本人有多年的区块链服务经验，为用户提供专业的服务信息。在这里，我将介绍如何调优jvm以及如何从什么开始调优jvm，调优效果如何。选择可以随时随地解决玩币遇到的各种问题。，让你不再担心繁琐的职称评定。

JVM调优的常用命令工具包括：

1)jps命令用于查询正在运行的JVM进程，

。

2)jstat可以实时显示本地或远程JVM进程中的类加载、内存、垃圾回收、JIT编译等数据

3)jinfo用于查询此处当前运行的JVM属性和参数的值。

4)jmap用于显示当前Java堆和永久生成的细节

5)jhat用于分析jmap生成的转储文件，是JDK自带的工具

。

6)jstack用于生成当前JVM的所有线程快照。线程快照是虚拟机的每个线程正在执行的方法，目的是定位线程长时间暂停的原因。

-xms256m；初始化堆大小为256m

-Xmx2g:堆的最大内存是2g；

-xmn50m；新生界大小为50m

-XX:printgdetails？打印千兆周详细信息

-XX:heapdumponotofmemoryerror？在发生OutOfMemoryError错误错误错误时，转储堆快照

-XX:NewRatio=4？将年轻一代和老一代的记忆比例设置为1:4；

-XX:SurvivorRatio=8？将新一代中伊甸人与幸存者的比例设定为8:2；

//参数上写的是新一代垃圾收集器

。

-XX:UseSerialGC？新生代和老一代都用串行集电极串行？旧系列

-XX:UseParNewGC？指定使用全新串行旧垃圾收集器组合；

-XX:UseParallelGC？新一代用的是并行清除，旧时代用的是串行olduJuye

//都是垃圾收集器

。

-xx:useparallelloldgc:parallelworldcombinationofthenewgenerationandtheoldageofcavenge;

-XX:useconcmasweepgc:parnew用于新一代和CMS是用在老年的；

-XX:newsize；新生代最小值；

-XX:MaxNewSize:新生代最大值

-xx:元空间大小的初始大小

-xx:max元空间大小最大值

。JstatName:JavaVirtualMachineStatisticsMonitoringTool

FunctionDescription:

JstatisalightweightgadgetthatcomeswithJDK.。它位于java的bin目录下，主要利用JVM的内置指令在命令行上实时监控Java应用的资源和性能，包括堆大小和垃圾收集。

命令用法：jstat[-命令选项][vmid][间隔/毫秒][查询次数]

注意：使用的jdk版本是jdk8。

c:usersadministratorjstat-helpusage:jstat-help|-optionsjstat-option[-t][-hlines]vmid[interval[count]]definition:optionvmidvirtualmachineidentifierreportedbyoption-options.Thevmidtakesthefollowingform:lvmid[@hostname[:port]],wherelvmidisthelocalvmidentifierofthetargetJavavirtualmachine,usuallytheprocessID;Hostnameisthenameofthehostrunningthetargetjavavirtualmachine;Portistheportnumberofrmiregistryonthetargethost.Foramorecompletedescriptionofthevirtualmachineidentifier,seethejvmstatdocumentation.Numberofsamplesbetweenheaderrows.Intervalsamplinginterval.Thefollowingformsareallowed:n["ms"|"s"],wherenisaninteger,andthesuffixspecifiesmilliseconds("ms")orseconds("s").Thedefaultunitis"milliseconds"Calculatethenumberofsamplestobecollectedbeforetermination.-Jflagpassestheflagdirectlytotheruntimesystem.

选项：参数选项

-t:可以在打印列中添加时间戳列，显示系统运行时间

-h:输出周期性数据时。，指定要输出多少行，然后输出一次。Header

VMID:虚拟机ID(进程的PID)

interval:以毫秒为单位的执行间隔

。

计数：用于指定输出记录的次数，

选项默认一直打印。您可以从以下参数中选择

jstat-options

。

-class用于查看类加载的统计数据

-编译器用于查看HotSpot中即时编译器编译的统计数据

-gc用于查看JVM中堆的垃圾收集的统计数据。

-gccapacity用于查看新世代、老世代和持久世代的存储容量

-gcmetacapity显示元空间的大小

。

-gcnew用于查看新一代的垃圾收集

-gcnewcapacity用于查看新一代的存储容量

-gcold用于查看老一代的垃圾收集和持久代

-gcoldcapacity用于查看老一代的容量

-gcutil显示垃圾收集信息

。

-gcause显示关于垃圾收集的信息(pass-
gcutil)，同时显示当前只发生的上一次垃圾收集的原因

-print编译输出JIT编译的方法信息。

例：

1。-类加载统计

[root@hadoop~]#jps#First,theJavalanguage(acomputerlanguage,Especiallyforcreatingwebsites)processnumber(here,itisacityzooprocess)3346quorumpermain7063JPS[root@Hadoop~]#jstat-class3346#Countingthenumberandsizeofclassesloadedinavirtualmachine(abbreviationofJavaVirtualMachine)Bytetimeofloadedbytesunloaded15272842.70.01.02.

Loaded:已加载类的数量

Bytes:已加载类的大小(以字节计)

Unloaded:已卸载类的数量

。

Bytes:以字节为单位的卸载类的大小

time:加载和卸载类所用的时间

2。-编译器编译统计

[root@Hadoop~]#jstat-compiler3346#Statisticalcompilationfailureinvalidtimefailuretypefailuremethod404000.190forviewingthecompilationsituationofreal-timecompilersinhotspots.

编译：编译任务执行数量

失败：编译任务执行失败数量

无效：编译任务执行失败数量

Time:编译任务消耗时间

FailedType:上次编译失败任务的类型

FailedMethod:上次编译失败任务的类和方法

第三章.-gc垃圾回收统计

[root@Hadoop~]#jstat-GC3346#用于查看虚拟机(Java虚拟机的缩写)中堆的垃圾收集情况的统计S0CS1CS0US1UECEUOCOUMCMUCCSCCCSUYGCYGCTFGCFGCTGCT128.0128.00.01024.0919.815104.02042.48448.08130.41024.0996.070.0190.0000.019

S0C:年轻一代中第一幸存者的容量(字节)

S1C:年轻一代中第二幸存者的容量(字节)

S0U:年轻一代中第一个幸存者的当前使用空间(字节)

S1U:年轻一代中第二个幸存者的当前使用空间(字节)

。

EC:年轻一代Eden的容量(字节)

EU:年轻一代Eden当前使用的空间(字节)

OC:老一代的容量(字节)

。

ou:旧代当前使用的空间(字节)

MC:元空间的容量(字节)

Mu:元空间当前使用的空间(字节)

CCSC:当前压缩类空间的容量(以字节为单位)

CCSU:当前压缩类空间的当前使用空间(以字节为单位)

Ygc:年轻一代从应用启动到采样的gc次数。

YgcT:年轻一代从应用程序启动到采样的GC时间

FGC:老一代(完全GC)从应用程序启动到采样的GC时间

FgcT:旧代GC(完全GC)从应用程序启动到采样所用的时间

GCT:GC从应用程序启动到采样所用的总时间

。

4。-gccapacityheapmemorystatistics

[root@Hadoop~]#jstat-gccapacity3346#usedtoviewtheCenozoic,Storagecapacityofoldgenerationandpersistentgeneration NGCMnNGCMXNGCS01C1CECMNOGCMXOGCOCMCMNMCMXMCCCSMNCCSMXCCSCYGCFGC1280.083264.01280.0128.01024.015104.0166592.015104.015104.001056768.08448.0.01048576Displaytheheaderevery5lines1000:displayiteverysecond.,单位为毫

秒NGCMnNGCMXNGCS0CS1CECOGCMnOGCMXOGCOCMCMNMCMXMCCCSMnCCSMXCCSCYGCFGC1280.083264.01280.0128.01024.015104.0166592.015104.015104.01056768.084480.0.0104857615104.015104.00.01056768.08448.000.01048576.01024.0701280.083264.01280.0128.0128.01024.015104.0166592.015104.015104.0015104.00000000.0104.0104.0104.0104.0105.10101280.083264.01280.0128.0128.01024.015104.0166592.015104.015104.00.01056768.08448.00.01048576.01024.0701280.083264.01280.0128.0128.01024.015104.0166592.015104.015104.00.01056768.08448.00.01048576.01024.070

NGCMN:年轻一代中初始化(最小)的大小(字节)

NGCMX:年轻一代的最大容量(字节)

NGC:年轻一代中的当前容量(字节)

S0C:年轻一代中第一个幸存者的容量(字节)

S1C:年轻一代中第二幸存者的容量(字节)

EC:年轻一代中伊甸园的容量(字节)

ogcmn:旧代中的初始化(最小)大小(字节)

ogcmx:旧代的最大容量(字节)

OGC:旧代的当前新生成容量(字节)

。

oc:老一代的容量(字节)

mcmn:元空间中初始化的大小(最小值)(字节)

mcmx:元空间的最大容量(以字节为单位)

MC:当前新生成的元空间的容量(以字节为单位)

CCSMN:最小压缩类空间大小。

CCSMX:最大压缩类空间大小

CCSC:当前压缩类空间大小

Ygc:年轻一代从应用启动到采样的gc时间

FGC:从应用程序启动到采样的老一代GC时间

5。-GCmetacapacity元数据空间统计数据

[root@Hadoop~]#jstat-GCmetacapacity3346#显示元数据空间的大小CCSCYGCFGCFGCTGCT0.01056768.08448.00.01048576.01024.0800.0000.020

MCMN:最小元数据容量

MCMX:最大元数据容量

MC:当前元数据空间大小

CCSMN:最小压缩类空间大小

。

CCSMX:最大压缩类空间大小

CCSC:当前压缩类空间大小

Ygc:年轻一代从应用启动到采样的gc时间

FGC:老一代(完全GC)从应用程序启动到采样的GC时间

FGCT:老一代(完全gc)gc从应用程序启动到采样所用的时间

gcT:GC从应用程序启动到采样所用的总时间

6。-gcnew垃圾收集统计信息

[root@Hadoop~]#jstat-gcnew3346#Usedtoviewthesituationofgarbagecollectioninthenewgeneration.S0CS1CS0USUTTMTTDSSEUYGCYCT128.0128.067.

80.011564.01024.0362.280.020.

S0C:年轻一代中第一幸存者的容量(字节)

S1C:年轻一代中第二幸存者的容量(字节)

S0U:年轻一代中第一个幸存者的当前使用空间(字节)

S1U:年轻一代中第二个幸存者的当前使用空间(字节)

。

TT:持有次数限制

MTT:最大持有次数限制

DSS:期望生存面积大小

EC:年轻一代伊甸园容量(字节)

。

EU:年轻一代中Eden当前使用的空间(字节)

Ygc:年轻一代中从应用程序启动到采样的gc时间

YgcT:年轻一代中的GC从应用程序启动到采样所用的时间

7。-新一代gcnewcapacity的内存统计

[root@Hadoop~]#jstat-gcnewcapacity3346#用于查看新生代存储容量的情况NGCMnNGCMXNGCS0CMXS0CS1cmxS1CECMXECYGCFGC1280.083264.01280.08320.0128.08320.0128.066624.01024.080

NGCMN:年轻一代中初始化(最小)的大小(字节)

NGCMX:年轻一代的最大容量(字节)

NGC:年轻一代中的当前容量(字节)

S0CMX:年轻一代中第一个幸存者的最大容量(字节)

S0C:年轻一代中第一幸存者(生存区域)的容量(字节)

S1CMX:年轻一代中第二幸存者(生存区域)的最大容量(字节)

S1C:年轻一代中第二幸存者(生存区域)的容量(字节)

ECMX:年轻一代中Eden的最大容量(字节)

EC:年轻一代中Eden的容量(字节)

Ygc:年轻一代中从应用程序启动到采样的gc时间

FGC:从应用程序启动到采样的老一代GC时间

8。-gcold垃圾收集统计信息

[root@Hadoop~]#jstat-gcold3346#用于检查陈年和持久代的垃圾收集。中国FGCGCT8448.08227.51024.01003.715104.02102.2800.0000.020。

MC:元空间容量(字节)

Mu:当前使用的元空间空间(字节)

CCSC:压缩类空间大小

CCSU:压缩类空间的已用大小

oc:老一代的容量(字节)

ou:老一代的当前已用空间(字节)

Ygc:年轻一代从应用程序启动到采样的gc时间

FGC:老一代(完全GC)从应用程序启动到采样的GC时间

FgcT:旧代GC(完全GC)从应用程序启动到采样所用的时间

GCT:GC从应用程序启动到采样所用的总时间

。9。-旧gcoldspace的内存统计信息

[root@Hadoop~]#jstat-gcoldcapacity3346#用于查看老年代的容量OGC锰OGCMXOGCOCYGCFGCFGCTGCT15104.0166592.015104.015104.0800.0000.020

ogcmn:老一代中的初始化(最小)大小(字节)ogcmx:老一代中的最大容量(字节)Ogc:老一代中当前新生成的容量(字节)OC:老一代中的容量(字节)Ygc:年轻一代中从应用启动到采样的GC时间FGC:从应用启动到采样。GctimesFgcT:老一代(fullgc)gc从应用启动到采样所用的时间(s)GCT:GC从应用启动到采样所用的总时间(s)这里给大家推荐一个架构学习交流圈。。假鑫：1253431195(里面有大量面试问答)会分享一些资深架构师录制的视频：Spring，MyBatis，Netty源代码分析，高并发，高性能，分布式，微服务架构的原理。、JVM性能优化、分布式架构等成为架构师必备的知识体系。还可以获得免费的学习资源，目前已经受益匪浅

10。-gcutil垃圾收集统计信息

[root@Hadoop~]#jstat-gcutil3346#显示垃圾收集信息s0S1EOMCCSYGCYGCTFGCFGCTGCT52.970.0042.1013.9297.3998.0280.0200.0000.020

S0:年轻一代中第一个幸存者(生存区域)使用的当前容量百分比

S1:年轻一代中第二个幸存者(生存区域)使用的当前容量百分比

。

E:年轻一代Eden当前容量的百分比

o:老一代Eden当前容量的百分比

M:元数据区当前容量的百分比

。

CCS:当前容量中使用的压缩类空间的百分比

Ygc:年轻一代中从应用程序启动到采样的gc时间

YgcT:年轻一代从应用程序启动到采样所用的GC时间

FGC:老一代(完整GC)从应用程序启动到采样所用的GC时间

FgcT:旧代GC(完全GC)从应用程序启动到采样所用的时间

GCT:GC从应用程序启动到采样所用的总时间

11。-gccause。

[root@Hadoop~]#jstat-GCcause3346#显示关于垃圾收集的信息(通过-gcutil)。同时显示上次或当前垃圾收集的诱因，s0s1EOmCCSYGCTFGCFGCFGCTGCTLGCCGCC52.970.0046.0913.9297.3998.0280.02000.0000.020分配失败无GC。

LGCC:上次GC原因

GCC:当前GC原因(没有GC表示当前没有执行GC)

12.-打印编译JVM编译方法统计信息

[root@Hadoop~]#jstat-printcompilation3346#OutputmethodinformationcompiledbyJiteDanceMusicCompilationsizetypemethod421601sun/nio/ch/Util$2clear

编译：编译任务数

大小：方法生成的字节码大小

类型：编译类型

方法：类名和方法名用于标识编译后的方法。类名使用/作为命名空间分隔符。方法名是给定类中的方法。上述格式由-xx:printcompilation选项

设置

远程监控

和jps一样，jstat也支持远程监控，也需要开启安全授权。有关方法，请参考jps。

用户管理员jps192。168.146.1283346quorumpeermain3475JstatdC:usersadministratorjstat-gcutil3346@192。168.146.128S0S1东部奥姆CCSYGCYGCTFGCFGCTGCT52.970.0065.1513.9297.3998.0280.020

JVM性能调优有很多设置，参考JVM参数即可。

[XY001]主要调谐目的：

控制GC的行为。GC是一个后台进程，但是它也消耗系统的性能，所以GC的行为往往会根据系统运行的程序的特性而改变

。

控制JVM堆栈大小。一般来说，你不会&#039；不需要修改JVM&#039的内存分配，(例如)。但是，当你的程序在一定时间内有很多新生代对象时，你就需要控制新生代的堆大小。同时，您需要控制JVM的总大小，以避免内存溢出。

控制JVM线程的内存分配。如果是多线程程序，线程消耗的内存和线程运行也是可以控制的，最优的结果需要经过一定时间的观察才能配置出来。

JVM调优的目的是减少gc次数，缩短gc时间。

从这两个维度优化。

倍：增加jvm内存空间，长时间持有对象，直接进入老年等。

时间：增强物理机性能或采用更好的gc收集器

如何调优jvm是很多人头疼的问题，尤其是在理解与现实的冲突中。如何对JVM进行调优，从什么方面入手，产生什么样的效果也是面临的类似问题。