# Shared Memory Consistency Models: A Tutorial

邹旭 201828013229112

2018 年 10 月 20 日

越来越多并行系统支持共享内存，为了保证读和写操作的正确性和有效性需要用到内存一致性模型。最直观的内存一致性模型是顺序一致性模型，但是顺序一致性模型限制了单处理器和编译器的很多优化。为了缓解这个问题，当前许多多处理器采用松散一致性模型。但是不同的松散一致性模型在一些细节但是很重要的方面存在差异。

在程序员和系统之间的每一级都需要有内存一致性模型规范。

**单处理器上的内存语义**

单处理器上，对内存的操作都是一个接一个发生的，因此对一个内存的读操作总是能读出最近的一次写操作写入的值。

**理解顺序一致性**

说一个多处理器系统是顺序一致的，如果程序任何一种执行方式的结果都和所有处理器的操作按某种顺序执行的结果一样，同时对每个处理器来说执行顺序都符合程序序。顺序一致性包含两个层面：（1）在每个处理器上保持程序序；（2）所有处理器上的操作保持一个简单的顺序次序。

**实现顺序一致性**

首先考虑没有 Cache 的情况，在写操作中使用旁路以及缓冲，重叠的写操作以及没有阻塞的读操作都会造成顺序的错乱，违背顺序一致性。

在有 Cache 的情况下，共享数据的复制会引入三个问题：多个副本的存在需要 Cache 一致性协议来将一个最新的对某个 Cache 位置的写的值传播给所有 Cache；其次，检测一个写操作的完成需要涉及更多的事物；将修改传播给多个副本是固有非原子性的操作，这使得向其他操作伪造原子性的写操作机具挑战性。

顺序一致性的程序顺序层面的交互关系在编译器上和在硬件上是一样的，编译器对一个内存并行程序的操作不能像单处理器上一样进行优化操作。

**松散内存模型**

考虑松散一致性模型的时候我们一般考虑如下两个方面：（1）它们是如何宽松对程序顺序的要求的，（2）他们是如何宽松对写操作原子性的要求的。关于程序顺序的松散化，我们一般分为读写、写写、写读、读读四种情况来分析；关于写原子性要求，我们通过以下方式来区分不同的模型：一个模型是否允许一个读操作在写操作的更新或者无效信息广播到其他 Cache 之前返回读取的值。

**取代松散内存模型的概念**

松散一致性模型引出了系统中心规范。该规范的特点是将程序员直接暴露在一个模型允许的重排序和原子

优化之下，这使得程序员需要考虑程序的行为在这些优化下是否正确。因此，我们需要一种更高的抽象概念来为程序员提供简洁的系统，同时允许系统设计者应用同样的优化。程序员中心规范要求程序员提供关于程序的确切信息，这些信息用于系统来判断某个优化操作是否会影响到程序的正确性。