Travail personnel : test des performances d'un serveur de calcul

1 Description du test

1.1 Objectifs

Le but de ce travail est de tester les performances d'un serveur de calcul accessible via le protocole UDP. Le logiciel serveur et le logiciel d'injection sont fournis. Le travail consiste :

- à mettre en place l'infrastructure de test : système sous test et système d'injection de charge ;
- à exécuter les plans de test fournis ;
- à analyser les mesures produites et produire un rapport de test.

1.2 Système sous test

Le Système Sous Test (SST) est un serveur de calcul qui s'exécute sur une machine dédiée à ce serveur. Il reçoit via UDP des opérations simples (addition `+`, soustraction `-`, multiplication `*`, division entière `/`) représentées en notation infixe sous forme d'une chaîne de caractères encodée en ASCII, calcule et retourne le résultat au client UDP qui a émis le paquet, également sous la forme d'une chaîne de caractères encodée en ASCII. Le résultat est préfixé par le signe `=`. Si la requête est mal formatée, un message d'erreur est retourné : `invalid number`, `invalid operation`.

Attention, il n'est pas impossible que le serveur fasse des erreurs de calcul...

1.3 Système d'injection de charge

Le système d'injection de charge s'exécute sur une machine utilisée exclusivement pour cet usage. L'outil d'injection de charge proposé pour réaliser ce test est le logiciel libre CLIF (clif.ow2.org). Il permet de déployer et contrôler :

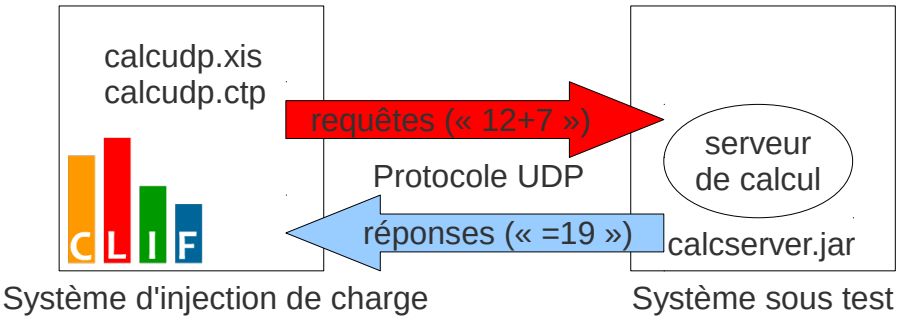
- des *injecteurs de charge*, qui génère des requêtes selon un certain *scénario*, et mesure les temps de réponse de chacune d'entre elles, i.e. le temps écoulé entre l'émission de la requête et la réception de la réponse
- des *sondes*, qui permettent de mesurer la consommation de certaines ressources, comme le processeur, la mémoire, l'interface réseau, etc.

Un test CLIF se définit à l'aide :

- d'un *plan de test* (fichier `.ctp`), qui spécifie la liste des injecteurs et des sondes à déployer
- d'un ou plusieurs scénarios (fichiers `.xis`), chaque scénario spécifiant le trafic de requêtes que doit générer un injecteur.

Un plan de test et un scénario sont fournis sous la forme d'une archive `.zip` de projet de test CLIF.

1.4 Schéma global du test



1.5 Résultats attendus

Suite à l'exécution de plusieurs tests avec différents niveaux de charge (voir section 3), l'analyse demandée est guidée par un certain nombre de questions posées.

Le rapport de test consiste à répondre aux questions posées en section 4. Ce rapport sera à déposer sur Teide (https://intranet.ensimag.fr/teide/interface_view_project.php?id=1601).

2 Mise en place du test

Les fichiers nécessaires sont à télécharger depuis Chamilo :

http://chamilo.grenoble-inp.fr/main/document/document.php?cidReq=ENSIMAG5MMTSL6&id=230134

2.1 Préparer le système sous test

- copier le fichier `calcserver.jar` sur la machine sous test
- lancer le serveur de calcul :

```
$ java -jar calcserver.jar 5432
Calculator listening on UDP socket at address 0.0.0.0/0.0.0.0:5432
```

NB : le paramètre 5432 désigne le numéro de port UDP sur lequel le serveur de calcul se met en écoute. Ce numéro peut être changé sous réserve d'adapter les paramètres du scénario d'injection de charge en conséquence.

2.2 Vérifier le bon fonctionnement du système sous test

- copier le fichier `calcclient.jar` sur la machine d'injection
- lancer le programme client pour vérification :

```
$ java -jar calcclient.jar server_addr server_port 12-7
=5
```

`server_addr` est l'adresse réseau de la machine sous test ;

`server_port` est le numéro de port UDP sur lequel le serveur de calcul écoute (5432 dans l'exemple ci-dessus) ;

les paramètres suivants sont des opérations à soumettre au serveur de calcul.

2.3 Installer la console CLIF

Télécharger l'archive de la console CLIF v2.3.5 correspondant au système d'exploitation/architecture du système sous test et de la machine d'injection (selon les cas, il peut

donc être nécessaire de télécharger 2 archives différentes). Il s'agit des paquets `clif-eclipse-`*`os-arch`* disponibles ici : http://forge.ow2.org/project/showfiles.php?group_id=57

L'installation de CLIF consiste simplement à dézipper l'archive téléchargée, et installer un environnement d'exécution Java de type JRE ou JDK, Oracle ou OpenJDK, en version 1.8 s'il n'est pas déjà présent.

2.4 Installer les plans de test et scénarios CLIF

Lancer la console CLIF (exécutable `clif-console`). Utiliser la fonction d'import de projet Eclipse pour charger le projet de test CLIF incluant les plans de test et les scénarios, à partir de l'archive `calctest.zip`. Dans ce projet, éditer le fichier `calcudp.props` afin de positionner la propriété `host` à l'adresse IP de votre système sous test.

Trois plans de tests sont proposés, permettant de déployer des scénarios avec différents profils de charge : 1, 20 et 40 utilisateurs virtuels, avec des rampes de montée en charge respectivement de 0, 2 et 4 secondes. Le comportement de chaque utilisateur virtuel est le même dans les 3 scénarios : connexion au serveur UDP, puis boucle sur envoi de requête et réception de réponse jusqu'à la fin du test (environ 1 minute).

Outre l'injecteur de charge, chaque plan de test déploie également une sonde pour observer la consommation mémoire de la JVM de l'injecteur. Le but est de vérifier que l'injecteur ne manque pas de ressources mémoire pour s'exécuter.

3 Exécution des tests

3.1 Instructions générales

Pour déployer et exécuter un plan de test (`calcudpXX.ctp`) à partir de la console CLIF :

1. activer la perspective Clif (Window > Open Perspective) si elle ne l'est pas déjà ;
2. déployer le plan de test à l'aide de l'assistant "*New test plan deployment*" ;
3. initialiser le test (bouton `init`) ;
4. démarrer le test (bouton `start`) ;
5. dans la vue monitor, observer les statistiques mobiles des différentes métriques remontées par l'injecteur : temps de réponse moyen, temps de réponse maximum, débit moyen (*action throughput*), écart-type du temps de réponse, nombre de requêtes (*actions*) et d'erreurs cumulé depuis le début du test ;
6. lorsque le test est terminé, déclencher la collecte des données et la production de la synthèse statistique sur les différents types de requêtes (bouton `collect`). Les mesures brutes (temps de réponse de chaque requête, consommation mémoire de la JVM) se trouvent dans le répertoire `report`, dans un sous-répertoire propre à chaque test. La synthèse statistique `quickstats.csv` se trouve dans le répertoire `stats`, dans un sous-répertoire propre à chaque test.
7. renouveler les étapes 3 à 6, afin d'obtenir au final 2 exécutions du plan de test.

IMPORTANT: contrairement à ce qu'indiquent les libellés des métriques de la vue monitor, les statistiques sur les temps de réponse sont exprimées en **microsecondes**, et non en millisecondes.

3.2 Premier test : évaluation du temps de traitement unitaire

Le premier test vise à mesurer le temps de traitement d'une requête isolée, en l'absence de tout autre trafic (pas de traitement parallèle, pas d'attente). Il est donc nécessaire d'avoir un unique utilisateur virtuel, et de s'abstraire de la latence réseau.

Déployer et exécuter le test `calcudp1.ctp` à partir de la console CLIF qui a été installée sur la machine du système sous test.

3.3 Deuxième test : évaluation de la latence réseau

Le deuxième test est identique au premier test, mais exécuté depuis la machine d'injection de charge. Les mesures permettront d'évaluer le temps de latence réseau, connaissant le temps de traitement mesuré précédemment.

Déployer et exécuter le test `calcudp1.ctp` à partir de la console CLIF qui a été installée sur la machine d'injection de charge.

3.4 Troisième test : test en charge avec 20 utilisateurs virtuels

Déployer et exécuter le test `calcudp20.ctp` à partir de la console CLIF qui a été installée sur la machine d'injection de charge.

3.5 Quatrième test : test en charge avec 40 utilisateurs virtuels

Déployer et exécuter le test `calcudp40.ctp` à partir de la console CLIF qui a été installée sur la machine d'injection de charge.

4 Rapport de test

4.1 Description de l'infrastructure de test

Donnez de manière précise les caractéristiques techniques des différents éléments qui composent votre infrastructure de test : machine d'injection et machine sous test (matériel, système d'exploitation), réseau.

4.2 Compréhension des scénarios

L'interprétation des résultats repose en premier lieu sur une bonne compréhension du trafic généré par le scénario. Dans la console CLIF, ouvrir le fichier scénario calcudp1.xis. En utilisant les onglets "Import" et "Behavior calc clients" en bas de l'éditeur, répondre aux questions suivantes :

4.2.1 Le bloc de comportement contenu dans la boucle « while(true) » débute par une instruction timer.period_begin et se termine par une instruction timer.period_end. Ce timer est réglé à 50ms lors de l'import du plug-in. L'instruction timer.period_end a pour effet de réaliser le temps d'attente réglé (50ms) diminué du temps écoulé depuis l'instruction timer.period_begin. Ainsi, chaque itération de la boucle « while(true) » est censée durer 50ms. Combien d'itérations par seconde ce comportement est-il censé effectuer ?

4.2.2 Chaque itération contient une instruction client.send, afin d'envoyer un paquet UDP contenant la requête (une opération extraite du fichier calcudp.csv), puis une instruction client.receive afin d'attendre la réponse du serveur de calcul. Combien de requêtes par seconde ce comportement est-il censé envoyer au serveur ?

4.2.3 Le temps d'attente réalisé par l'instruction timer.period_end est nul dans le cas particulier où le temps écoulé depuis l'instruction timer.period_begin est supérieur ou égal au réglage

(50ms). Si l'on néglige le temps d'exécution des instructions autres que client.receive, quelle hypothèse faut-il faire sur le temps de réponse du serveur pour que le débit attendu puisse être atteint ? Que se passe-t-il cette hypothèse n'est pas respectée ?

4.2.4 Du point de vue de CLIF, les instructions client.send et client.receive sont toutes les deux vues comme des « actions », en particulier par le monitoring qui les agrège indifféremment dans le graphique des statistiques mobiles de l'injecteur. A quel débit moyen d'actions/s doit-on s'attendre sur ce graphique ?

4.2.5 En début de comportement, l'instruction client.setTimeOut fixe le délai maximum d'attente de la réponse UDP à 5 secondes. Par conséquent, toute attente supérieure à 5 secondes sur l'instruction client.receive s'interrompt et un rapport de requête en erreur est enregistré dans le fichier « action » de l'injecteur. De même, le graphique des statistiques mobiles du monitoring affiche des erreurs. Sachant que UDP est un protocole non fiable, quelles sont les 3 raisons possibles pour que ce mécanisme de time-out se déclenche ?

4.2.6 Le profil de charge associé au comportement permet de spécifier le nombre d'utilisateurs virtuels actifs au cours du temps. La fin du profil de charge signifie que tous les utilisateurs virtuels actifs doivent s'arrêter dès que l'instruction en cours se termine. En consultant ce profil de charge, quelle borne supérieure peut-on donner à la durée d'exécution de ce scénario ?

4.3 Premier test (1 utilisateur virtuel local)

Copier-coller les 2 tableaux `quickstats.csv` obtenus.

Observer les synthèses statistiques de temps de réponse au niveau des requêtes de type RECEIVE.

4.3.1 Sachant qu'il n'y a ni latence réseau (injection locale), ni contention ni délais de file d'attente (pas de concurrence), en déduire le temps de traitement moyen d'une requête.

Le serveur UDP n'a qu'un seul fil d'exécution pour traiter les requêtes. En considérant le temps de traitement moyen d'une requête, pouvez-vous déduire :

4.3.2 le débit maximum théorique admissible par le serveur en nombre de requêtes par seconde ?

4.3.3 le nombre d'utilisateurs virtuels que ce serveur de calcul pourrait admettre au maximum avant saturation ?

4.3.4 le taux d'utilisation du serveur de calcul dans ce test (pourcentage du temps passé à traiter des requêtes) ?

Dans la définition du comportement des utilisateurs virtuels (`calcudp1.xis`), les deux blocs "`if-then`" imbriqués servent à vérifier la validité du résultat retourné par le serveur. En effet, les opérations soumises au serveur sont extraites du fichier « `calcudp.csv` », qui contient également les résultats des opérations, ce qui permet de faire la vérification. Lorsqu'une erreur est détectée, l'instruction `common.log` est exécutée, ce qui a pour effet de produire un rapport de requête de type « LOG » en erreur (champ `successful` à `false`) au sein du fichier `action` de l'injecteur, ainsi que l'apparition d'une erreur dans le graphique des statistiques mobiles du monitoring.

4.3.5 Combien d'erreurs de calcul ont été commises, et quel pourcentage cela représente-t-il par rapport au nombre total de réponses reçues ?

4.4 Deuxième test (1 utilisateur virtuel distant)

Copier-coller les 2 tableaux `quickstats.csv` obtenus.

Observer les synthèses statistiques de temps de réponse au niveau des requêtes de type RECEIVE.

4.4.1 Sachant qu'il n'y a ni contention ni délais de file d'attente (pas de concurrence), et connaissant le temps de traitement moyen d'une requête évalué précédemment, donner une estimation de la latence réseau entre la machine d'injection de charge et la machine sous test.

4.5 Troisième test (20 utilisateurs virtuels distants)

Copier-coller les 2 tableaux `quickstats.csv` obtenus.

Observer les synthèses statistiques de temps de réponse au niveau des requêtes de type RECEIVE.

4.5.1 Donner une estimation du temps de réponse moyen.

4.5.2 En supposant que la latence réseau est fixe (pas de contention réseau), donner une estimation du temps de traitement. Le comparer au temps de traitement unitaire (1 seul utilisateur virtuel).

4.5.3 Combien d'erreurs de calcul ont été commises, et quel pourcentage cela représente-t-il par rapport au nombre total de réponses reçues ?

4.5.4 Y a-t-il d'autres erreurs qui s'ajoutent aux erreurs de calcul ? Si oui, à quoi correspondent-elles ?

4.6 Quatrième test (40 utilisateurs virtuels distants)

Observer en particulier ce qui s'affiche dans la vue « console ».

4.6.1 Le test fonctionne-t-il ? Si non, pourquoi ?

4.6.2 Que faudrait-il faire pour pouvoir injecter davantage de charge ?