**Rapport\_Java Performance\_TP1**

**- LIU Yuanyuan && GUO Xiaoqing**

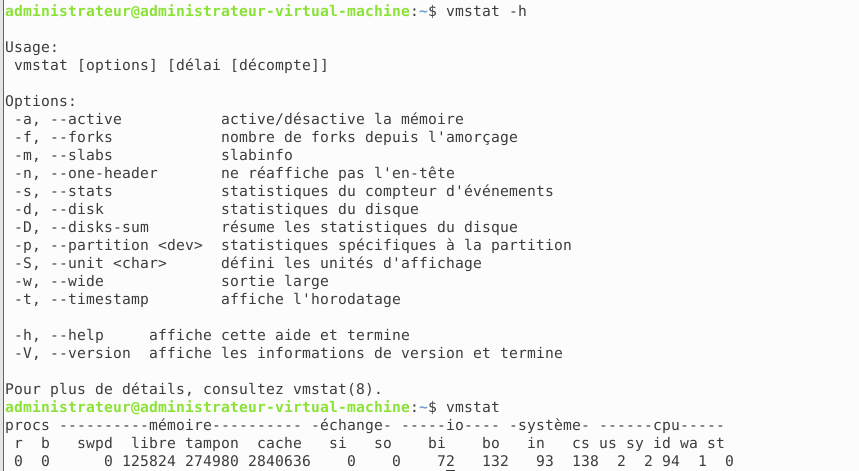
**TP1 : Prise en main des outils de monitoring**

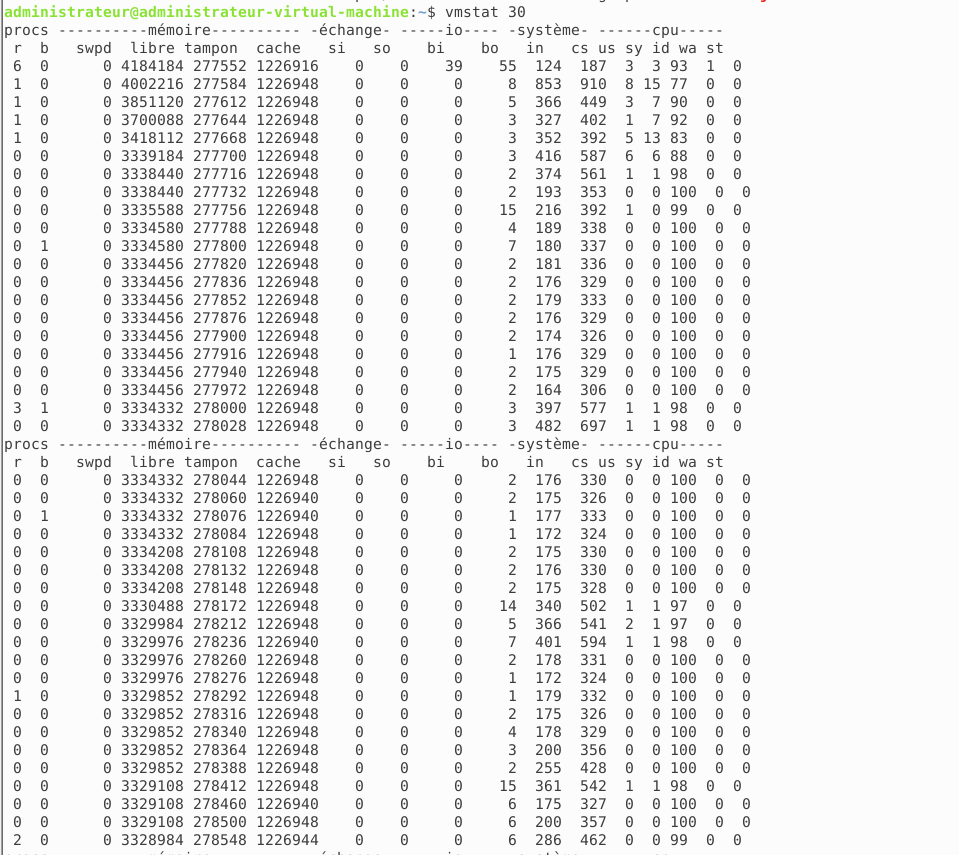
Ce TP sert à comprendre le fonctionnement de différents outils permettant de monitorer et d’analyser une application JAVA. Vous pouvez trouver ci-dessous les applications de monitoring qu’on peut utiliser :

* **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 :



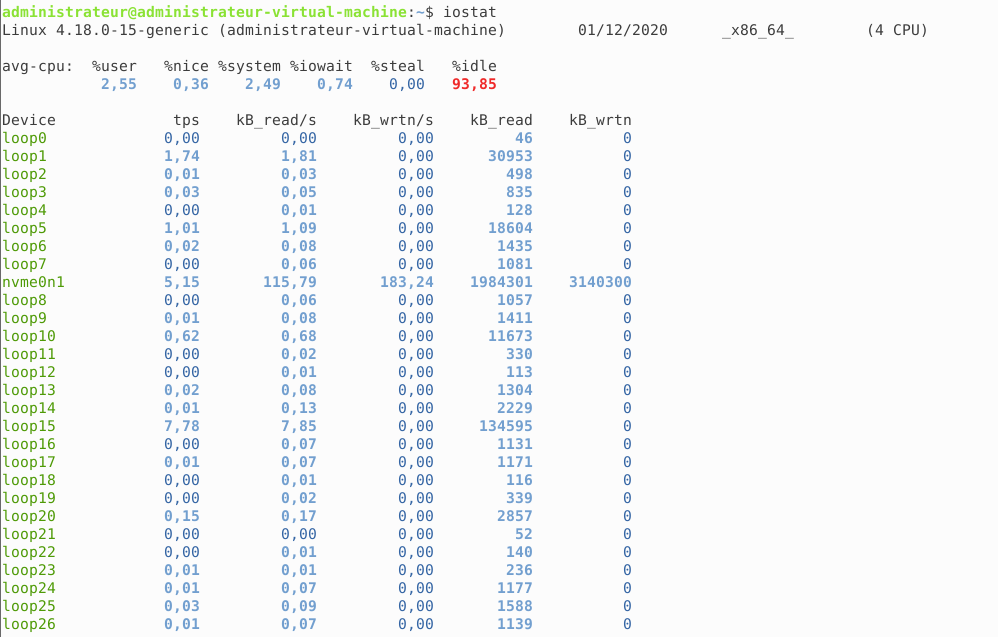


La signification de chaque paramètre :

* %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]



Les significations des valeurs d'attribut du cpu:

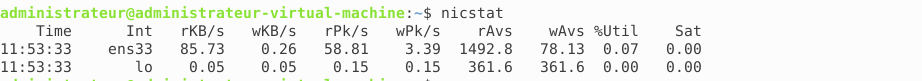
* % utilisateur: pourcentage de temps pendant lequel la CPU est en mode utilisateur.
* % nice: Le pourcentage de temps pendant lequel le processeur est en mode utilisateur avec une valeur NICE.
* % système: pourcentage de temps pendant lequel la CPU est en mode système.
* % iowait: pourcentage de temps 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.

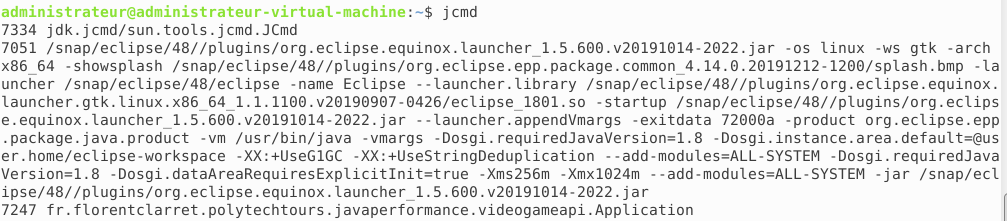
* Test :

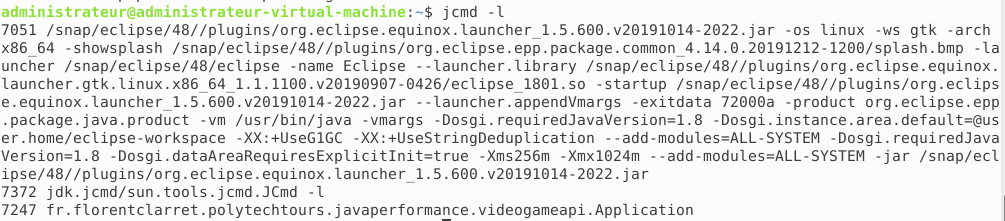


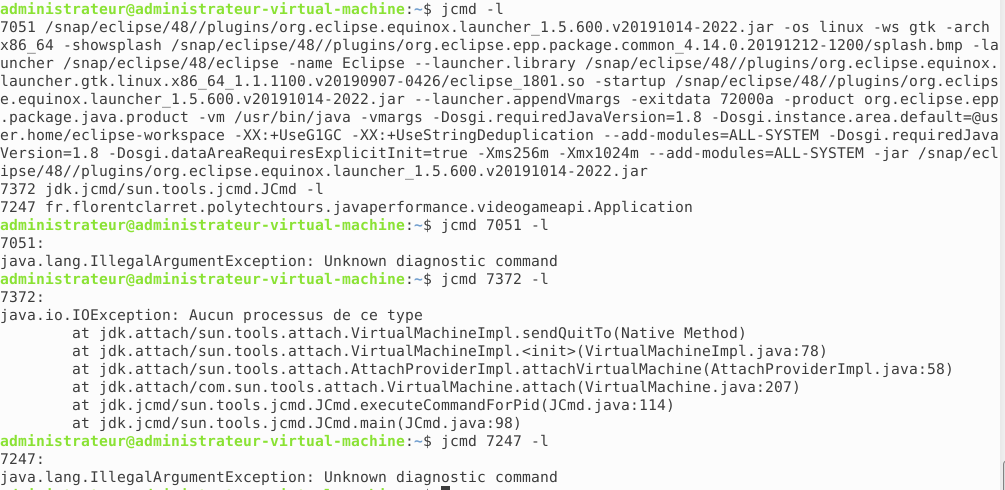
* **jcmd**
* Fonction :

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 :





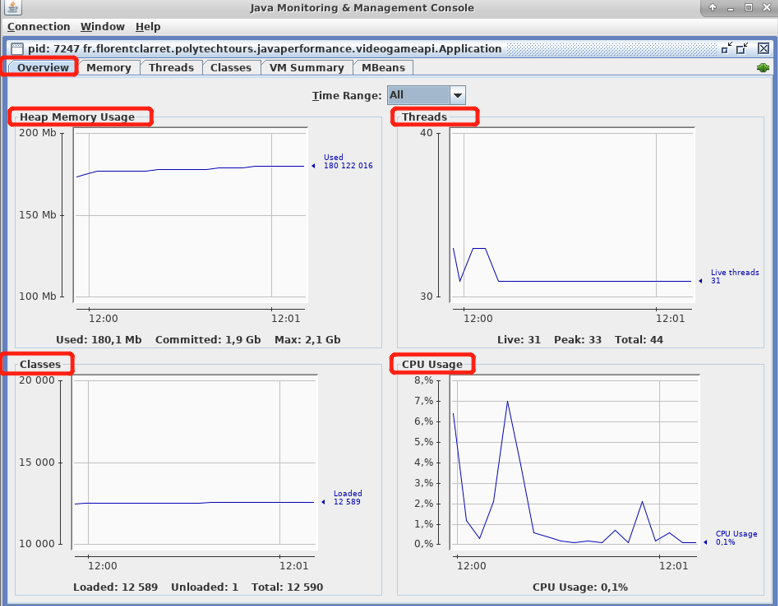


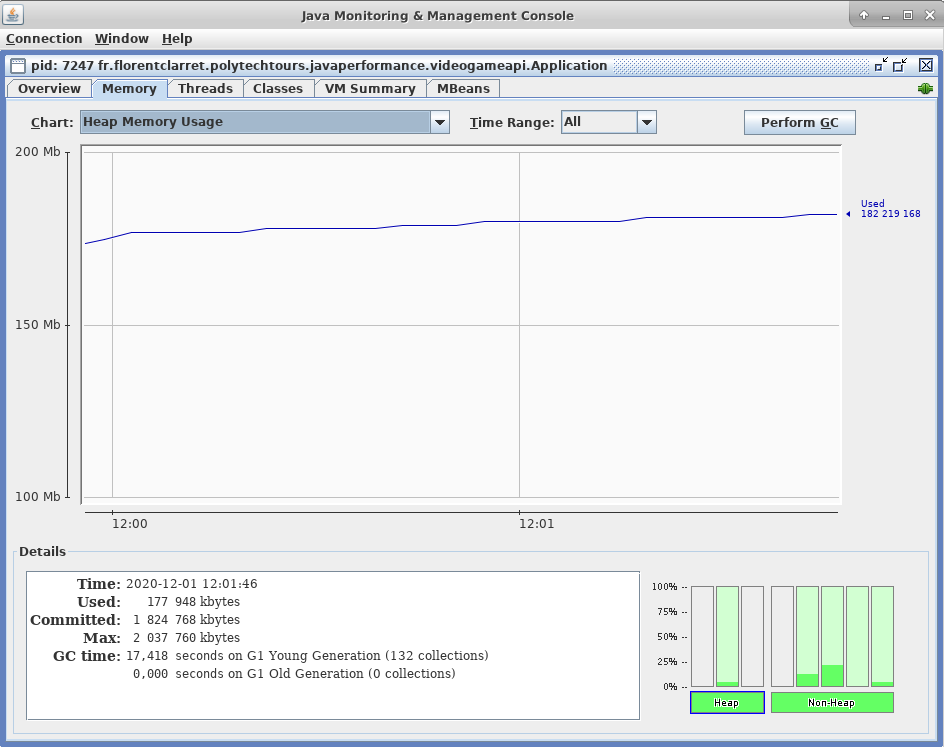
jcmd -l :

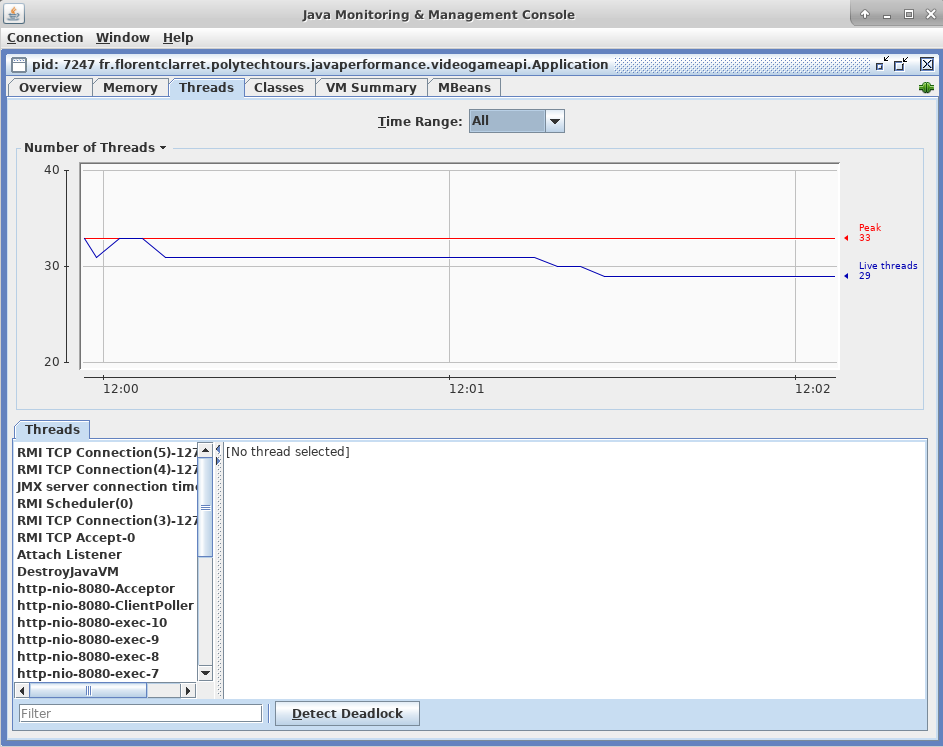
* Affiche la liste des identificateurs de processus de la machine virtuelle Java qui ne sont pas exécutés dans un processus docker distinct, ainsi que la classe principale et les arguments de ligne de commande qui ont été utilisés pour lancer le processus. Si la JVM est dans un processus docker, vous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

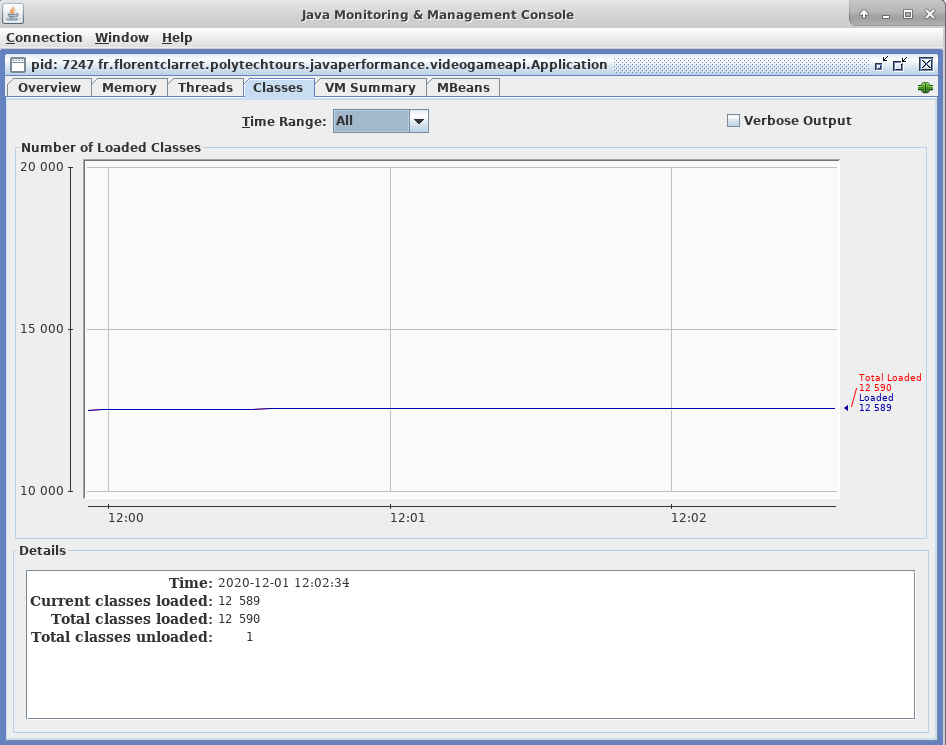
JConsole (Java™ Monitoring and Management Console) est un outil graphique qui permet à l'utilisateur de contrôler et de gérer le comportement des applications Java. Lorsque JConsole se connecte à une application Java, il affiche des informations la concernant. Ces informations comprennent l'utilisation 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.

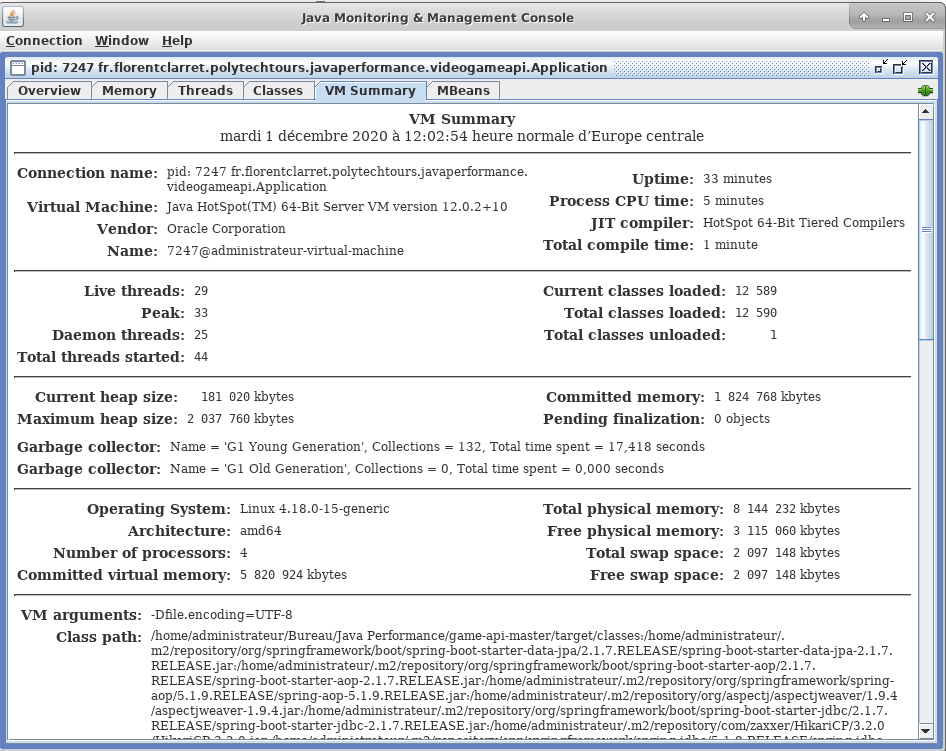
* Test :







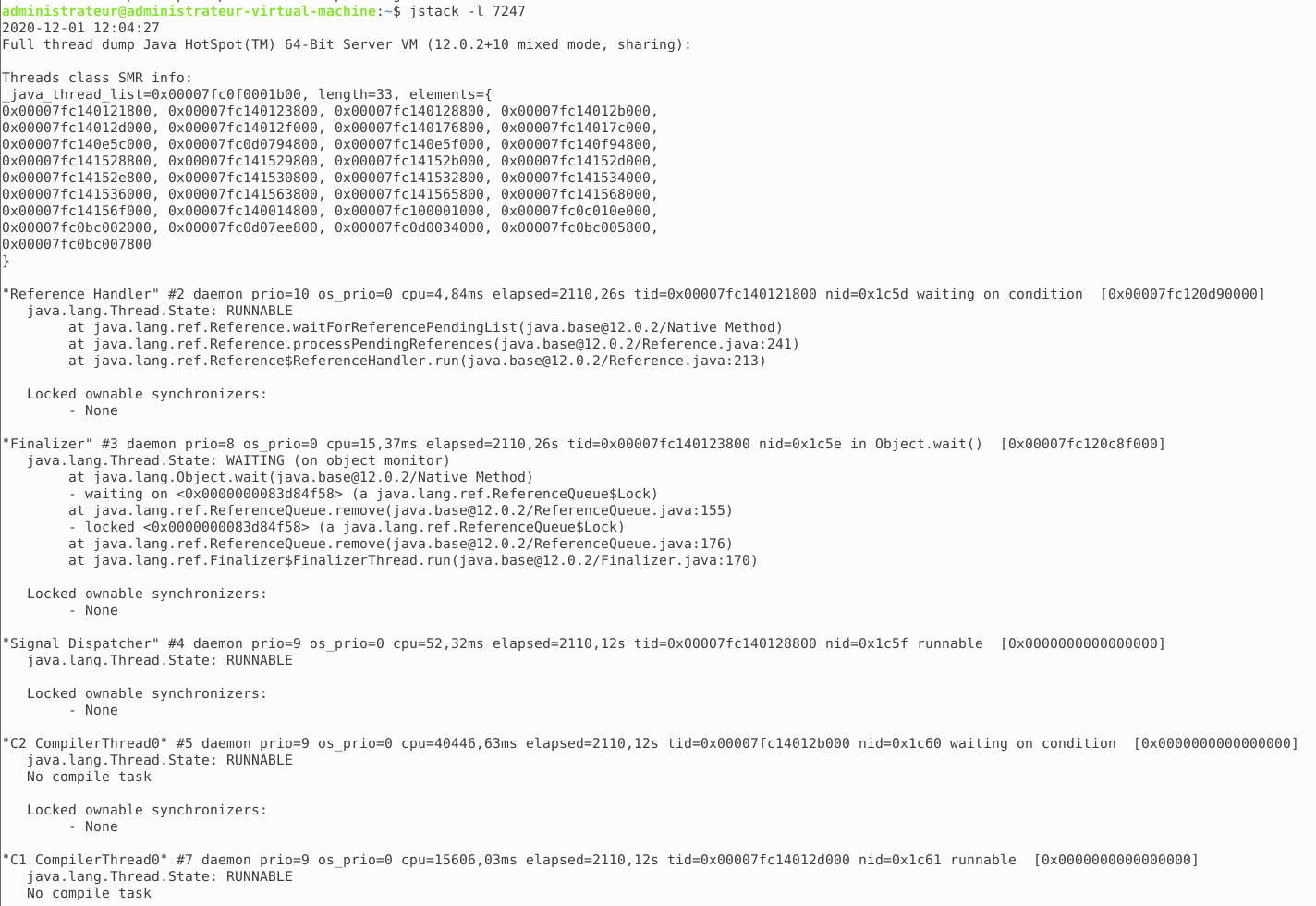




* **jstack**
* Fonction :

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 :



... ... ...

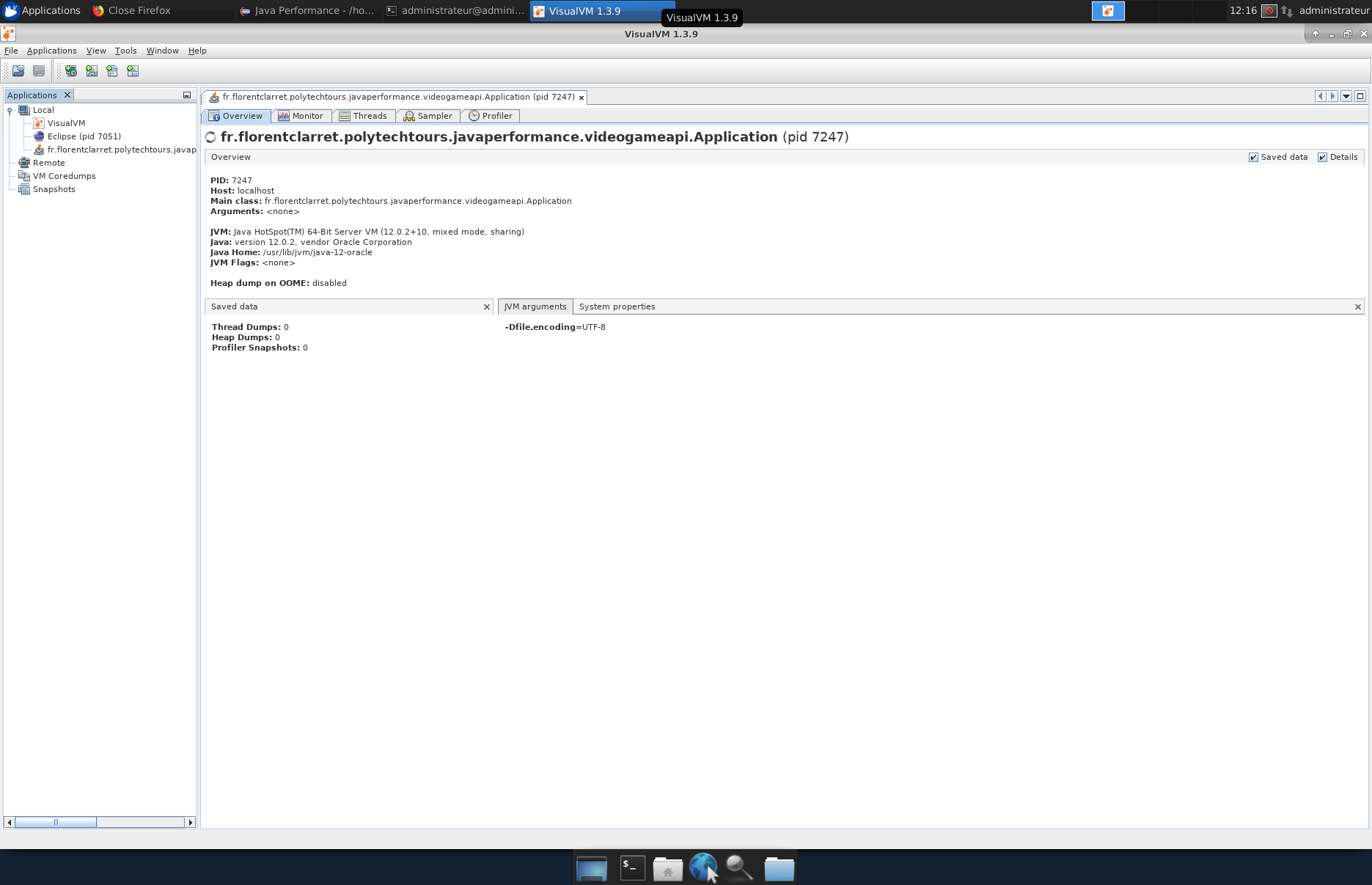


jstack -l: 🡺 jstack [ option ] pid

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

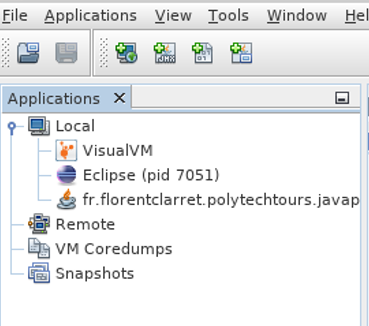
VisualVM monitors and troubleshoots applications running on Java 1.4+ from many vendors using various technologies including jvmstat, JMX, Serviceability Agent (SA) and Attach API.

* Test:



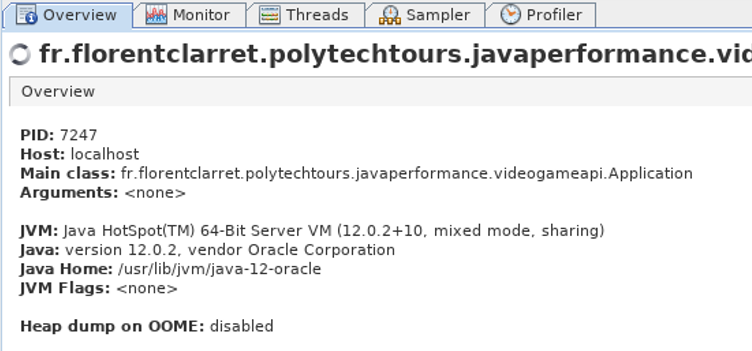
Afficher les processus Java locaux et distants

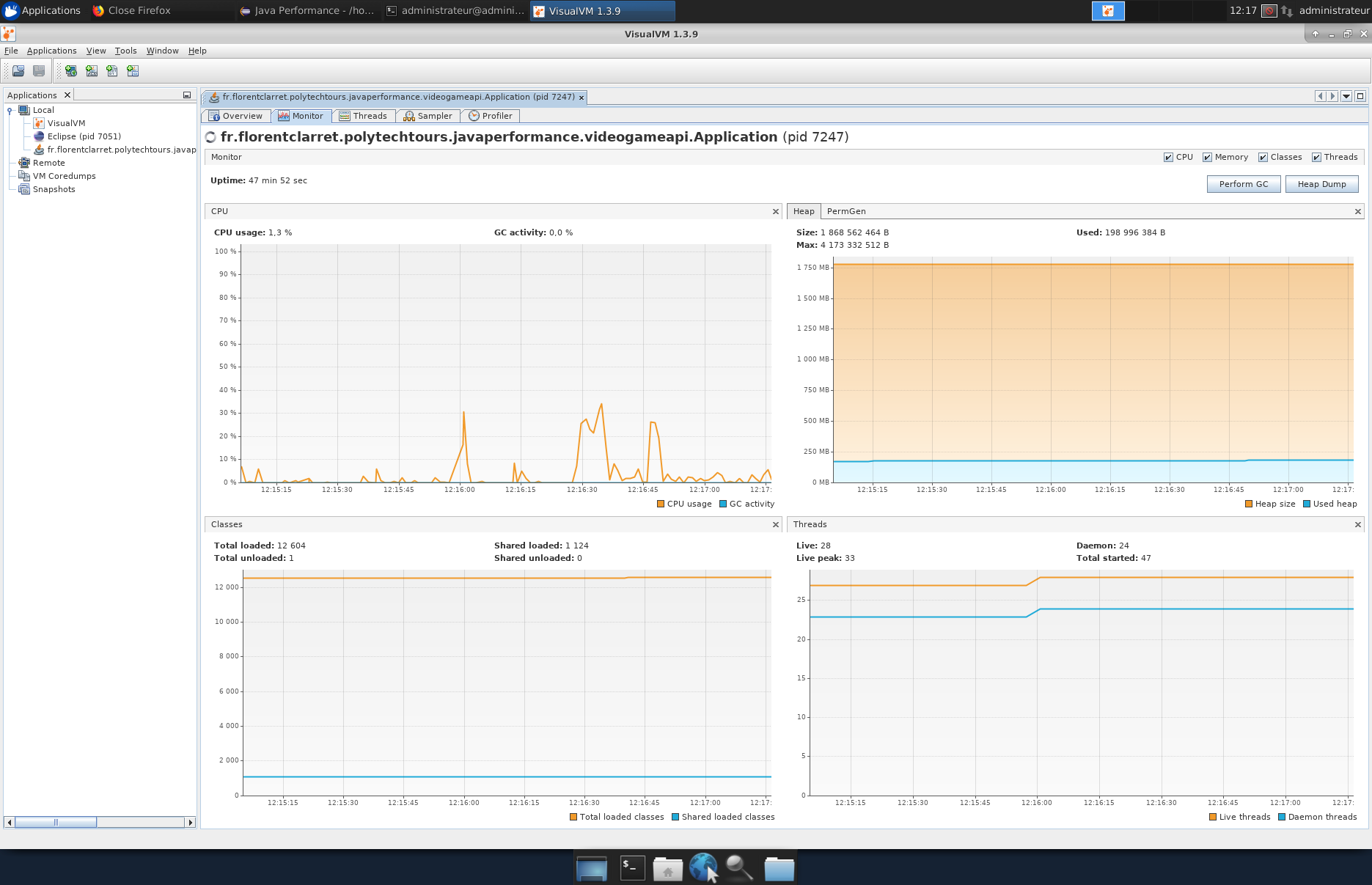
* VisualVM détecte et répertorie automatiquement les applications Java exécutées localement et à distance (jstatd doit être exécuté sur l'hôte distant). Vous pouvez également définir des applications manuellement par connexion JMX.



Afficher la configuration et l'environnement du processus

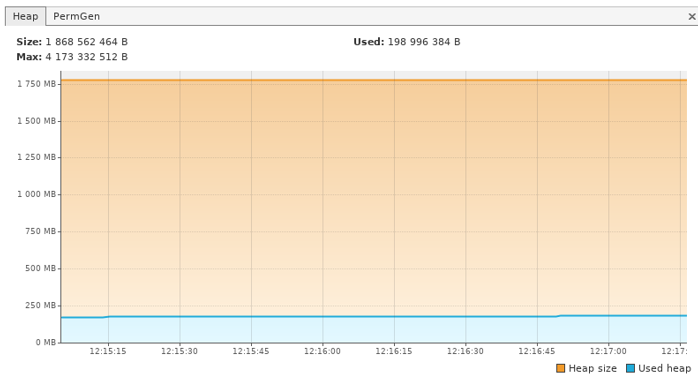
* Pour chaque processus, VisualVM affiche les informations d'exécution de base: PID, classe principale, arguments passés au processus java, version JVM, accueil JDK, indicateurs et arguments JVM et propriétés système.





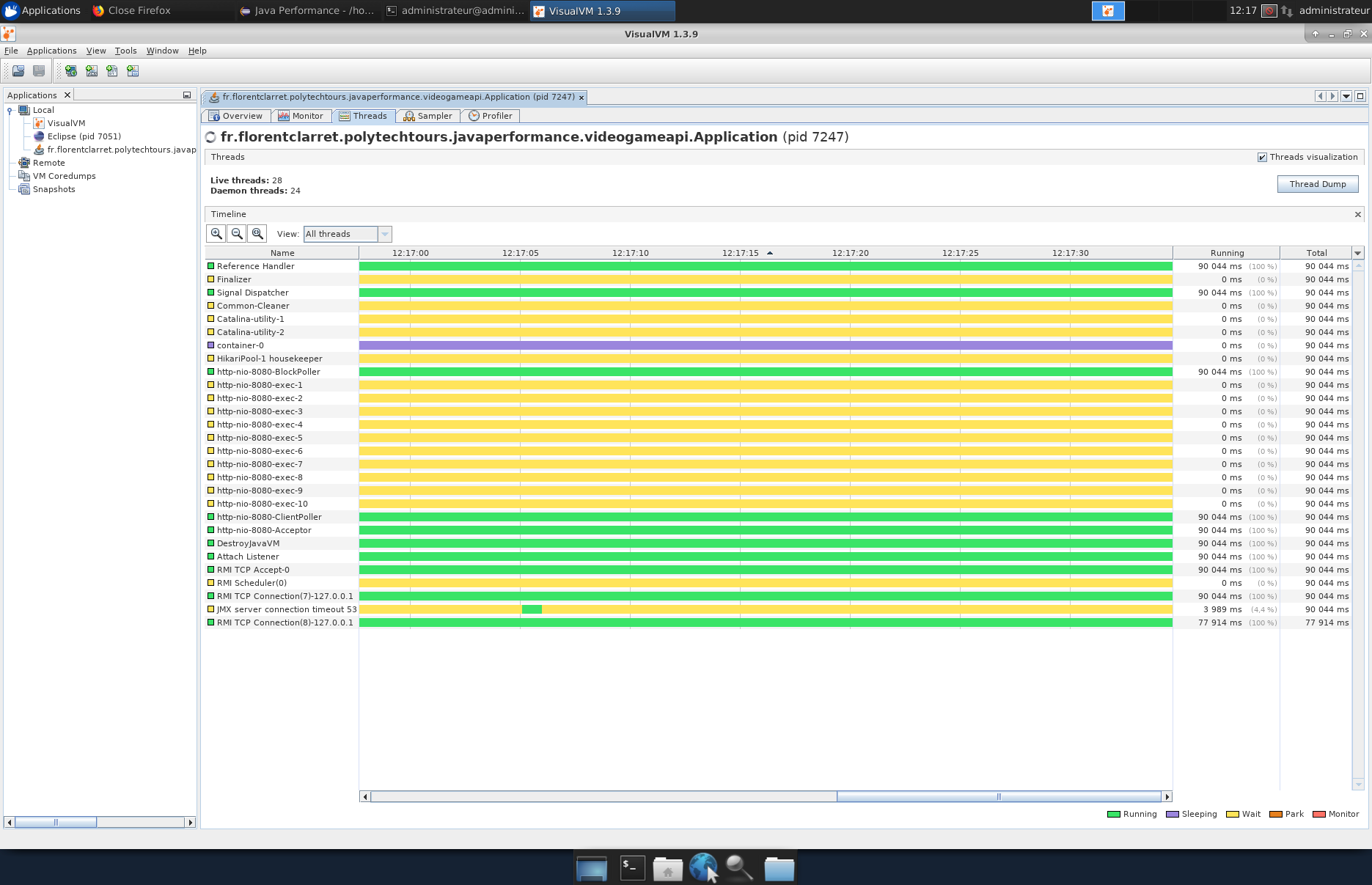
Surveiller les performances des processus et la mémoire

* 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

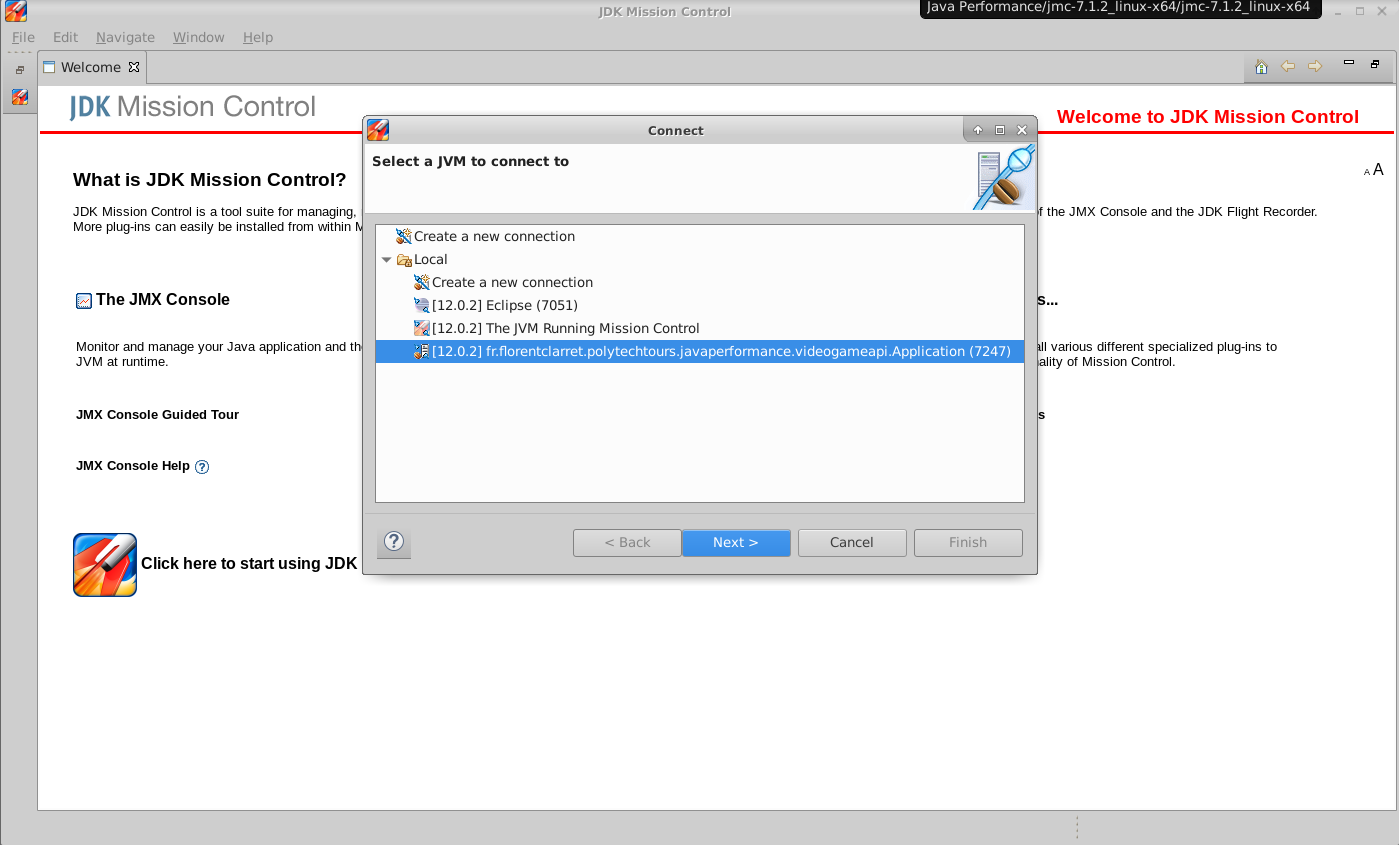
* Tous les threads en cours d'exécution dans un processus Java sont affichés dans une chronologie avec les temps cumulés d'exécution, de mise en veille, d'attente, de parcage et de surveillance.



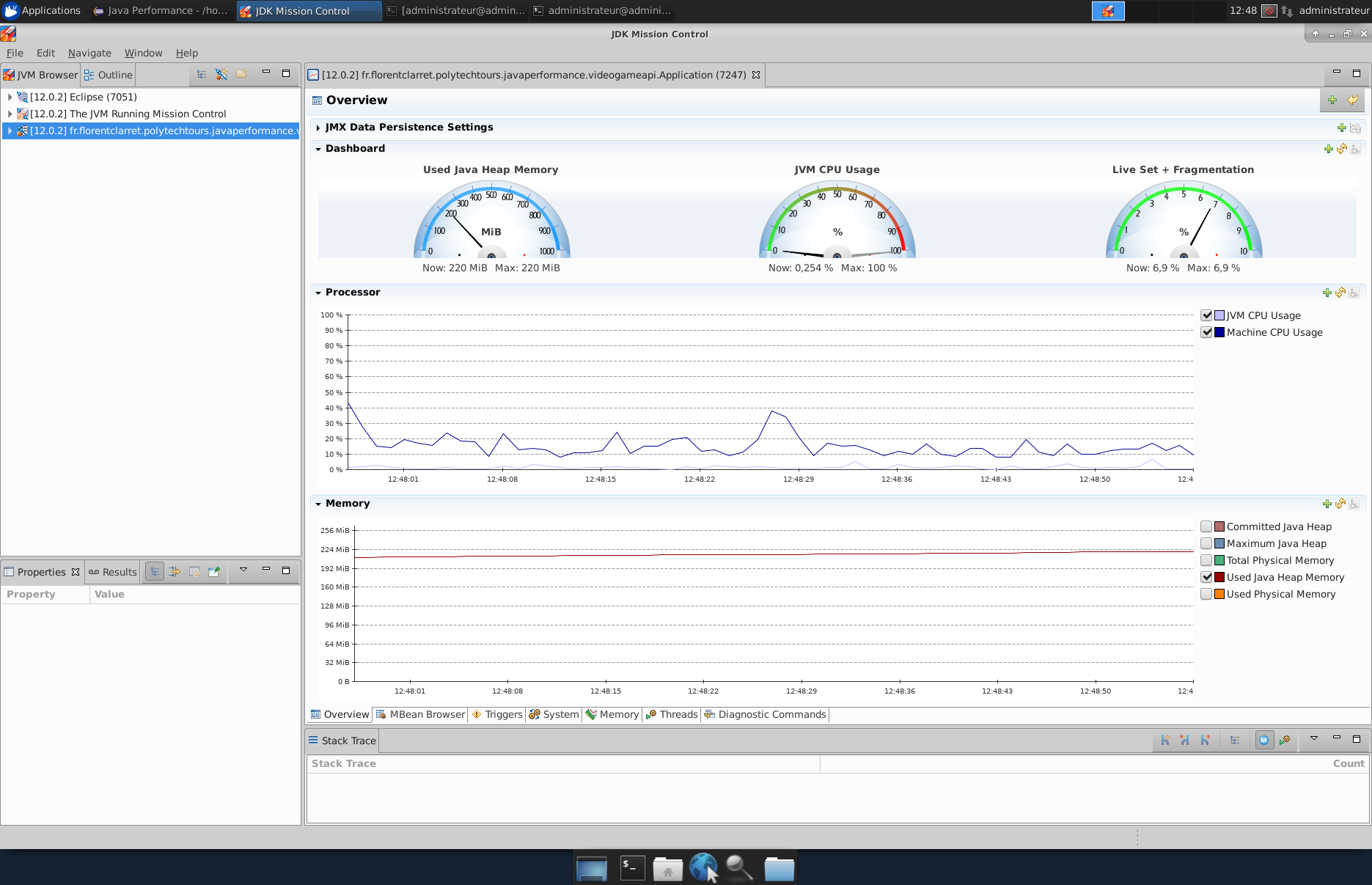
* **JMC (JDK Mission Control)**
* Fonction :

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 :

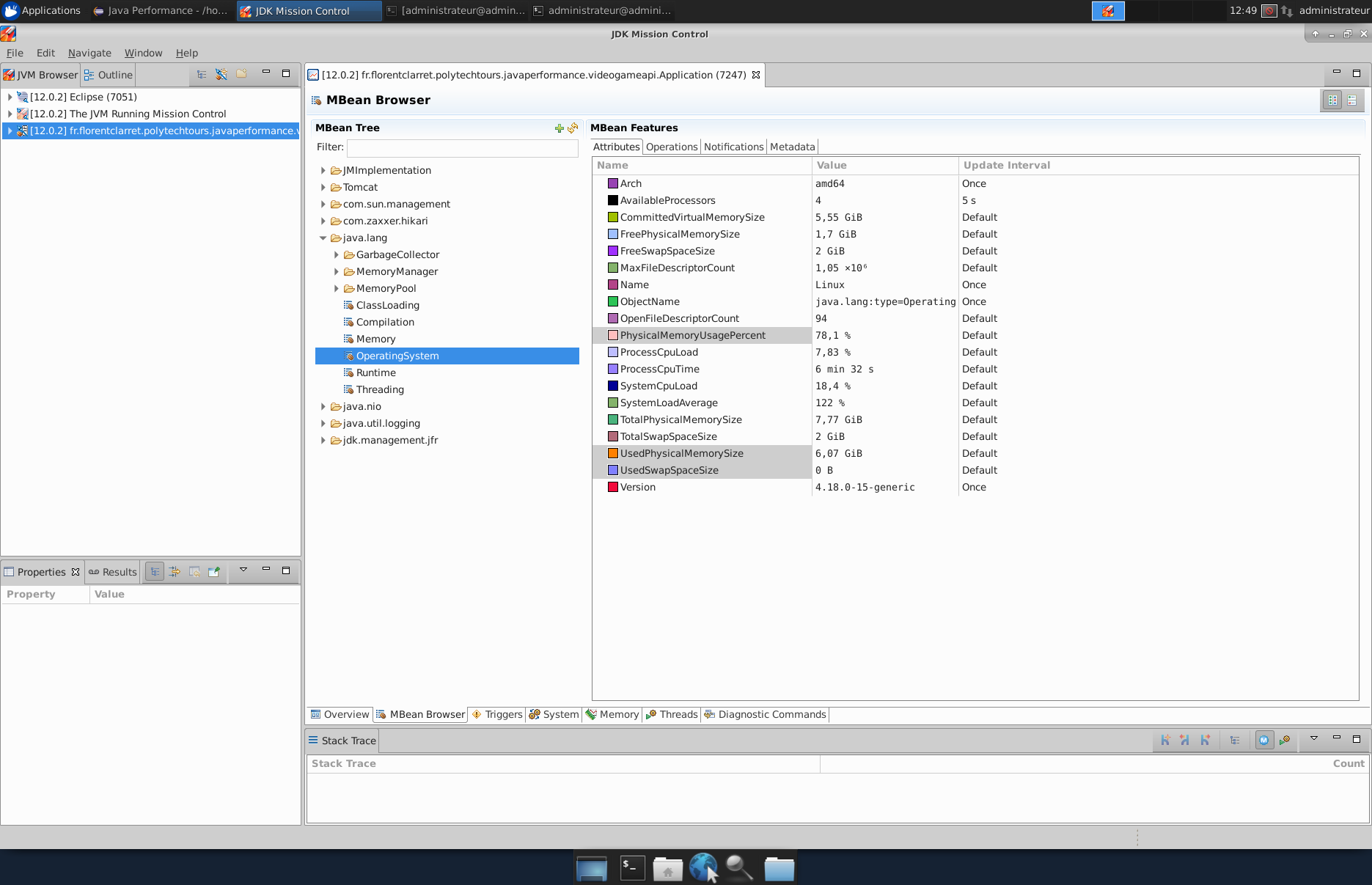


Analyse :

* 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:

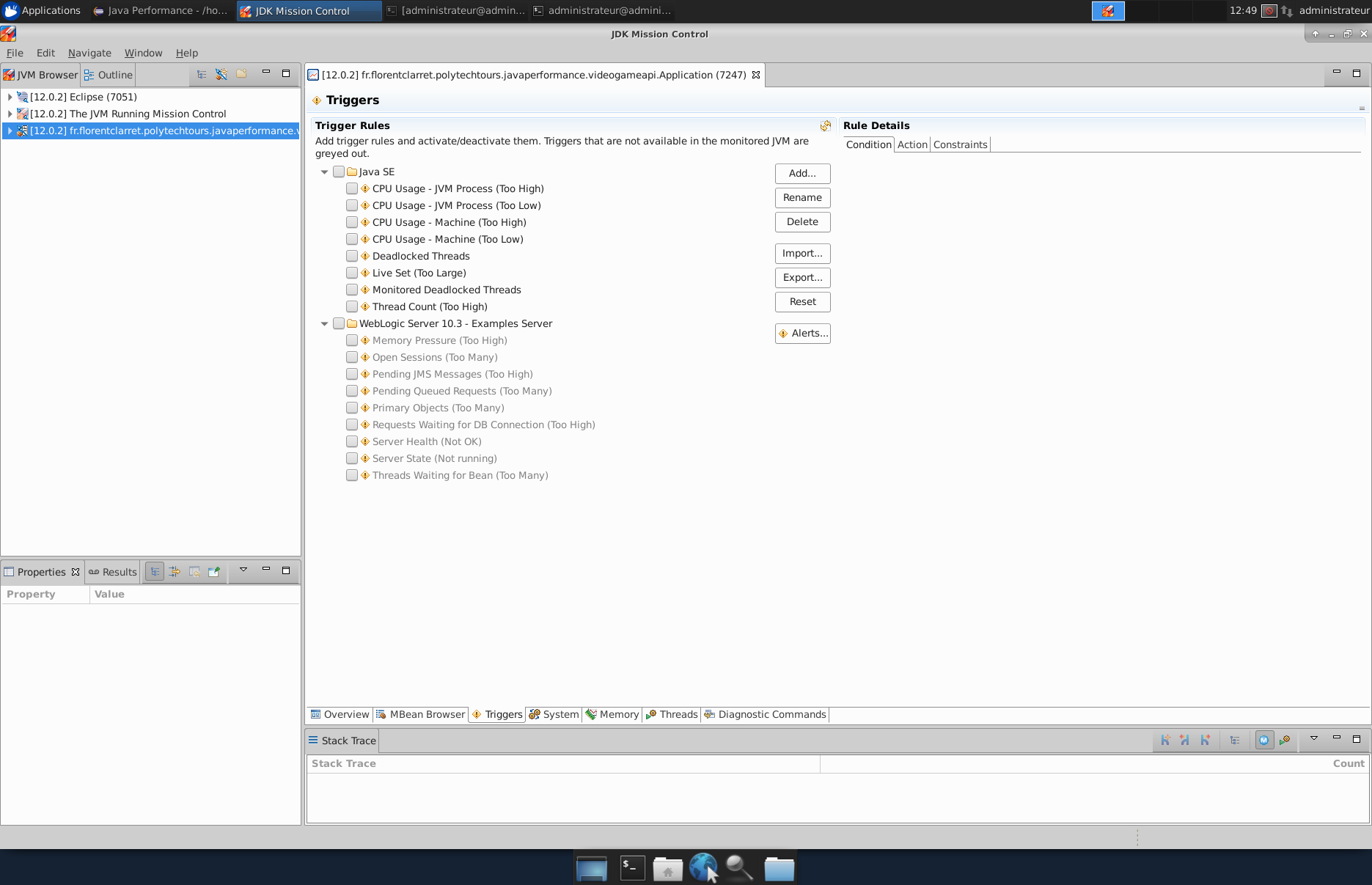
La courbe d'utilisation du processeur montre clairement que l'utilisation du processeur testée a atteint 10% et que l'utilisation de la mémoire était d'environ 210 Mo. En bas, vous avez le choix entre des balises, des balises de propriété système, des balises d'informations JVM, etc. Les boutons sur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

Navigateur Mbean :



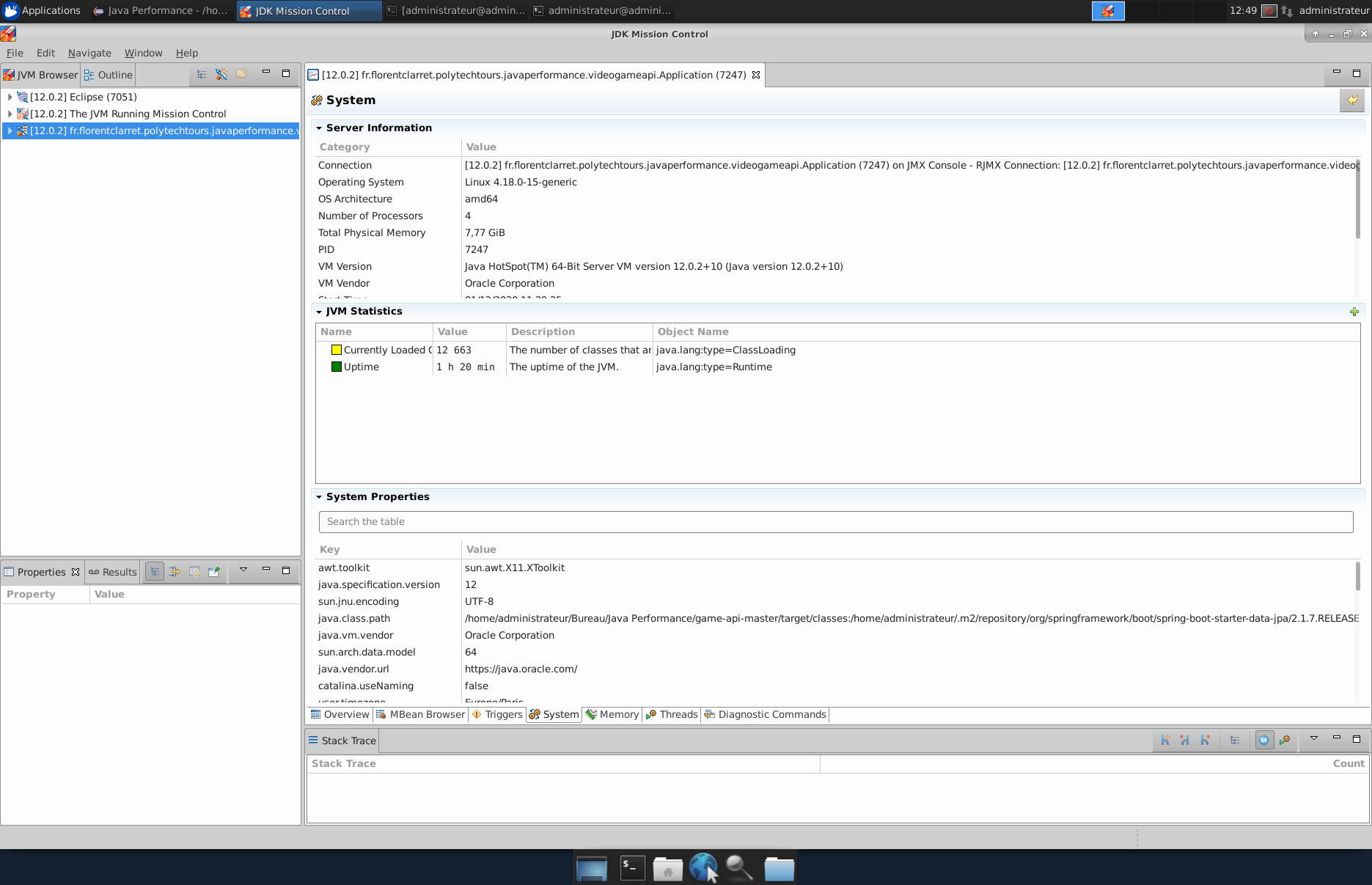
Mbean présente les informations système que jvm peut fournir à la console JMX. L'état de ces beans peut changer avec le fonctionnement du système. Vous pouvez l'ajouter à votre propre vue créée en fonction des informations MBean auxquelles vous devez prêter attention.

Déclencheur :



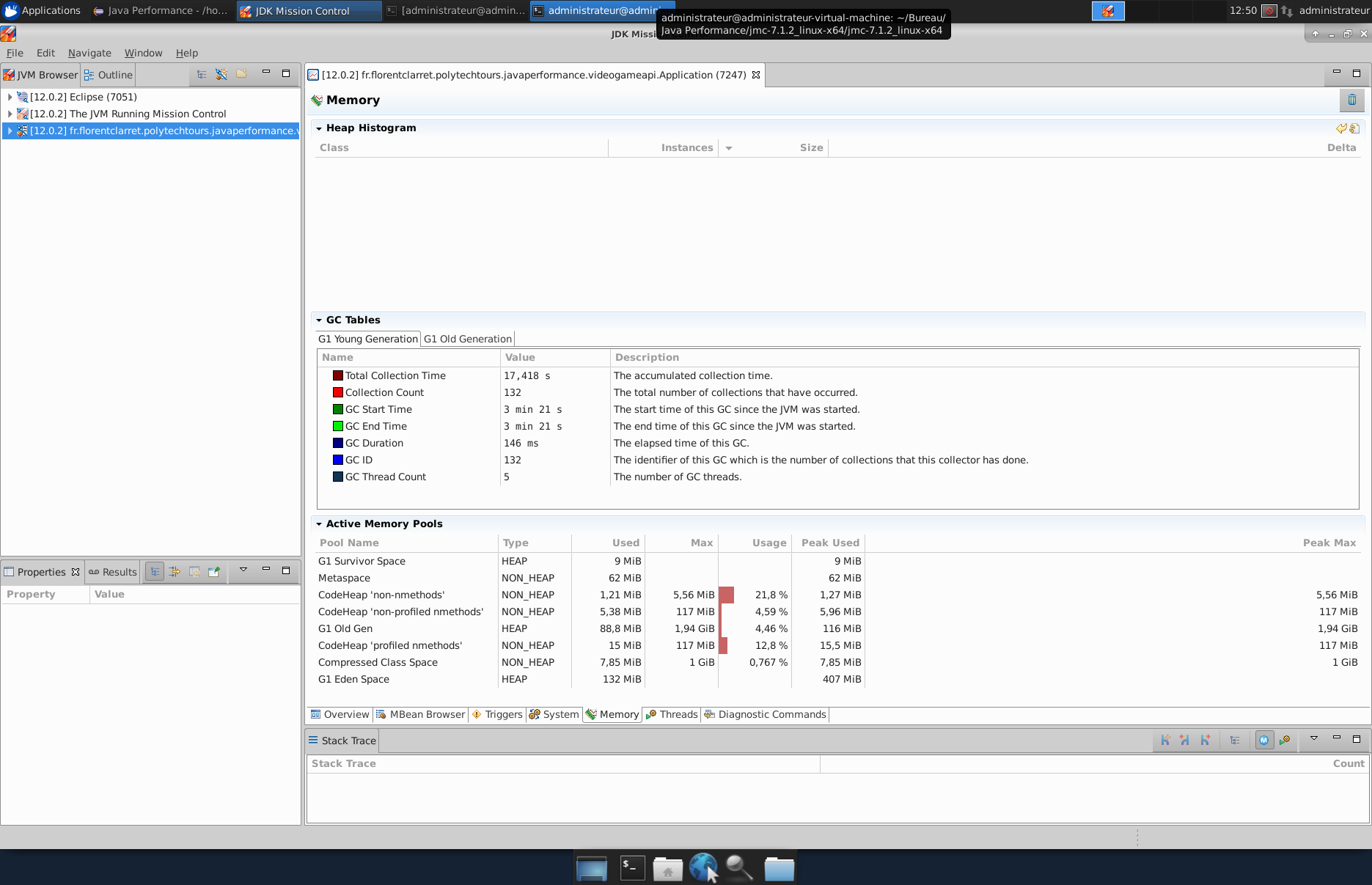
Il est pratique de surveiller et de rappeler un seuil d'état du système via le déclencheur.

Système :



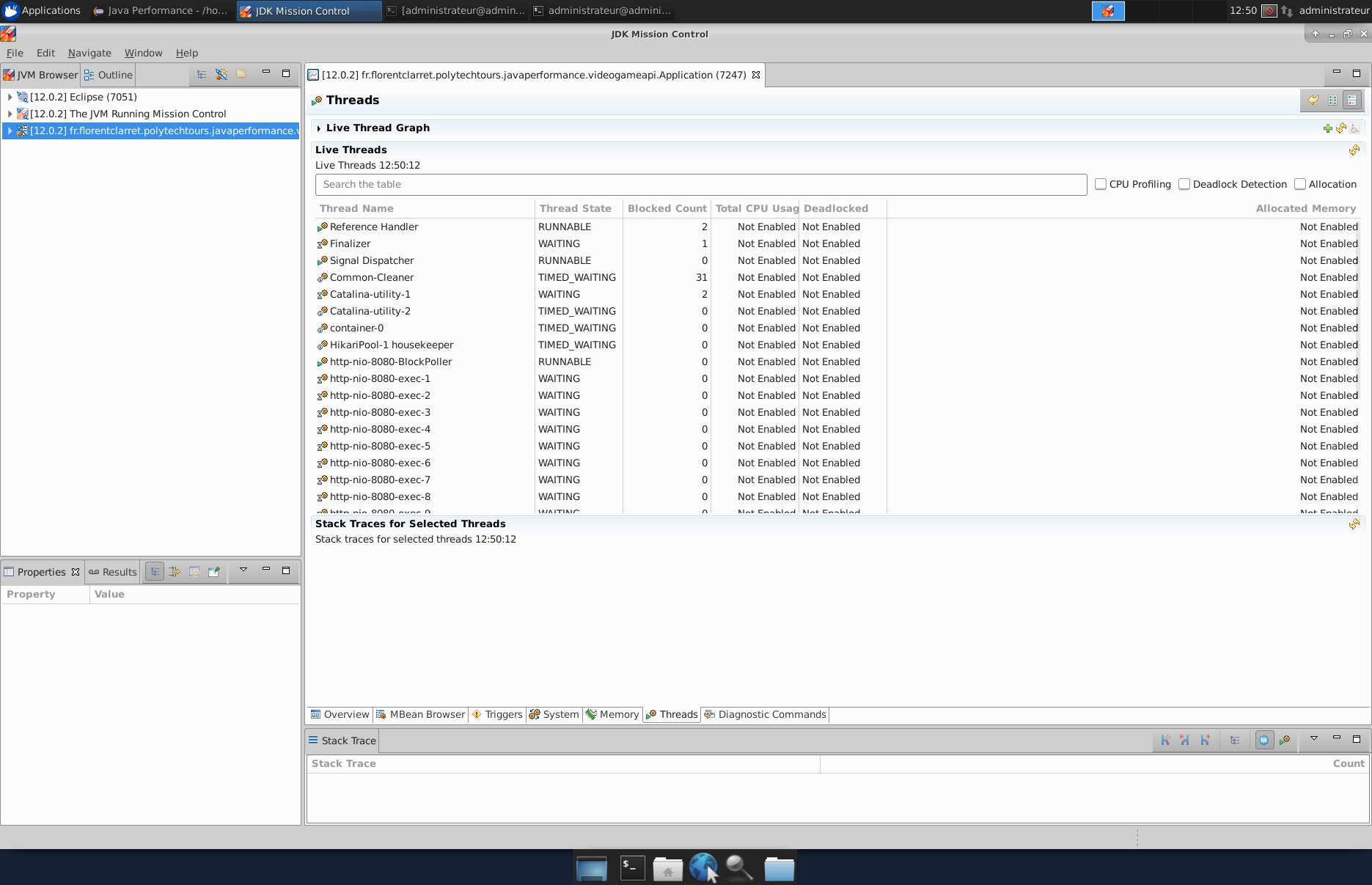
Dans la vue système, vous pouvez afficher les informations de configuration de l'application jvm, et vous pouvez afficher les propriétés de java par filtrage de nom, par exemple: java.class. Vous pouvez afficher le chemin de la classe de configuration de l'application java.

Mémoire :



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

Threads :



Diagnostic :

