JVM参数调优

区域大小

1. `-Xmn`：设置新生代大小，将NewSize与MaxNewSize设为一致，如`-Xmn256m`同下：

```
-XX:NewSize=256m
-XX:MaxNewSize=256m
```

   建议：如果将新生代的值设置过大，那么一次新生代GC的时间就会过长；如果将新生代的值设置过小，又会引起频繁新生代GC。官方建议该值设置在堆最大值的25%-50%区间内。

2. `-Xss`：设置虚拟机线程栈大小，linux默认为`1m`，windows的默认值为`0`（实际上，虚拟机会根据虚拟内存计算出默认值）；效果同`-XX:ThreadStackSize=size`；一般几个系统的默认值如下：

> Linux/ARM (32-bit): 320 KB Linux/i386 (32-bit): 320 KB Linux/x64 (64-bit): 1024 KB OS X (64-bit): 1024 KB Oracle Solaris/i386 (32-bit): 320 KB Oracle Solaris/x64 (64-bit): 1024 KB

3. 设置永久代：`-XX:MaxMetaspaceSize=size`（jdk8之前：`-XX:MaxPermSize=size`）；
4. 错误转储：`-XX:+HeapDumpOnOutOfMemoryError`，还可使用选项`-XX:HeapDumpPath=path`指定文件地址，示例如下(%p表示进程PID)：

```
-XX:HeapDumpPath=/data/log/java_pid%p.hprof
```

5. 内存溢出后执行命令：`-XX:OnOutOfMemoryError=string`；

> 同理`OnError`

6. `-XX:ErrorFile=filename`指定Java进程发生不可恢复的错误时，保存错误文件的地址和文件名。默认情况下，错误文件存储在当前路径下，文件名为hs_err_pid[pid].log。一个示例如下（%p表示进程PID）：

```
-XX:ErrorFile=/data/log/java_error_%p.log
```

   如果文件由于权限问题、硬盘已满或是其他问题而不能在指定目录下创建，那么该文件将创建在临时目录下。对于Linux环境，临时目录为/tmp；对于Windows环境，临时目录由环境变量TMP指定，如果该环境变量没有指定，那么将使用TEMP环境变量的值。

收集器设置

策略设置

- `-d32`、`-d64`：设置启动模式是32位或64位，-d64模式和-server模式是绑定的，也就是说-d64声明了就默认是server模式。
- 断言：

```
-enableassertions[:[packagename]...|:classname]
-ea[:[packagename]...|:classname]

disableassertions[:[packagename]...|:classname]
-da[:[packagename]...|:classname]
```

系统包下的断言：

```
-enablesystemassertions
-esa

-disablesystemassertions
-dsa
```

- 信息打印

```
-verbose:class
-verbose:gc
-verbose:jni
```

- gc日志：`-Xloggc:filename`
- 关于解释器模式：`-Xmixed`、`-Xint`
- 关闭对class（方法区、元数据区）的gc：`-Xnoclassgc`
- 自行理解`-XshowSettings`
- 错误日志文件：`-XX:ErrorFile=filename`

> 当不可恢复的错误发生时，错误信息记录到哪个文件。默认是在当前目录的一个叫做hs_err_pid pid.log的文件。如果指定的目录没有写权限，这时候文件会创建到/tmp目录下。使用如下：

- 对象内存对齐大小`-XX:ObjectAlignmentInBytes=alignment`，默认是8字节，JVM实际计算堆内存上限的方法是`4GB * ObjectAlignmentInBytes`；
- 指针压缩`XX:+UseCompressedOops -XX:+UseCompressedClassPointers`

> 对于大小在4G和32G之间的堆，应该使用压缩的oop

- `XX:CompressedClassSpaceSize`（参考：JVM源码分析之Metaspace解密）

> Starting with JDK 8, the permanent generation was removed and the class metadata is allocated in native memory. Java Hotspot VM explicitly manages the space used for metadata. Space is requested from the OS and then divided into chunks. A class loader allocates space for metadata from its chunks (a chunk is bound to a specific class loader). 链接

`CompressedClassSpaceSize指定的内存只保存对象的压缩指针`

更多

1. 查看默认值

命令：`java -XX:+PrintFlagsFinal -version`

2. 命令行参数格式

- `-Dproperty=value` JVM全局属性设置，可通过`System.getProperty("name")`获取；
- 前缀`-X`表示：HotSpot实现常用功能的选项，其他JVM不一定实现。如：`-Xmx`、`-Xmn`等

- Boolean型设置 布尔Boolean型，表示开启或关闭某一功能。使用加号+表示开启某一功能，减号-表示关闭某一功能，如：`-XX:+OptionName -XX:-OptionaName`

未解之谜

javaagent使用指南 参数`-XshowSettings` jvm的`attach`模式

更多参考

Java8 JVM参数解读