

# 第七节 JDK8升级到JDK11导致的问题

背景

初步解决

进一步解决

思路

已有参数配置

## 背景

因jdk8漏洞比较多，应安全部门要求，升级到jdk11，在升级过程中，出现了如下问题，应用启动失败。

```
robotaxi-trip-provider-deployment-6d89f98c84-sk5tj 0/1 Running 0 15d
[root@cceworker-dev-02 ~]# kubectl logs -f robotaxi-account-provider-deployment-5bf6cc7df-29mwg -n tiger-dev
Unrecognized VM option 'PrintGCTimeStamps'
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

## 初步解决

后来我们粗暴的去除了启动参数中的所有的 -XX 参数。应用启动成功了。原因是 -X 是非标参数，-XX 参数是不稳定参数。前者是任意JDK任意厂商必须支持的参数，但是实现方式可能存在差别。后者可能随着JDK的升级而不复存在。

## 进一步解决

粗暴的去除了启动参数中的所有的 -XX 参数显然是不可取的，因为有些参数是被支持的，而且对于我们应用来说，对之后的排查问题有很大的帮助，那么有没有什么其他的办法可以找到这个参数呢？

## 思路

基于测试的目的与理论基础，业界认为：- 是标准参数，-X非标参数，-XX不稳定参数。

1. 统一去除 -XX 参数，应用启动成功，可以确保最开始我们的猜测结果是正确的。因为JVM的启动参数是逐个检查报错的，不是全部报错，如果我们逐一去除，很难一招定位问题。
2. 具体JVM支持的哪些参数，我们可以通过运行命令获取：  
java -XX:+PrintFlagsInitial
3. openjdk1.8 与 openjdk1.11 参数的对标，需要理论知识的推敲。一般而言，名字未做变更的，含义是一样的，名字变更或者直接没有的，理论上是有取代参数可以使用。

## 已有参数配置

```
java_jvm = "-server -Xmx1024m -Xms128m -XX:MetaspaceSize=256m -
XX:MaxHeapFreeRatio=70 -XX:MinHeapFreeRatio=40 -XX:+HeapDumpOnOutOfMemoryError -
XX:HeapDumpPath=/tmp -XX:ErrorFile=/tmp/jvm_err.log -XX:+PrintGCDetails -
XX:+PrintGCTimeStamps -Xloggc:/tmp/gc.log -XX:+UseGCLogFileRotation -
XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=10M"
```

因为这些参数是我们升级之前配置的，所以我们基于这个去找就可以了。

```
uintx MaxMetaspaceFreeRatio = 70 (product) {default}
size_t MaxMetaspaceSize = 18446744073709551615 (product) {default}
size_t MaxNewSize = 18446744073709551615 (product) {default}
intx MaxNodeLimit = 80000 {C2 product} {default}
uint64_t MaxRAM = 137438953472 {pd product} {default}
uintx MaxRAMFraction = 4 (product) {default}
double MaxRAMPercentage = 25.000000 (product) {default}
intx MaxRecursiveInlineLevel = 1 {C2 product} {default}
uintx MaxTenuringThreshold = 15 (product) {default}
intx MaxTrivialSize = 6 {C2 product} {default}
intx MaxVectorSize = 64 {C2 product} {default}
size_t MetaspaceSize = 21810376 {pd product} {default}
```

按照这个思路，找到了一个典型的参数。并且我认为，关于GC的控制的参数，肯定是有变更的。自1.8之后，推出了ZGC，官网一直在推这个算法，其他的关于分代的配置，，应该还是没有变化的。

通过这个思路，我们排查到以下参数的支持情况(Y支持，N不支持)。

```
X:MetaspaceSize=256m Y
X:MaxHeapFreeRatio=70 Y
X:MinHeapFreeRatio=40 Y
X:+HeapDumpOnOutOfMemoryError Y
X:HeapDumpPath=/tmp Y
X:ErrorFile=/tmp/jvm_err.log Y
X:+PrintGCDetails Y
X:+PrintGCTimeStamps -Xloggc:/tmp/gc.log N
X:+UseGCLogFileRotation N
X:NumberOfGCLogFiles=5 N
X:GCLogFileSize=10M" N
```

从上述结果可以看到，如预期结果差不多，GC的参数有变动。