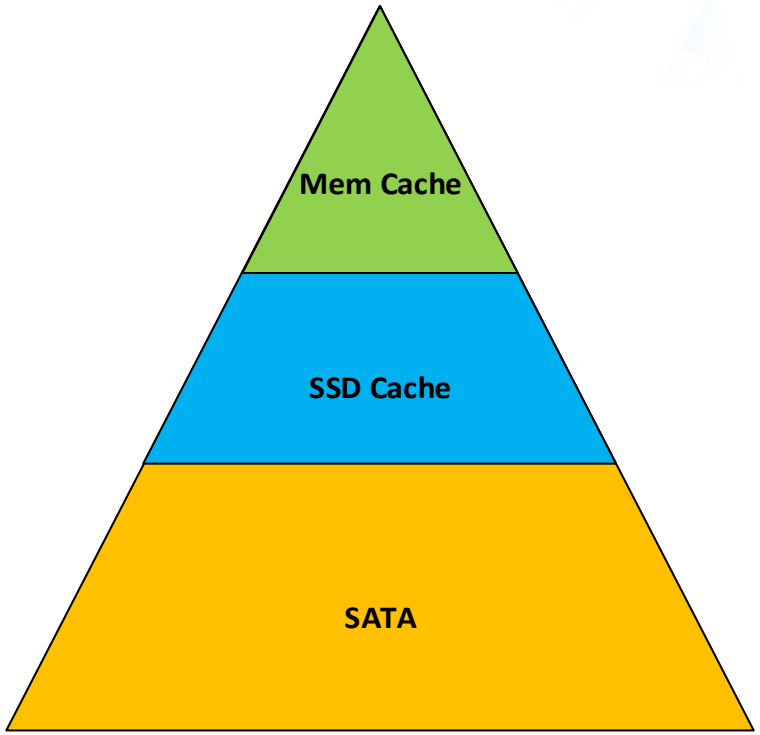


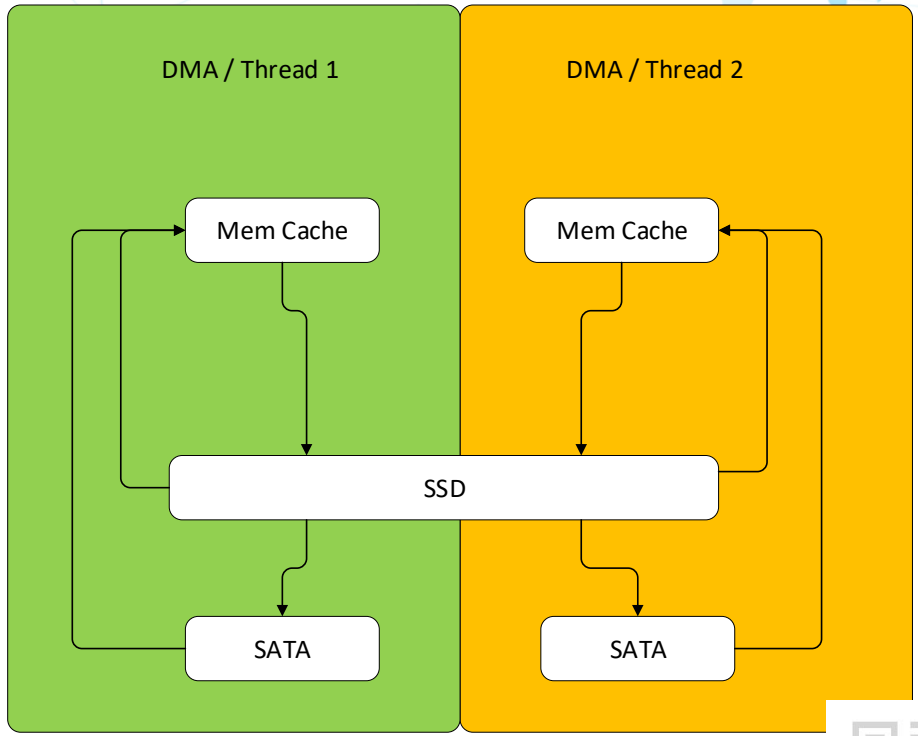
- 依赖Page Cache
- 多线程并发读写盘
- 回源数据先落SATA
- SATA多次命中后升级到SSD

性能上限被SATA提前锁定

SSD写缓存模式DMA
（ SSD Cache + Mem Cache + AIO ）

提升: 40%

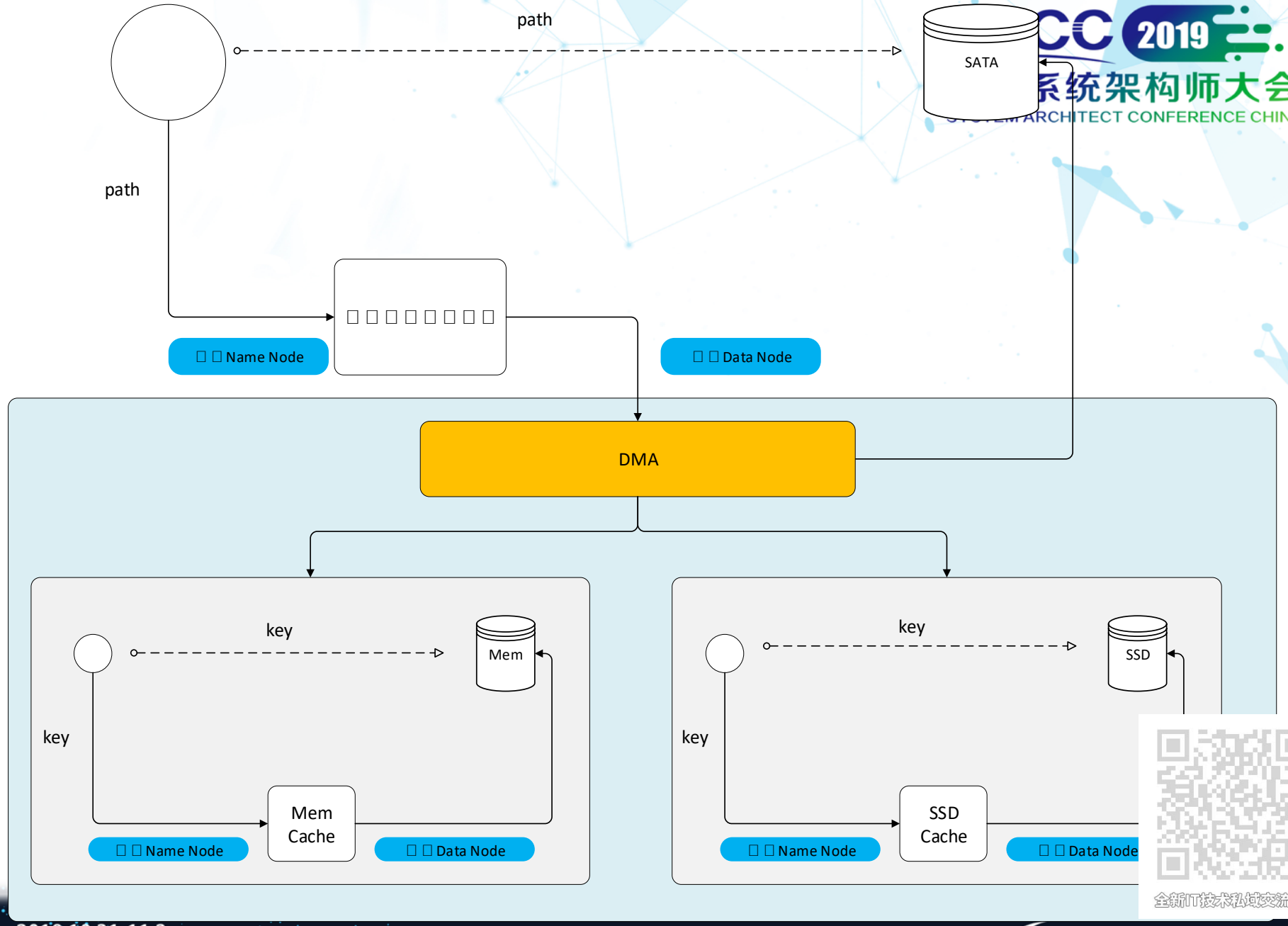




性能优先、读写优先

DMA架构

Mem Cache和SSD Cache是简化版RFS（Random access File System）



DMA之AIO

```
/* 创建一个异步IO上下文（io_context_t是一个句柄） */
int io_setup(int maxevents, io_context_t *ctxp);

/* 销毁一个异步IO上下文（如果有正在进行的异步IO，取消并等待它们完成） */
int io_destroy(io_context_t ctx);

/* 提交异步IO请求 */
long io_submit(aio_context_t ctx_id, long nr, struct iocb **iocbpp);

/* 取消一个异步IO请求 */
long io_cancel(aio_context_t ctx_id, struct iocb *iocb, struct io_event *result);

/* 等待并获取异步IO请求的事件（也就是异步请求的处理结果） */
long io_getevents(aio_context_t ctx_id, long min_nr, long nr, struct io_event *events, struct timespec *timeout)
```

Linux原生
AIO技术

DMA之AIO

struct iocb主要包含以下字段：

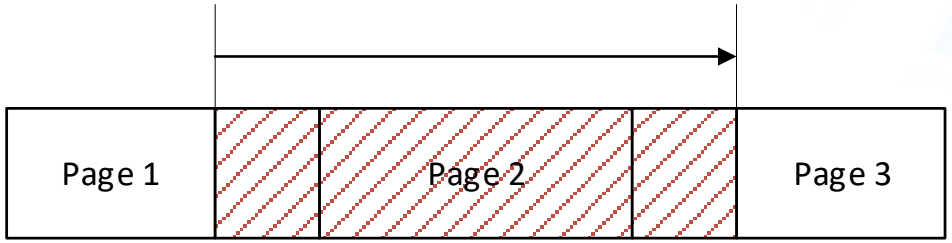
```
__u16    aio_lio_opcode;   /* 请求类型（如：IOCB_CMD_PREAD=读、IOCB_CMD_PWRITE=写、等） */
__u32    aio_fildes;       /* 要被操作的fd */
__u64    aio_buf;          /* 读写操作对应的内存buffer */
__u64    aio_nbytes;       /* 需要读写的字节长度 */
__s64    aio_offset;       /* 读写操作对应的文件偏移 */
__u64    aio_data;         /* 请求可携带的私有数据（在io_getevents时能够从io_event结果中取得） */
__u32    aio_flags;        /* 可选IOCB_FLAG_RESFD标记，表示异步请求处理完成时使用eventfd进行通知 */
__u32    aio_resfd;        /* 有IOCB_FLAG_RESFD标记时，接收通知的eventfd */
```

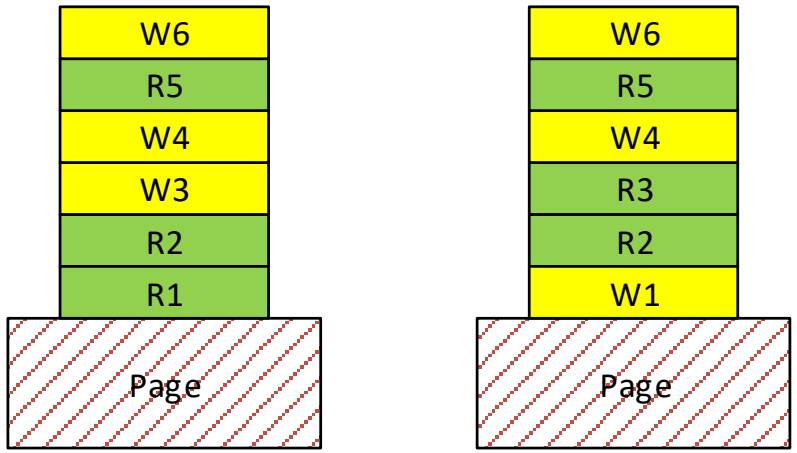
零拷贝

其中，struct io_event主要包含以下字段：

```
__u64    data;             /* 对应iocb的aio_data的值 */
__u64    obj;              /* 指向对应iocb的指针 */
__s64    res;              /* 对应IO请求的结果（>=0: 相当于对应的同步调用的返回值；<0: -errno） */
```

DMA之读写合并





- 偏移量不对齐，需要额外1~2次读盘操作
- 承诺严格时序
- 最终一致性：如果首次操作为R，至多需要两次操作：R, W
- 最终一致性：如果首次操作为W，仅需要一次操作：W

DMA之内存拷贝

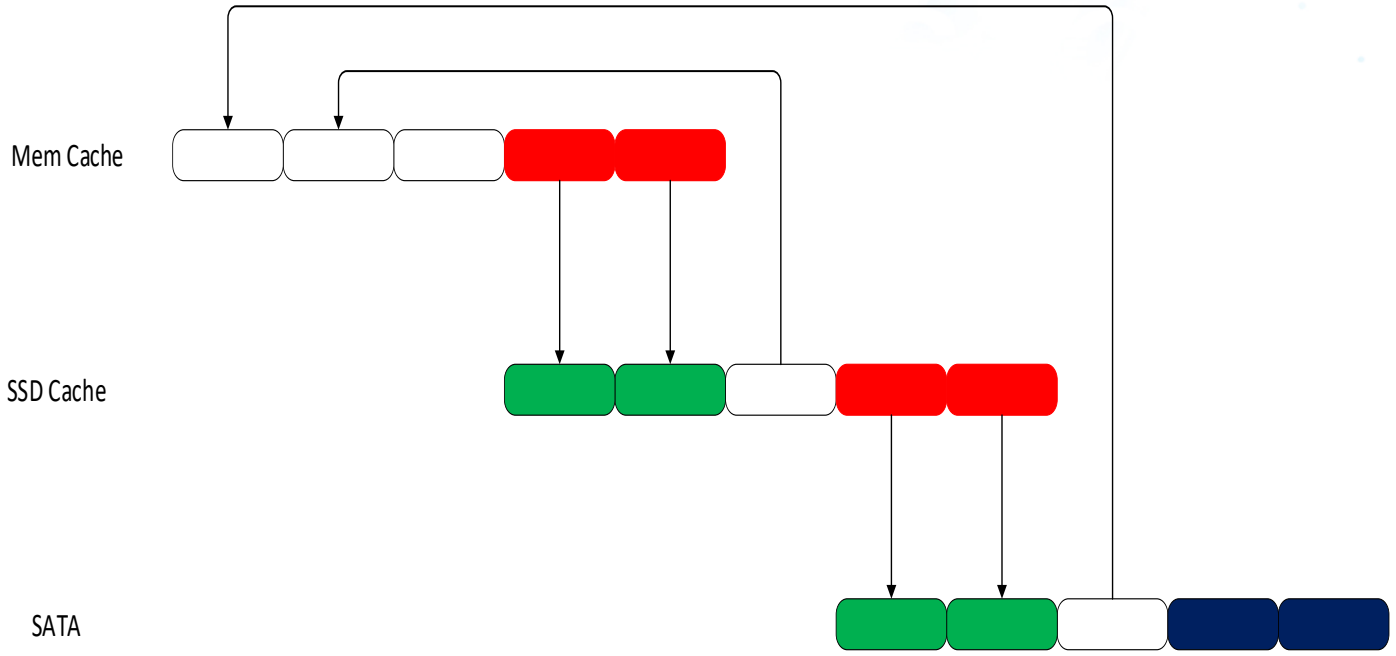
总共需要两次内存拷贝：

1. AIO的内存对齐约束，产生一次内存拷贝

2. 读写合并功能，产生一次内存拷贝

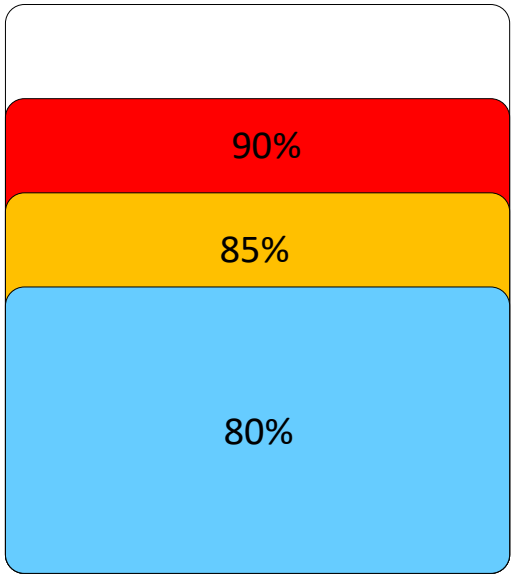
思考：能减少一次内存拷贝吗？

DMA之策略



- 降级策略
- 流控策略
- LRU策略
- 淘汰策略

DMA之流控



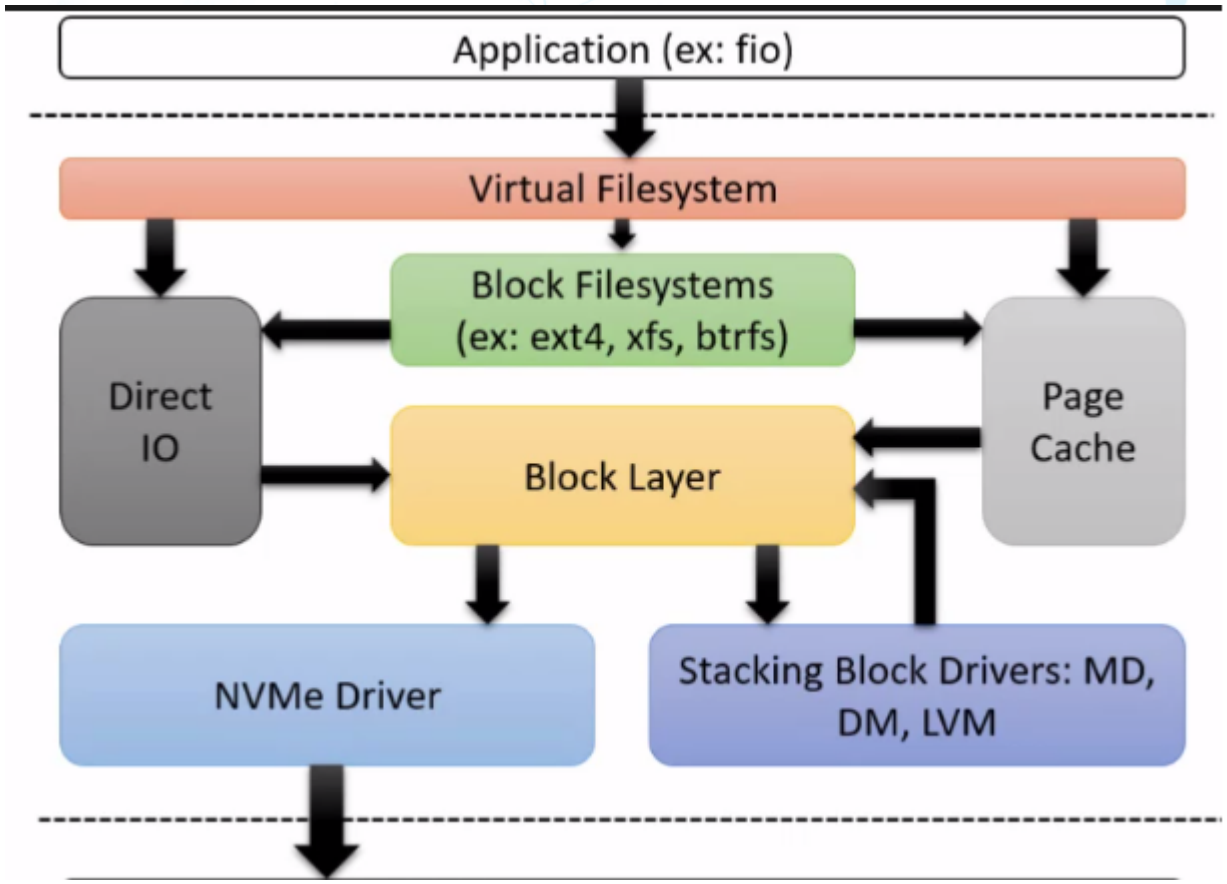
- 高、中、低三级水位
- 水位不同，流速不同
- DMA和外部流控联动

流控的目的是防止系统被打穿，以及确保系统运行流畅

DMA之优化（展望）

- 大页技术

连续（虚拟、物理）内存，防止在block layer额外切出更多的IO请求



Thanks