根据1和2的演示结果，JVM 中GC总结如下：

第一种是 Serial 串行垃圾回收器，这是 JVM 早期的垃圾回收器，只有一个线程执行垃圾回收。这种GC的处理效能很慢，在多核CPU的情况下，只能在单核上运作，没有并发优势。

第二种是 Parallel 并行垃圾回收器，它启动多线程执行垃圾回收。如果 JVM 运行在多核 CPU 上，那么显然并行垃圾回收要比串行垃圾回收效率高。在串行和并行垃圾回收过程中，当垃圾回收线程工作的时候，必须要停止用户线程的工作，否则可能会导致对象的引用标记错乱，因此垃圾回收过程也被称为 stop the world，在用户视角看来，所有的程序都不再执行，整个世界都停止了。在可以存着短暂STW的场景中，这个GC效能比较高具备优势。但是如果STW的时间过长，还是会对业务产生影响。如果堆过大导致每次FULLGC的时间过长，那么这种GC方式的劣势就会出现。

第三种 CMS 并发垃圾回收器，在垃圾回收的某些阶段，垃圾回收线程和用户线程可以并发运行，因此对用户线程的影响较小。这种GC针对Parallel并行垃圾回收器，性能没有后者高，但是对业务的影响又比后者小。Web 应用这类对用户响应时间比较敏感的场景，适用 CMS 垃圾回收器。

第四种是 G1 垃圾回收器，它将整个堆空间分成多个子区域，然后在这些子区域上各自独立进行垃圾回收，在回收过程中垃圾回收线程和用户线程也是并发运行。G1 综合了以前几种垃圾回收器的优势，适用于各种场景，是未来主要的垃圾回收器。但是从演示的情况来看，在堆不是太大的场景中，效能不如Parallel并行垃圾回收器。G1的综合能力可能是最强的GC，但是并不是在所有场景中都是最优的选择。

综上所述针对GC的选择还需要根据场景而定和压测结果而定。