缓存一致性

CPU Cache的数据写入

Cache由多个Cache Line组成的，
CPU Line是CPU从内存读取到的基本单位
CPU Line是由各种标志（Tag）+数据块（Data Block）

写直达

保持内存和Cache一致性最简单的方式，「将数据同时写入内存和Cache中」，这种方式成为「写直达」

写之前会判断数据是否已经在CPU Cache里

如果已经存在，先将数据更新到Cache，在写入内存里

如果不存在，直接把数据更新到内存里

无论数据在不在Cache里，每次写操作都会写回到内存，这样会消耗大量时间，性能受损

写回

当发生写操作时，新的数据仅仅被写入「Cache Block」，只有当修改过的Cache Block「被替换」时才需要写到内存中

减少数据写回的频率，提高系统性能

实现

当发生写操作，数据已经在CPU Cache，则把数据更新到CPU Cache，同时标记Cache中的Block为脏页（Dirty）

当发生写操作，数据所对应的Cache Block存放的是「别的内存地址数据」，此时就要检查是这个Block是否是脏页，如果是脏的话就需要写回内存，然后把当前写入的数据，写入到Cache Block 同时标记为脏页，如果没有标记为脏页直接将当前数据写入，标记为脏页

缓存一致性问题

由于CPU中的L1/L2Cache 是多个核心独有的，
那么多核心存在「缓存一致性问题」

写传播（Wreite Propagation）

写传播还是可能会导致其他核心收到的传播数据不同而导致缓存一致性问题

事务的串行化（Transaction Serialization）

CPU核心对于Cache中数据操作，同步给其他CPU核心

如果两个CPU核心有相同的Cache，那么对于Cache 的更新需要通过加锁完成

总线嗅探（Bus Snooping）

写传播原则就是当某个CPU核心更新Cache广播，最常见的时间方式就是「总线嗅探」

CPU需要时刻监听总线上的一切活动，不管其他核心的Cache是否缓存相同的数据，都要发出一个广播时间，会加重总线负载

总线嗅探保证了某个CPU更新数据事件能被其他CPU核心知道，但不能保证事务串行化

基于总线嗅探的MESI实现了事务串行化

MESI协议

Modified-已修改

代表Cache Block上的数据已经被修改过，但是还没有写到内存里。

Exclusive-独占

数据只存储在一个CPU核心的Cache里，这时候如果向独占的Cache写入数据，就可以自由写入，不需要通知其他核心

在独占状态下如果有其他核心从内存读取了相同数据到各自的Cache，那么这个时候，独占状态下的数据就会变成共享

Shared-共享

共享状态代表相同的数据在多个CPU核心的Cache里面都有麦当我们要更新Cache数据的时候，不能直接修改，要向其他CPU核心广播一个请求，使其他核心的Cache中对应的CacheLine标记为「无效」，然后再更新当前Cache里面的数据

invalidated-已失效

代表ChaCha Block上的数据已经失效，不可读取

当前状态	事件	行为
已失效 I(Invalid)	Local Read	如果其它核心的 Cache 没有这份数据，本地核心的 Cache 从内存中读取数据，Cache Line 状态变成 E； 如果其它核心的 Cache 有这份数据，且状态为 M，则将数据更新到内存，本地核心的 Cache 再从内存中取数据，这 2 个 Cache 的 Cache Line 状态都变成 S； 如果其它核心的 Cache 有这份数据，且状态为 S 或者 E，本地核心的 Cache 从内存中取数据，这些 Cache 的 Cache Line 状态都变成 S；
	Local Write	从内存中读取数据，缓存到 Cache，再在 Cache 中更新数据，状态变成 M； 如果其它核心的 Cache 有这份数据，且状态为 M，则要先将其他核心的 Cache 里的数据写回到内存； 如果其它核心的 Cache 有这份数据，则其它 Cache 的 Cache Line 状态变成 I；
	Remote Read	既然是 Invalid，其他核心的操作与它无关
	Remote Write	既然是 Invalid，其他核心的操作与它无关
独占 E(Exclusive)	Local Read	从 Cache 中取数据，状态不变
	Local Write	修改 Cache 中的数据，状态变成M
	Remote Read	数据和其它核心共用，状态变成了 S
	Remote Write	数据被修改，本地核心的 Cache Line 中的数据不能再使用，状态变成 I
共享 S(Shared)	Local Read	从 Cache 中取数据，状态不变
	Local Write	修改 Cache 中的数据，状态变成 M， 其它核心共享的 Cache Line 状态变成 I
	Remote Read	状态不变
	Remote Write	数据被修改，本地核心的 Cache Line 中的数据不能再使用，状态变成 I
已修改 M(Modified)	Local Read	从 Cache 中取数据，状态不变
	Local Write	修改 Cache 中的数据，状态不变
	Remote Read	Cache Line 的数据被写到内存中，使其它核能使用到最新的数据，状态变成 S
	Remote Write	Cache Line 的数据被写到内存中，使其它核能使用到最新的数据，由于其它核会修改这行数据，状态变成 I