## 一、基础概念

并发：

同时拥有两个或多个线程，如果程序在单核处理器上运行，多个线程将交替地换入或者换出内存，这些线程是同时“存在”的，每个线程都处于执行过程中的某个状态，如果运行在多核处理器上，此时，程序中的每个线程都将分配到一个处理器核上，因此可以运行

高并发：（High Concurrency）

是互联网分布式系统架构设计中必须考虑的因素之一，它通常是指，通过设计保证系统能够同时并行处理很多请求

对比：

并发：多个线程操作相同的资源，保证线程安全，合理使用资源

高并发：服务能同时处理很多请求，提高程序性能（如：双十一，12306抢票）

Java内存模型（Java Memory Model，JMM）

Java内存模型规范：规定了一个线程如何和何时可以看到其他线程修改过后的值，以及在必须的时如何同步的访问共享变量

同步的八种操作：

Lock（锁定）：作用于主内存的变量，把一个变量标识为一条线程独占状态

Unlock（解锁）：作用于主内存的变量，把一个处于锁定状态的变量释放出来，释放后的变量才可以被其他线程锁定

Read（读取）：作用于主内存的变量，把一个变量值从主内存传输到线程的工作内存中，以便随后的load动作使用

Load（载入）：作用于工作内存的变量，它把read操作从主内存中得到的变量值放入工作内存的变量副本中

Use（使用）：作用于工作内存的变量，把工作内存中的一个变量值传递给执行引擎

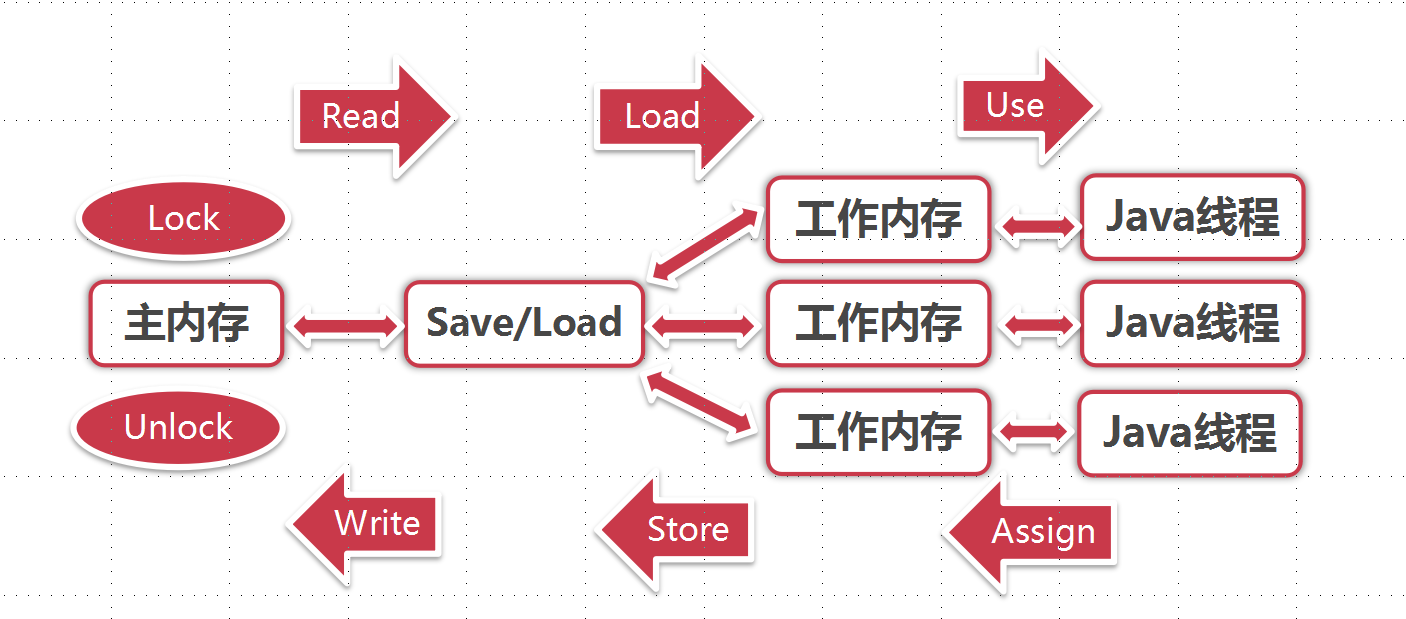
Assign（赋值）：作用于工作内存的变量，它把一个从执行引擎接收到的值赋值给工作内存的变量

Store：（存储）：作用于工作内存的变量，把工作内存中的一个变量的值传送到主内存中，以便随后的write的操作

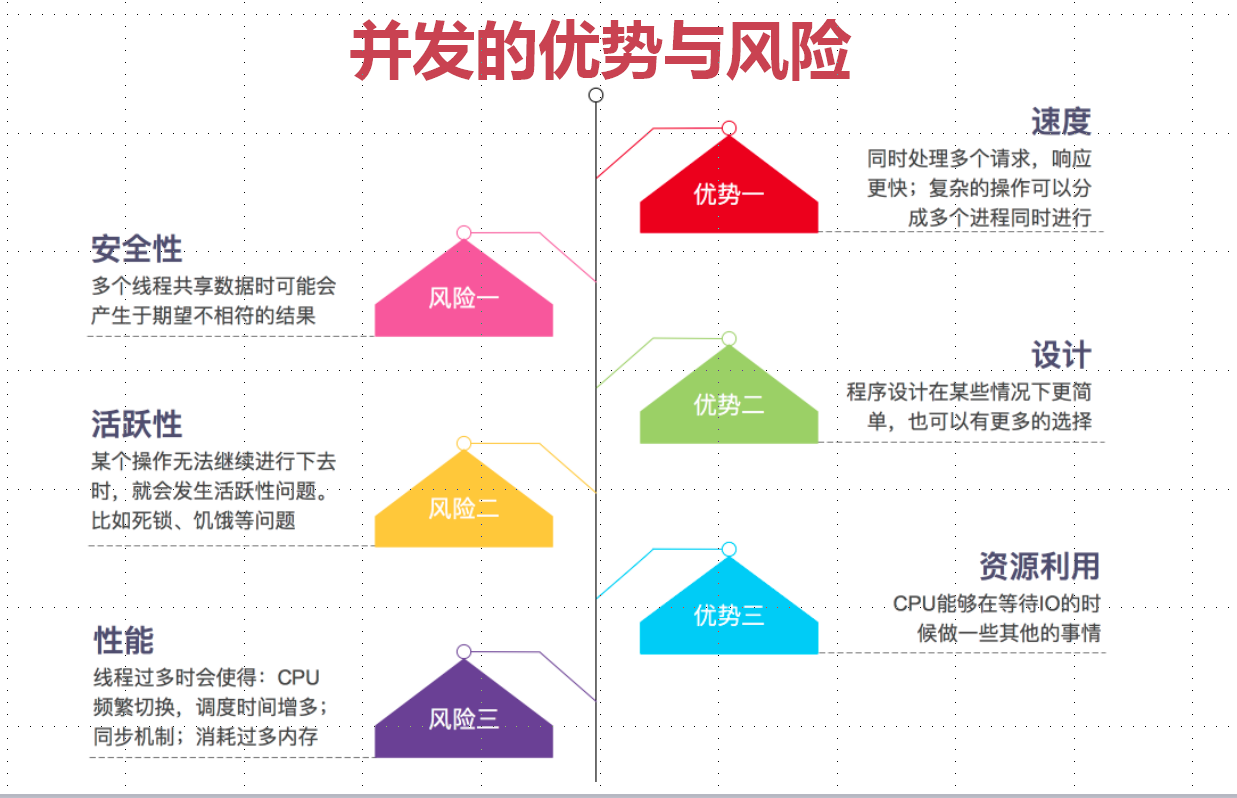
Write（写入）：作用于主内存的变量，它把store操作从工作内存中一个变量的值传递到主内存的变量中

同步规则：

1. ：如果把一个变量从主内存中复制到工作内存，就需要按顺序的执行read和load操作，如果把变量从工作内存中同步回主内存中，就要按顺序地执行store和write操作，但Java内存模型只要求上述操作必须按顺序执行，而没有保证必须是连续执行
2. ：不允许read和load、store和write操作之一单独出现
3. ：不允许一个线程丢弃它的最近assign的操作，即变量在工作内存总改变了之后必须同步到主内存中
4. ：不允许一个线程无原因的（没有发生任何assign操作）把数据从工作内存中同步回主内存中
5. ：一个新的变量只能从主内存中诞生，不允许在工作内存中直接使用一个未被初始化（load或assign）的变量。即就是对一个变量事实use和store操作之前，必须先执行过了assign和load操作
6. ：一个变量在同一时刻只允许一条线程对其进行lock操作，但lock 操作可以同时被同一条线程重复执行多次，多次执行lock后，只有执行相同次数的unlock操作，变量才会被解锁。Lock和unlock必须成对出现
7. ：如果对一个变量执行lock操作，将会清空工作内存中此变量的值，在执行引擎使用这个变量前需要重新执行load或assign操作初始化变量的值



多线程的优势与劣势



## 二、并发编程与线程安全

线程安全性：

定义：当多个线程访问某个类时，不管运行时环境采用何种调度方式或者这些进程将如何交替执行，并且在主调代码中不需要任何额外的同步活协同，这个类都能表现出正确的行为，那么就称这个类是线程安全的

原子性：提供了互斥访问，同一时刻只能有一个进程对它进行操作

可见性：一个线程对主内存的修改可以及时的被其他线程观察到

有序性：一个线程观察其他线程中的指令执行顺序，由于指令重排序的存在，

该观察结果一般杂乱无序