CPU缓存和内存屏障

CPU性能优化手段-缓存

1. 为了提高程序运行性能，现代CPU在很多方面对程序进行了优化。例如CPU高速缓存，尽可能地避免处理器访问主内存的时间开销，处理器大多会利用缓存以提高性能。

2. 多级缓存

   L1 Cache(一级缓存)是CPU第一层高速缓存，分为数据缓存和指令缓存。一般服务器CPU的L1缓存容量通常在32-4096KKB。

   L2 Cache(二级缓存)由于L1级高速缓存容量的限制，为了再次提高CPU的运算速度，在CPU外部放置一高速存储器，即二级缓存。

   L3 现在都是内置的。而它的实际作用即是，L3缓存的应用可以进一步降低内存延迟，同时提升大数据量计算时处理器的性能。具有较大L3缓存的处理器提供更有效的文件系统缓存行为计较短消息和处理器队列长度。一般是多核共享一个L3缓存。

   CPU在读数据时，现在L1中寻找，再从L2寻找，再从L3寻找，然后是内存，再后是外存储器。

缓存同步协议

多CPU读取同样的数据进行缓存，，进行不同的运算之后，最终写入主内存以那个CPU为准。在这种高速缓存回写的场景下，有一个缓存一致性协议，多数CPU厂商对它进行了实现。

MESI协议，它规定每条缓存有个状态位，同时定义了下面四个状态：

修改态(Modified) 此cache行已经被修改过，内容已不同于主存，为此cache专有；

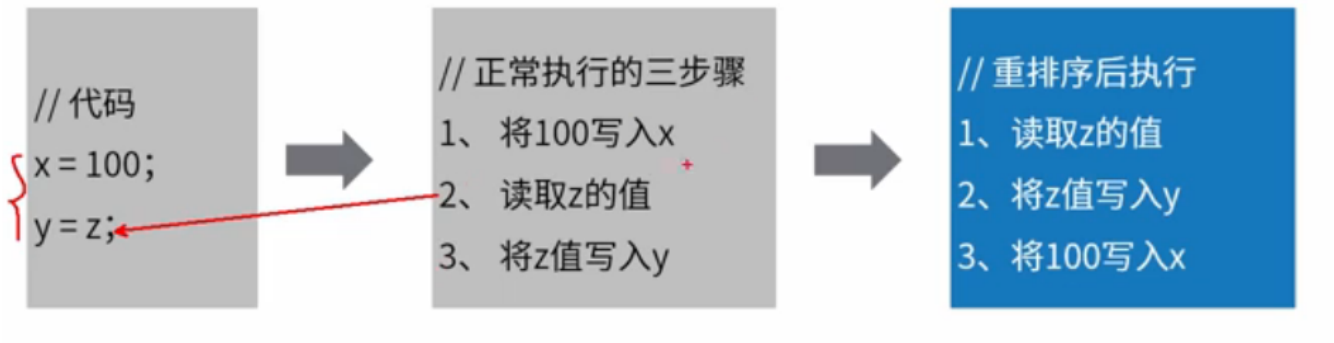
专有态(Exclusive) 此cache行内容同于主存，但不出现在其它cache中；

共享态(Shared) 此cache行内容同于主存，但也出现在其它cache中；

无效态(Invalid) 此cache行内容无效(空行)；

多处理器时，单个CPU对缓存中数据进行了改动，需要通知给其他CPU。也就意味着，CPU处理要控制自己的读写操作，还要监听其他CPU发出的通知，从而保证最终一致。

CPU性能优化手段-运行时指令重排序



指令重排的场景：当CPU写缓存时发现缓存区域块正在被其他CPU占用，为了提高CPU处理性能，可能将后面的读缓存命令优先执行。

注意并非随意重排，需要遵守as-if-serial语义。

as-if-serial语义的意思指：不管怎么重排序，单线程程序的执行结果不能被改变。编译器，runtime和处理器都必须遵守as-if-serial语义。也就是说：编译器和处理器不会对存在数据依赖关系的操作做重排序。

高速缓存和指令重排的两个问题

1. CPU高速缓存下有一个问题

   缓存中的数据与主内存的数据并不是实时同步的，各CPU间缓存的数据也不是实时同步的。在同一个时间点，各CPU所看到同一内存地址的数据的值可能是不一致的。

2. CPU指令重排序优化下有一个问题

   虽然遵守了as-if-serial语义，单仅在单CPU自己执行的情况下保证结果正确。多核线程中，指令逻辑无法分辨因果关联，可能出现乱序执行，导致程序运行结果错误。

内存屏障

处理器提供了两个内存屏障用于解决上面两个问题。

写内存屏障(Store Memory Barrier)：在指令后插入Store Barrier，能让写入缓存中的最新数据更新写入主内存，让其他线程可见。强制写入主内存，这种显示调用，CPU就不会因为性能问题考虑而对指令重排。

读内存平障(Load Memory Barrier)：在指令之前插入Load Barrier，可以让高速缓存中的数据实效，强制重新从主内存加载数据。强制读取主内存内容，让CPU缓存与主内存保持一致，避免缓存导致的一致性问题。