JVM里99%的人不知道的那些事儿

-- JVM的10个常见误区

秦金卫(kimmking)

Apache Dubbo/ShardingSphere PMC

讲师介绍

- 极客时间《Java 进阶训练营》讲师
- 资深开源参与者/前阿里巴巴架构师
- 《高可用可伸缩微服务架构》作者
- 阿里云 MVP/腾讯云 TVP/TGO 会员

为什么要了解JVM误区

作为一个Java程序员，我们所有的代码最终都运行在JVM之上，但是实际上我们经常在工作中因为自己对JVM的一些常识不了解，导致掉进陷阱或者误区，最终引起系统运行的故障，甚至是产生宕机的风险。

了解了JVM的常见误区，就能在工作中降低这种风险，从而更深入地理解JVM，在实践中更高效可靠地使用JVM。

JVM十大误区

1 误区一：默认GC策略到底是什么？

（*JDK6-17 默认GC策略有何不同？*）

JDK默认GC

JDK 6、7、8	JDK 9、10、11、12、13、14、15、16、17、18
Parallel GC	G1 GC

GC发展历史:

Serial GC->Parallel GC->CMS GC->G1 GC->ZGC/Shenandoah GC

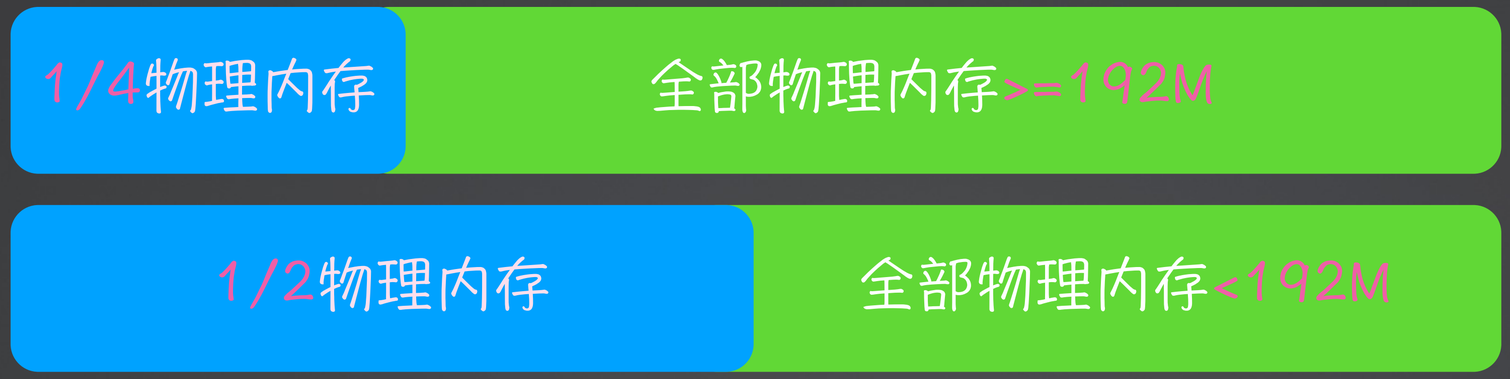
其中8、11、17为LTS版本。

JVM十大误区

2 误区二：默认的最大堆内存大小到底是多少？

（到底是物理内存的多少百分比？）

JVM 默认 最大堆内存



必须是1024的倍数，且不能低于2M。

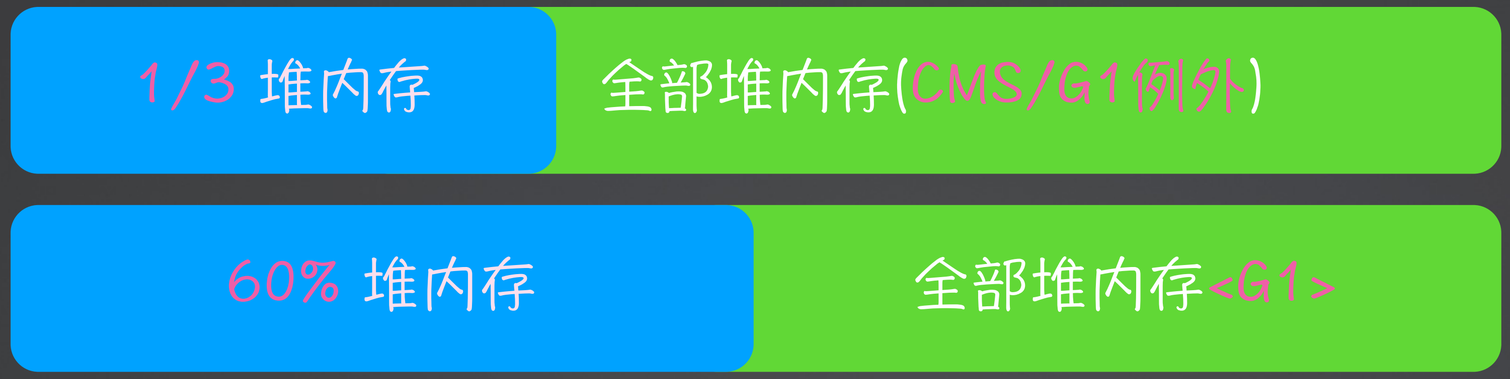
32位机器，最大1G/4G；64位机器，最大可以超过32G/128G。

JVM十大误区

3 误区三：默认的年轻代最大值是多少？

（是不是一定是最大堆内存的1/3？）

JVM 默认 最大年轻代大小



-XX:NewRatio默认是2，所以年轻代占比1(1+2) = 1/3

-Xmn/-XX:MaxNewSize，直接制定大小

G1默认是5%~60%：

-XX:G1NewSizePercent=5 -XX:G1MaxNewSizePercent=60

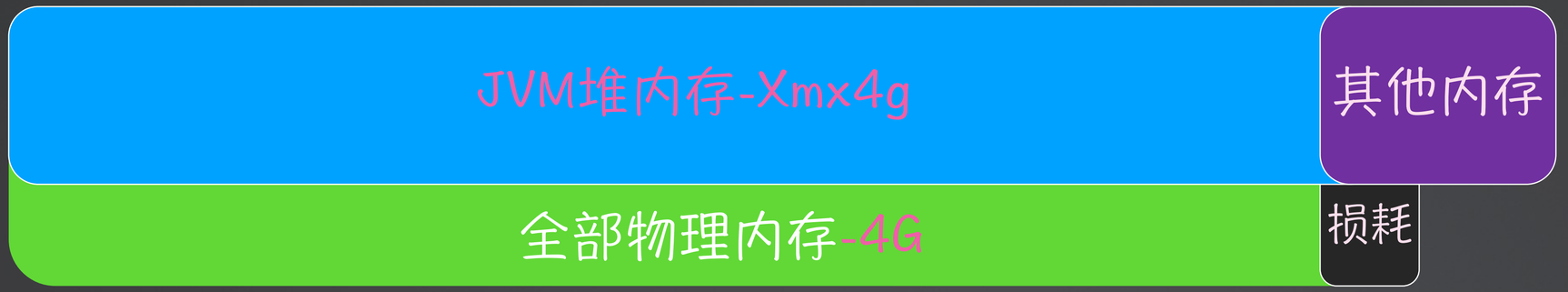
*CMS在内存较小时为：64M * 并发GC线程数(4) * 13 / 10 =332.8M*

JVM十大误区

4 误区四：给JVM分配内存越多越好吗？

（JVM进程到底使用了哪些内存？）

JVM 最大堆内存 配置经验



其他内存 = 虚拟机（少量） + 非堆 + 堆外

系统可用内存 = 全部内存 - 系统损耗（少量）

所以，如果配置-Xmx4g，一定会某个时候OOM

永远建议 -Xmx=最大物理内存[0.6~0.8]*

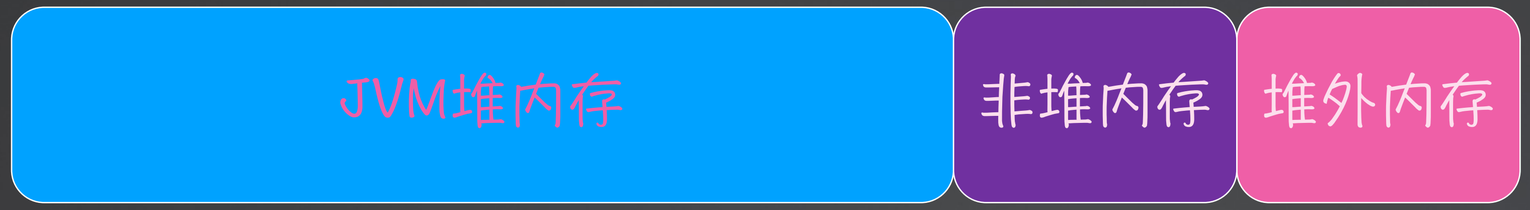
物理内存4G，Xmx为2.4g~3.2g为佳。

JVM十大误区

5 误区五：堆内存和堆外内存的关系？

（设置JVM时堆外内存是否需要考虑？）

堆内存与堆外内存



堆外内存 = 一般指Direct Memory，不受GC控制。

一般来说，JVM/Netty等都可能会使用堆外内存。

默认堆外内存可以跟堆内存一样大！！！

可以通过 -XX:MaxDirectMemorySize 限制。

JVM十大误区

6 误区六：GC过程是不是都产生STW停顿

（GC并发和并行处理是怎么回事？）

GC 停顿与并发执行

所有GC都可能会Stop The World:

- *Serial GC/Parallel GC*
- *CMS GC/G1*
- *ZGC/Shennandoah GC*

JVM十大误区

7 误区七：并发线程和并行线程的默认值如何计算？

（到底是固定值还是跟CPU核心数有关系？）

默认并发线程与并行线程数

GC是CPU密集型操作，所以垃圾回收使用所有CPU时效率最高。

1. ParallelGCThreads 参数的默认值是：

CPU核心数 <= 8，则为 ParallelGCThreads=CPU核心数

CPU核心数 > 8，则为 ParallelGCThreads = CPU核心数 * 5/8 + 3

向下取整

- 16核的情况下，ParallelGCThreads = 13
- 32核的情况下，ParallelGCThreads = 23
- 64核的情况下，ParallelGCThreads = 43

2. ConcGCThreads的默认值则为：

ConcGCThreads = (ParallelGCThreads + 3)/4 向下去整

- ParallelGCThreads = 1~4时，ConcGCThreads = 1
- ParallelGCThreads = 5~8时，ConcGCThreads = 2
- ParallelGCThreads = 13~16时，ConcGCThreads = 4

JVM十大误区

8 误区八：是不是GC停顿越短系统性能就越好？

（GC时间/停顿与延迟/吞吐量的关系如何？）

是不是 GC 延迟越短越好？

吞吐量最大？

延迟最低？

GC 是 吞吐量 和延迟 之间平衡的艺术。

一般情况下选择：

业务是关注吞吐还是延迟？堆内存是大还是小？

当然，相同的GC比如G1，在更高版本JDK表现更好。

JVM十大误区

9 误区九：G1会不会发生长时间停顿的Full GC?

（G1是否就一定比CMS/ParallelGC好？）

G1 GC 的种类

Young GC

Mixed GC

Full GC

① Young GC：Young区满了。

② Mixed GC：同时回收Young和Old。

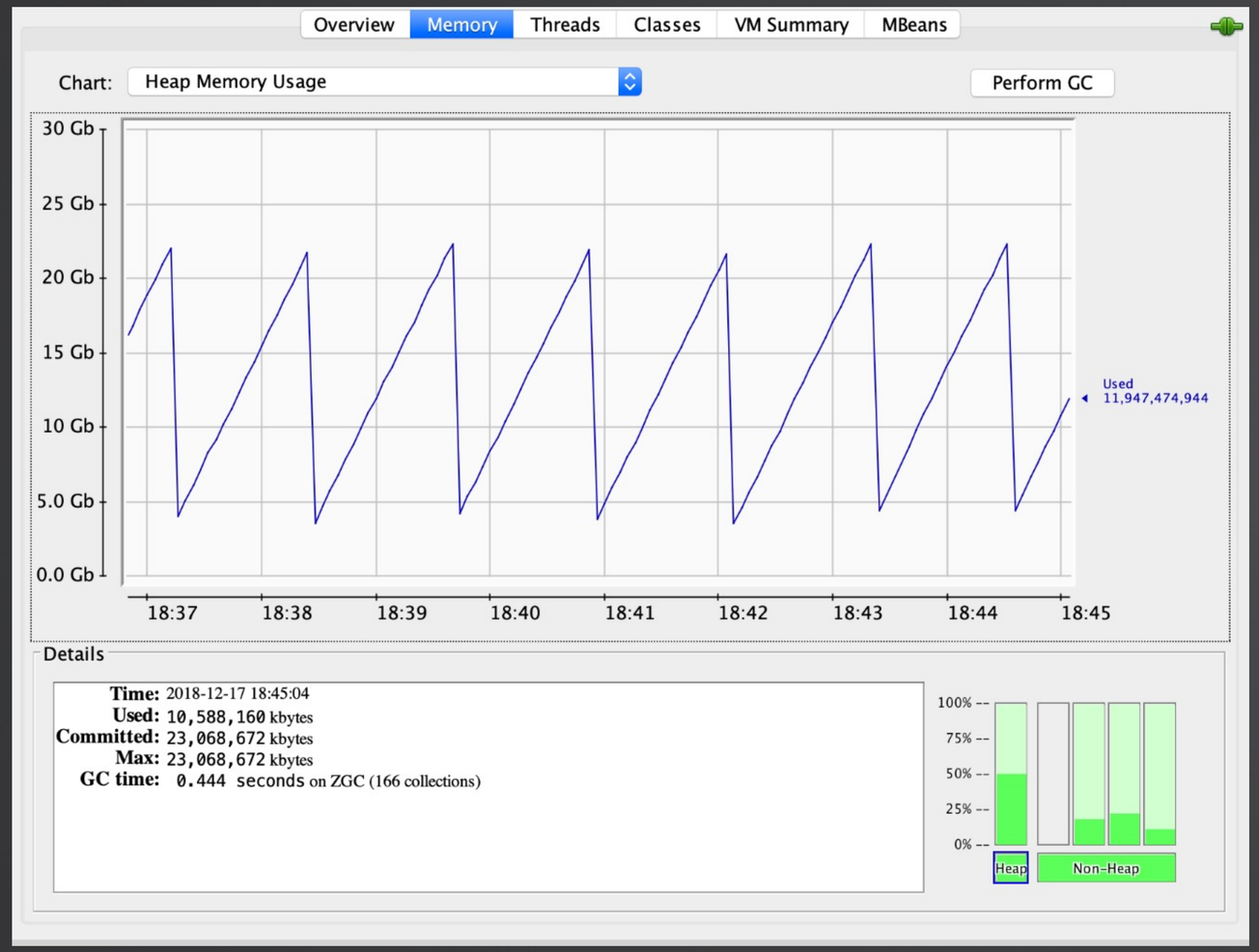
③ Full GC：极端条件下退化成串行处理（JDK10后改成并行）。

JVM十大误区

10 误区十：ZGC到底比G1/CMS/ParallelGC好在哪儿？

（到底该如何选择合适的GC策略？）

ZGC 到底好在哪里？



非常低的延迟，适用于内存较大，延迟敏感的系统。

示例：32 G堆平均GC不到 3 ms

其他：

1. *吞吐量问题*
2. *返还系统内存*
3. *JDK15全系统*

总结

选择正确的GC算法，唯一可行的方式就是去尝试，并找出不合理的地方，一般性的指导原则：

- 如果考虑吞吐优先，CPU尽量都用于处理业务，用 Parallel GC；
- 如果考虑有限的低延迟，且每次GC时间尽量短，用 CMS GC；
- 如果堆较大，同时希望整体来看平均GC时间可控，使用 G1 GC。

对于内存大小的考量：

- 一般4G以上，算是比较大，用 G1 GC 的性价比较高。
- 一般超过8G，比如16G-64G内存，非常推荐使用 G1 GC。
- 更大内存或者低延迟要求非常苛刻，用 ZGC 。

THANKS