TP1
Prise en main des outils de monitoring
Compte Rendu

PENG Hanyuan
YAN Wenli

Note: Pour chaque outil, nous n'avons pas testé et listé tous les commandes ou des options (Ils sont dans le manuel, nous savons tous chercher sur internet…..)

VMstat

vmstat fournit des informations sur les processus, la mémoire, la pagination, les entrées / sorties de blocs, les interruptions et l'activité du processeur.

L'inconvénient: Il n'y a pas d'analyse d'un processus particulier.

Commande exécutée: `vmstat -w 1`
Option -w agrandit la largeur du champ pour les grandes tailles de mémoire.

```
administrateur@VM-ubuntu-javaperf:~$ vmstat -w 1
procs -----------------------mémoire---------------------- -échange- -----io---- -système- --------cpu--------
 r  b         swpd         libre        tampon        cache   si   so    bi    bo   in   cs  us  sy  id  wa  st
 1  0            0       1544136        48532      1081024    0    0   127    12  254  424   3   3  93   2   0
 0  0            0       1544160        48532      1081024    0    0     0     0 1023 1741   6   2  91   0   0
 0  0            0       1544260        48532      1081024    0    0     0     0 1130 1845   5   2  93   0   0
 2  0            0       1544260        48532      1081024    0    0     0     0 1317 2155   5   1  94   0   0
 0  0            0       1544260        48532      1081024    0    0     0    32 1812 2718   8   1  91   0   0
 0  0            0       1543268        48532      1081024    0    0     0     0 1127 1890   6   1  93   0   0
 0  0            0       1543328        48540      1081024    0    0     0    36 1400 2215   7   2  91   0   0
 4  0            0       1488228        48540      1081056    0    0     0     0 2415 3666  13   6  81   0   0
 1  0            0       1531680        48540      1081000    0    0     0     0 1920 3097   9   4  87   0   0
 4  0            0       1502572        48540      1081416    0    0     0    56 3927 7436  41  16  42   0   0
 2  0            0       1462396        48540      1081416    0    0     0     4 2842 4655  45  10  46   0   0
 2  0            0       1426272        48540      1081416    0    0     0     0 2887 4656  42  14  44   0   0
 1  0            0       1403192        48540      1081496    0    0     0     0 3075 7117  55  14  31   0   0
 4  0            0       1358412        48548      1081616    0    0     0   120 3039 7502  59  15  26   0   0
 2  0            0       1350212        48548      1081552    0    0     8     0 2743 4916  49   2  48   1   0
 0  0            0       1349964        48548      1081552    0    0     0     0 1593 2713   8   2  90   0   0
 1  0            0       1348972        48548      1081552    0    0     0     0 1599 2637  10   2  88   0   0
 0  0            0       1348848        48548      1081552    0    0     0     0 1283 2124   6   1  93   0   0
 0  0            0       1348848        48548      1081552    0    0     0     0 1478 2400   5   2  93   0   0
 0  0            0       1348848        48556      1081552    0    0     0    52 1619 2732   7   4  89   0   0
 1  0            0       1349220        48556      1081552    0    0     0     0 2061 3158   7   5  88   0   0
 1  0            0       1348972        48556      1081552    0    0     0     0 1634 2749   7   3  90   0   0
 0  0            0       1348972        48556      1081552    0    0     0     0  925 1646   4   2  94   0   0
 2  0            0       1348972        48556      1081552    0    0     0     0  927 1632   5   2  93   0   0
 0  0            0       1348972        48556      1081552    0    0     0     0  971 1686   5   0  95   0   0
```

Image 1 - vmstat

Cas d'utilisation:
On veut savoir que les informations sur le système rapidement, par exemple le mémoire libre et l'usage de cpu.
Parfois on doit changer le hardware pour améliorer la performance du programme. Cet outil nous aide d'évaluer le système.
C'est utilisé pour analyser le système Linux mais pas un programme Java particulier.

IOstat

La commande iostat est utilisée pour surveiller le périphérique d'entrée / sortie du système.
L'inconvénient: Il n'y a pas d'analyse d'un processus particulier.

Commande exécutée: `iostat -d -k 1`
Option -d: Affichier l'information sur disque dur.
Option -k: Utiliser le Kilobytes.

Cas d'utilisation:
On veut savoir le I/O status des périphériques du système. Mais il peut aussi afficher l'information sur l'usage de cpu (avec option -c).

nicstat

Analyser les statistiques de trafic réseau.
L'inconvénient: Il n'y a pas d'analyse d'un processus particulier.
Nous avons ouvert une page web.

```
administrateur@vm-ubuntu-javaperf:~$ nicstat 1
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
13:40:38       lo    6.94    6.94   11.49   11.49   618.7   618.7  0.00   0.00
13:40:38    ens33    4.91    0.38    4.84    2.58  1040.1   149.8  0.43   0.00
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
13:40:39       lo    0.00    0.00    0.00    0.00    0.00    0.00  0.00   0.00
13:40:39    ens33    0.20    0.15    2.99    2.00   68.33   77.00  0.03   0.00
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
13:40:40       lo    0.00    0.00    0.00    0.00    0.00    0.00  0.00   0.00
13:40:40    ens33    0.00    0.00    0.00    0.00    0.00    0.00  0.00   0.00
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
13:40:41       lo    6.68    6.68   59.98   59.98   114.0   114.0  0.00   0.00
13:40:41    ens33   85.94   26.82   148.9   119.0   590.8   230.8  9.24   0.00
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
13:40:42       lo    1.88    1.88   15.99   15.99   120.6   120.6  0.00   0.00
13:40:42    ens33    5.59    4.94   28.97   27.98   197.5   180.9  0.86   0.00
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
13:40:43       lo    0.00    0.00    0.00    0.00    0.00    0.00  0.00   0.00
13:40:43    ens33    0.00    0.00    0.00    0.00    0.00    0.00  0.00   0.00
```

Image 2 - nicstat

Cas d'utilisation:
On peut analyser l'usage du réseau d'un système.
On peut aussi analyser les statistiques du protocole TCP ou UDP avec les options **-t** ou **-u**.
Avec l'option **-I** on peut savoir l'information de la carte réseau dans la machine.

Jcmd

jcmd est utilisé pour envoyer des demandes de commande de diagnostic à la machine virtuelle Java.

Nous pouvons faire l'analyse sur un JVM particulier.

Commandes utilisées:
jcmd -l: lister toutes les processus JVM.
jcmd pid VM.flags: Imprimer tous les flags utilisés pour une VM

Par exemple nous avons modifié le VM de eclipse (fichier eclipse.ini) pour tester.

MaxHeapSize -> 600mb,
OldSize -> 30 mb

```
administrateur@vm-ubuntu-javaperf:~$ jcmd 23149 VM.flags
23149:
-XX:CICompilerCount=3 -XX:InitialHeapSize=41943040 -XX:MaxHeapSize=629145600 -XX:MaxNewSize=10485760 -XX:MinHeapDeltaBytes=524288
 -XX:NewSize=10485760 -XX:OldSize=31457280 -XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseFastUnorderedTimeStamps
 -XX:+UseParallelGC
```

Image 3 - jcmd

Nous pouvons utiliser les autres options sur des aspects différents:

- **JFR.***: Java Flight Recorder (JFR) est un outil de collecte de données de diagnostic et de profilage concernant une application Java en cours d'exécution.
- **GC.***: l'information sur Gabage collector, GC.class_stats, GC.class_histogram, GC.heap_dump etc.
- **VM.*:** JVM status, comme VM.flags, VM.system_properties

jconsole

Nous avons analysé eclipse avec JConsole.

JConsole peut surveiller des applications locaux ou en remote. L'utilisation de JConsole pour surveiller une application locale est utile pour le développement, mais n'est pas recommandée pour les environnements de production, car JConsole lui-même consomme d'importantes ressources. La surveillance à distance est recommandée pour isoler l'application JConsole de la plate-forme surveillée.

Nous pouvons analyser l'usage du mémoire, cpu et threads pour une application particulière avec JConcole. Il a l'interface graphique qui nous permet d'analyser plus facilement.

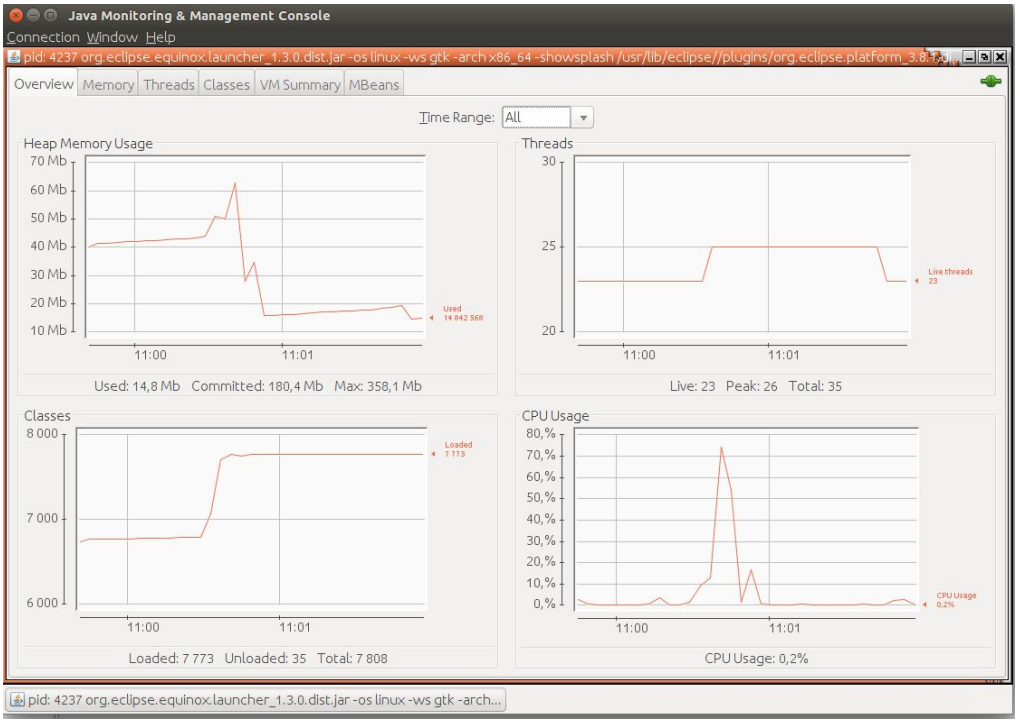

Image 4 - JConcole

Jhat

Java Head Analyse Tool (Jhat), est un outil pour l'analyse de tas de java.
La commande jhat analyse un fichier de segment de mémoire java et lance un serveur Web. jhat vous permet de parcourir des vidages de tas à l'aide de votre navigateur Web préféré.

Nous avons utilisé le heap-dump-file dumpé par Jcmd et l'ouvert avec Jhat.
Parfois le fichier heap-dump peut être très large, nous pouvons utiliser l'option: `-J-Xmx512m` pour augmenter le mémoire.
Mais comme le Jhat consomme d'importantes ressources, c'est mieux d'isoler l'outil de surveillance pour les environnements de production.

Utilisation:

- jhat [options] heap-dump-file
- Ouvrir le ficher et le analyser par jhat
- Regarter le .html par le serveur HTTP http://<IP>:7000

Jstack

Jstack est principalement utilisé pour afficher des informations sur la pile de threads dans un processus Java. Jstack peut obtenir des informations sur la pile Java et la pile native exécutant le programme Java. Il est facile de connaître le thread en cours d'exécution.

Comme indiqué ci-dessous

```
jstack [option] pid
jstack [option] executable core
jstack [option] [server-id@]remote-hostname-or-ip
```

Image 5 - JStack

Cas d'utilisation:
On peut savoir par exemple le temps CPU consommé par chaque thread. Il a le possibilité de dumper le fichier d'un server en remote donc il est conseillé pour l'environnement de production.

JVisualVM

Java VisualVM est un outil graphique intuitive fournissant des informations détaillées sur les applications Java.

JVisualVM est utile aux développeurs d'applications Java pour dépanner les applications et pour surveiller et améliorer les performances des applications. JVisualVM permet de générer et d'analyser des heap dump de segments de mémoire, de localiser les fuites de mémoire, d'effectuer et de surveiller la récupération de place, et de créer un profilage léger de la mémoire et du processeur.

Points forts:

a. Gabage collector (GC)
   Dans jvisualvm, GC est très visible, comme indiqué ci-dessous:

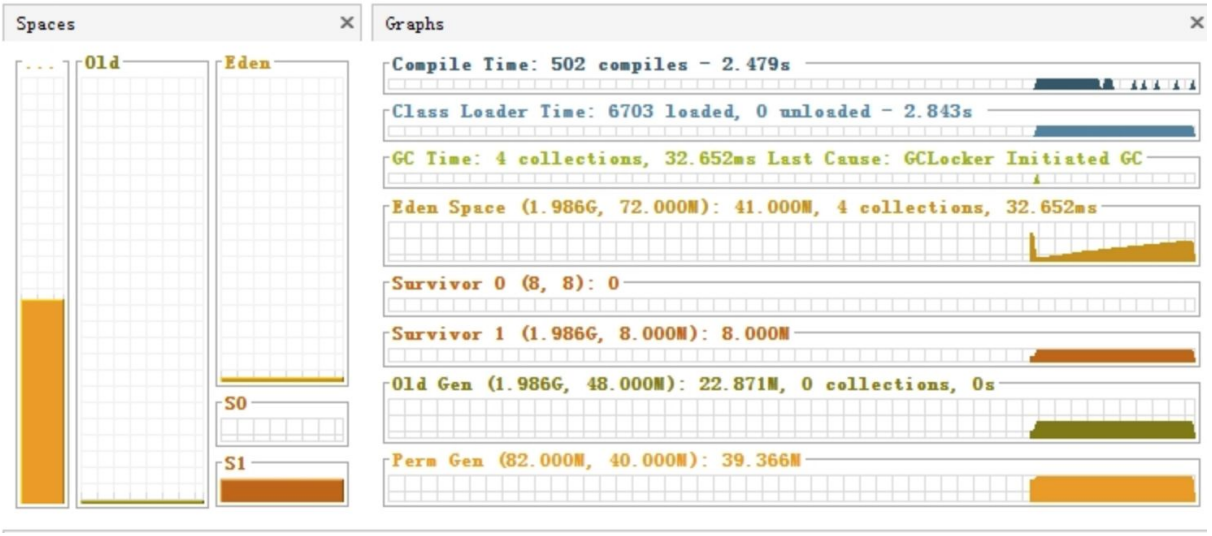

Image 6 - JVisualvm GC-1

Pour ce mécanism, il y a trois espaces importantes pour les tas de java processus: young generation, old generation et permanent generation. L'algorithme de collecte le plus approprié peut être utilisé en fonction des caractéristiques de chaque époque.

La nouvelle génération est divisée en trois régions, une Eden Space et deux régions Survivor, ce ratio pouvant également être modifié. Habituellement, les objets sont principalement affectés dans la zone Eden de la nouvelle génération et, lorsque la mémoire de la Eden Space est saturée, ils sont attribués à un survivant. Deux survivants ne peuvent pas occupés en même temps, il faut garantir au moins une space est disponible.

L'ancienne génération stocke tous les objets qui ont survécu pendant longtemps et qui sont de grande taille. L'ancienne génération utilise donc l'algorithme de balisage. Lorsque l'ancienne capacité de génération est pleine, un GC majeur (GC complet) sera activé pour récupérer les ressources d'objet qui ne sont plus utilisées par les générations anciennes et jeunes.

Le ramassage des ordures de **la génération permanente** recycle principalement deux parties: les constantes obsolètes et les classes inutiles.

On peut configurer des espaces des régions, par exemple:
Si on a mis 10M pour Young Generation (-Xmn):

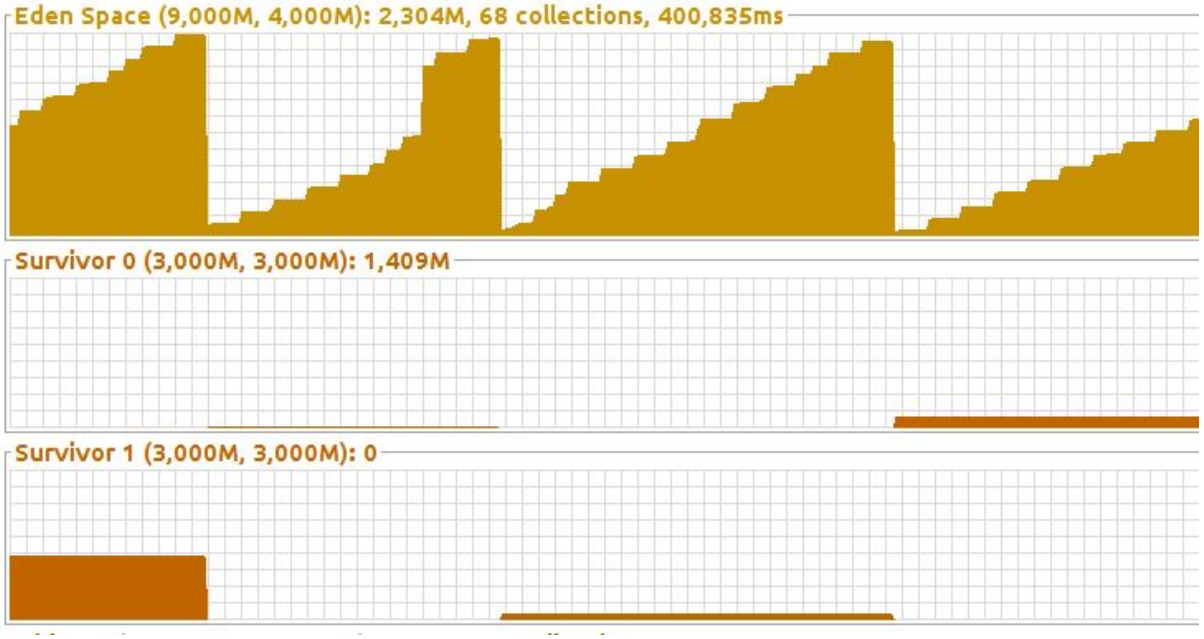

Image 7 - JVisualvm GC-2 (Young Generation: 10M)

Si on a changé à 300M:

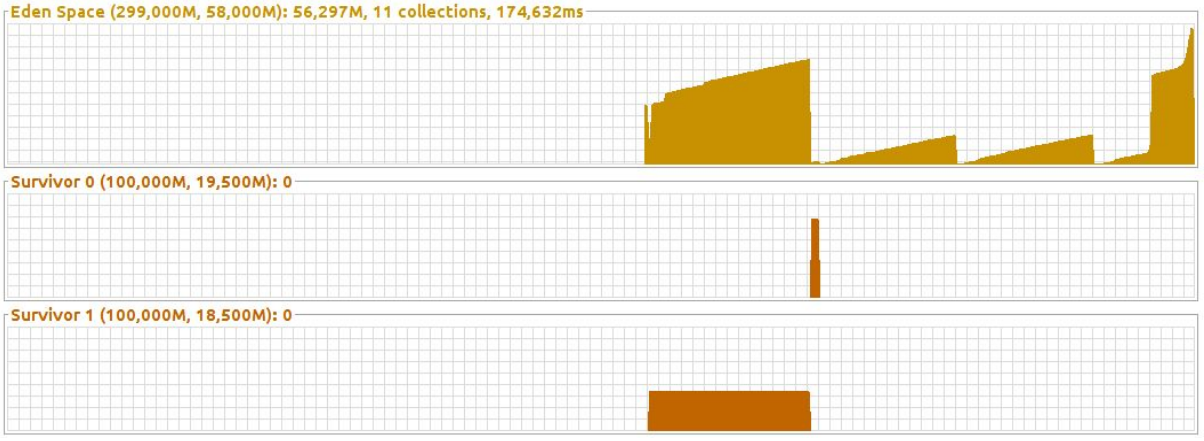

Image 8 - JVisualvm GC-2 (Young Generation: 300M)

b. Surveillance du temps CPU
   Par cet outil, nous pouvons surveiller dynamiquement le temps CPU utilisé par chaque méthode. Comme indiqué ci-dessous:

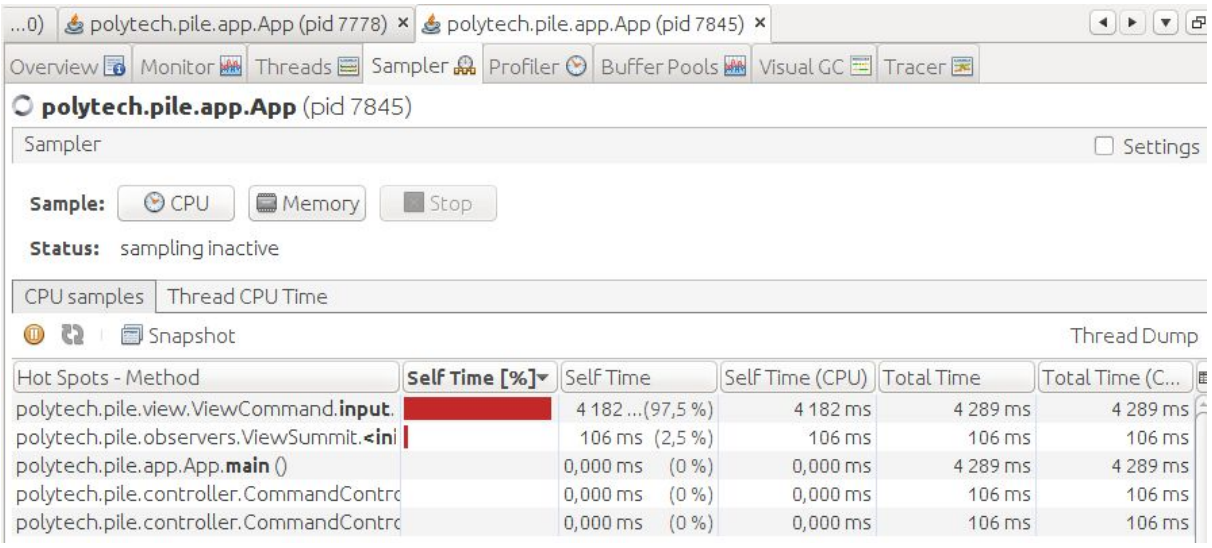

Image 9 - JVisualvm CPU

c. Surveillance de fuite de mémoire.
Elle effectue aussi un heap dump de l'application Java détectée à intervalles réguliers.
Comme l'image ci-dessous, on peut regarder la mémoire par rapport chaque class, instance dans des processus.

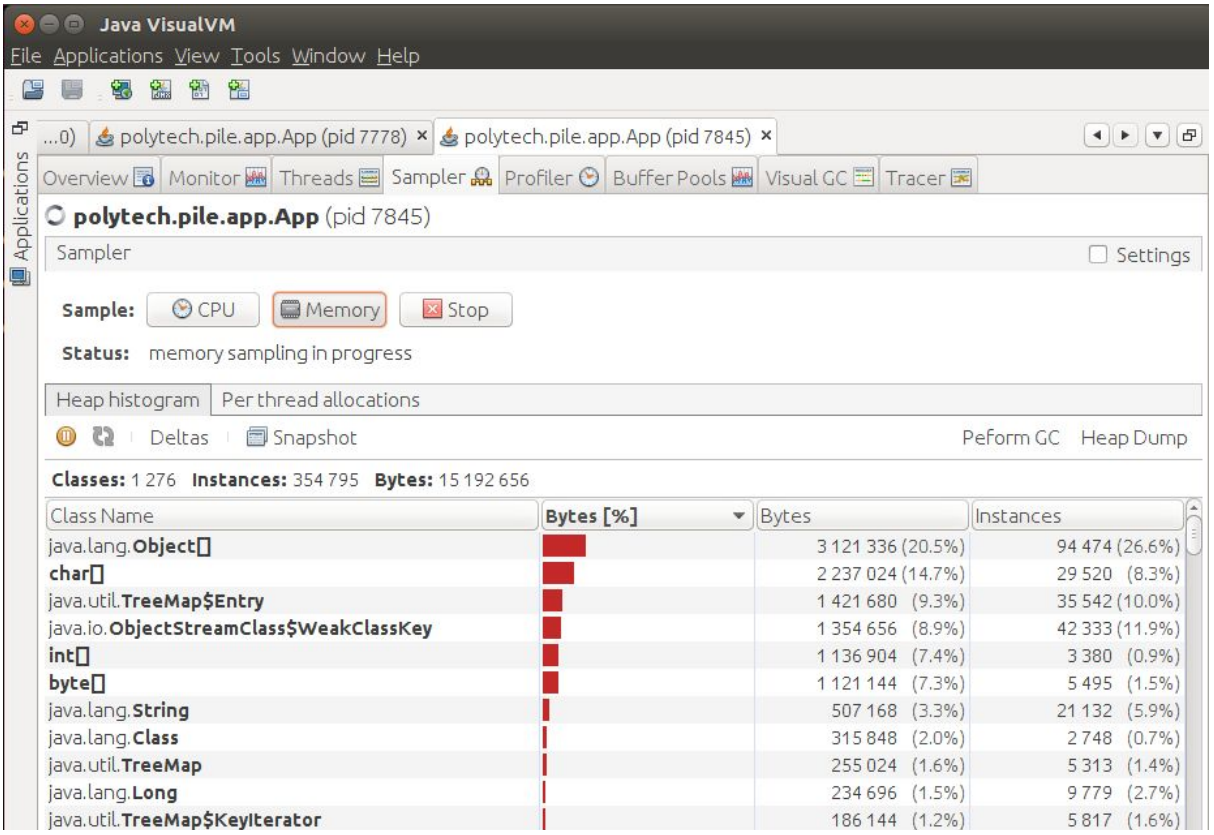

Image 10 - JVisualvm Heap histogram

Cas d’utilisations:

a. Surveillance des programmes locaux
    - Peut surveiller le processeur, la classe, le thread, la consommation du processus java local, heap dump.
b. Surveillance à distance
    - Comme Jconsole, JVisualVM permet aux utilisateurs de surveiller des performances(cpu, class, fuite de mémoire etc...) des applications java à distance. (plugin)

Java Mission Control

Java Mission Control ressemble beaucoup à JVisualVM sauf qu'il a un "meilleur" UI. Nous n'avons pas trouvé des "usecases" différents.