LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 5

Simulations de tests de charge

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

Hiver 2023

1 Introduction

Dans ce laboratoire, vous apprendrez ce qu'est un test de performance, comment utiliser un outil de test de performance, comment concevoir un test de performance et exécuter des tests de performance. En particulier, vous effectuerez des tests de charge sur le système Web OpenMRS.

Les tests de performance sont un type de test permettant de déterminer la vitesse d'un ordinateur, d'un réseau ou d'un périphérique. Ils vérifient la performance des composants d'un système en transmettant différents paramètres dans différents scénarios de charge.

Le test de charge est le processus qui simule la charge réelle d'un utilisateur sur une application ou site Web. Il vérifie le comportement de l'application lors de charges normales et élevées. Ce type de test est appliqué lorsqu'un projet de développement touche à sa fin.

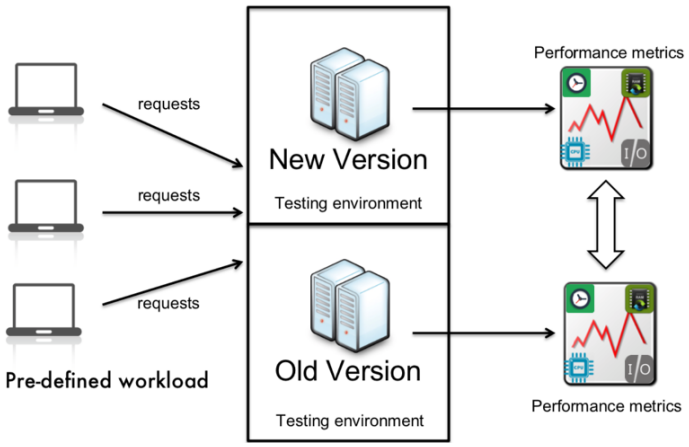
2 Objectifs

Les objectifs généraux de ce laboratoire sont :
- Apprendre à déployer un système pour faire des tests de charge
- Apprendre à configurer un pilote de charge
- Apprendre à exécuter un test de charge
- Pratiquer l'analyse des résultats de ce test de charge

3 Les tests de performance

Les tests de performance/charge consistent en trois processus : (1) concevoir une charge appropriée, (2) exécuter un test de charge et (3) analyser les résultats d'un test de performance/charge.

En pratique, les testeurs configurent l'environnement de test et définissent la charge de travail. Ensuite, les mêmes demandes sont envoyées pour tester le système et les mesures de performance sont collectées. Enfin, les testeurs comparent les deux mesures de performance et identifient la régression des performances (le cas échéant). Ci-dessous, une figure montre le mécanisme des tests de performance/charge :



Les outils

Il existe plusieurs outils pour aider à effectuer des tests de performance/charge.

— **JMeter** - Un outil open source qui peut être utilisé pour les tests de performances et de charge afin d'analyser et de mesurer les performances d'une variété de services.

— **LoadRunner** - Une version de test de performances d'entreprise de Loadrunner et une plate-forme ont permis la normalisation mondiale.

— **ReadLine13** - Une plate-forme de test de charge qui apporte la puissance à faible coût du cloud à JMeter et à d'autres outils de test de charge open source.

Vous pouvez consulter ce lien pour plus de détails : https://jmeter.apache.org/usermanual//

Outils Requis

Veuillez vous assurer que **Python**, **Java JDK**, **Apache JMeter**, et **psutil** sont installés sur votre ordinateur.

Système d'exploitation : macOS/Windows (Linux non testé)

Remarque : macOS 10.15 (Catalina) désactive les applications 32 bits. Par conséquent, si votre système est macOS 10.15 (ou+), veuillez installer docker. https://hub.docker.com/editions/community/docker-ce-desktop-mac

Java : version 1.8. Télécharger ici.

Python : version 3.7 (ou+)

Apache JMeter : version 5.2.1

Psutil : dernière version https://psutil.readthedocs.io/

Remarque : seul l'environnement ci-dessus est testé, nous ne savons pas si d'autres versions fonctionnent.

Les tâches

Partie A - *Déployer le système autonome OpenMRS et utiliser Jmeter pour effectuer une charge de travail stable*

Remarque : Uniquement pour macOS 10.15+ (ne fonctionne pas avec processeur M1 & M2) : veuillez ignorer les étapes a,b,c. Si votre système est macOS 10.15, veuillez utiliser le projet docker à partir du zip du labo (demo.zip), puis décompressez le fichier demo.zip. Veuillez accéder au dossier de démonstration et utiliser les commandes ci-dessous pour déployer OpenMRS :

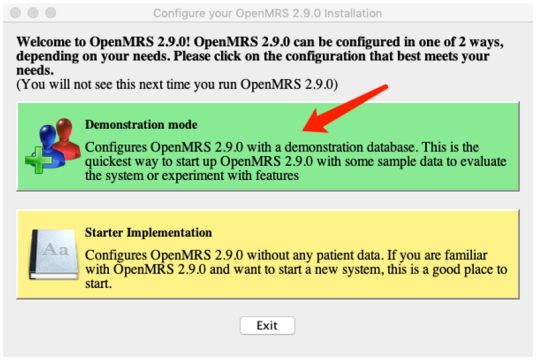
```
$ docker pull openhmis/openmrs-docker
$ docker-compose down -v
$ OPENMRS_VERSION=2.5 docker-compose up
```

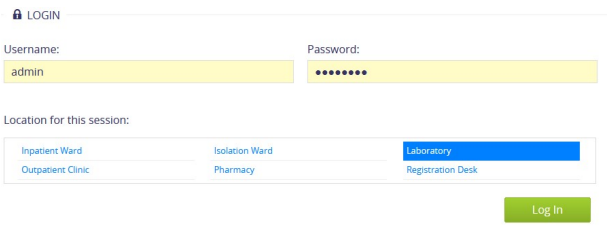
Remarque Windows : Si votre système est Windows, veuillez démarrer OpenMRS dans un terminal avec le rôle d'administrateur

(a) Décompressez le fichier loadTests.zip dans le zip du labo et décompressez les deux sous-fichiers apache-jmeter-5.2.1.zip et OpenMRS-2.9.0.zip.

(b) Démarrez le système autonome OpenMRS *(cela peut prendre quelques minutes)*
- Accédez au dossier décompressé OpenMRS-2.9.0 et exécutez la commande suivante :
  - `java -jar openmrs-standalone.jar`

(c) Vérifiez le serveur OpenMRS
- Accédez à `http://localhost:8081/openmrs-standalone` dans votre navigateur et connectez-vous en tant que *admin* et *Admin*123 pour l'ID utilisateur et le mot de passe, respectivement.
- Remarque : macOS 10.15, veuillez vous rendre sur http ://localhost :8081/openmrs, idem pour les dernières étapes.





(d) Lancez JMeter :
- Ouvrez un autre terminal, accédez au dossier JMeter et lancez jmeter
- `$./bin/jmeter (mac os)`
- Double-cliquez sur "ApacheJMeter.jar" (Windows)

Essayez d'utiliser d'utiliser le script `jmeter Steady-Workload.jmx` à partir du zip du labo (File - Load - Cliquez avec le bouton droit Steady-Workload - Start). Vous pouvez voir certaines instructions et les exemples sur la façon de créer ce script dans le steady-workload-instructions.pdf à partir du zip du labo. Vous pouvez utiliser ces exemples pour compléter les prochaines tâches.

(e) Ajoutez une nouvelle configuration dans $Steady-Workload$ à partir du fichier csv (CSV Data Set Config) nommée $PersonIDList$. Utilisez le fichier $data/personid.csv$.

(f) Ajouter une nouvelle requête HTTP (HTTP request) dans $Steady-Workload$ nommée $SearchPersonByUUID$. C'est une requête GET qui doit retouner l'information sur la personne selon son ID. Utilisez le fichier csv avec les IDs que vous avez ajouté dans la configuration $PersonIDList$. Vous pouvez trouvez le format

de requête d'envoyer ici `https://wiki.openmrs.org/display/docs/REST+Web+Service+Resources+in+OpenMRS+1.9`.

(g) Ajouter une nouvelle requête HTTP (HTTP request) dans $Steady - Workload$ nommée $EditPersonByUUID$. C'est une requête $POST$ qui doit modifier l'information de la personne selon son ID. Pour modifier l'information comme le nom et prénom vous pouvez utilise la fonction du JMeter $\_\_RandomString()$ et $\_\_Random()$ pour modifier l'age. Vous pouvez trouver plus d'information sur ces fonctions ici : `https://jmeter.apache.org/usermanual/functions.html#__Random`.

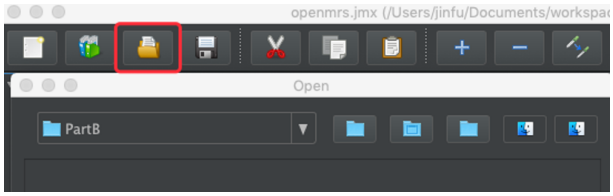
(h) Enregistrez le plan de test (File - Save) comme $testA.jmx$ à l'emplacement de votre choix.

(i) Enregistrez les résultats d'exécution de plan de test comme $resA.jtl$ à l'emplacement de votre choix.

Partie B - *Utiliser Jmeter pour effectuer une charge de travail aléatoire*

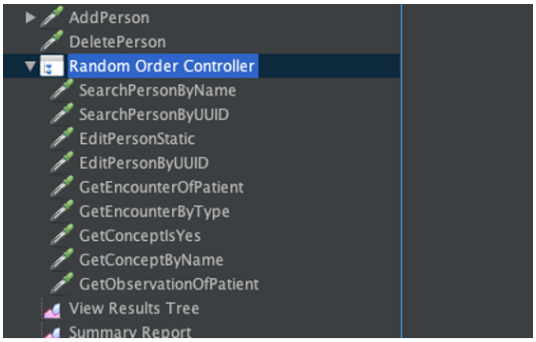
(a) Lancez JMeter
- Allez dans le dossier JMeter et exécutez ./bin/jmeter
- `$./bin/jmeter`

(b) Importez le script Jmeter de la partie A dans Jmeter.



(c) Nommez le groupe de threads "Random-Workload"

(d) Ajoutez Random Order Controller sous Random-Workload :
- Cliquez avec le bouton droit sur Random-Workload - Add - Logic Controller - Random Order Controller
- Placez les requêtes HTTP sous le Random Order Controller selon la capture d'écran en bas :
- https://jmeter.apache.org/usermanual/test_plan.html#logic_controller



(e) Nous sommes maintenant prêts à enregistrer notre scénario de test pour "Random-Workload" :
— Cliquez avec le bouton droit sur Random-Workload - Start

(f) Enregistrez le plan de test (File - Save) comme *testB.jmx* à l'emplacement de votre choix.

(g) Enregistrez les résultats d'exécution de plan de test comme *resB.jtl* à l'emplacement de votre choix.

Partie C - *Utilisez JMeter pour effectuer un test de charge sur OpenMRS en mélangeant deux charges de travail*

Les deux charges de travail :

Workload-1 : SearchPerson - EditPerson - GetEncounter...

Workload-2 : AddPerson - DeletePerson - SearchPerson - GetObservationOfPatient...

(a) Lancez JMeter

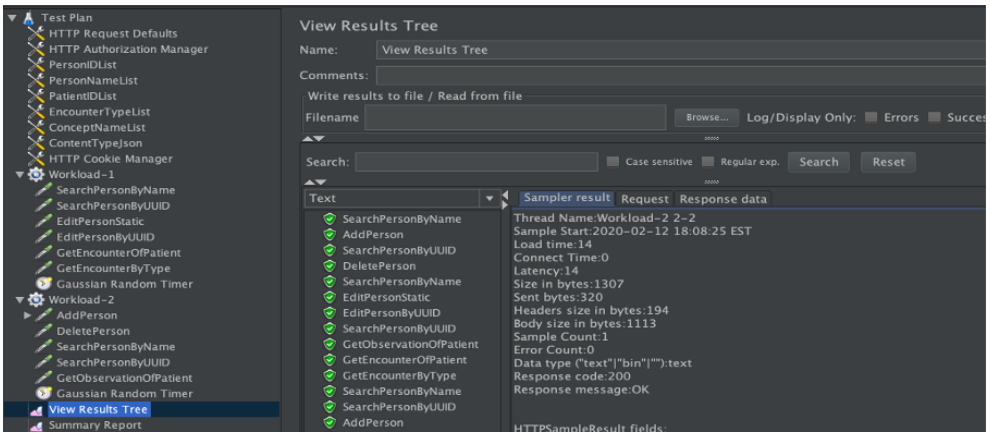
(b) Créez deux groupes de threads "Workload-1" et "Workload-2"

(c) Pour chaque charge de travail, ajoutez les requêtes qu'on a crée dans la partie A. La différence est que vous devez créer différentes requêtes HTTP en fonction de la charge de travail-1 ou de la charge de travail-2. Vous devez créer les workloads selon la capture d'écran en bas ou selon votre choix. La quantité minimale des étapes (requêtes HTML) dans le workload est 4.

(d) Nous sommes maintenant prêts à enregistrer notre plan de test.

(e) Enregistrez le plan de test (File - Save) comme *testC.jmx* à l'emplacement de votre choix.

(f) Enregistrez les résultats d'exécution de plan de test comme *resC.jtl* à l'emplacement de votre choix.



Partie D - *Utilisez JMeter pour effectuer un test de charge sur OpenMRS avec une* ***charge de travail variée*** *à l'aide du mode CLI et surveillez les performances du processus openmrs-standalone.jar à l'aide de l'outil* ***psutil***

Charges de travail variées : La première minute, 5 utilisateurs accèdent à OpenMRS. La deuxième minute 10 utilisateurs accèdent à OpenMRS. La troisième minute, 15

utilisateurs accèdent à OpenMRS. La quatrième minute 20 utilisateurs accèdent à OpenMRS.

(a) Cette partie peut être effectuée en créant quatre groupes de threads (un groupe de threads représente un type de charge de travail) sous Plan de test. Pour chaque charge de travail, spécifiez le nombre d'utilisateurs.

(b) Configurez l'ordre d'exécution de ces quatre charges de travail :
- Cliquez sur Plan de test
- Cochez "Run Thread Groups consecutively"

(c) Enregistrez le plan de test (File - Save,testD.jmx) à l'emplacement de votre choix.

(d) Effectuez une charge de travail variée à l'aide du mode CLI JMeter :
- Ouvrez un terminal et accédez au dossier JMeter
- `$ ./jmeter -n -t vary-workload.jmx -l testresults.jtl`
  - -n : spécifie que JMeter doit s'exécuter en mode non-gui
  - -t : Nom du fichier JMX qui contient le plan de test
  - -l : nom du fichier JTL (journaux de texte JMeter) pour enregistrer les résultats
  - -j : nom du fichier journal d'exécution de JMeter

(e) Capturez les données de performances du processeur avec *psutil*
- Obtenez l'ID de processus du processus openmrs-standalone.jar, puis utilisez psutil pour capturer les données de performances du processeur comme utilisation du CPU et mémoire. (Exemple dans perfMonitor.py dans le zip du labo). Sur *Windows* vous pouvez utilisez la commande :
  jps -l | findstr "org.openmrs.standalone.ApplicationController"
  pour trouver le PID du processus de OpenMRS.
- Enregistrez le plan de test (File - Save) comme $testD.jmx$ à l'emplacement de votre choix.
- Enregistrez les résultats d'exécution de plan de test comme $resD.jtl$ à l'emplacement de votre choix.

Rapport Répondre au questions suivantes :

Question 1 : Dans la partie A, combien de demandes d'URL sont envoyées ? Toutes les demandes sont-elles traitées avec succès ? Vous devez fournir des captures d'écran de votre plan de test, avec les captures d'ecran pour les parties A.f, A.g et A.h (votre plan de test), ainsi les captures des résultats e.g. de View Results Tree et Summary Report dans votre rapport. Ajouter aussi les fichiers $testA.jmx$ et $resA.jtl$.

Question 2 : Pour effectuer une charge de travail aléatoire dans la partie B, nous utilisons Random Order Controller. Pouvez-vous utiliser d'autres composants pour effectuer des charges de travail aléatoires ? Vous devez fournir une capture d'écran de **deux possibilités** dans votre rapport. Ajouter aussi les fichiers $testB.jmx$ et $resB.jtl$ pour chaque charge de travail (avec les noms un peu différents).

Question 3 : Fournissez votre script de plan de test dans la partie C nommée $testC.jmx$, et votre fichier de résultat $resC.jtl$. Ajoutez les captures d'écran avec les tests et les résultats.

Question 4 : Collectez les données de performances du processeur (en mode utilisateur) de la partie D et fournissez un graphique d'utilisation du processeur. Ajouter aussi les fichiers *testD.jmx* et *resD.jtl* et votre script pour créer le graphique et collecter les donnes de la performance (cpu, mémoire).

4 Livrables attendus

Les livrables suivants sont attendus :

- Un rapport pour le laboratoire. Le rapport doit contenir :
  - Vos réponses à la question 1 : (4 points).
  - Vos réponses à la question 2 : (4 points).
  - Vos réponses à la question 3 : (3 points).
  - Vos réponses à la question 4 : (8 points).
  - Qualité du rapport : (1 point).
- Le dossier COMPLET contenant la réponse à chaque question et les captures d'écran (comme indiqué dans chaque question), ainsi que le code source nécessaire pour effectuer les questions 3 et 4. Vous pouvez créer le fichier séparé pour chaque question. Si le code est manquant pour une question qui le demande, la question ne sera pas notée.

Le tout à remettre dans une seule archive **zip** avec le titre matricule1_matricule2_lab1.zip sur Moodle. Seulement une personne de l'équipe doit remettre le travail.

Le rapport doit contenir le titre et numéro du laboratoire, les noms et matricules des coéquipiers ainsi que le numéro du groupe.

5 Information importante

1. Consultez le site Moodle du cours pour la date et l'heure limites de remise des fichiers.
2. Un retard de [0,24h] sera pénalisé de 10%, de [24h, 48h] de 20% et de plus de 48h de 50%.
3. Aucun plagiat n'est toléré. Vous devez soumettre uniquement le code et les rapports de couverture de code réalisé par les membres de votre équipe.