Documentation de la Tâche 3

Richard Lao (20278343) et Paul Boulesteix (20198839)

12 novembre 2024

1 Introduction

La **Tâche 3** consiste à effectuer des tests sur divers environnements en utilisant différentes configurations de la JVM (Java Virtual Machine). L'objectif principal est d'évaluer l'impact de divers **flags JVM** sur la qualité, la performance et l'observabilité de l'application. Cette documentation détaille les critères respectés pour accomplir cette tâche, ainsi que les modifications apportées aux actions GitHub pour supporter cette configuration. Le code source de cette tâche est disponible sur GitHub :

— Workflow GitHub Actions : ICI
— Code complet : ICI

1.1 Flags JVM Diversifiés

Description : L'action exécute la compilation et les tests en utilisant **6 flags JVM différents**, chacun appartenant à une catégorie distincte pour assurer une couverture variée des fonctionnalités JVM.

Flags Utilisés :

Nom	Flag JVM
Utiliser Garbage Collector G1	`-XX:+UseG1GC`
Définir CICompilerCount à 2	`-XX:CICompilerCount=2`
Activer GC Details Printing	`-XX:+PrintGCDetails`
Désactiver les gestionnaires de signaux utilisateur	`-XX:-AllowUserSignalHandlers`
Définir le chemin du HeapDump	`-XX:HeapDumpPath=./heapdumps`
Options de mémoire et déduplication de chaînes	`-Xms1g -Xmx2g -XX:+UseCompressedOops -XX:+UseStringDeduplication`

Justification : Chaque flag a été sélectionné pour sa capacité à influencer différents aspects de l'exécution de l'application, tels que la gestion de la mémoire, les performances du compilateur, et la capacité de diagnostic.

1.2 Structure des Logs

Description : L'action génère des **logs clairs et structurés** qui documentent quels flags sont exécutés lors de chaque build. Pour améliorer la lisibilité et la traçabilité des logs, **les noms descriptifs des flags ont été ajoutés dans le code**. Cela permet de facilement identifier quelle configuration est utilisée lors de chaque exécution.

Exemple de Log :

```
Building and Testing with flag -XX:+UseG1GC (Utiliser Garbage
   Collector G1)
...
Building and Testing with flag -XX:CICompilerCount=2 (D finir
   CICompilerCount    2)
...
```

1.3 Motivation et Justification des Flags

Description : La documentation comprend une section qui **justifie le choix de chaque flag** en expliquant son impact potentiel sur la qualité, la performance et l'observabilité de l'application. Les flags sélectionnés couvrent différents types de configurations JVM pour assurer une évaluation complète et diversifiée.

Justification des Flags JVM

1. **-XX :+UseG1GC**
   - **Type :** Gestion du Garbage Collector
   - **Impact :** Optimisation de la gestion du garbage collector pour réduire les pauses de l'application.
   - **Qualité :** Améliore la réactivité et la stabilité de l'application en minimisant les interruptions liées au nettoyage de la mémoire.
2. **-XX :CICompilerCount=2**
   - **Type :** Configuration du Compilateur
   - **Impact :** Limitation du nombre de threads utilisés par le compilateur Just-In-Time (JIT).
   - **Performance :** Réduit la contention des ressources sur les machines avec des capacités multi-thread limitées, optimisant ainsi l'utilisation des ressources CPU.
3. **-XX :+PrintGCDetails**
   - **Type :** Logging du Garbage Collector
   - **Impact :** Active l'affichage détaillé des logs du garbage collector.
   - **Observabilité :** Facilite le diagnostic des problèmes liés à la mémoire en fournissant des informations détaillées sur les opérations de collecte des ordures.
4. **-XX :-AllowUserSignalHandlers**
   - **Type :** Gestion des Signaux Utilisateur
   - **Impact :** Désactive les gestionnaires de signaux utilisateur, renforçant la sécurité et la stabilité en empêchant les applications de définir leurs propres gestionnaires de signaux.
   - **Qualité :** Réduit les risques de comportement inattendu dû à des gestionnaires de signaux non fiables.
5. **-XX :HeapDumpPath=./heapdumps**
   - **Type :** Gestion des Dumps de Mémoire
   - **Impact :** Définit le chemin où les dumps de heap seront enregistrés en cas de crash ou d'erreur.
   - **Observabilité :** Permet une analyse post-mortem des problèmes de mémoire en facilitant l'accès aux dumps de heap.
6. **-Xms1g -Xmx2g -XX :+UseCompressedOops -XX :+UseStringDeduplication**

- **Type :** Optimisation de la Mémoire
- **Impact :**
  - `-Xms1g` et `-Xmx2g` : Définit la taille initiale et maximale du heap, optimisant l'utilisation de la mémoire.
  - `-XX:+UseCompressedOops` : Active les pointeurs compressés, réduisant l'empreinte mémoire des objets.
  - `-XX:+UseStringDeduplication` : Active la déduplication des chaînes, économisant de la mémoire en évitant les duplications inutiles.
- **Performance :** Améliore l'efficacité de l'utilisation de la mémoire et réduit la pression sur le garbage collector.
- **Qualité :** Contribue à une meilleure gestion de la mémoire, diminuant les risques de fuites et optimisant les performances globales.

Diversité des Types de Flags : Les flags sélectionnés couvrent plusieurs types de configurations JVM, notamment la gestion du garbage collector, la configuration du compilateur, le logging, la gestion des signaux utilisateur, la gestion des dumps de mémoire, et l'optimisation de la mémoire. Cette diversité garantit une évaluation complète des différents aspects pouvant impacter la qualité, la performance et l'observabilité de l'application.

1.4 Élément d'Humour

Humour :

Pour la partie humoristique, nous avons ajouté une section à la fin de chaque test, qui, selon le succès ou l'échec de celui-ci, affichera un petit bout d'art ASCII différent (Wikipedia : ASCII art).

- Celui en cas de succès est défini comme le mème "LETS GOOOO" (LetGO)
- Celui d'échec, lui, montre un "sad trollge" (Sad trollge)

1.5 Lolcommits

Lolcommits : ICI