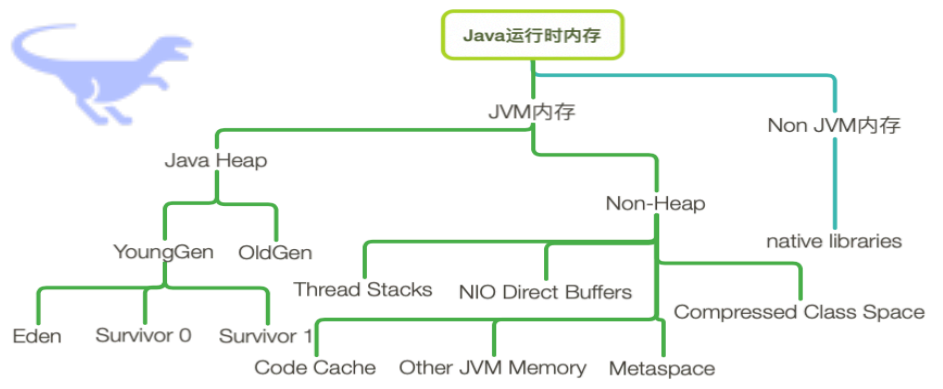
JDK 11 HotSpot JVM 调优秘籍

程序那些事 www.flydean.com

HotSpot JVM可用GC类型		
年轻代回收算法	老年代回收算法	GC启用参数
Serial(DefNew))	Serial Mark Sweep Compact(PSOldGen)	-XX:+UseSerialGC
Parallel scavenge(PSYoungGen)	Serial Mark Sweep Compact(PSOldGen)	-XX:+UseParallelGC
Parallel scavenge(PSYoungGen))	Parallel Mark Sweep Compact(ParOldGen)	-XX:+UseParallelOldGC(随-XX:+UseParallelGC开启)
G1(Garbage First)		-XX:+UseG1GC(默认)



VM通用参数	
VM参数	说明
-XX:ObjectAlignmentInBytes=alignment	Java对象的对齐字节大小
-XX:-UseBiasedLocking	禁止使用偏向锁
-XX:-UseCompressedOops	停止使用对象指针压缩
-verbose:class	loaded class的信息
-verbose:module	使用的JPMS module信息
-verbose:jni	native methods 使用信息
-Xbatch	禁用JVM的后台编译功能
-Xcomp -XX:CompileThreshold	强制JIT编译
-Xint	强制使用解释模式，不使用JIT
-Xmixed	comp和int的混合模式
-Xprof	Profiles运行的程序，只推荐在开发中使用
-XX:AllocateHeapAt=path	允许将java heap空间分配在其他的内存设备中，比如NV-DIMM
-XX:LargePageSizeInBytes=size	设置java heap使用的 large pages大小

Application Class Data Sharing配置	
-XX:+UnlockCommercialFeatures	解锁商业功能
-Xshare:mode	设置CDS的mode(auto,on,off)
-XX:DumpLoadedClassList	dump加载的class list
-XX:SharedClassListFile	需要共享的class list
-XX:SharedArchiveFile	需要创建的shared archive文件

VM通用参数	
VM参数	说明
-Xrs	减少JVM对操作系统信号的捕获，用户自行处理接收到的信号
-XX:-CompactStrings	取消JDK9中的字符串压缩功能
-XX:+UseLargePages -XX:+UseHugeTLBFS	开启largepage

JIT调优	
VM参数	说明
-XX:ReservedCodeCacheSize=size	JIT最大的code cache size
-XX:InitialCodeCacheSize=size	初始化的codeCacheSize
-XX:AllocateInstancePrefetchLines=lines	设置要在实例分配指针之前预取的行数。 默认值为1
-XX:AllocatePrefetchDistance=size	设置对象分配的预取距离的大小（以字节为单位）
-XX:AllocatePrefetchInstr=instruction	设置prefetch指令，值为0-3
-XX:AllocatePrefetchLines=1	使用JIT编译代码生成的预取指令在最后一个对象分配后加载的缓存行数。
-XX:AllocatePrefetchStyle=1	生成预取指令的代码样式：0,没有预取指令生成；1,在每次分配之后执行预取指令； 2,当预取指令执行的时候，使用TLAB分配水印指针
-XX:AllocatePrefetchStepSize=size	预取指令的step size
-XX:+BackgroundCompilation	开启后台编译
-XX:CICompilerCount=threads	设置编译线程的个数
-XX:CompileCommand=command,method[,option]	指定具体方法的编译行为
-XX:CompileCommandFile=filename	从文件指定具体方法的编译行为
-XX:CompileOnly=methods	只有指定的方法才会被编译
-XX:CompileThreshold=invocations	设置在编译之前需要被解释执行的次数
-XX:CompileThresholdScaling=scale	CompileThreshold的比例，小于1表示提前编译，大于1表示延后编译

JIT调优	
VM参数	说明
-XX:+DoEscapeAnalysis	开启逃逸分析
-XX:+Inline	开启inline方法
-XX:InlineSmallCode=size	设置需要inline方法的最大大小
-XX:MaxInlineSize=size	设置最大inline大小
-XX:MaxNodeLimit=nodes	单方法编译的最大节点个数
-XX:NonNMethodCodeHeapSize=size	code segment中非方法code的大小
-XX:NonProfiledCodeHeapSize=size	code segment中nonprofiled methods的大小
-XX:ProfiledCodeHeapSize=size	code segment中profiled methods的大小
-XX:MaxTrivialSize=size	普通方法内联的最大大小
-XX:+OptimizeStringConcat	String连接优化
-XX:+SegmentedCodeCache	code cache分区
-XX:-TieredCompilation	取消分层编译
-XX:+UseCondCardMark	更新card table之前检查card是否已经被标记过
-XX:+UseCountedLoopSafepoints	将safe point保持在计数循环中。
-XX:+LogCompilation	输出编译日志
-XX:+PrintAssembly	打印汇编代码
-XX:+PrintCompilation	输出编译的方法信息
-XX:+PrintInlining	输出inlining信息

Thread配置	
VM参数	说明
-XX:TLABSize=size	TLAB的初始大小
-XX:+UseTLAB	开启TLAB
-XX:+ResizeTLAB	允许JVM对TLAB进行调整
-XX:MinTLABSize=64k	最小TLAB大小

内存大小调整	
VM参数	说明
-Xmnsize	young gen的初始化和最大值
-XX:NewSize	young gen的初始化大小
-XX:MaxNewSize	young gen的最大值
-Xmssize -XX:InitialHeapSize=size	heap的初始值
-Xmxsize -XX:MaxHeapSize	heap的最大值
-XX:MaxHeapFreeRatio=percent	GC过后允许的最大free heap比例
-XX:MinHeapFreeRatio=percent	GC过后允许的最小free heap比例
-XX:-ShrinkHeapInSteps	默认开启，和-XX:MaxHeapFreeRatio配合使用，逐步压缩Heap空间大小
-Xsssize -XX:ThreadStackSize	Thread stack size
-XX:MaxDirectMemorySize=size	设置NIO的最大direct-buffer size
-XX:MaxMetaspaceSize=size	元数据区域的最大大小
-XX:MetaspaceSize=size	首次触发GC的class元数据区域大小
-XX:NewRatio=ratio	young和old区域的大小比例
-XX:InitialSurvivorRatio=ratio	survivor占用的比例,随-XX:+UseAdaptiveSizePolicy开启 计算公式：S=Y/(R+2)
-XX:SurvivorRatio=ratio	eden和survivor大小的比例,随-XX:-UseAdaptiveSizePolicy开启
-XX:TargetSurvivorRatio=percent	youngGC之后，survivor的目标使用比例
-XX:+UseAdaptiveSizePolicy	使用自适应的分代大小策略
-XX:CompressedClassSpaceSize=1g	compressed class space大小
-XX:InitialCodeCacheSize=256m	codeCache的初始化大小
-XX:ReservedCodeCacheSize=512m	codeCache的最大大小
-XX:MaxDirectMemorySize=2g	NIO direct buffer的最大值

G1调优参数	
VM参数	说明
-XX:G1HeapRegionSize=32m	heap区域的大小
-XX:G1ReservePercent=10	设置预留空闲内存百分比，以降低内存溢出的风险
-XX:G1MixedGCCountTarget=8	一个混合收集周期中包含多少次混合收集
-XX:G1MixedGCLiveThresholdPercent=percent -XX:G1OldCSetRegionLiveThresholdPercent	old region被包含在一个混合收集周期的阈值
-XX:G1HeapWastePercent=10	设置浪费的堆内存百分比，当可回收百分比小于浪费百分比时，JVM就不会启动混合垃圾收
-XX:MaxGCPauseMillis=500	GC的最大暂停时间
-XX:DefaultMaxNewGenPercent=percent -XX:G1MaxNewSizePercent=percent	young gen占用heap的最大比例
-XX:G1NewSizePercent=percent -XX:DefaultMinNewGenPercent	young gen占用heap的最小比例

GC日志详情	
VM参数	说明
-verbose:gc -Xlog:gc	打印基本的GC信息
-Xlog:gc*	打印详细的GC信息
-Xlog:task*=debug	输出GC work thread task的timestamps
-Xlog:gc+heap=trace	GC的heap信息
-Xlog:age*=level	young gen的age信息
-Xlog:ref*=debug	STW阶段打印reference processing
-Xlog:ergo*=level	输出自适应的分代大小
-XX:+PrintPromotionFailure	输出promotion失败的信息
-Xlog:safepoint	是应用程序在不停止的情况下工作的时间,即两个连续安全点之间的时间
-Xlog:gc+region=trace	输出G1 region分配和回收信息

GC日志输出	
VM参数	说明
-Xlog:gc:garbage-collection.log	将GC日志输出到文件
-XX:ErrorFile=filename	GC错误日志重定向
-XX:LogFile=path	JVM日志重定向

其他GC日志信息	
-XX:+PrintTLAB	输出TLAB信息
-XX:+PrintPLAB	输出PLAB信息
-XX:+PrintOldPLAB	输出old space的PLAB信息
-Xlog:stringdedup*=debug	输出String去重的信息
-XX:+PrintHeapAtSIGBREAK	接收到signal的时候的heap信息
-XX:+PrintClassHistogramAfterFullGC	full GC后的class直方图信息
-XX:+PrintClassHistogramBeforeFullGC	full GC之前的class直方图信息

通用GC参数	
VM参数	说明
-Xnoclassgc	禁用classes的GC
-XX:ActiveProcessorCount=x	重写VM使用的CPU核数
-XX:+AggressiveHeap	开启java heap优化
-XX:+AlwaysPreTouch	在main方法执行之前将所有的page都加载到heap中
-XX:InitiatingHeapOccupancyPercent=percent	触发GC的heap使用比例
-XX:MaxGCPauseMillis=time	最大的GC暂停时间
-XX:MaxHeapFreeRatio=percent	GC之后最大的Heap释放比例
-XX:MinHeapFreeRatio=percent	GC之后最小的Heap释放比例
-XX:TargetSurvivorRatio=percent	young GC之后Survivor的目标比例
-XX:+ScavengeBeforeFullGC	在fullGC之前运行youngGC
-XX:+UseStringDeduplication	开启字符串去重
-XX:StringDeduplicationAgeThreshold=threshold	字符串去重的最小Age数
-XX:+DisableExplicitGC	禁止显式调用System.gc()

GC并发线程控制	
VM参数	说明
-XX:ParallelGCThreads=threads	设置STW的垃圾收集线程数
-XX:+ParallelRefProcEnabled	开启并发reference processing
-XX:+UseGCOverheadLimit	OutOfMemoryError之前JVM在GC上使用的时间比例
-XX:ConcGCThreads = n	设置并行标记线程的数量

Young space tenuring	
-XX:InitialTenuringThreshold=8	设置保有年龄阀值,就是一个对象多少age之后会被升级到old space
-XX:MaxTenuringThreshold=15	最大的保有年龄阈值
-XX:PretenureSizeThreshold=2m	超出该阈值，对象将会被直接分配到old space
-XX:+AlwaysTenure	将young space中的survivor对象直接提升到old space
-XX:+NeverTenure	除非survivor space不能容纳该对象了，否则不会提升到old space