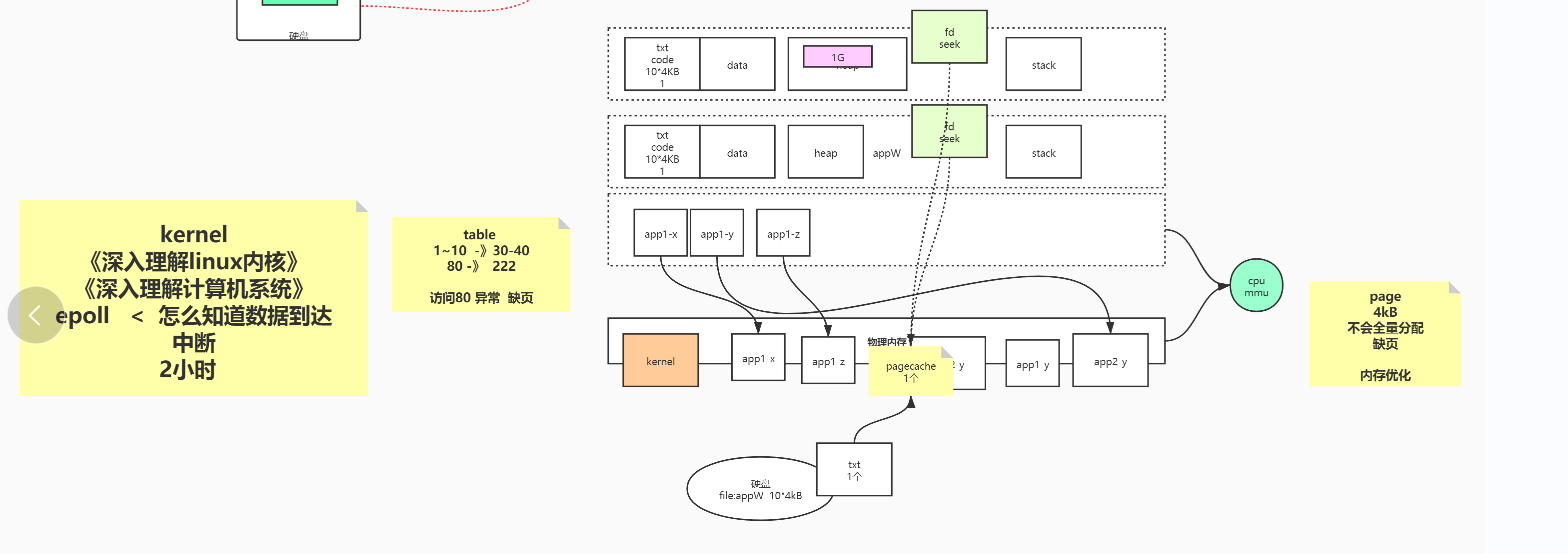
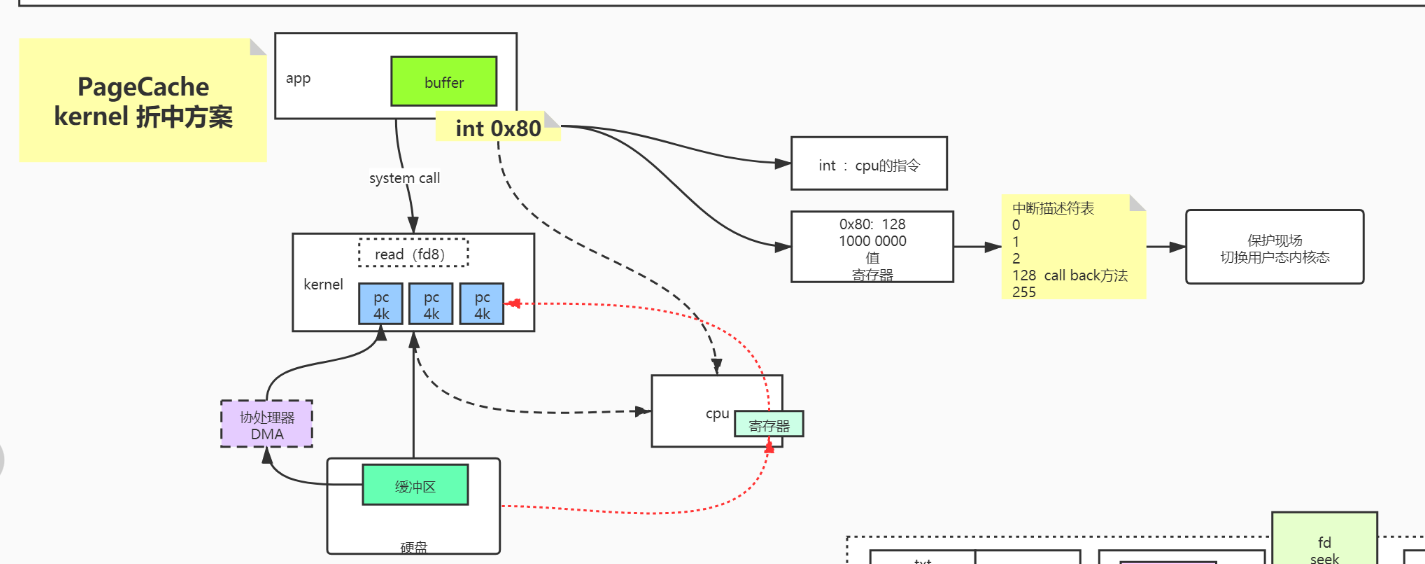
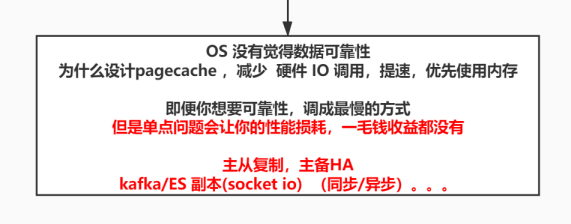
1. 虚拟地址与物理地址



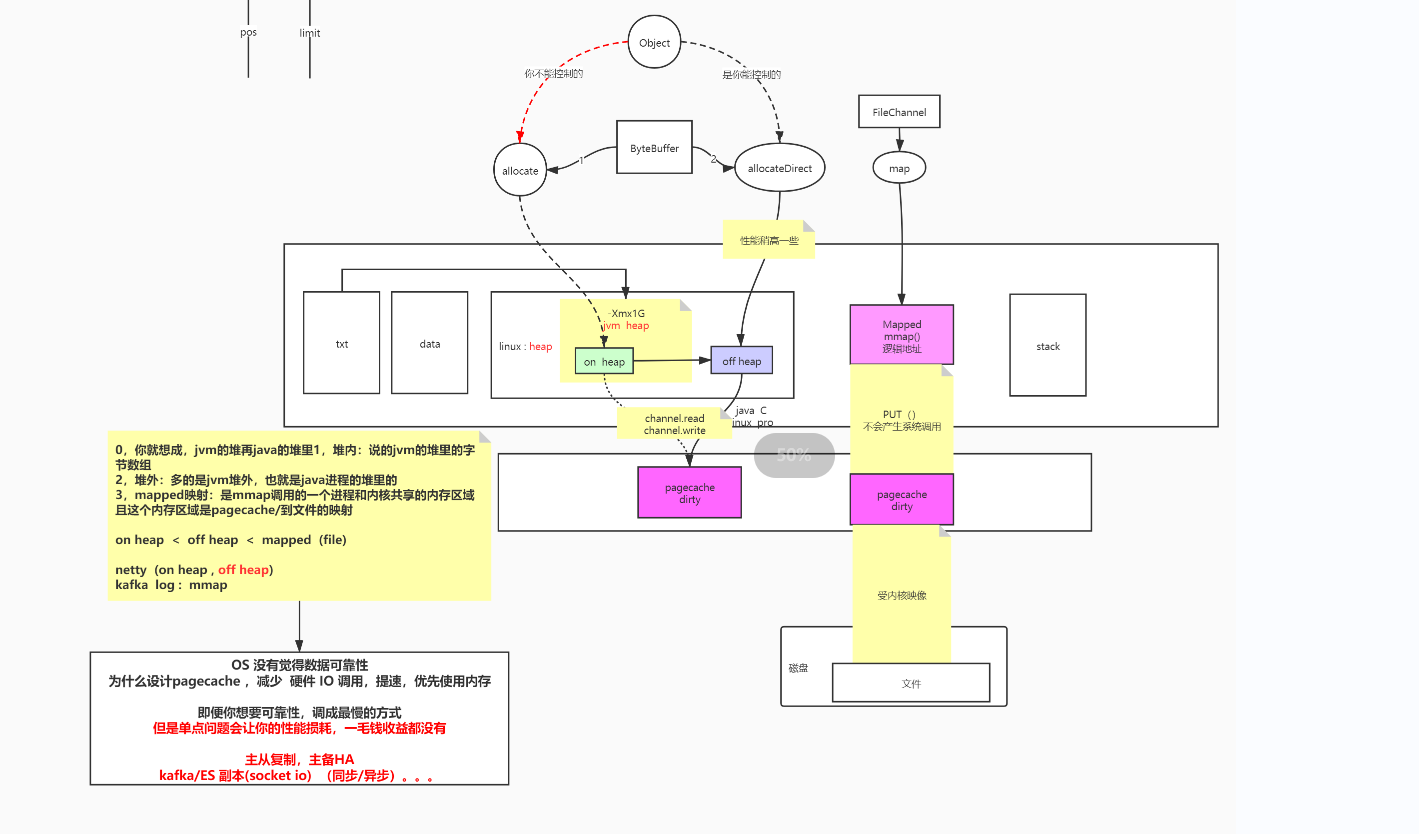
1. 物理内存：内核（有pageCache），程序物理地址（不一定是连续的，分配内存空间是随用随分配，以4K为单位所以不一定都加载到内存）
2. 程序虚拟空间：程序所有数据，虚拟地址（线性地址，由于程序空间不是全量分配的，所以可能访问某个虚拟页的时候可能内存中还未分配，就发生缺页中断）
3. 物理地址与虚拟地址的映射：通过CPU的MMU单元完成映射，以Page 4K为单位；当不同程序访问相同文件的时候，通过映射关系可能指向同一块内存空间。
4. pageCache

pageCache：减少硬件IO的调用，提速，优先使用内存

就会涉及到何时写磁盘，如果每写一次pageCache就flush一次，注重可靠性，那么会使得虚拟损耗，收益少。因此，通过权衡，就有主从复制，主备HA。



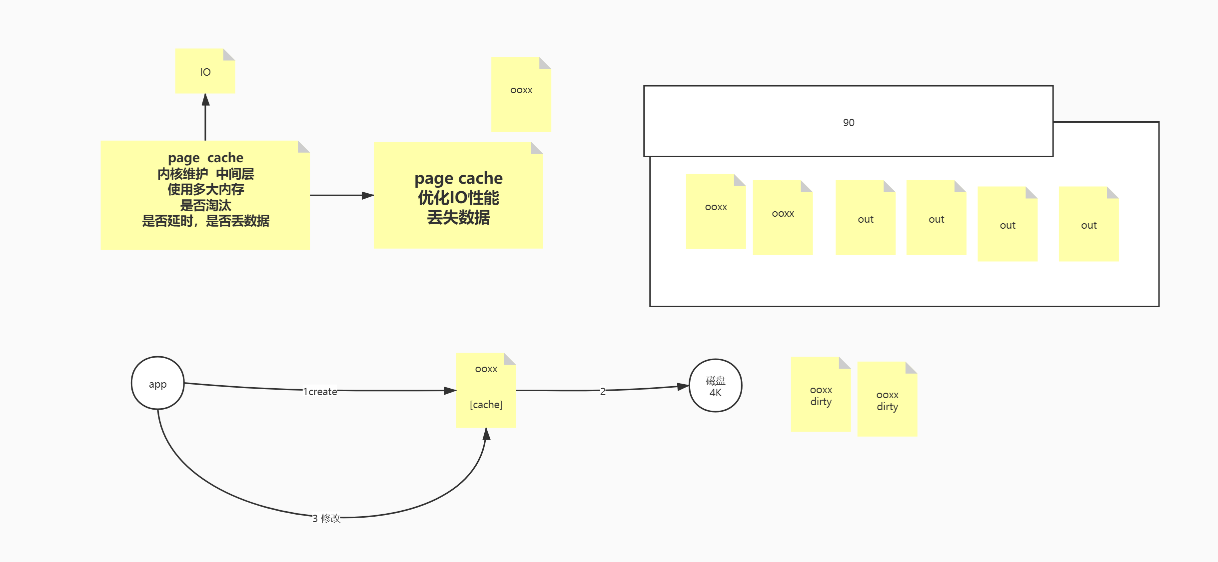
1. 程序写数据流程



1. 没有使用mmap：两次系统调用

当程序调用write时，通过系统调用，先将数据写到内核的pageCahce中，然后将此页标记为脏页。当调用flush或者空间达到设定的阈值时，通过系统调用将脏页写到磁盘中。如果内存不够用了，就通过LRU机制将页从内存中删除。

弊端：掉电时，可能为来得及将数据写回磁盘中，造成数据丢失。



1. 使用mmap：（只有文件有，socket没有）一次系统调用

调用MappedByteBuffer的put方法（不需要系统调用），数据直接到达内核的pageCache；最终需要通过系统调用，将pageCache的内容写到磁盘中。

1. 随机读写与非随机读写

随机读写：可以通过seek改变指针偏移，写到指定位置

非随机读写：只能往后追加写

1. MMAP

没有mmap：需要将用户空间的数据，通过Buffer拷贝到内核空间，即用户空间和内核空间不是共用的。

有mmap：用户空间的逻辑地址是直接映射到内核空间pageCache的，它们之间传递数据不需要再次拷贝，比查都可以访问。