redhat TRAINING

CHAPTER 10

CONFIGURATION OF THE JAVA VIRTUAL MACHINE

General description	
Meta	Configure JVM settings on a standalone server and in a managed domain. •
Goals	Perform JVM configuration on a separate server. • Perform JVM configuration in a domain managed.
sections	• JVM configuration on a standalone server (and quiz) • Configuring the JVM in a managed domain (and guided exercise)
Laboratory work	• Configuring the Java virtual machine

JVM configuration on a standalone server

Goals

After completing this section, students should be able to do the following:

- **Perform JVM configuration on a separate server.**

JVM memory architecture

EAP is written in Java, a popular object-oriented, cross-platform programming language. Java programs run on a Java Virtual Machine (JVM). It is important for a JBoss system administrator to understand, at a minimum, how Java programs work within the JVM. In this section, we will discuss how to configure JVM memory settings for your EAP servers. EAP 7 requires a JDK 1.8 compliant JVM to run. Older JVMs, such as JDK 1.7, are no longer supported.

The JVM runs as a regular user-space process in the OS, and strictly observes the memory limits defined by the OS. To ensure consistent cross-platform runtime support, JVM memory flags and values are identical between different operating systems. During the JVM startup process, some validation is performed to ensure that valid parameters have been set (for example, the minimum heap size must be less than or equal to the maximum heap size).

The JVM uses sophisticated memory management mechanisms and can automatically deallocate unused Java data structures (objects) from heap memory through a process called garbage collection.

The memory of a Java Virtual Machine (JVM) can be divided into two categories:

- **Dynamic memory – A block of memory that dynamically increases and decreases in size, in which application Java data structures (objects) reside.**

- **Non-dynamic memory: Consists of the stack, class and method metadata, tag cache, and metaspace.**

The size of these different blocks of memory can be specified as arguments to the JVM at startup. Some of the values are shown in the following graph:

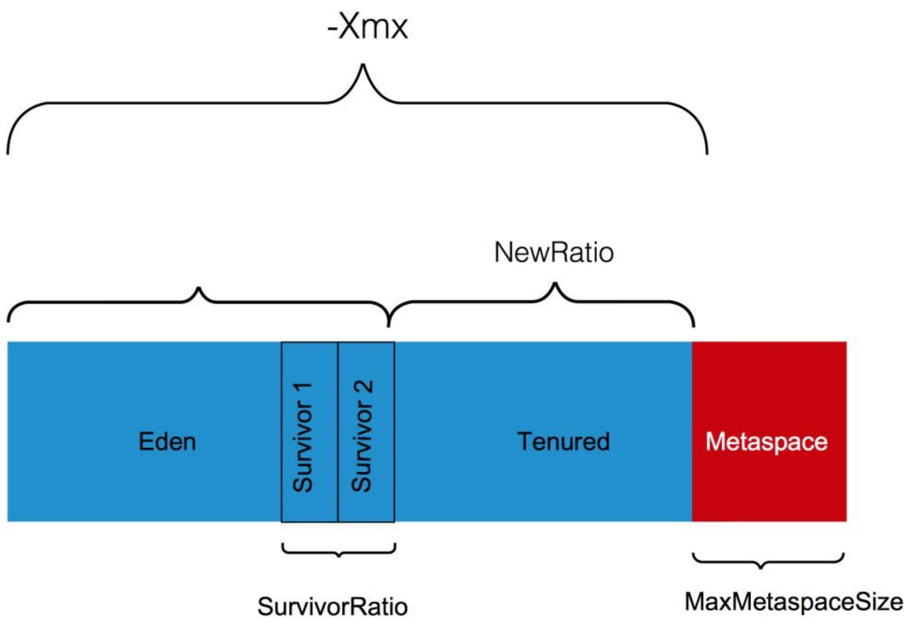

Figure 10.1: JVM memory regions

- In the Eden space, new objects appear. If an item has a short duration, the garbage collector in the Eden space can collect it quickly.
- The survivor space is where objects that survived the garbage collection cycles in the young generation are stored. Generally, there are two survivor spaces.
- The Tenured space is where "old" objects that survived multiple garbage collection cycles are stored in the Eden and Survivor spaces.
- The metaspace is where class and method metadata is stored, as well as other internal JVM data structures.

use

One of the major changes between JDK 1.7 and JDK 1.8 is the removal of the permanent build of JDK 1.8. Permanent generation has been replaced by metaspace in JDK 1.8.

There are several settings available to tune the performance of the JVM. The most frequent are included below:

- -Xms is used to indicate the minimum heap size. It must always be less than or equal to the maximum heap size.
- -Xmx is used to indicate the maximum size of dynamic memory. If the application allocates memory larger than the maximum heap size, this will cause JVM process swapping and eventually an out-of-memory (OOM) error.

- -XX:MaxMetaspaceSize is used to indicate the maximum size of the metaspace.
- -XX:NewRatio is used to indicate the ratio between the sizes of the generation old and new generation.
- -XX:NewSize is used to indicate the size of the Eden generation.

Properly configuring the JVM memory slot sizes can help prevent out-of-memory exceptions and, in turn, improve the performance of any application. Configuring the JVM is different for a standalone server than it is for servers in a managed domain:

- On a standalone server, the installed JVM starts the server instance, and that process is responsible for determining the values of the JVM.
- In a managed domain, instead, the host controller starts the server that the JVM processes, so the host controller is responsible for determining the JVM settings for each individual server.

Configure JVM settings for a standalone server

The JVM memory settings for a standalone server are located in the standalone.conf file in the *JBOSS_HOME/ bin folder.* The JVM values defined in the JAVA_OPTS variable are passed to the system JVM process running the standalone server.

The line to modify in standalone.conf is the following:

```
# Specify options to pass to the Java VM. #

if [ "x$JAVA_OPTS" = "x" ]; then
    JAVA_OPTS="-Xms1303m -Xmx1303m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m ..."
```

Within the double quotes of the JAVA_OPTS variable, you can edit, add, or remove any value regarding the JVM memory and garbage collection options for your standalone server. Changes to the JVM settings in standalone.conf require the server to be restarted.

Quiz: Configuring the JVM on a standalone server

Choose the correct answer to the following questions:

1. How many survivor slots are in the young generation of the JVM? (Choose one option).

a. 1
b. 2
c. 3
d. 4

2. Which of the following statements are true? (Choose two options.)

a. **The metaspace and the permanent generation should be the same size in JDK 1.8.**
b. **There is no permanent build in JDK 1.8.**
c. **EAP 7 can run on JDK 1.7 as well as JDK 1.8.**
d. **EAP 7 can only run on JDK 1.8 and later.**
It is. **In JDK 1.8, the minimum (-Xms) and maximum (-Xmx) sizes of the JVM heap should always be set to the same value.**

3. Which of the following statements are false? (Choose two options.)

a. **For JDK 1.8, there are different JVM flags for different operating systems.**
b. **For an EAP 7 standalone server, you cannot change the JVM settings without restarting the server.**
c. **The memory size of the young generation (Eden) is always less than the maximum heap size (-Xmx) configured for the JVM.**
d. **You cannot run more than one EAP 7 standalone server per operating system.**
It is. **The old generation memory size (Tenured) is always less than the maximum dynamic memory size (-Xmx) configured for the JVM.**

4. Which of the following statements are correct? (Choose two options.)

a. **The minimum heap size (-Xms) of a JVM can be larger than the maximum size (-Xmx).**
b. **The minimum heap size (-Xms) is always less than or equal to the maximum size (-Xmx).**
c. **The minimum heap size (-Xms) is always less than, and can never be equal to, the maximum size (-Xmx).**
d. **You can set the minimum and maximum heap sizes of a JVM to a value greater than the memory allocated (RAM) to the operating system.**

5. Consider the following configuration in a server's standalone.conf file EAP 7

```
JAVA_OPTS="-Xms4096m -Xmx2048m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=4098m"
```

Which of the following statements is correct? (Choose one option).

a. **The server will fail to start because the maximum metaspace size is greater than the maximum heap size (-Xmx).**

b. **The server fails to start because the minimum dynamic memory value (-Xms) is too low.**

c. **The server fails to start because the minimum heap size (-Xms) is greater than the maximum size (-Xmx).**

d. **The server starts successfully without any error The JVM**

It is. **immediately allocates 2 GB as the maximum heap size (-Xmx) is set to this value.**

Solution

Choose the correct answer to the following questions:

1. How many survivor slots are in the young generation of the JVM? (Choose one option).

a.

b. 1 **2**

c. 3

d. 4

2. Which of the following statements are true? (Choose two options.)

a. The metaspace and the permanent generation should be the same size in JDK 1.8.

b. There is no permanent build in JDK 1.8.

c. EAP 7 can run on JDK 1.7 as well as JDK 1.8.

d. **EAP 7 can only run on JDK 1.8 and later.**

It is. In JDK 1.8, the minimum (-Xms) and maximum (-Xmx) sizes of the JVM heap should always be set to the same value.

3. Which of the following statements are false? (Choose two options.)

a. **For JDK 1.8, there are different JVM flags for different operating systems.**

b. For an EAP 7 standalone server, you cannot change the JVM settings without restarting the server.

c. The memory size of the young generation (Eden) is always less than the maximum heap size (-Xmx) configured for the JVM. **d. You cannot run more than one EAP 7 standalone server per system operational.**

It is. The old generation memory size (Tenured) is always less than the maximum dynamic memory size (-Xmx) configured for the JVM.

4. Which of the following statements are correct? (Choose two options.)

a. The minimum heap size (-Xms) of a JVM can be larger than the maximum size (-Xmx).

b. **The minimum heap size (-Xms) is always less than or equal to the maximum size (-Xmx).**

c. The minimum heap size (-Xms) is always less than, and can never be equal to, the maximum size (-Xmx).

d. **You can set the minimum and maximum heap sizes of a JVM to a value greater than the memory allocated (RAM) to the operating system.**

5. Consider the following configuration in a server's standalone.conf file EAP 7

```
JAVA_OPTS="-Xms4096m -Xmx2048m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=4098m"
```

Which of the following statements is correct? (Choose one option).

a. The server will fail to start because the maximum metaspace size is greater than the maximum heap size (-Xmx).

b. The server fails to start because the minimum dynamic memory value (-Xms) is too low.

c. **The server fails to start because the minimum heap size (-Xms) is greater than the maximum size (-Xmx).**

d. The server starts successfully without any error The JVM

It is. immediately allocates 2 GB as the maximum heap size (-Xmx) is set to this value.

JVM configuration in a managed domain

Goals

After completing this section, students should be able to do the following:

• Perform JVM configuration in a managed domain.

Overview of JVM configuration in a managed domain

In an EAP managed domain, the host controller is responsible for starting the JVM process for each server in the managed domain. JVM settings for each individual server can be configured at three different levels:

• host controller level: the <jvm> section within host.xml

• server group level: the <server-group> section in domain.xml

• server level: the <server> section within host.xml

use

Similar to the $JBOSS_HOME/bin/standalone.conf file for a standalone server, domain.conf exists in the same folder. The JVM memory settings in domain.conf refer to the memory provided to the Java process that runs the host controller and NOT the individual EAP servers within that host.

A JVM memory value appearing at a lower level (closer to the server level) overrides any value inherited from a higher level. More specifically:

• A JVM memory value at the pool level of domain.xml overrides any values inherited from the corresponding <jvm> definition in host.xml.

• A JVM memory value at the server level of host.xml overrides every value inherited from the <jvm> section and also the pool level.

Definition of JVMs

JVMs can be defined and configured at all three levels using the EAP Management Console and the EAP CLI.

A JVM can be defined at the host controller level in the host.xml file within the <jvms> tag. A sample JVM definition in host.xml looks like this:

```
<jvms>
        <jvm name="small_jvm">
```

```
            <heap size="64m" max-size="128m"/> </jvm>
        <jvm
        name="production_jvm"> <heap
            size="2048m" max-size="2048m"/> <jvm-options>
            <option value="-
               server"/> </jvm-options> </jvm> </
            jvms>
```

The corresponding CLI command to define the JVMs is as follows:

```
/host=servera/jvm=small_jvm:add(heap-size=64m,max-heap-size=128m)
```

```
/host=servera/jvm=production_jvm:add( heap-
    size=2048m, max-
    heap-size=2048m, jvm-
    options=["-server"]
)
```

In the definition above, the JVM named small_jvm will have a minimum heap size of 64 MB and a maximum heap size of 128 MB. The JVM named production_jvm will have a minimum and maximum heap size of 2 GB, and an option named -server is set for the JVM. You can pass multiple JVM options this way.

use

Once you've defined a JVM, you can reference it in both the server-group definition and the server definition. Values specific to a given JVM can be overridden by the server-group or server definitions.

JVM Values in Server Groups

In a server-group definition in the domain.xml file, the <jvm> tag is used to specify which <jvm> definition to use. For example, the following server-group definition (in domain.xml) uses a JVM named production_jvm:

```
<server-group name="groupA" profile="default"> <jvm
    name="production_jvm"/> <socket-
    binding-group ref="standard-sockets"/> </server-group>
```

The corresponding CLI command is as follows:

```
/server-group=groupA:add \
      (profile=default,socket-binding-group=standard-sockets) \ /server-group=groupA/
      jvm=production_jvm:add()
```

Any server in the groupA server group will inherit the JVM settings from the production_jvm. A server-group can override the values of the <jvm> definition.

For example, the following server-group definition changes the heap size of the production_jvm:

```
<server-group name="groupB" profile="default"> <jvm
            name="production_jvm"> <heap
                   size="1024m" max-size="1024m"/> </jvm> <socket-
            binding-
            group ref="standard-sockets"/> </server-group>
```

Any server in server group groupB will have a minimum and maximum heap size of 1024 MB, and all other JVM settings are inherited from the production_jvm definition.

use

A new JVM configuration can be defined at the server group level without inheriting any values from a JVM defined at the host controller level. This new JVM can be referenced to a server level and can be overridden. For example, a new JVM named groupa-jvm can be defined at the server group level:

```
/server-group=groupA/jvm=groupa-jvm:add \
    (heap-size=512m,max-heap-size=512mm,jvm-options=["-server"])
```

JVM values on servers

JVM settings can be configured at the server level within the <server> definition in the host.xml file. A new <jvm> can be defined, which can also override any JVM values that the server inherits from its pool definition or its jvm definition at the host controller level. Here is a sample definition at the server level:

```
<server name="test_server" group="groupB" auto-start="true">
       <socket-binding-group ref="standard-sockets" port-offset="300"/>
    <jvm name="production_jvm"> <heap
            size="256m"/> </jvm> </
       server>
```

In the example above, test_server will use the values from production_jvm, only the initial heap size will be 256 MB instead of the 1024 MB size defined in server group groupB (which overrides the 2048 MB heap size of the definition of production_jvm at the host controller level.

use

You can also define a new JVM configuration at the server level without inheriting any values from a JVM defined at the host controller level or server group level. For example, a new JVM named serverX-jvm can be defined at the server level:

```
/host=host3/server-config=serverX \ /jvm=serverX-
    jvm:add \ (heap-size=512m,max-
    heap-size=1500m, \ jvm-options=["-server","-
    XX:ParallelGCThreads=4"])
```