安全点：

作用：解决在枚举根节点时引用关系变化的问题

特征：是否具有让程序长时间执行的特征

条件：指令序列复用，如方法调用、循环跳转、异常跳转

**什么是safepoint**

safepoint**可以用在不同地方**，比如GC、Deoptimization，在Hotspot VM中，GC safepoint比较常见，需要一个数据结构记录每个线程的调用栈、寄存器等一些重要的数据区域里什么地方包含了GC管理的指针。

从线程角度看，safepoint可以理解成是在**代码执行过程中的一些特殊位置**，当线程执行到这些位置的时候，说明虚拟机当前的状态是安全的，如果有需要，可以在这个位置暂停，**比如发生GC时，需要暂停所有活动线程，但是该线程在这个时刻，还没有执行到一个安全点，所以该线程应该继续执行，到达下一个安全点的时候暂停，然后才开始GC，该线程等待GC结束**。

又比如从偏向锁，升级为轻量级锁的时候，可以让持有锁的线程来到这儿暂停，然后把锁变换后再重新执行。

**什么地方可以放safepoint**

下面以Hotspot为例，简单的说明一下什么地方会放置safepoint

1、理论上，在解释器的**每条字节码的边界都可以放一个safepoint**，不过挂在safepoint的调试符号信息要占用内存空间，如果每条机器码后面都加safepoint的话，需要保存大量的运行时数据，所以要尽量少放置safepoint，在safepoint会生成polling代码询问VM是否要“进入safepoint”，polling操作也是有开销的，polling操作会在后续解释。

2、通过JIT编译的代码里，**会在所有方法的返回之前**，以及所有非counted loop的**循环**（无界循环）**回跳之前放置一个safepoint**，为了防止发生GC需要STW时，该线程一直不能暂停。另外，JIT编译器在生成机器码的同时会为每个safepoint生成一些“调试符号信息”，为GC生成的符号信息是OopMap，指出栈上和寄存器里哪里有GC管理的指针。

**线程如何被挂起**

如果触发GC动作，VM thread会在VMThread::loop()方法中调用SafepointSynchronize::begin()方法，最终使所有的线程都进入到safepoint。

**线程有五种不同的状态对应五种挂起的措施**

**1、执行Java code**

**在执行字节码时会检查safepoint状态**，因为在begin方法中会调用Interpreter::notice\_safepoints()方法，通知解释器更新dispatch table

**2、执行native code**

如果VM thread发现一个Java thread正在执行native code，并不会等待该Java thread阻塞，不过当该Java thread从native code返回时，必须检查safepoint状态，看是否需要进行阻塞。

**3、执行complied code**

如果想进入safepoint，则设置polling page不可读，当Java thread发现该内存页不可读时，最终会被阻塞挂起。在SafepointSynchronize::begin()方法中，通过os::make\_polling\_page\_unreadable()方法设置polling page为不可读。

**4、线程处于Block状态**

即使线程已经满足了block condition，也要等到safepoint operation完成，如GC操作，才能返回。

**5、线程正在转换状态**

会去检查safepoint状态，如果需要阻塞，就把自己挂起。

**安全点**  
   
 

**1.安全点的选取**

在OppMaps的帮助下，虚拟机能够迅速的完成GCRoots的枚举，但是如果每一条指令都生成对应的OppMaps，那就需要大量的额外空间。

 所以，**程序在执行的时候并非在所有地方都能停顿下来gc，只有到达安全点才能停顿。**安全点的选定是以“是否具有让程序长时间执行的特性”为标准，因为安全点过少的话gc停顿时间就会很长，安全点过多又会增加运行时负荷。”长时间执行“最明显的特征就是指令序列复用，如方法调用，循环跳转，异常跳转等。

所有这些功能的指令前面才会产生安全点。   
**2.线程的停顿**  
   
 在gc发生时让所有线程跑到最近的安全点后停顿。  
 两种思路：  
**第一种,抢先式中断,gc发生时，让所有线程中断，如果有线程不在安全点，那么让线程跑到安全点。**  
**第二种,主动式中断，设置一个标识，各个线程执行时不断轮询这个标志，发现标志时就自动挂起，轮询标志的地方和安全点重合。**  
  
 **安全区域**  
  
  
**安全点机制保证了程序执行的时候，在不太长的时间就会遇到可进入gc的安全点。但是如果线程处于运行状态或者blocked状态的时候，这时线程无法响应jvm的中断请求，就需要安全区域。**  
  
安全区域是指在一段代码片段中，引用关系不会发生变化，**在该区域的任何地方发生gc都是安全的。**  
当代码执行到安全区域时，首先标示自己已经进入了安全区域，那样如果在这段时间里jvm发起gc，就不用管标示自己在安全区域的那些线程了，在线程离开安全区域时，会检查系统是否正在执行gc，如果是那么就等到gc完成后再离开安全区域。