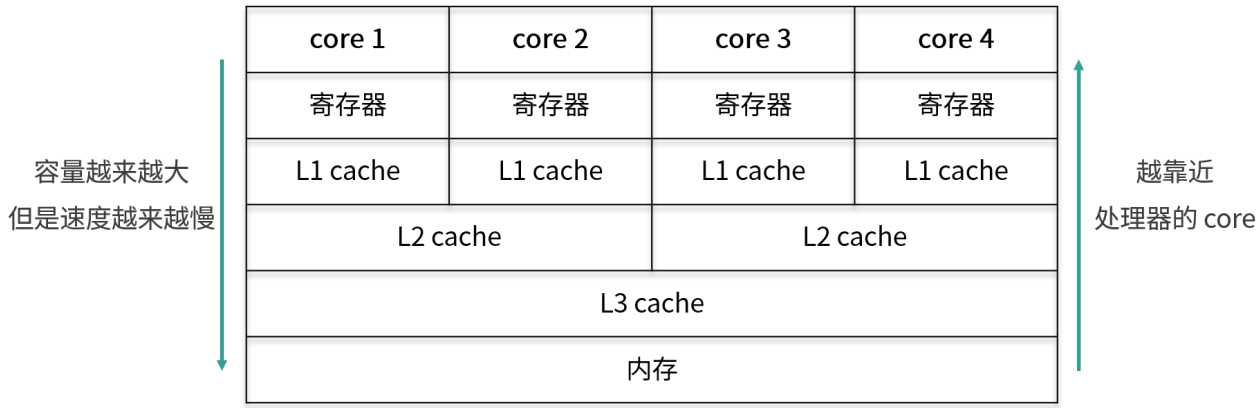
60 主内存和工作内存的关系?

本课时我们主要讲解主内存和工作内存的关系。

CPU 有多级缓存，导致读的数据过期

由于 CPU 的处理速度很快，相比之下，内存的速度就显得很慢，所以为了提高 CPU 的整体运行效率，减少空闲时间，在 CPU 和内存之间会有 cache 层，也就是缓存层的存在。虽然缓存的容量比内存小，但是缓存的速度却比内存的速度要快得多，其中 L1 缓存的速度仅次于寄存器的速度。结构示意图如下所示：



在图中，从下往上分别是内存，L3 缓存、L2 缓存、L1 缓存，寄存器，然后最上层是 CPU 的 4个核心。从内存，到 L3 缓存，再到 L2 和 L1 缓存，它们距离 CPU 的核心越来越近了，越靠近核心，其容量就越小，但是速度也越快。正是由于缓存层的存在，才让我们的 CPU 能发挥出更好的性能。

其实，线程间对于共享变量的可见性问题，并不是直接由多核引起的，而是由我们刚才讲到的这些 L3 缓存、L2 缓存、L1 缓存，也就是**多级缓存**引起的：每个核心在获取数据时，都会将数据从内存一层层往上读取，同样，后续对于数据的修改也是先写入到自己的 L1 缓存中，然后等待时机再逐层往下同步，直到最终刷回内存。

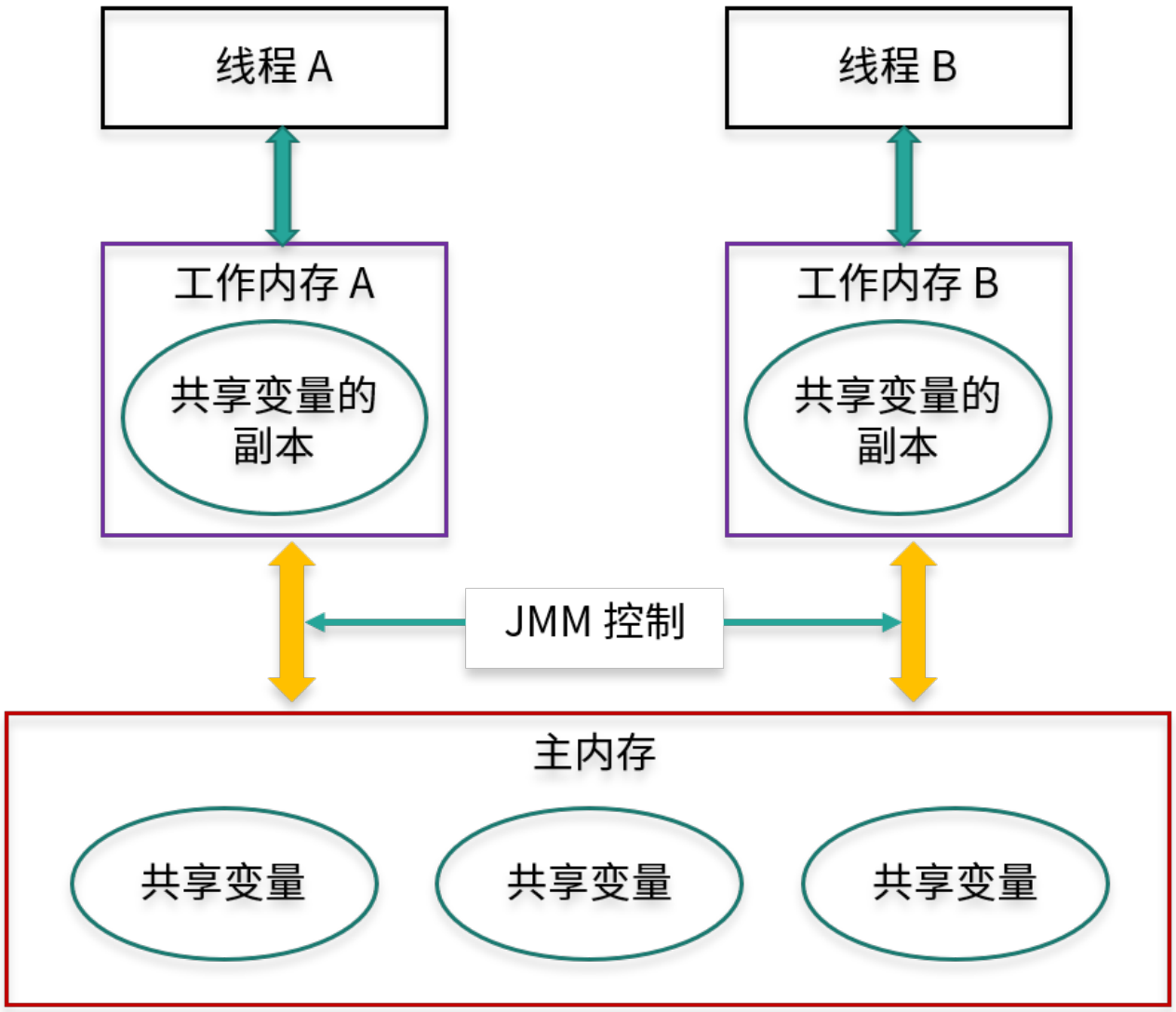
假设 core 1 修改了变量 a 的值，并写入到了 core 1 的 L1 缓存里，但是还没来得及继续往下同步，由于 core 1 有它自己的的 L1 缓存，core 4 是无法直接读取 core 1 的 L1 缓存的值的，那么此时对于 core 4 而言，变量 a 的值就不是 core 1 修改后的最新的值，core 4 读

取到的值可能是一个**过期**的值，从而引起多线程时可见性问题的发生。

JMM的抽象：主内存和工作内存

什么是主内存和工作内存

Java 作为高级语言，屏蔽了 L1 缓存、L2 缓存、L3 缓存，也就是多层缓存的这些底层细节，用 JMM 定义了一套读写数据的规范。我们不再需要关心 L1 缓存、L2 缓存、L3 缓存等多层缓存的问题，我们只需要关心 JMM 抽象出来的主内存和工作内存的概念。为了更方便你去理解，可参考下图：



每个线程只能够直接接触到工作内存，无法直接操作主内存，而工作内存中所保存的正是主内存的共享变量的副本，主内存和工作内存之间的通信是由 JMM 控制的。

主内存和工作内存的关系

JMM 有以下规定：

（1）所有的变量都存储在主内存中，同时每个线程拥有自己独立的工作内存，而工作内存中的变量的内容是主内存中该变量的拷贝；

（2）线程不能直接读 / 写主内存中的变量，但可以操作自己工作内存中的变量，然后再同步到主内存中，这样，其他线程就可以看到本次修改；

（3） 主内存是由多个线程所共享的，但线程间不共享各自的工作内存，如果线程间需要通信，则必须借助主内存中转来完成。

听到这里，你对上图的理解可能会更深刻一些，从图中可以看出，每个工作内存中的变量都是对主内存变量的一个拷贝，相当于是一个副本。而且图中没有一条线是可以直接连接各个工作内存的，因为工作内存之间的通信，都需要通过主内存来中转。

正是由于所有的共享变量都存在于主内存中，每个线程有自己的工作内存，其中存储的是变量的副本，所以这个副本就有可能是过期的，我们来举个例子：如果一个变量 x 被线程 A 修改了，只要还没同步到主内存中，线程 B 就看不到，所以此时线程 B 读取到的 x 值就是一个过期的值，这就导致了可见性问题。

以上就是本课时的内容了，本课时主要介绍了 CPU 的多层缓存结构，以及由此抽象出来的 JMM 主内存和工作内存的结构图，并且还介绍了主内存和工作内存之间的关系。听完本课时，你会更加深刻的理解为什么会发生可见性问题。

上一页 下一页