



United States ▼

Communities ▼

I am a... ▼

I want to... ▼

Secure Search



Products and Services

Downloads

Store

Support

Education

Partners

About

Oracle Technology Network ▼

Oracle Technology Network &gt; Java &gt; Java SE &gt; Community &gt; Bug Database

**JDK-8043516 : After "ulimit -v", the JVM can not start without extra GC command line args.****Details****Type:** Bug**Status:** Closed**Project Name:** JDK**Component:** hotspot**Sub-Component:** gc**Priority:** P4**Resolution:** Won't Fix**Affected Versions:** 9**Fixed Versions:** 9**Submit Date:** 2014-05-20**Updated Date:** 2016-04-15**Resolved Date:** 2016-04-15**OS:** linux**CPU:****Related Reports****Relates:** [JDK-8044054 - nsk/jvmti/Allocate/alloc001 OOMs again](#)**Relates:** [JDK-7112912 - Message "Error occurred during initialization of VM" on boxes with lots of RAM](#)**Sub Tasks****Description**

If virtual memory has been limited with "ulimit -v", and the server has a lot of RAM, then the JVM can not start without extra command line arguments to the GC.

Both XX:MaxHeapSize and XX:CompressedClassSpaceSize have to be specified. (there may be other combinations that work, I have not tried all)

It would be good if the jvm could start without extra command line args, even if virtual memory has been limited.

There is an old bug for this that was resolved in jdk8, but it is still failing in jdk9.

This log is from a server with 32G ram:

```
// Before "ulimit -v", it works ok.
$ java -version
java version "1.9.0-ea-fastdebug"
Java(TM) SE Runtime Environment (build 1.9.0-ea-fastdebug-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b62-fastdebug, mixed mode)

// After "ulimit -v" The jvm does not start with default command line.
$ ulimit -S -v 4194304
$ java -version
Error occurred during initialization of VM
Could not allocate metaspace: 1073741824 bytes

// Different command line args gives different error messages.
$ java -XX:CompressedClassSpaceSize=64m -version
Error occurred during initialization of VM
Cannot create VM thread. Out of system resources.

$ java -XX:MaxHeapSize=512m -version
#
# There is insufficient memory for the Java Runtime Environment to continue.
# pthread_getattr_np
# An error report file with more information is saved as:
# /home/egahlin/mtobiass/hs_err_pid24449.log[thread 139641639245568 also had an error]
[thread 139641638192896 also had an error]
[thread 139641637140224 also had an error]
[thread 139641636087552 also had an error]
[thread 139641635034880 also had an error]
Error occurred during initialization of VM
java.lang.OutOfMemoryError: unable to create native thread: possibly out of memory or process/resource limits reached

// This command line works.
$ java -XX:MaxHeapSize=512m -XX:CompressedClassSpaceSize=64m -version
$ java version "1.9.0-ea-fastdebug"
Java(TM) SE Runtime Environment (build 1.9.0-ea-fastdebug-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b62-fastdebug, mixed mode)
```

**Comments**

Closing as "Won't fix" for the reasons as mentioned in my first comment above.

2016-04-15

The problem seems to be that we must specify MALLOC\_ARENA\_MAX.

If I set the environment variable MALLOC\_ARENA\_MAX=4, then the jvm can start without any extra arguments.

I guess that this is not something that can be fixed from the jvm. If so we can close this bug.

2014-06-11

When using "UseConcMarkSweepGC" then the command line above does not work.

I have tried to add MaxMetaspaceSize=128m, but it does not help.

I am sure there are an argument that makes it work, but I have not found one.

Configuring the GC with limited virtual memory is not very user friendly.

```
ulimit -S -v 4194304
java -XX:MaxHeapSize=512m -XX:InitialHeapSize=512m -XX:CompressedClassSpaceSize=64m -XX:MaxMetaspaceSize=128m -XX:+UseConcMarkSweepGC -version
#
# There is insufficient memory for the Java Runtime Environment to continue.
# pthread_getattr_np
# An error report file with more information is saved as:
# /export/home/mtobiass/ute/hs_err_pid1936.log
```

15.6.2017

Bug ID: JDK-8043516 After "ulimit -v", the JVM can not start without extra GC command line args.

```
[thread 139650370844416 also had an error]
[thread 139650369791744 also had an error]
[thread 139650368739072 also had an error]
[thread 139650367686400 also had an error]
[thread 139650366633728 also had an error]
[thread 139650365581056 also had an error]
java version "1.9.0-ea"
Java(TM) SE Runtime Environment (build 1.9.0-ea-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b62, mixed mode)
```

2014-06-09


Unclear to me of this is something we really need/want to support. We already have some logic to take virtual memory restrictions into account when calculating heap sizes, but you can theoretically set lots of restrictions on the environment (i.e. not just memory), which the JVM don't understand or handle well.

ILW=MML=P4

2014-05-23

---

**Hardware and Software, Engineered to Work Together**

[Subscribe](#) | [About Oracle](#) | [Careers](#) | [Contact Us](#) | [Site Maps](#) | [Legal Notices](#) | [Terms of Use](#) | [Your Privacy Rights](#) |  [Oracle Mobile](#) |