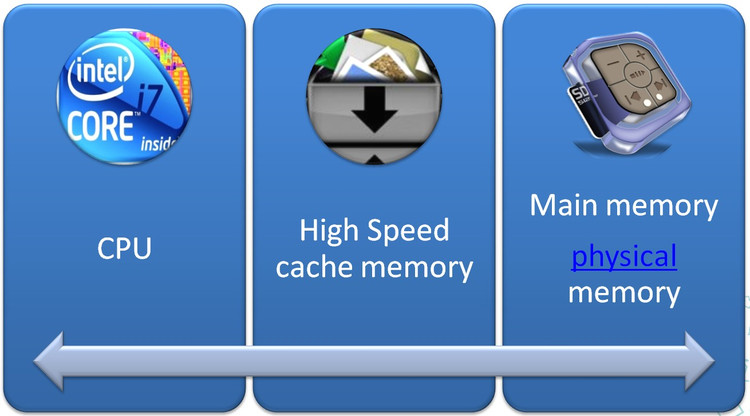
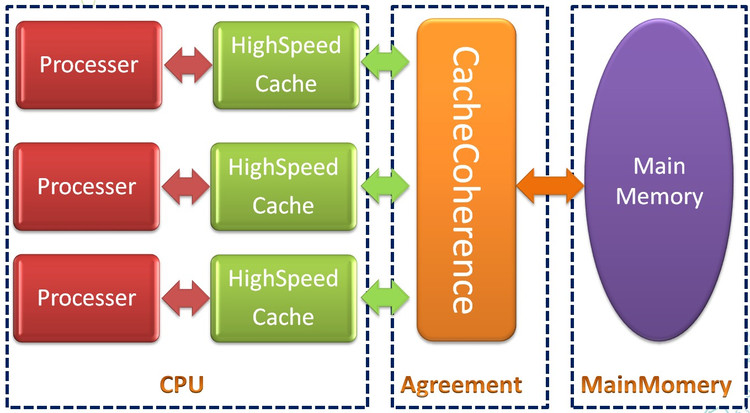
1、首先简单说一下CPU与内存之间的关系



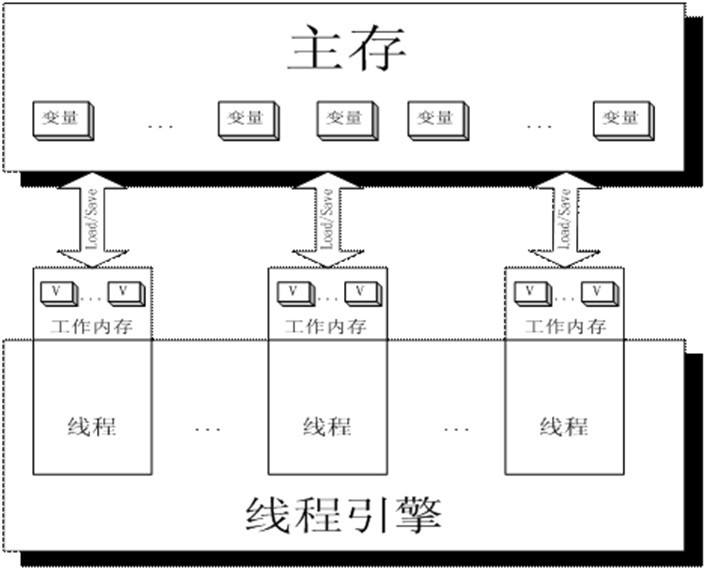
CPU运转速度快，磁盘的读写速度远远不及CPU运转速度，所以设计了内存来缓冲CPU等待磁盘读写；随着CPU的发展，内存读写也远远跟不上CPU的读写速度，CPU生产商就在每颗CPU上加了高速缓存来缓解这种症状，便出现了上图结构。高速缓存的出现很好的解决了CPU与内存之间的矛盾。

多处理器的出现引入缓存不一致的新问题，也就是多个CPU同时处理一块内存区域的时候就可能发生缓存不一致现象。为了解决这一问题，那就要求处理器运行的时候遵循某些协议来保证数据一致性。如下图



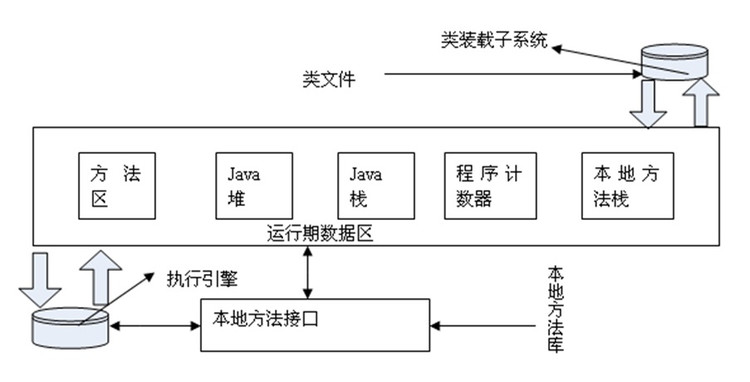
协议就是规定CPU对主存数据操的作状态如何记录,便于CPU正确读取内存数据

2、JVM内存模型与物理机处理操作基本一致



java 通过多线程机制使得多个任务同时执行处理，所有线程共享JVM内存区域的main memory（主存），每个线程又有自己的独立工作内存，线程与内存交互时，数据先从主内存拷贝到线程工作内存，然后再交给线程处理。

2、JVM逻辑内存模型，即JVM如何划分内存，并处理数据的



（1）、程序计数器

一小块内存，可以看作是当前线程所执行字节码的行号指示器，字节码解释器工作时就是通过改变这个计数器的值来选取下一条要执行的字节码指令，分支、跳转、循环、异常处理、线程恢复等基础功能都需要依赖这个计数器。

JVM多线程是通过轮流切换并分配处理器执行时间方式来实现，一个处理器(单核)某时刻只能执行一条线程指令，在多线程切换后能恢复到正确的执行位置，每个线程就需要有独立的程序计数器，线程之间互不影响。

（2）、java虚拟机栈（java栈）

java栈和程序计数器一样，是线程私有的，生命周期与线程相同。虚拟机栈描述的是java方法执行的存储模型，每个方法执行的时候都会创建一个栈帧，存储局部变量、操作栈、动态链接、方法出口等信息，方法执行至结束的过程，也就是栈帧从入栈到出栈的过程。

局部变量内存空间在编译期间就分配好了，如果运行期间内存不够可以动态扩展，但是如果请求内存大于虚拟机规定内存大小抛出异常StackOverflowError，如果动态扩展，请求内存不够用则抛异常OutOfMemoryError。

（3）、本地方法栈

本地方法栈与虚拟机栈发挥作用一样，区别就是本地方法栈是为调用Native方法（非java方法，如操作系统内方法或其他程序C、C++等方法）服务，虚拟机栈是为调用java方法服务

（4）、java堆

对于大多数应用来说，java堆是JVM管理内存中最大的一块，在启动的时候就创建好了（主流虚拟机都可以通过-Xmx和Xms动态扩展），而且为所有线程共享。java堆唯一目的就是存放对象实例，虽然虚拟机规范中描述，所有的对象实例及数组都要在堆上分配，但随着技术发展，栈上分配、标量替换技术将会导致微妙变化。GC堆（垃圾回收管理器）的主要区域也在堆上。java堆中数据存储物理上是可以不连续的，而逻辑上连续。

（5）、方法区

方法区与java堆一样，为所有线程共享，不需要连续内存管理数据，主要用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等。为了与java堆区别开，别名叫NonHeap（非堆）

（6）、运行时常量

属于方法区一部分，Class文件除了有类的版本、字段、方法、接口等描述信息外，还有一项是信息常量池，用于存放编译期生成的各种字面量和符号引用，这部分内容将在类加载后存放到方法区的运行时常量池中。