Title: Spring boot is consuming too much RAM

Post Body:

I have created some services in spring boot, I have 11 fat jars and I deploy them in docker containers, my doubt was that every jar was consuming between 1 and 1.5 GB of RAM without any use, I check the RAM by running:

```
docker stats containername
```

At first I thought that it was the java container and I tried to change to one that uses alpine but nothing changed, so I think the only problem is my jar. Is there a way to change the RAM that the jar is using? Or this behavior is normal because every jar has an embedded tomcat? Or maybe is better to put some jars together and deploy them as war and use only one tomcat for a group of 'jars'? Can someone share his/her experience?,

Thanks in advance.

Accepted Answer:

You can set memory usage of docker container using `-e JAVA_OPTS='-Xmx64M -Xms64M'`.

docker file:

```
FROM openjdk:8-jre-alpine VOLUME ./mysql:/var/lib/mysql ADD /build/libs/application.jar app.jar ENTRYPOINT exec java $JAVA_OPT
```

image run:

```
 docker run -d --name container-name -p 9100:9100 -e JAVA_OPTS='-Xmx512M -Xms512M'   imagename:tag
```

Here i set 512Mb memory usage . you can set 1g or as per your requirement. After run using this check your memory usage. it will max 512Mb.

Highest Rated Answer:

This is how Java behaves in general. The JVM takes as much memory as you give it, and it will perform a process called **Garbage collection** (What is the garbage collector in Java) to free up space once it decides it should do so.

However, if you don't tell your JVM how much memory it can use, it will use the system defaults, which depend on your systems memory and the amount of cores you have. You can verify this using the following command (How is the default Java heap size determined):

```
java -XX:+PrintFlagsFinal -version | grep HeapSize
```

On my machine, that's an initial heap memory of 256MiB and a maximum heap size of 4GiB. However, that doesn't mean that your application needs it.

A good way of measuring your memory is by using a monitoring tool like jvisualvm. Additionally, you could use actuator's `/health` endpoint to see the heap memory usage as well.

Your heap memory usage will normally have a sawtooth pattern (Why a sawtooth shaped graph), where the memory is gradually being used, and eventually freed by the garbage collector.

The memory that is left over after a garbage collection are usually objects that cannot be destroyed because they're still in use. You could see this as your working memory. Now, to configure your `-Xmx` you'll have to see how your application behaves after trying it out:

- Configure it below your normal memory usage and your application will go out of memory, throwing an `OutOfMemoryError`.
- Configure it too low but above your minimal memory usage, and you will see a huge performance hit, due to the garbage collector continuously having to free memory.
- Configure it too high and you'll reserve memory you won't need in most of the cases, so wasting too much resources.

From the screenshot above, you can see that my application reserves about 1GiB of memory for heap usage, while it only uses about 30MiB after a garbage collection. That means that it has a way too high `-Xmx` value, so we could change it to different values and see how the application behaves.

People often prefer to work in powers of 2 (even though there is no limitation, as seen in jvm heap setting pattern). In my case, I need to go with at least 30MiB, since that's the amount of memory my application uses at all times. So that means I could try `-Xmx32m`, see how it performs, and adjust if it goes out of memory or performs worse.