Rapport_Java Performance_TP1

- LIU Yuanyuan && GUO Xiaoqing

TP1 : Prise en main des outils de monitoring

Ce TP sert à comprendre le fonctionnement de différents outils permettant de monitorer et d'analyser une application JAVA. Vous pouvez trouver ci-dessous les applications de monitoring qu'on peut utiliser :

- **vmstat**
  - Fonction :
    Commande vmstat permet d'afficher des statistiques concernant la charge du système, en particulier l'utilisation de la mémoire virtuelle. Les données rapportées par vmstat proviennent d'une structure maintenue par le kernel et dépendent donc de l'Unix considéré. Les valeurs remontées sont des moyennes sur l'intervalle de mesure, ce qui peut produire un effet de lissage sur des intervalles trop longs.
  - Test :

```
administrateur@administrateur-virtual-machine:~$ vmstat
procs -----------mémoire---------- -échange- -----io---- -système- ------cpu-----
 r  b   swpd  libre tampon  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 125824 274980 2840636    0    0    72   132   93  138  2  2 94  1  0
```

```
administrateur@administrateur-virtual-machine:~$ vmstat 30
procs -----------mémoire---------- -échange- -----io---- -système- ------cpu-----
 r  b   swpd  libre tampon  cache   si   so    bi    bo   in   cs us sy id wa st
 6  0      0 4184184 277552 1226916    0    0    39    55  124  187  3  3 93  1  0
 1  0      0 4002216 277584 1226948    0    0     0     8  853  910  8 15 77  0  0
 1  0      0 3851120 277612 1226948    0    0     0     5  366  449  3  7 90  0  0
 1  0      0 3700088 277644 1226948    0    0     0     3  327  402  1  7 92  0  0
 1  0      0 3418112 277668 1226948    0    0     0     3  352  392  5 13 83  0  0
 0  0      0 3339184 277700 1226948    0    0     0     3  416  587  6  6 88  0  0
 0  0      0 3338440 277716 1226948    0    0     0     2  374  561  1  1 98  0  0
 0  0      0 3338440 277732 1226948    0    0     0     2  193  353  0  0 100  0  0
 0  0      0 3335588 277756 1226948    0    0     0    15  216  392  1  0 99  0  0
 0  0      0 3334580 277788 1226948    0    0     0     4  189  338  0  0 100  0  0
 0  1      0 3334580 277800 1226948    0    0     0     7  180  337  0  0 100  0  0
 0  0      0 3334456 277820 1226948    0    0     0     2  181  336  0  0 100  0  0
 0  0      0 3334456 277836 1226948    0    0     0     2  176  329  0  0 100  0  0
 0  0      0 3334456 277852 1226948    0    0     0     2  179  333  0  0 100  0  0
 0  0      0 3334456 277876 1226948    0    0     0     2  176  329  0  0 100  0  0
 0  0      0 3334456 277900 1226948    0    0     0     2  174  326  0  0 100  0  0
 0  0      0 3334456 277916 1226948    0    0     0     1  176  329  0  0 100  0  0
 0  0      0 3334456 277940 1226948    0    0     0     2  175  329  0  0 100  0  0
 0  0      0 3334456 277972 1226948    0    0     0     2  164  306  0  0 100  0  0
 3  1      0 3334332 278000 1226948    0    0     0     3  397  577  1  1 98  0  0
 0  0      0 3334332 278028 1226948    0    0     0     3  482  697  1  1 98  0  0
procs -----------mémoire---------- -échange- -----io---- -système- ------cpu-----
 r  b   swpd  libre tampon  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 3334332 278044 1226948    0    0     0     2  176  330  0  0 100  0  0
 0  0      0 3334332 278060 1226940    0    0     0     2  175  326  0  0 100  0  0
 0  1      0 3334332 278076 1226940    0    0     0     1  177  333  0  0 100  0  0
 0  0      0 3334332 278084 1226948    0    0     0     1  172  324  0  0 100  0  0
 0  0      0 3334208 278108 1226948    0    0     0     2  175  330  0  0 100  0  0
 0  0      0 3334208 278132 1226948    0    0     0     2  176  330  0  0 100  0  0
 0  0      0 3334208 278148 1226948    0    0     0     2  175  328  0  0 100  0  0
 0  0      0 3330488 278172 1226948    0    0     0    14  340  502  1  1 97  0  0
 0  0      0 3329984 278212 1226948    0    0     0     5  366  541  2  1 97  0  0
 0  0      0 3329976 278236 1226940    0    0     0     7  401  594  1  1 98  0  0
 0  0      0 3329976 278260 1226948    0    0     0     2  178  331  0  0 100  0  0
 0  0      0 3329976 278276 1226948    0    0     0     1  172  324  0  0 100  0  0
 1  0      0 3329852 278292 1226948    0    0     0     1  179  332  0  0 100  0  0
 0  0      0 3329852 278316 1226948    0    0     0     2  175  326  0  0 100  0  0
 0  0      0 3329852 278340 1226948    0    0     0     4  178  329  0  0 100  0  0
 0  0      0 3329852 278364 1226948    0    0     0     3  200  356  0  0 100  0  0
 0  0      0 3329852 278388 1226948    0    0     0     2  255  428  0  0 100  0  0
 0  0      0 3329108 278412 1226948    0    0     0    15  361  542  1  1 98  0  0
 0  0      0 3329108 278460 1226940    0    0     0     6  175  327  0  0 100  0  0
 0  0      0 3329108 278500 1226948    0    0     0     6  200  357  0  0 100  0  0
 2  0      0 3328984 278548 1226944    0    0     0     6  286  462  0  0 99  0  0
```

La signification de chaque paramètre :

- %sys: consommation en mode système,
- %usr: consommation en mode utilisateur,
- %idle: pourcentage de temps CPU non consommé.
- b: nombre de processus "bloqués",
- r: nombre de processus dans la run queue,
- w: nombre de processus en "wait".
- Des informations concernant l'activité de la mémoire virtuelle:
- free: nombre de pages mémoires disponibles. Suivant les Unix, les pages allouées au cache du système de fichiers peuvent ou non être incluses dans cette valeur.
- avm: active virtual memory, nombre de pages mémoires actives au cours d'un intervalle dépendant du système.
- pi: page in, nombre de pages par seconde chargées en mémoire depuis le disque, lors du lancement d'un processus par exemple.

- po: page out, nombre de pages par seconde écrites sur le disque depuis la mémoire, parfois appelé swap.
- sr: nombre de pages par seconde scannées par le daemon de libération de pages.
- fr: nombre de pages par seconde libérées par le daemon de libération de pages.

- **iostat**
  - Fonction **:**
    iostat est utilisé pour collecter et afficher les statistiques d'entrée et de sortie de stockage du système d'exploitation. Il est souvent utilisé pour identifier les problèmes de performances avec les périphériques de stockage, y compris les disques locaux ou les disques distants accessibles via des systèmes de fichiers réseau tels que NFS. Il peut également être utilisé pour fournir des informations sur l'entrée et la sortie du terminal (TTY), et comprend également des informations de base sur le processeur.
  - Test : [Afficher toutes les conditions de charge du dispositif]

```
administrateur@administrateur-virtual-machine:~$ iostat
Linux 4.18.0-15-generic (administrateur-virtual-machine)        01/12/2020      _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2,55    0,36    2,49    0,74    0,00   93,85

Device             tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
loop0             0,00         0,00         0,00         46          0
loop1             1,74         1,81         0,00      30953          0
loop2             0,01         0,03         0,00        498          0
loop3             0,03         0,05         0,00        835          0
loop4             0,00         0,01         0,00        128          0
loop5             1,01         1,09         0,00      18604          0
loop6             0,02         0,08         0,00       1435          0
loop7             0,00         0,06         0,00       1081          0
nvme0n1           5,15       115,79       183,24    1984301    3140300
loop8             0,00         0,06         0,00       1057          0
loop9             0,01         0,08         0,00       1411          0
loop10            0,62         0,68         0,00      11673          0
loop11            0,00         0,02         0,00        330          0
loop12            0,00         0,01         0,00        113          0
loop13            0,02         0,08         0,00       1304          0
loop14            0,01         0,13         0,00       2229          0
loop15            7,78         7,85         0,00     134595          0
loop16            0,00         0,07         0,00       1131          0
loop17            0,01         0,07         0,00       1171          0
loop18            0,00         0,01         0,00        116          0
loop19            0,00         0,02         0,00        339          0
loop20            0,15         0,17         0,00       2857          0
loop21            0,00         0,00         0,00         52          0
loop22            0,00         0,01         0,00        140          0
loop23            0,01         0,01         0,00        236          0
loop24            0,01         0,07         0,00       1177          0
loop25            0,03         0,09         0,00       1588          0
loop26            0,01         0,07         0,00       1139          0
```

Les significations des valeurs d'attribut du cpu:

- % utilisateur: pourcentage de temps pendant lequel la CPU est en mode utilisateur.
- % nice: Le pourcentage de temps pendant lequel le processeur est en mode utilisateur avec une valeur NICE.
- % système: pourcentage de temps pendant lequel la CPU est en mode système.
- % iowait: pourcentage de temps pendant lequel le processeur attend que l'entrée et la sortie se terminent.

- % voler: pourcentage de temps d'attente inconscient du processeur virtuel lorsque l'hyperviseur gère un autre processeur virtuel.
- % inactif: pourcentage du temps d'inactivité du processeur.

Les significations des valeurs d'attribut du disk:

- rrqm/s: nombre d'opérations de lecture de fusion par seconde. C.-à-d. rmerge/s
- wrqm/s: le nombre d'opérations d'écriture de fusion par seconde. C'est-à-dire wmerge/s
- r/s: le nombre de périphériques d'E/S de lecture achevés par seconde. C'est-à-dire rio/s
- w/s: nombre de périphériques d'E/S d'écriture terminés par seconde. C'est-à-dire wio/s
- rsec/s: Le nombre de secteurs lus par seconde. C'est-à-dire rsect/s
- wsec/s: le nombre de secteurs écrits par seconde. C'est-à-dire wsect/s
- rkB/s: le nombre de K octets lus par seconde. C'est la moitié de rsect/s car chaque secteur a une taille de 512 octets.
- wkB/s: le nombre de K octets écrits par seconde. C'est la moitié de wsect/s.
- avgrq-sz: taille moyenne des données (secteur) de chaque opération d'E/S de périphérique.
- avgqu-sz: longueur moyenne de la file d'attente d'E/S.
- wait: temps d'attente moyen (millisecondes) pour chaque opération d'E/S de périphérique.
- svctm: temps de service moyen (millisecondes) pour chaque opération d'E/S de périphérique.
- % util: quel pourcentage de seconde est utilisé pour les opérations d'E/S, c'est-à-dire le pourcentage de processeur consommé par io

- **nicstat**
  - Fonction :
    nicstat permets d'imprimer les statistiques réseau pour toutes les cartes réseau (NIC), y compris les paquets, kilo-octets par seconde, paquet moyen.
  - Test :

```
administrateur@administrateur-virtual-machine:~$ nicstat
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util     Sat
11:53:33    ens33   85.73    0.26   58.81    3.39  1492.8   78.13  0.07   0.00
11:53:33       lo    0.05    0.05    0.15    0.15   361.6   361.6  0.00   0.00
```

- **jcmd**
  - Fonction :
    jcmd envoie des requêtes de commande de diagnostic à une machine virtuelle Java (JVM) en cours d'exécution (Vous pouvez l'utiliser pour exporter le tas, afficher le processus Java, exporter les informations de thread, effectuer GC et effectuer une analyse d'échantillonnage).
  - Test :

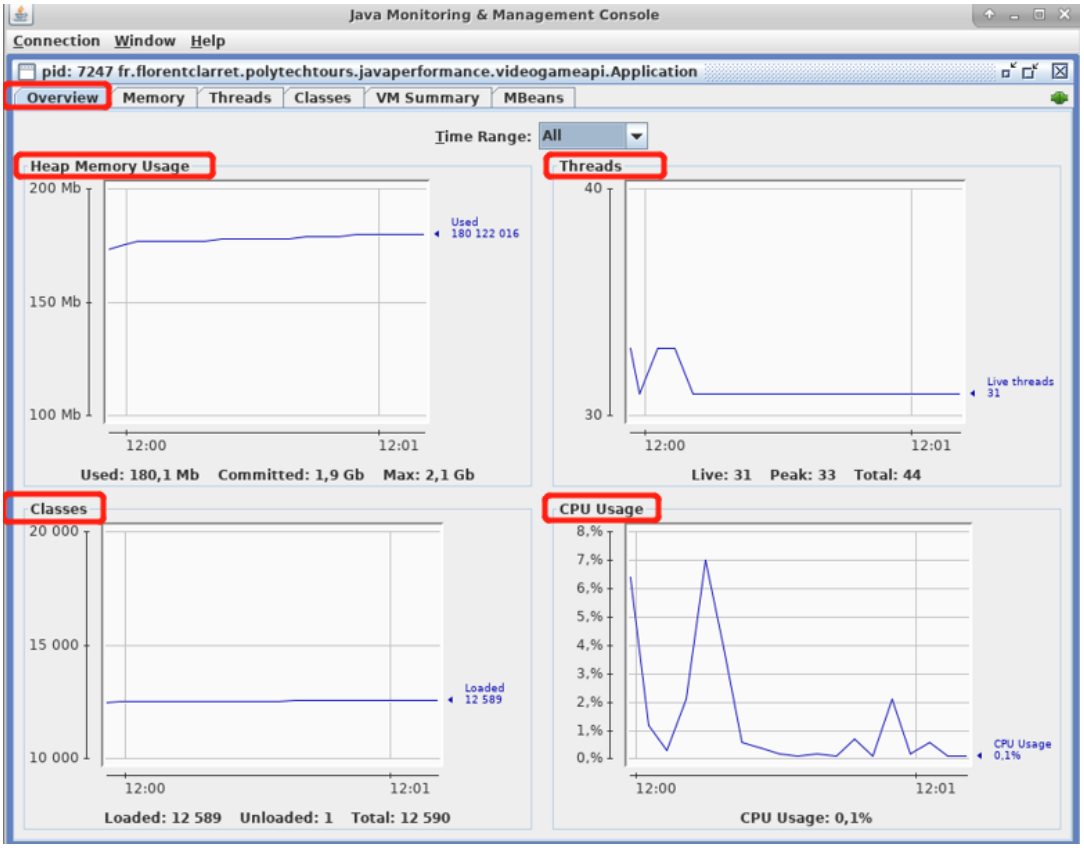
```
administrateur@administrateur-virtual-machine:~$ jcmd
7334 jdk.jcmd/sun.tools.jcmd.JCmd
7051 /snap/eclipse/48//plugins/org.eclipse.equinox.launcher_1.5.600.v20191014-2022.jar -os linux -ws gtk -arch
x86_64 -showsplash /snap/eclipse/48//plugins/org.eclipse.epp.package.common_4.14.0.20191212-1200/splash.bmp -la
uncher /snap/eclipse/48/eclipse -name Eclipse --launcher.library /snap/eclipse/48//plugins/org.eclipse.equinox.
launcher.gtk.linux.x86_64_1.1.1100.v20190907-0426/eclipse_1801.so -startup /snap/eclipse/48//plugins/org.eclips
e.equinox.launcher_1.5.600.v20191014-2022.jar --launcher.appendVmargs -exitdata 72000a -product org.eclipse.epp
.package.java.product -vm /usr/bin/java -vmargs -Dosgi.requiredJavaVersion=1.8 -Dosgi.instance.area.default=@us
er.home/eclipse-workspace -XX:+UseG1GC -XX:+UseStringDeduplication --add-modules=ALL-SYSTEM -Dosgi.requiredJava
Version=1.8 -Dosgi.dataAreaRequiresExplicitInit=true -Xms256m -Xmx1024m --add-modules=ALL-SYSTEM -jar /snap/ecl
ipse/48//plugins/org.eclipse.equinox.launcher_1.5.600.v20191014-2022.jar
7247 fr.florentclarret.polytechtours.javaperformance.videogameapi.Application
administrateur@administrateur-virtual-machine:~$ jcmd -l
7051 /snap/eclipse/48//plugins/org.eclipse.equinox.launcher_1.5.600.v20191014-2022.jar -os linux -ws gtk -arch
x86_64 -showsplash /snap/eclipse/48//plugins/org.eclipse.epp.package.common_4.14.0.20191212-1200/splash.bmp -la
uncher /snap/eclipse/48/eclipse -name Eclipse --launcher.library /snap/eclipse/48//plugins/org.eclipse.equinox.
launcher.gtk.linux.x86_64_1.1.1100.v20190907-0426/eclipse_1801.so -startup /snap/eclipse/48//plugins/org.eclips
e.equinox.launcher_1.5.600.v20191014-2022.jar --launcher.appendVmargs -exitdata 72000a -product org.eclipse.epp
.package.java.product -vm /usr/bin/java -vmargs -Dosgi.requiredJavaVersion=1.8 -Dosgi.instance.area.default=@us
er.home/eclipse-workspace -XX:+UseG1GC -XX:+UseStringDeduplication --add-modules=ALL-SYSTEM -Dosgi.requiredJava
Version=1.8 -Dosgi.dataAreaRequiresExplicitInit=true -Xms256m -Xmx1024m --add-modules=ALL-SYSTEM -jar /snap/ecl
ipse/48//plugins/org.eclipse.equinox.launcher_1.5.600.v20191014-2022.jar
7372 jdk.jcmd/sun.tools.jcmd.JCmd -l
7247 fr.florentclarret.polytechtours.javaperformance.videogameapi.Application
administrateur@administrateur-virtual-machine:~$ jcmd 7051 -l
7051:
java.lang.IllegalArgumentException: Unknown diagnostic command
administrateur@administrateur-virtual-machine:~$ jcmd 7372 -l
7372:
java.io.IOException: Aucun processus de ce type
        at jdk.attach/sun.tools.attach.VirtualMachineImpl.sendQuitTo(Native Method)
        at jdk.attach/sun.tools.attach.VirtualMachineImpl.<init>(VirtualMachineImpl.java:78)
        at jdk.attach/sun.tools.attach.AttachProviderImpl.attachVirtualMachine(AttachProviderImpl.java:58)
        at jdk.attach/com.sun.tools.attach.VirtualMachine.attach(VirtualMachine.java:207)
        at jdk.jcmd/sun.tools.jcmd.JCmd.executeCommandForPid(JCmd.java:114)
        at jdk.jcmd/sun.tools.jcmd.JCmd.main(JCmd.java:98)
administrateur@administrateur-virtual-machine:~$ jcmd 7247 -l
7247:
java.lang.IllegalArgumentException: Unknown diagnostic command
```

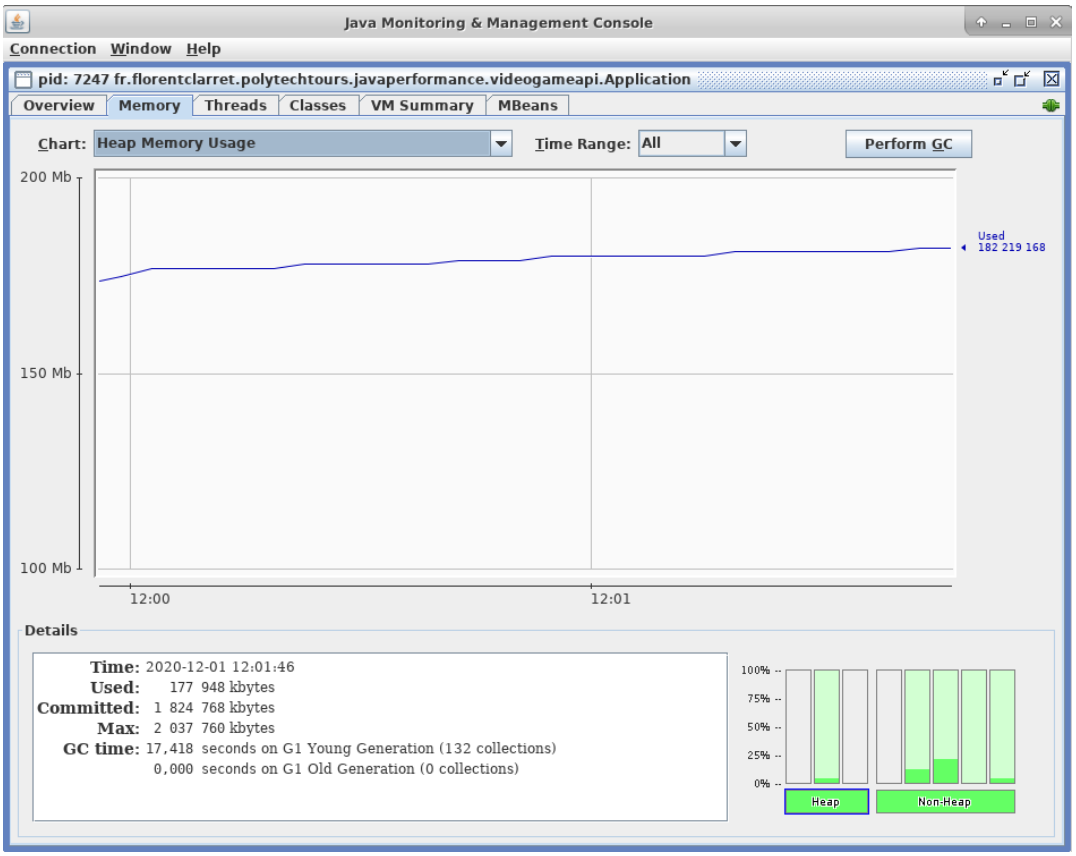
jcmd -l :

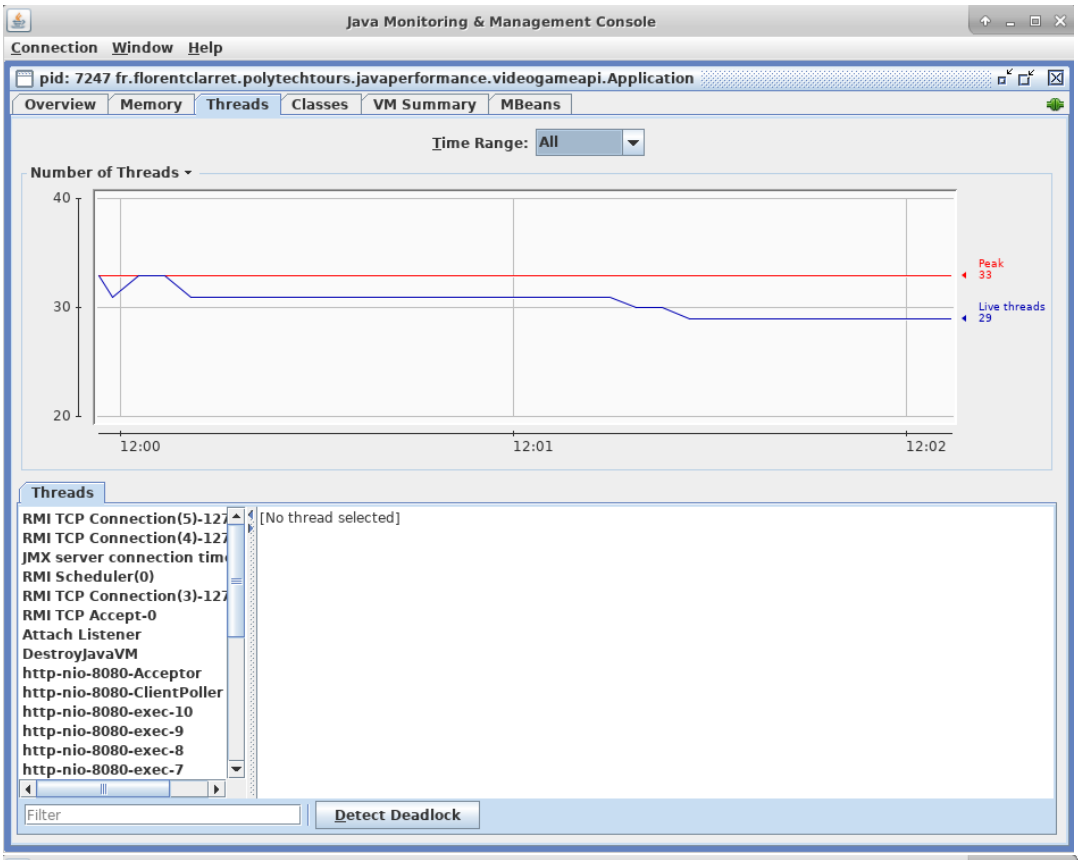
- Affiche la liste des identificateurs de processus de la machine virtuelle Java qui ne sont pas exécutés dans un processus docker distinct, ainsi que la classe principale et les arguments de ligne de commande qui ont été utilisés pour lancer le processus. Si la JVM est dans un processus docker, vous devez utiliser des outils tels que ps pour rechercher le PID.

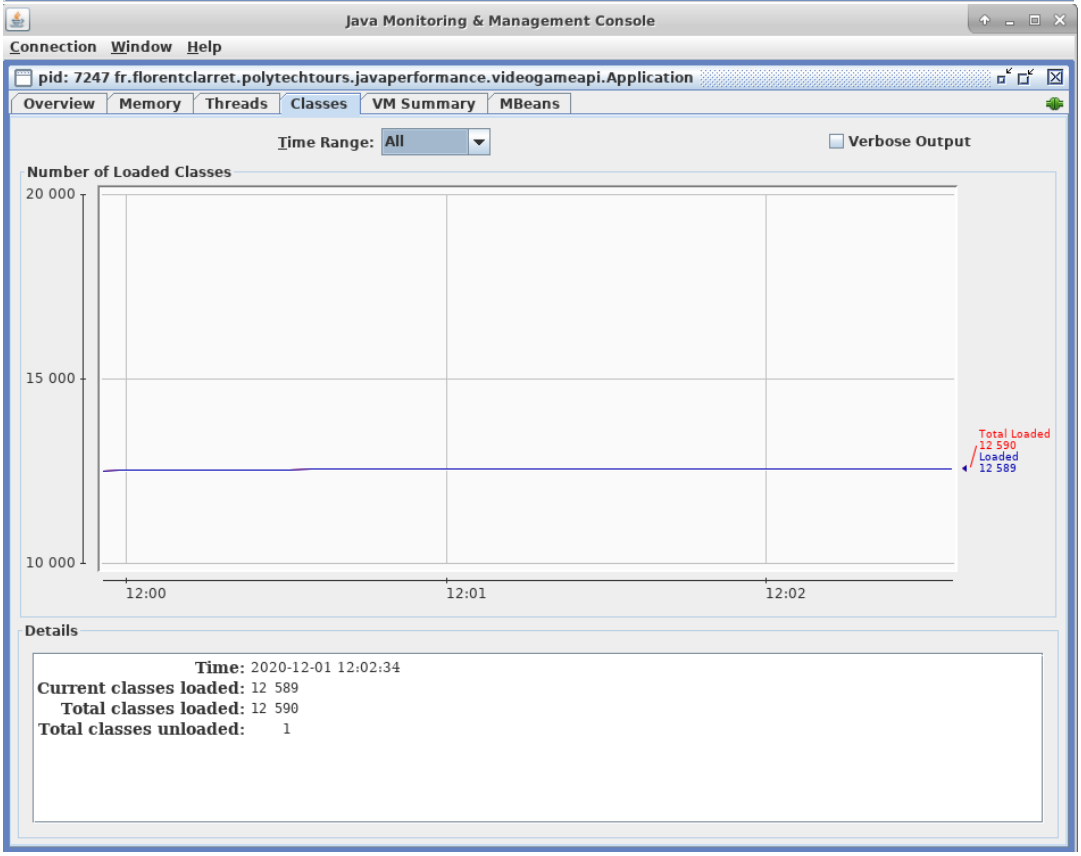
- **jconsole**
  - Fonction :

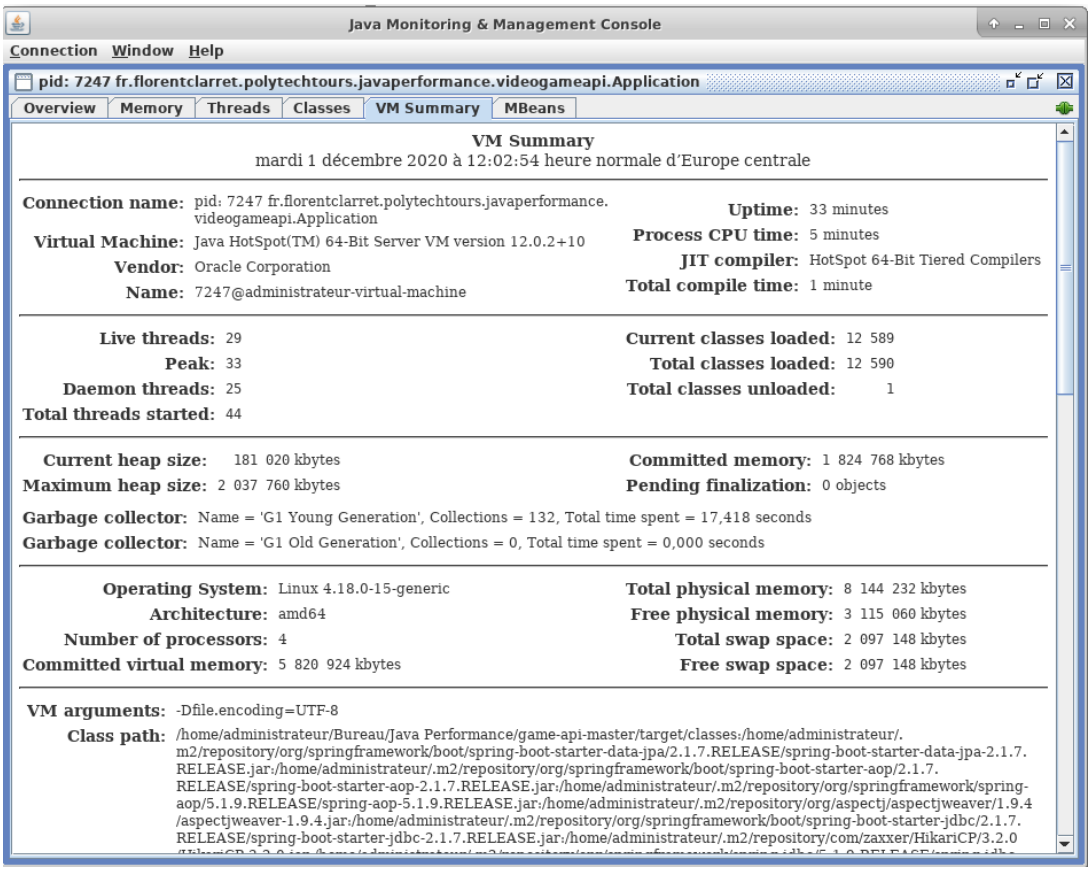
    JConsole (Java™ Monitoring and Management Console) est un outil graphique qui permet à l'utilisateur de contrôler et de gérer le comportement des applications Java. Lorsque JConsole se connecte à une application Java, il affiche des informations la concernant. Ces informations comprennent l'utilisation de la mémoire, les unités d'exécution en cours de fonctionnement et les classes chargées. Ces données vous aident à surveiller le comportement de votre application et de la JVM. Elles sont utiles à la compréhension des problèmes de performances, d'utilisation de la mémoire, de blocage ou d'interblocage.
  - Test :


Java Monitoring & Management Console
Connection Window Help
pid: 7247 fr.florentclarret.polytechtours.javaperformance.videogameapi.Application
Overview Memory Threads Classes VM Summary MBeans
Time Range: All
Heap Memory Usage
200 Mb
150 Mb
100 Mb
Used
180 122 016
12:00
12:01
Used: 180,1 Mb Committed: 1,9 Gb Max: 2,1 Gb
Threads
40
30
Live threads
31
Live: 31 Peak: 33 Total: 44
Classes
20 000
15 000
10 000
Loaded
12 589
Loaded: 12 589 Unloaded: 1 Total: 12 590
CPU Usage
8,%
7,%
6,%
5,%
4,%
3,%
2,%
1,%
0,%
CPU Usage
0,1%
CPU Usage: 0,1%


Java Monitoring & Management Console
Connection Window Help
pid: 7247 fr.florentclarret.polytechtours.javaperformance.videogameapi.Application
Overview Memory Threads Classes VM Summary MBeans
Chart: Heap Memory Usage
Time Range: All
Perform GC
200 Mb
150 Mb
100 Mb
Used
182 219 168
12:00
12:01
Details
Time: 2020-12-01 12:01:46
Used: 177 948 kbytes
Committed: 1 824 768 kbytes
Max: 2 037 760 kbytes
GC time: 17,418 seconds on G1 Young Generation (132 collections)
0,000 seconds on G1 Old Generation (0 collections)
100%
75%
50%
25%
0%
Heap
Non-Heap


Java Monitoring & Management Console
Connection Window Help
pid: 7247 fr.florentclarret.polytechtours.javaperformance.videogameapi.Application
Overview Memory Threads Classes VM Summary MBeans
Time Range: All
Number of Threads
40
30
20
12:00
12:01
12:02
Peak 33
Live threads 29
Threads
RMI TCP Connection(5)-127
RMI TCP Connection(4)-127
JMX server connection tim
RMI Scheduler(0)
RMI TCP Connection(3)-127
RMI TCP Accept-0
Attach Listener
DestroyJavaVM
http-nio-8080-Acceptor
http-nio-8080-ClientPoller
http-nio-8080-exec-10
http-nio-8080-exec-9
http-nio-8080-exec-8
http-nio-8080-exec-7
[No thread selected]
Filter
Detect Deadlock


Java Monitoring & Management Console
Connection Window Help
pid: 7247 fr.florentclarret.polytechtours.javaperformance.videogameapi.Application
Overview Memory Threads Classes VM Summary MBeans
Time Range: All
Verbose Output
Number of Loaded Classes
20 000
15 000
10 000
12:00
12:01
12:02
Total Loaded 12 590
Loaded 12 589
Details
Time: 2020-12-01 12:02:34
Current classes loaded: 12 589
Total classes loaded: 12 590
Total classes unloaded: 1



- **jstack**
  - Fonction :

    jstack est l'outil de trace de pile Java fourni avec la JVM, qui est utilisé pour générer un instantané de thread de la machine virtuelle au moment actuel et pour imprimer les informations de pile Java de l'ID de processus java, du fichier principal et du service de débogage à distance.
  - Test :

```
administrateur@administrateur-virtual-machine:~$ jstack -l 7247
2020-12-01 12:04:27
Full thread dump Java HotSpot(TM) 64-Bit Server VM (12.0.2+10 mixed mode, sharing):

Threads class SMR info:
_java_thread_list=0x00007fc0f0001b00, length=33, elements={
0x00007fc140121800, 0x00007fc140123800, 0x00007fc140128800, 0x00007fc14012b000,
0x00007fc14012d000, 0x00007fc14012f000, 0x00007fc140176800, 0x00007fc14017c000,
0x00007fc140e5c000, 0x00007fc0d0794800, 0x00007fc140e5f000, 0x00007fc140f94800,
0x00007fc141528800, 0x00007fc141529800, 0x00007fc14152b000, 0x00007fc14152d000,
0x00007fc14152e800, 0x00007fc141530800, 0x00007fc141532800, 0x00007fc141534000,
0x00007fc141536000, 0x00007fc141563800, 0x00007fc141565800, 0x00007fc141568000,
0x00007fc14156f000, 0x00007fc140014800, 0x00007fc100001000, 0x00007fc0c010e000,
0x00007fc0bc002000, 0x00007fc0d07ee800, 0x00007fc0d0034000, 0x00007fc0bc005800,
0x00007fc0bc007800
}

"Reference Handler" #2 daemon prio=10 os_prio=0 cpu=4,84ms elapsed=2110,26s tid=0x00007fc140121800 nid=0x1c5d waiting on condition  [0x00007fc120d90000]
   java.lang.Thread.State: RUNNABLE
        at java.lang.ref.Reference.waitForReferencePendingList(java.base@12.0.2/Native Method)
        at java.lang.ref.Reference.processPendingReferences(java.base@12.0.2/Reference.java:241)
        at java.lang.ref.Reference$ReferenceHandler.run(java.base@12.0.2/Reference.java:213)

   Locked ownable synchronizers:
        - None

"Finalizer" #3 daemon prio=8 os_prio=0 cpu=15,37ms elapsed=2110,26s tid=0x00007fc140123800 nid=0x1c5e in Object.wait()  [0x00007fc120c8f000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(java.base@12.0.2/Native Method)
        - waiting on <0x0000000083d84f58> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@12.0.2/ReferenceQueue.java:155)
        - locked <0x0000000083d84f58> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(java.base@12.0.2/ReferenceQueue.java:176)
        at java.lang.ref.Finalizer$FinalizerThread.run(java.base@12.0.2/Finalizer.java:170)

   Locked ownable synchronizers:
        - None

"Signal Dispatcher" #4 daemon prio=9 os_prio=0 cpu=52,32ms elapsed=2110,12s tid=0x00007fc140128800 nid=0x1c5f runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"C2 CompilerThread0" #5 daemon prio=9 os_prio=0 cpu=40446,63ms elapsed=2110,12s tid=0x00007fc14012b000 nid=0x1c60 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

   Locked ownable synchronizers:
        - None

"C1 CompilerThread0" #7 daemon prio=9 os_prio=0 cpu=15606,03ms elapsed=2110,12s tid=0x00007fc14012d000 nid=0x1c61 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task
```

… … …

```
        - parking to wait for  <0x00000000ef13ebc8> (a java.util.concurrent.SynchronousQueue$TransferStack)
        at java.util.concurrent.locks.LockSupport.parkNanos(java.base@12.0.2/LockSupport.java:235)
        at java.util.concurrent.SynchronousQueue$TransferStack.awaitFulfill(java.base@12.0.2/SynchronousQueue.java:462)
        at java.util.concurrent.SynchronousQueue$TransferStack.transfer(java.base@12.0.2/SynchronousQueue.java:361)
        at java.util.concurrent.SynchronousQueue.poll(java.base@12.0.2/SynchronousQueue.java:937)
        at java.util.concurrent.ThreadPoolExecutor.getTask(java.base@12.0.2/ThreadPoolExecutor.java:1053)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@12.0.2/ThreadPoolExecutor.java:1114)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@12.0.2/ThreadPoolExecutor.java:628)
        at java.lang.Thread.run(java.base@12.0.2/Thread.java:835)

"VM Thread" os_prio=0 cpu=13818,80ms elapsed=2126,84s tid=0x00007fc14011e800 nid=0x1c5c runnable

"GC Thread#0" os_prio=0 cpu=12286,16ms elapsed=2127,59s tid=0x00007fc140064000 nid=0x1c57 runnable

"GC Thread#1" os_prio=0 cpu=12161,22ms elapsed=2118,36s tid=0x00007fc104001000 nid=0x1c68 runnable

"GC Thread#2" os_prio=0 cpu=11777,26ms elapsed=2118,36s tid=0x00007fc104002000 nid=0x1c69 runnable

"GC Thread#3" os_prio=0 cpu=10825,12ms elapsed=2115,76s tid=0x00007fc104004800 nid=0x1c6b runnable

"G1 Main Marker" os_prio=0 cpu=13,35ms elapsed=2127,56s tid=0x00007fc140072000 nid=0x1c58 runnable

"G1 Conc#0" os_prio=0 cpu=3696,81ms elapsed=2127,56s tid=0x00007fc140074000 nid=0x1c59 runnable

"G1 Refine#0" os_prio=0 cpu=596,20ms elapsed=2127,22s tid=0x00007fc1400ed800 nid=0x1c5a runnable

"G1 Young RemSet Sampling" os_prio=0 cpu=985,33ms elapsed=2127,21s tid=0x00007fc1400ef800 nid=0x1c5b runnable
"VM Periodic Task Thread" os_prio=0 cpu=1881,98ms elapsed=2126,15s tid=0x00007fc140179000 nid=0x1c64 waiting on condition

JNI global refs: 29, weak refs: 0
```

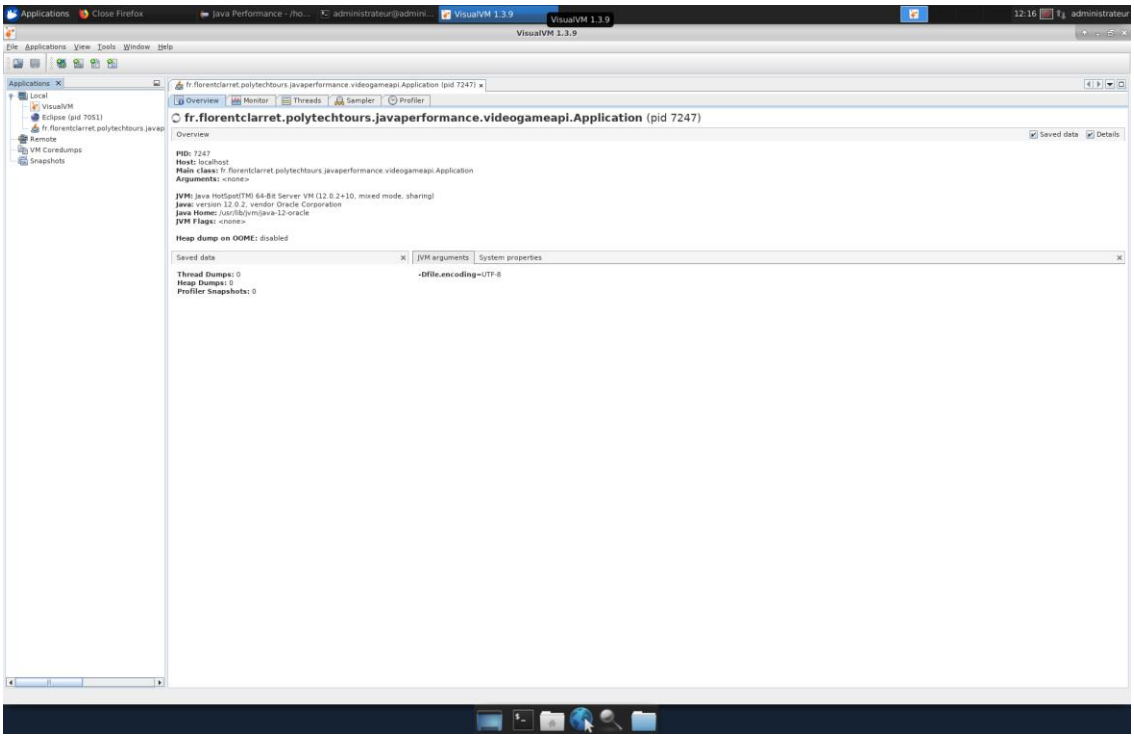
jstack -l: ➔ jstack [option] pid

- Liste en détail. Imprime des informations supplémentaires sur les verrous, telles que la liste des synchroniseurs possédables java.util.concurrent.

- **VisualVM**
  - Fonction :

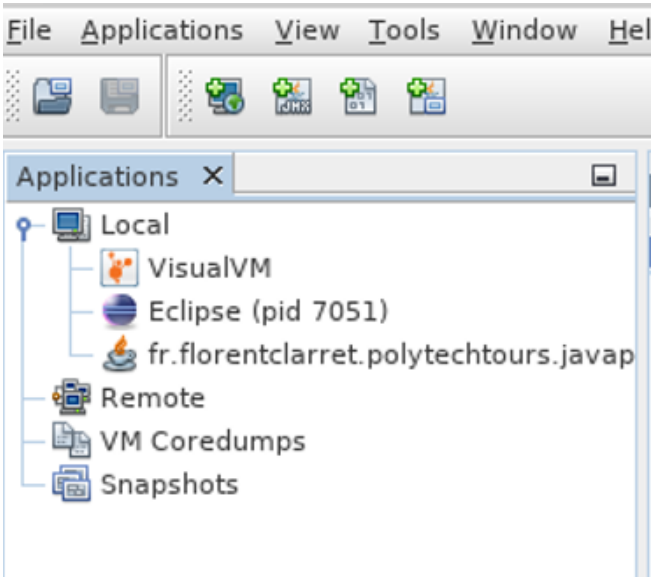
VisualVM monitors and troubleshoots applications running on Java 1.4+ from many vendors using various technologies including jvmstat, JMX, Serviceability Agent (SA) and Attach API.

- Test:



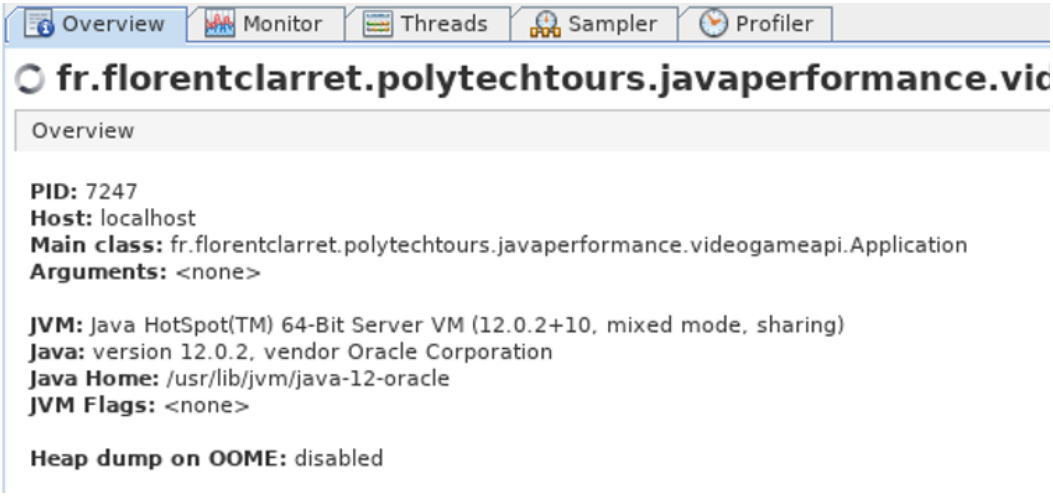
Afficher les processus Java locaux et distants

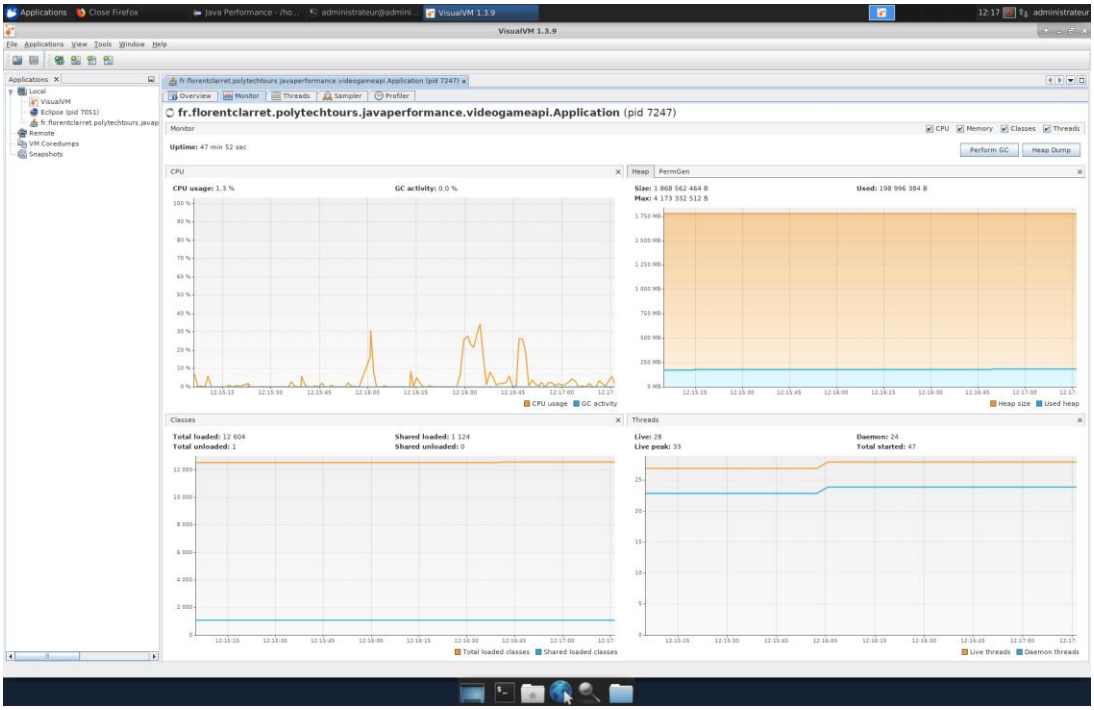
- VisualVM détecte et répertorie automatiquement les applications Java exécutées localement et à distance (jstatd doit être exécuté sur l'hôte distant). Vous pouvez également définir des applications manuellement par connexion JMX.



Afficher la configuration et l'environnement du processus

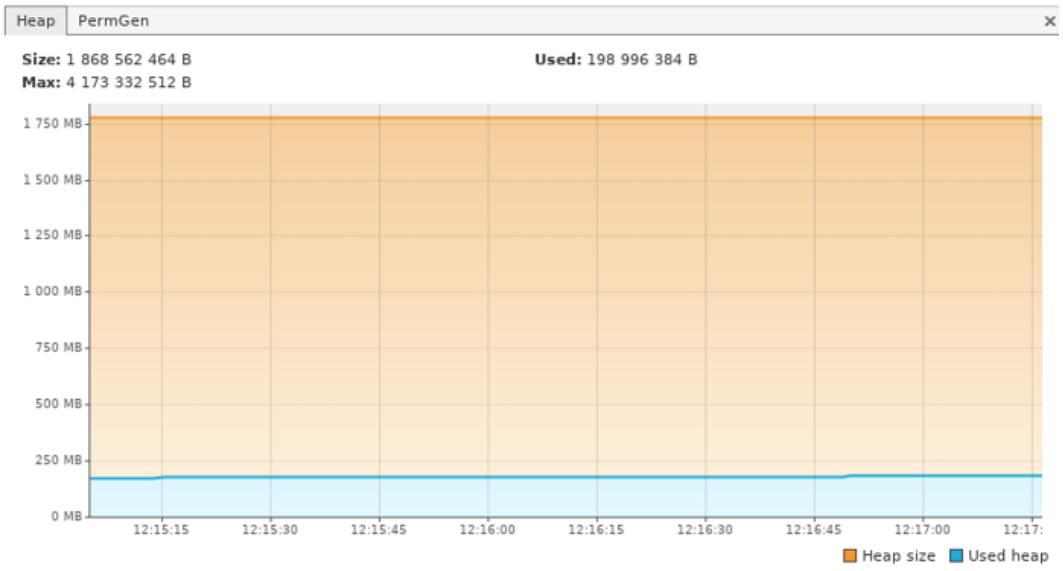
- Pour chaque processus, VisualVM affiche les informations d'exécution de base: PID, classe principale, arguments passés au processus java, version JVM, accueil JDK, indicateurs et arguments JVM et propriétés système.





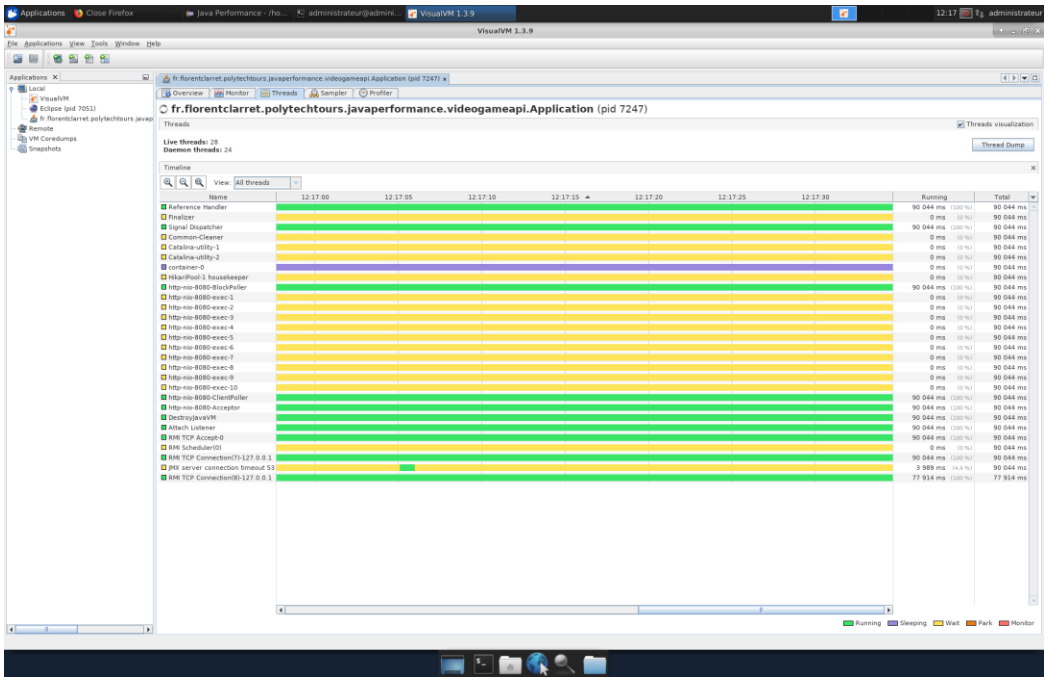
Surveiller les performances des processus et la mémoire

- VisualVM surveille l'utilisation du processeur de l'application, l'activité du GC, le tas et la mémoire de génération de méta-espace / permanente, le nombre de classes chargées et les threads en cours d'exécution.



Visualiser les threads de processus

- Tous les threads en cours d'exécution dans un processus Java sont affichés dans une chronologie avec les temps cumulés d'exécution, de mise en veille, d'attente, de parcage et de surveillance.

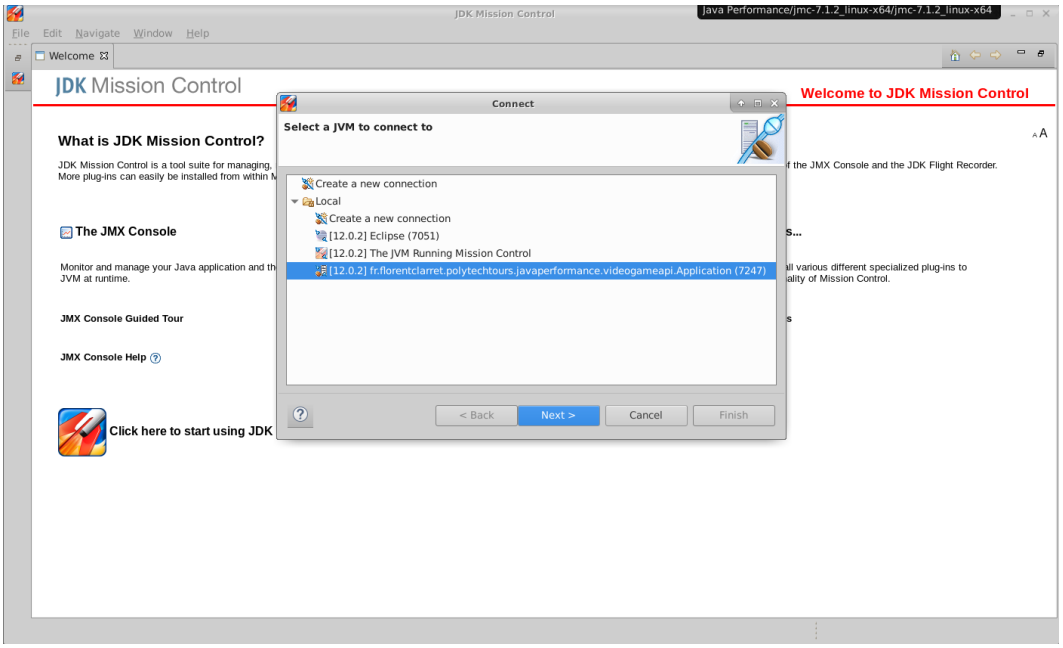


- **JMC (JDK Mission Control)**
  - Fonction :

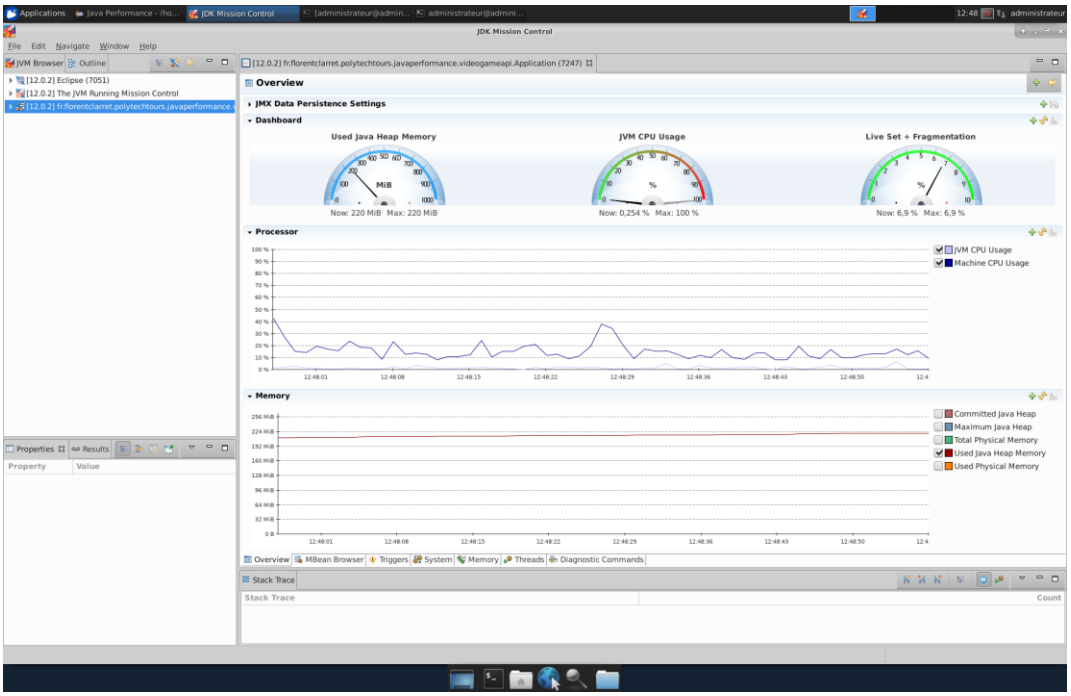
    JDK Mission Control est une suite d'outils open source pour la machine virtuelle Java. Les outils aident à détecter les problèmes et à optimiser les programmes exécutés sur la machine

virtuelle Java en production. JDK Mission Control prend en charge OpenJDK 11 (et supérieur) et Oracle JDK 7u40 (et supérieur).

- Test :



Console JMX :

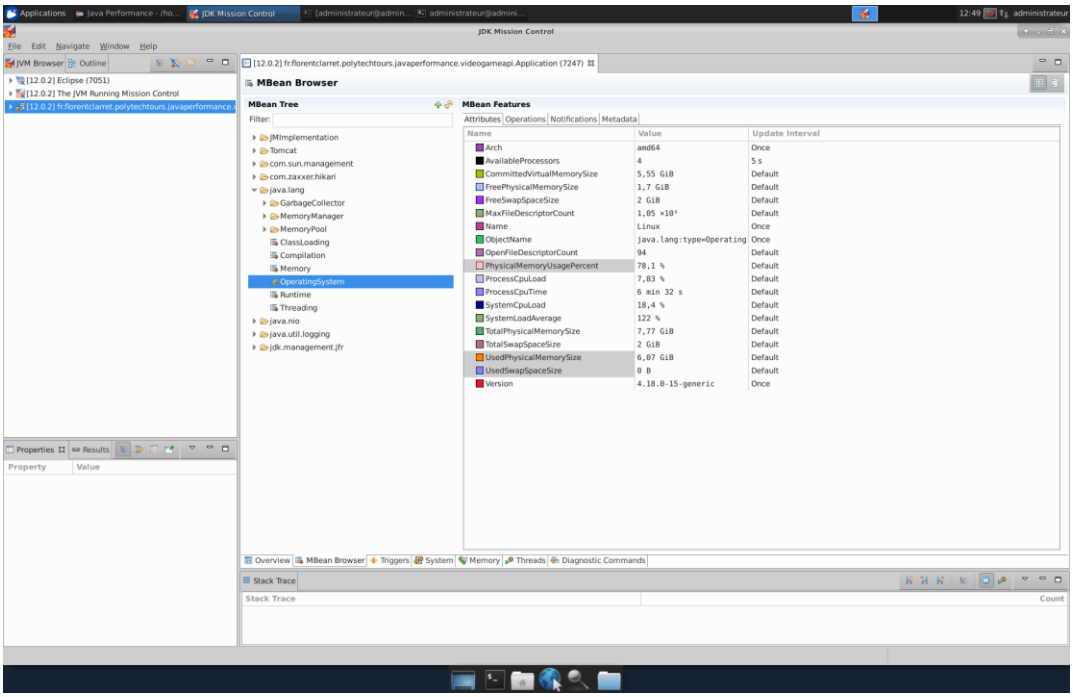


Analyse :

- La " Dashboard " partie montre l'utilisation du processeur et l'utilisation du tas de mémoire. Au-dessus, il y a une chronologie indiquant l'ordre des événements et les barres verticales représentent les événements. Vous pouvez zoomer sur la partie intéressante de la chronologie pour une analyse détaillée.

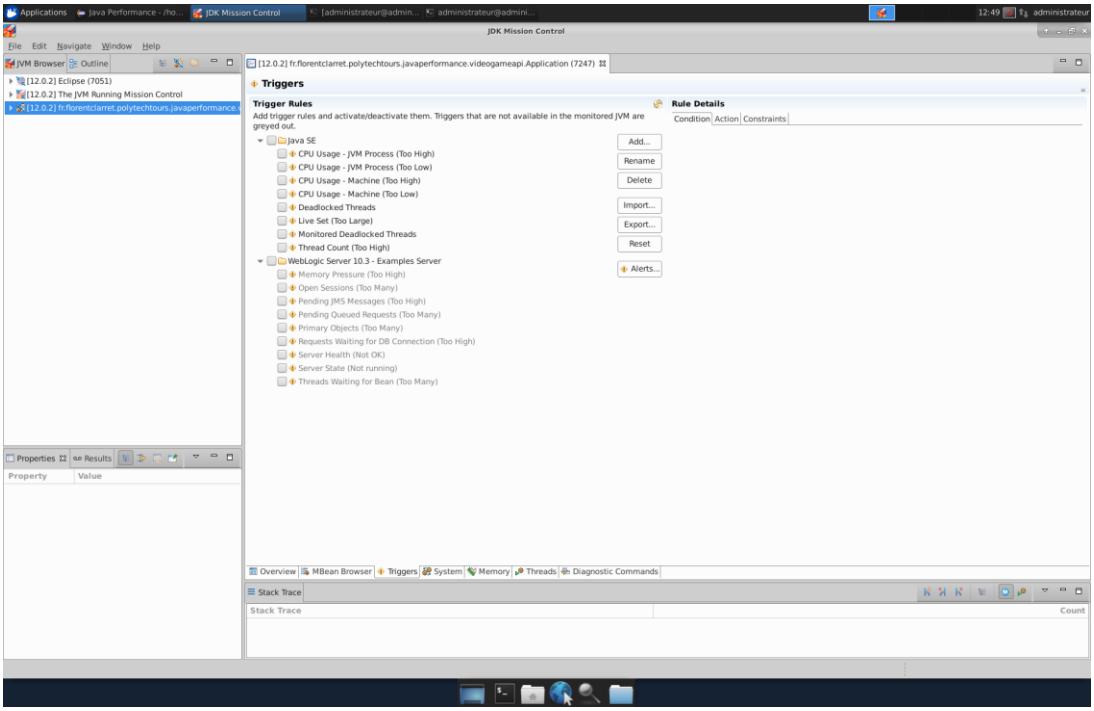
- La partie courbe suivante:
  La courbe d'utilisation du processeur montre clairement que l'utilisation du processeur testée a atteint 10% et que l'utilisation de la mémoire était d'environ 210 Mo. En bas, vous avez le choix entre des balises, des balises de propriété système, des balises d'informations JVM, etc. Les boutons sur le côté gauche de l'écran fournissent un statut d'exécution plus détaillé de l'application.

Navigateur Mbean :



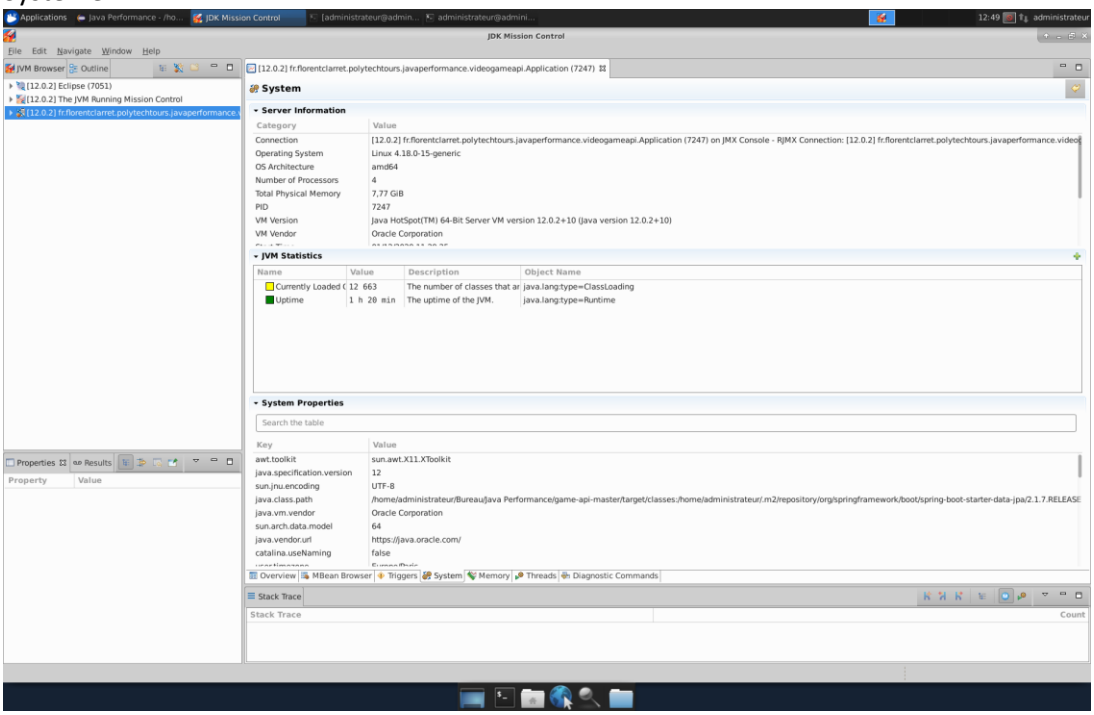
Mbean présente les informations système que jvm peut fournir à la console JMX. L'état de ces beans peut changer avec le fonctionnement du système. Vous pouvez l'ajouter à votre propre vue créée en fonction des informations MBean auxquelles vous devez prêter attention.

Déclencheur :



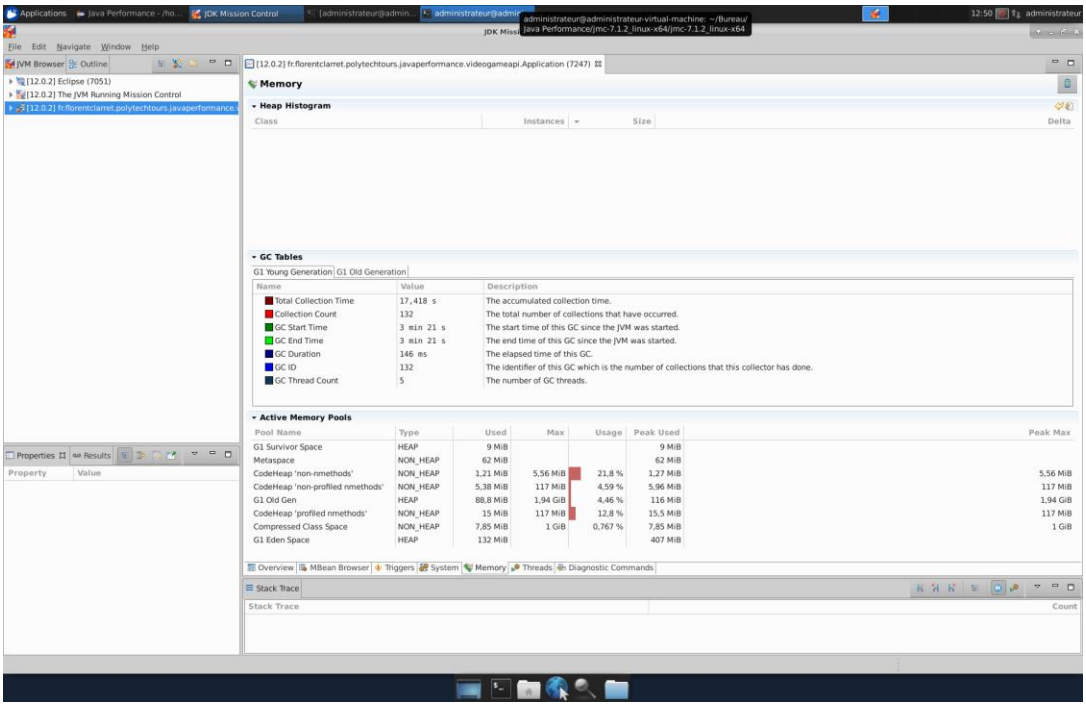
Il est pratique de surveiller et de rappeler un seuil d'état du système via le déclencheur.

Système :



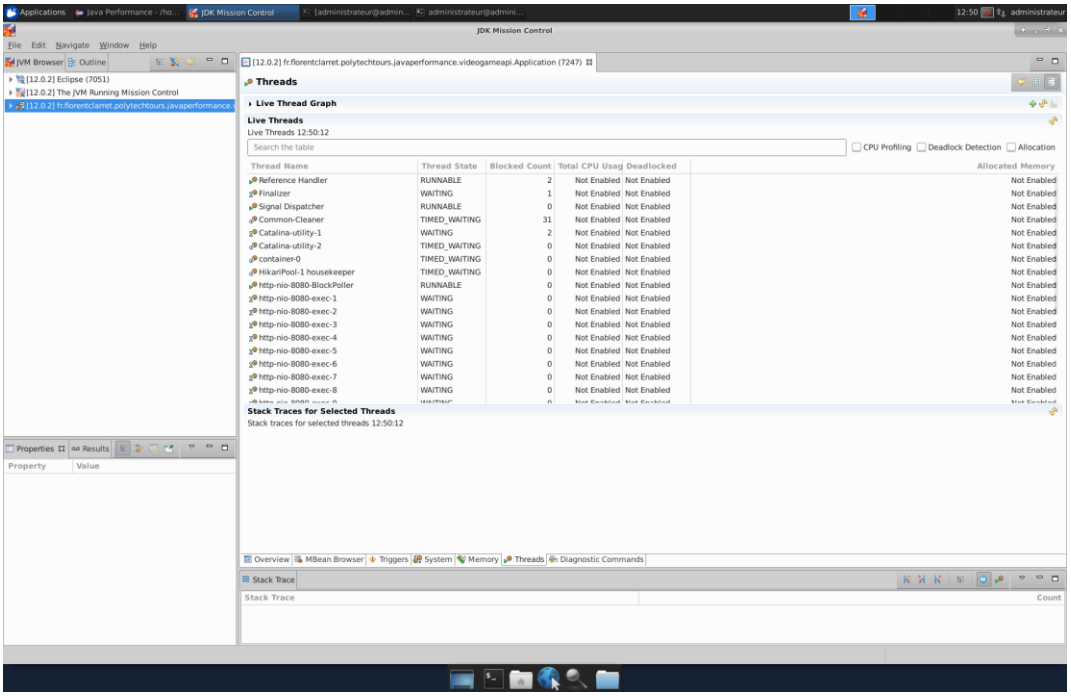
Dans la vue système, vous pouvez afficher les informations de configuration de l'application jvm, et vous pouvez afficher les propriétés de java par filtrage de nom, par exemple: java.class. Vous pouvez afficher le chemin de la classe de configuration de l'application java.

Mémoire :

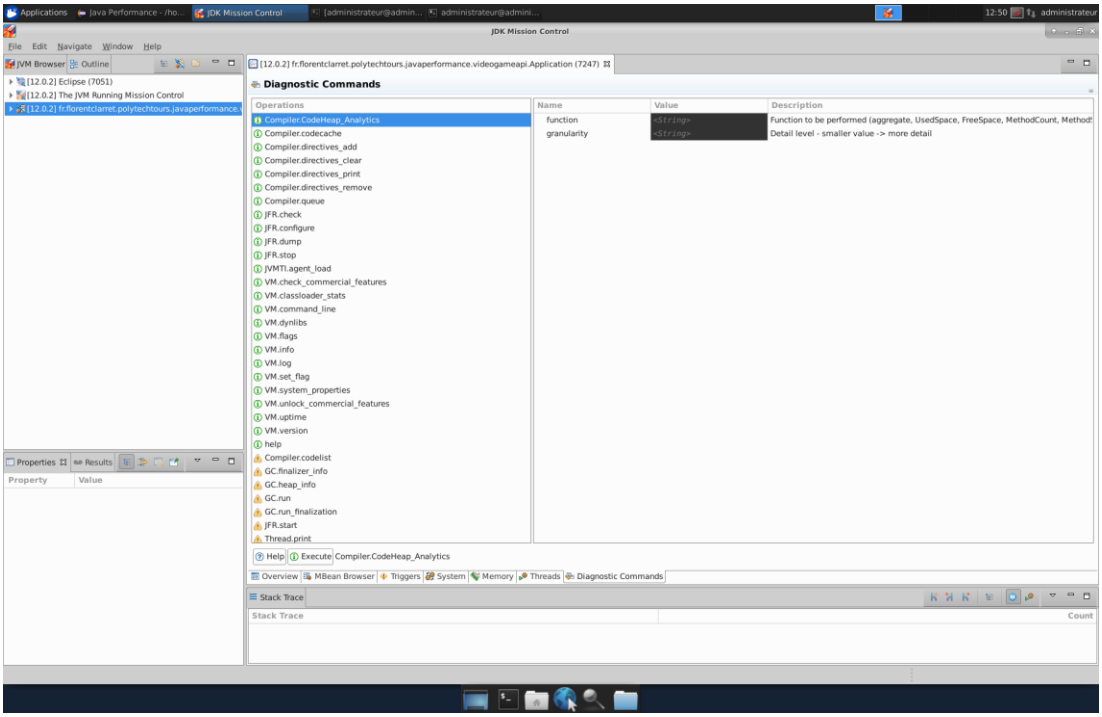


Il est plus important d'afficher l'histogramme du tas et les informations de la table gc via la vue mémoire.

Threads :



Diagnostic :


Applications
Java Performance - /ho...
JDK Mission Control
[administrateur@admin...
administrateur@admini...
12:50
administrateur
JDK Mission Control
File Edit Navigate Window Help
JVM Browser
Outline
[12.0.2] Eclipse (7051)
[12.0.2] The JVM Running Mission Control
[12.0.2] fr.florentclarret.polytechtours.javaperformance.
[12.0.2] fr.florentclarret.polytechtours.javaperformance.videogameapi.Application (7247)
Diagnostic Commands
Operations
Compiler.CodeHeap_Analytics
Compiler.codecache
Compiler.directives_add
Compiler.directives_clear
Compiler.directives_print
Compiler.directives_remove
Compiler.queue
JFR.check
JFR.configure
JFR.dump
JFR.stop
JVMTI.agent_load
VM.check_commercial_features
VM.classloader_stats
VM.command_line
VM.dynlibs
VM.flags
VM.info
VM.log
VM.set_flag
VM.system_properties
VM.unlock_commercial_features
VM.uptime
VM.version
help
Compiler.codelist
GC.finalizer_info
GC.heap_info
GC.run
GC.run_finalization
JFR.start
Thread.print
Name
Value
Description
function
<String>
Function to be performed (aggregate, UsedSpace, FreeSpace, MethodCount, Method
granularity
<String>
Detail level - smaller value -> more detail
Help
Execute
Compiler.CodeHeap_Analytics
Overview
MBean Browser
Triggers
System
Memory
Threads
Diagnostic Commands
Properties
Results
Property
Value
Stack Trace
Stack Trace
Count