We have seen how to set the heap memory via -Xms & -Xmx flags. For large server applications, it is recommended to grant as much memory as possible to the virtual machine. To avoid excessive page faults and thrashing ensure that maximum heap size is always smaller than the amount of memory installed on the machine. Below are few other flags that can be passed to JVM.

Permanent Generation

In pre-Java 8 world, method area was part of Permanent Generation or PermGen space. If there is insufficient memory allocated for PermGen, then we would encounter an error that says: **java.lang.OutOfMemoryError: Permgen space**.

The main cause for this error is that either ***too many classes or classes that are too large*** are loaded into the permgen area. One way to address this is to increase the permgen space, which can be done by setting MaxPermSize flag as follows:

java **-XX:MaxPermSize**=512m ClassName

Metaspace

From Java 8 onwards, method area is part of a new space called metaspace, which is part of the native memory. By default, metaspace allocation is limited by the amount of available native memory. For restricting this allocation, a new flag called MaxMetaspaceSize can be used as follows:

java -XX:MaxMetaspaceSize=512m ClassName

If we runout of metaspace and GC is of no help, then we would encounter an error that says: **java.lang.OutOfMemoryError: Metadata space**

Below is a reference that includes several command-line options that can be passed to affect the performance of JVM.

http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html

Below is also another good resource on setting of several different flags that you might find useful:

http://www.oracle.com/technetwork/java/javase/gc-tuning-6-140523.html

I may expand this document in future in which case you will be notified on the updates.