Testez vos applications web avec JMeter

GNU/Linux Magazine n° 153 | octobre 2012 | Par Hage Jérémy

Réseau

DANS CET ARTICLE

À la fin du développement d'un site internet, il est toujours intéressant voire essentiel de tester son application avant sa mise en production en lui faisant subir un test de charge afin d'évaluer ses performances. Pour ce faire, je vous propose de découvrir JMeter, un outil permettant d'effectuer ce type de tests.

L'un des critères les plus importants pour qu'un site soit apprécié de ses visiteurs est que son chargement soit performant. En effet, bien que le plus important sur un site web soit le contenu qu'il propose, les internautes sont généralement vite lassés par les éventuelles lenteurs qu'ils peuvent rencontrer. Pour ne pas avoir de mauvaises surprises, et ne pas risquer de perdre des visiteurs, il est important de savoir si l'architecture dédiée à votre site est adaptée aux exigences de votre application en matière de consommation des ressources en situation de fort trafic.

Un test de charge vous permettra donc d'avoir une estimation du nombre de requêtes par seconde que votre infrastructure pourra supporter et d'avoir une idée du nombre de visiteurs simultanés que votre site pourra accueillir.

Bien qu'ils puissent être utilisés à tout moment, ces tests sont généralement effectués avant la mise en production d'un site ou d'une nouvelle version d'un site, ou encore avant de grands événements tels que les soldes pour les sites marchands, les grands championnats pour les sites de sport, les élections pour les sites d'info, etc. Les

AU SOMMAIRE DU MÊME NUMÉRO

résultats de ces tests permettent d'anticiper d'éventuels problèmes de charge des serveurs et ainsi d'ajuster les ressources matérielles au trafic attendu. Ils peuvent également mettre en avant les possibles faiblesses de l'application testée et orienter les recherches d'optimisation.

Après une présentation de JMeter, nous verrons la façon de créer un scénario et l'exécuter. Nous verrons ensuite comment exploiter et interpréter les résultats que vous obtiendrez.

1. Présentation de JMeter

JMeter est un logiciel permettant d'effectuer des tests de charge de plusieurs services différents (ftp, mail, ldap, …), mais nous nous concentrerons ici sur la mise en œuvre d'un test de charge d'une application web. JMeter est écrit en Java et son développement est géré par la fondation Apache.

Son utilisation consiste dans un premier temps à établir un scénario de test, c'est-à-dire un exemple de session de navigation d'un visiteur, puis à faire répéter ce scénario autant de fois et par autant de visiteurs (ou *threads*) simultanés que vous l'aurez décidé.

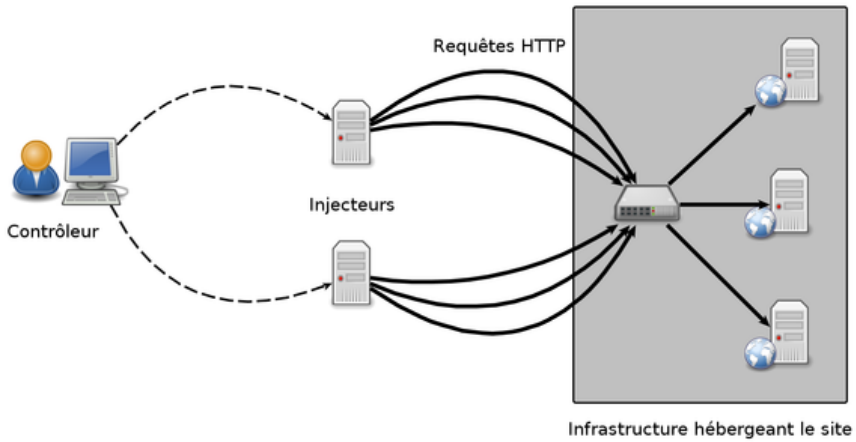


Illustration 1 : Fonctionnement global de JMeter

La figure 1 représente de manière générale le fonctionnement de JMeter. Le contrôleur est la machine à partir de laquelle vous avez créé votre scénario. Cette instance vous permettra également de contrôler les injecteurs qui eux se chargent d'exécuter le scénario. Lorsque vous fournissez un scénario à plusieurs injecteurs à la fois, le même scénario est exécuté par chacun d'entre eux, c'est-à-dire qu'il n'y a pas de répartition du nombre de threads entre les 2 instances.

Ce schéma représente le cas idéal où vous posséderiez des ressources matérielles disponibles. Seulement ce n'est pas toujours le cas et on pourrait plus simplement se contenter d'une seule instance de JMeter à partir de laquelle vous élaborerez le scénario et lancerez le test de charge. Cependant, il faut noter que lors d'un test, la charge générée par JMeter n'est pas négligeable, donc plus l'infrastructure de votre site est robuste et plus vous aurez besoin d'injecteurs et de bande passante afin de pousser vos tests le plus loin possible.

Les résultats obtenus sur JMeter sont représentatifs de ce que l'on obtiendra côté client mais ne vous permettront pas de déterminer clairement quelles ressources arrivent à saturation lors de la montée en charge. C'est pourquoi il est important pour l'analyse de vos résultats d'avoir des graphes de comportement de vos serveurs. Il serait intéressant d'avoir au moins les informations portant sur le *load average*, la consommation mémoire et la bande passante. Dans le cas de ce test, j'ai utilisé Cacti qui est un outil simple permettant d'obtenir ces informations.

2. Avant de commencer

Il est essentiel d'avoir un objectif avant de commencer un test afin de savoir si votre infrastructure est dimensionnée selon vos attentes de trafic. C'est pourquoi vous devez vous poser les questions suivantes :

- Quel est le nombre moyen de visiteurs simultanés attendus :

```
- Lors d'un trafic normal ?

- Lors d'un pic de trafic ?
```

- Quel scénario de navigation sera le plus représentatif d'une visite « standard » ?

Le choix du scénario est essentiel dans les tests de charge et il mérite que l'on y réfléchisse sérieusement afin que les résultats obtenus collent le plus à la réalité.

L'application de test que j'utiliserai dans cet article est un « Dotclear » hébergé sur un petit serveur physique possédant 2 CPU dual core, 2Go de mémoire vive et 1Go de bande passante. J'ai fait le choix de tester un blog car les scénarios possibles ne sont pas vraiment

nombreux, donc un seul test nous permettra d'avoir des résultats significatifs. Comme il s'agit d'un test d'un site n'existant que pour cet article, je n'ai pas réellement d'objectifs. Je me contenterai donc de fixer le nombre de threads simultanés assez haut pour atteindre le seuil critique et me faire une idée de sa valeur.

Le scénario que l'on utilisera se divise en fait en 2 parties :

1 - Consultation de 2 articles (200 threads simultanés maximum)

- L'internaute arrive sur la page d'accueil et y reste 10 secondes.

- Il clique sur le premier article et le lit durant 30 secondes.

- Il retourne sur la page d'accueil durant 5 secondes.

- Il clique sur le deuxième article et le lit durant 30 secondes.

- Il recommence.

2 - Consultation d'un article et dépôt d'un commentaire (50 threads maximum)

- L'internaute arrive sur la page d'accueil et y reste 10 secondes.

- Il clique sur le premier article, le lit et rédige un commentaire en 50 secondes.

- Il attend une minute et recommence.

La division en deux parties est nécessaire car la fréquence des deux scénarios n'est pas identique puisque la consultation d'article est généralement plus fréquente que le dépôt d'un commentaire.

L'exécution du scénario se fera de manière progressive, c'est-à-dire que nous augmenterons le nombre de threads de manière périodique jusqu'à atteindre la limite spécifiée par notre configuration. Il est

important de ne pas monter trop vite en charge afin que nous puissions visualiser la manière dont réagit notre infrastructure. Notre test s'étalera donc sur un peu plus de 8 heures.

3. Installation et utilisation de JMeter

3.1. Installation

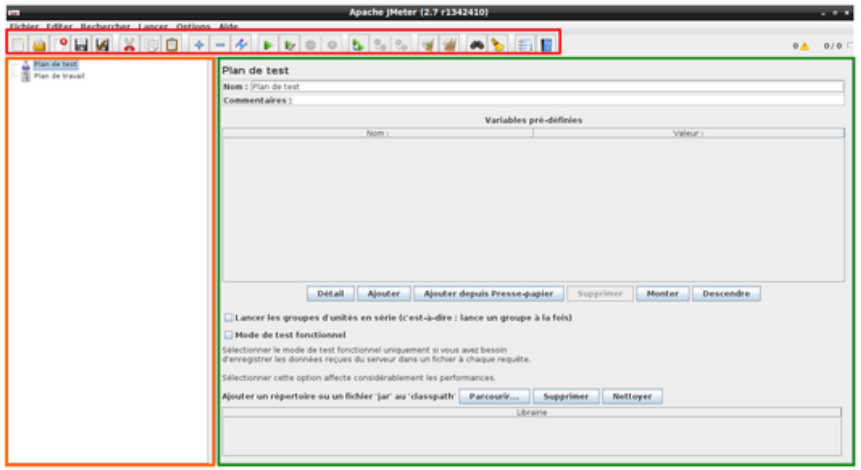
JMeter est présent dans les dépôts de la plupart des distributions, mais dans cet article, j'utiliserai la dernière version récupérée depuis le site de JMeter (http://jmeter.apache.org/download_jmeter.cgi). Il s'agit de la version 2.7. L'installation est très simple, il suffit de télécharger l'archive, la décompresser et lancer l'exécutable `jmeter` qui se trouve dans le répertoire `bin` de l'archive. JMeter étant un programme Java, il faut installer le JRE préalablement.

Le graphage des résultats n'est pas vraiment le point fort de JMeter. Par défaut, ils ne sont pas très lisibles. Heureusement, il existe un plugin qui gère très bien cela. Vous pouvez le télécharger à cette adresse : http://code.google.com/p/jmeter-plugins/. Pour l'installer, il suffit de copier le fichier `JmeterPlugins.jar` dans le répertoire `lib/ext` du répertoire d'installation de JMeter.

3.2. Présentation de l'interface

Bien que JMeter nous offre un bel environnement graphique, son utilisation n'est pas très intuitive. La configuration d'un scénario passe par celle de différents éléments plus ou moins indépendants que vous

pouvez ajouter selon vos besoins. Commençons par une présentation de l'écran d'accueil de JMeter :



Écran d'accueil de JMeter

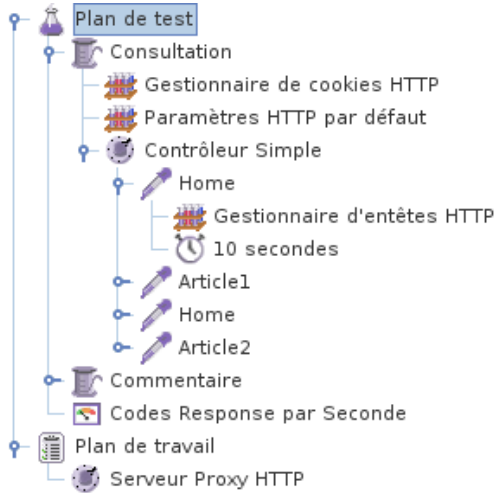
L'encadré rouge représente les raccourcis des menus. Vous pourrez à partir de là lancer vos tests, les stopper ou encore les réinitialiser.

L'encadré orange est la partie où seront listés les différents éléments du test. Nous verrons par la suite quelques-uns des éléments les plus utilisés dans les tests de performance web.

Enfin, c'est dans l'encadré vert que s'effectuera la configuration des éléments sélectionnés dans la partie orange.

3.3. Élaboration du scénario

Nous pouvons commencer à ajouter des éléments. Pour un test de charge de site web, les éléments visibles ci-dessous sont suffisants pour créer un scénario, gérer le nombre de threads simultanés et visualiser les résultats du test.



Arbre de configuration du test

On peut déjà voir que nous avons 2 parties principales : le plan de test et le plan de travail.

Le plan de test contiendra toutes les informations nécessaires au test de charge, c'est-à-dire les éléments de configuration du nombre de threads, la gestion des requêtes ou encore des *cookies*.

Quant au plan de travail, il contiendra tous les éléments hors test et tout ce que vous souhaitez conserver temporairement. Cette partie n'est pas sauvegardée à l'enregistrement d'un scénario.

Le groupe d'unités est l'élément obligatoire pour tout test de charge avec JMeter. Nous en avons 2 dans notre exemple : Consultation et Commentaire. Ils représentent les 2 parties du scénario que nous avons vu plus haut. Cet élément permet de :

- fixer le nombre de threads simultanés.

- la durée de montée en charge, c'est-à-dire le temps que devra mettre JMeter pour lancer tous les threads.

- le nombre d'itérations, c'est-à-dire le nombre de fois que le scénario devra être répété par chaque thread. Il est possible d'effectuer une répétition infinie. Dans ce cas, je vous conseille de cliquer sur ***Programmateur de démarrage*** afin de configurer la durée du test.

Le groupe d'unités s'obtient en cliquant droit sur ***Plan de test***, puis ***Ajouter -> Moteurs d'utilisateurs -> Groupe d'unités***.

L'élément « Paramètres HTTP par défaut » vous permettra de fournir au groupe d'unités des paramètres HTTP tels que le nom de domaine du site (ou son IP), le port à utiliser, etc. Pour ajouter cet élément, il faut cliquer droit sur ***Groupe d'unités*** puis : ***Ajouter -> Configurations -> Paramètres HTTP par défaut***.

Le gestionnaire de cookies n'est réellement important que pour les sites utilisant des cookies ou des sessions gérées par des cookies. Pour ajouter cet élément, il faut cliquer droit sur ***Plan de test*** puis : ***Ajouter -> Configurations -> Gestionnaire de cookie HTTP***. Il pourrait être intéressant pour la configuration de cet élément de cocher ***Nettoyer les cookies à chaque itérations ?*** afin que lors de la répétition d'un scénario par un même thread, le cookie soit généré à nouveau.

Un contrôleur est l'élément qui vous permettra de gérer un groupe de requêtes HTTP. Il existe une quinzaine de types de contrôleurs différents. Ces éléments permettent de spécifier comment seront gérées les requêtes d'un groupe. Le scénario que nous présenterons ici étant assez simple, nous nous contenterons d'un contrôleur simple. Pour l'ajouter, il faut cliquer droit sur ***Groupe d'unités*** puis : ***Ajouter -> Contrôleurs Logiques -> Contrôleur Simple***.

L'élément « Serveur Proxy HTTP » nous permettra d'enregistrer un scénario simplement en naviguant sur les pages que nous voulons y inclure. Pour ajouter cet élément, il faut cliquer droit sur ***Plan de travail*** puis : ***Ajouter -> Éléments Hors Test -> Serveur Proxy HTTP***.

Le principe est que JMeter deviendra notre proxy le temps de l'enregistrement du scénario. Ainsi, il pourra enregistrer toutes les requêtes que nous faisons. Pour ce faire, il faut dans un premier temps configurer votre navigateur afin qu'il utilise le proxy local écoutant sur le port 8080. Ensuite, au niveau de JMeter, il faut sélectionner l' élément « Serveur Proxy HTTP » puis choisir le contrôleur dans lequel s'enregistrera la liste de requêtes. Vous pouvez exclure des URL ou des types de fichiers en spécifiant les motifs dans le champ « Motifs à exclure ». De manière générale, il n'est pas très utile de conserver tout le contenu statique, sauf si vous souhaitez tester votre bande passante. En effet, la charge générée par les requêtes d'éléments statiques est négligeable face à celle générée par PHP ou autres langages de script. Enfin, il faut cliquer sur ***Lancer***. Vous pouvez maintenant naviguer dans votre site en suivant le scénario que vous avez établi préalablement. Pour terminer, vous appuyez sur ***Arrêter***. N'oubliez pas de restaurer la configuration de votre navigateur dans son état d'origine.

Si vous déroulez le contrôleur sélectionné, vous pourrez voir la liste de toutes les requêtes effectuées lors de votre navigation.

Comme la visite d'un site est ponctuée par des temps de lecture du contenu des pages, il faut ajouter des compteur de temps afin de marquer des pauses après chaque page, comme nous l'avons vu dans la présentation du scénario. Il existe une dizaine de compteurs de temps. Nous allons utiliser un compteur de temps fixe. Pour l'ajouter, il faut cliquer sur la requête concernée puis ***Ajouter -> Compteur de temps -> Compteur de temps fixe***. Sa configuration est simple, il suffit de spécifier le nombre de millisecondes à patienter.

Tous les éléments permettant de faire le test ont été ajoutés. Il faut maintenant ajouter l'élément qui nous permettra d'enregistrer les résultats obtenus dans un fichier que nous pourrons exploiter après le test. Pour ce faire, nous allons ajouter un des éléments du plugin que nous avons installé en cliquant droit sur ***Plan de test*** : ***Ajouter -> Recepteur -> jp@gc - Response Codes per Second***.

Il faut noter que durant les tests de charge, JMeter est assez gourmand en ressources, surtout lorsqu'il doit afficher les résultats en temps réel. Je vous conseille donc de toujours lancer vos tests en ligne de commandes. Vous pourrez exploiter les résultats obtenus après le test. Pour ce faire, il faut spécifier le fichier dans lequel vous enregistrerez vos résultats. Un seul récepteur est nécessaire pour recueillir les résultats, mais plusieurs pourront être utilisés pour les visionner une fois que le test sera fini. Pour cela, il faut juste spécifier le chemin du fichier dans le champ ***Nom du fichier***. Lorsque le test sera fini, nous ouvrirons ce même fichier en cliquant sur ***Parcourir*** et le graphe sera généré.

Si vous cliquez sur ***Configurer***, vous pourrez sélectionner ce qui sera enregistré dans le fichier. Pour ce test, je n'ai eu à ajouter que « Nombre d'unités active » en plus de la sélection par défaut. Cette option permettra d'obtenir des graphes de temps de réponse en fonction du nombre de threads actifs.

Pour faire ce test, j'ai utilisé 2 instances Jmeter : un contrôleur et un injecteur. Je ne m'attarderai pas sur la configuration d'un test en utilisant un injecteur, la procédure n'est pas très compliquée et est bien expliquée sur le site du projet JMeter.

Nous pouvons enfin enregistrer notre plan de test et entrer la commande suivante pour lancer le test :

```
./jmeter -n -X -t /chemin/vers/fichier.jmx
```

4. Exploitation des résultats

Une fois le test terminé, on peut visualiser les résultats obtenus en lançant JMeter et en ajoutant les différents récepteurs que l'on souhaite. Les récepteurs que j'utilise le plus souvent sont :

- jp@gc - *Response Codes per Second* ;

- jp@gc - *Response Times Over Time* ;

- jp@gc - *Response Times vs Threads* ;

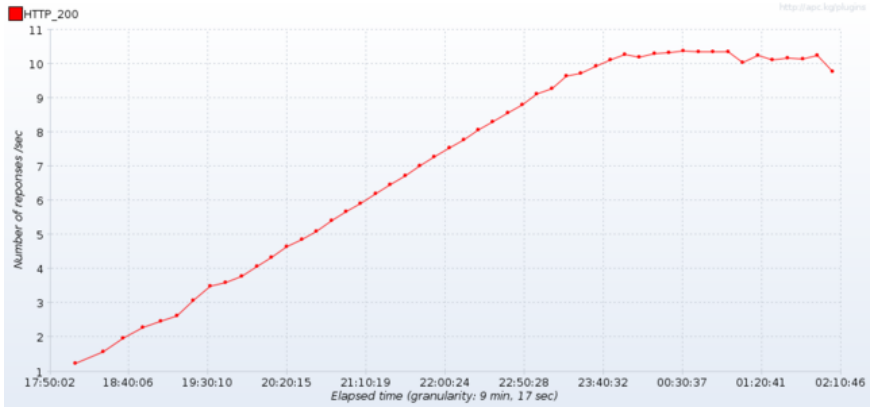
- jp@gc - *Active Threads Over Time* ;

Vous trouverez ci-dessous les résultats obtenus pour ce test.

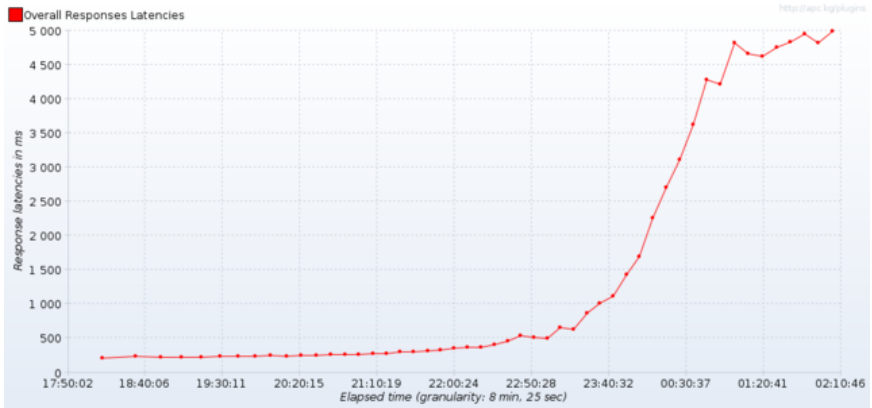
4.1. Les graphes JMeter

Tout d'abord, voici le graphe du nombre de réponses obtenues par seconde :

On remarque que l'on atteint vers 23h40 un seuil de 10 requêtes par seconde. Si nous jetons un œil au détail du temps de réponse moyen, on retrouve le phénomène. À peu près au même moment, vers 23h00, le temps de réponse des requêtes subit une forte augmentation :

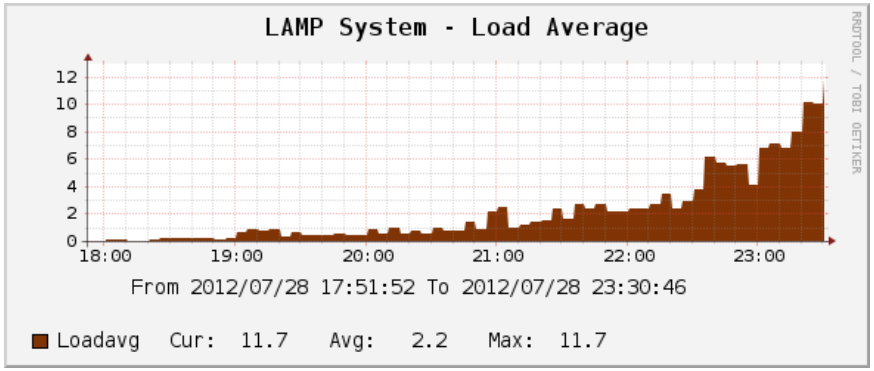


Nombre de réponses par seconde (code 200)



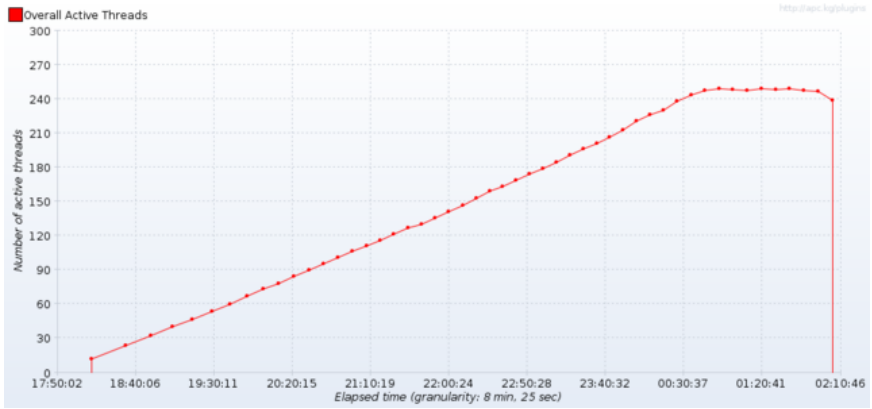
Temps de réponse en fonction du temps

En regardant le graphe de load average du serveur obtenu grâce à Cacti, on peut voir que les 4 de load sont franchis vers 23h également, et comme nous avons 4 cœurs, cela indique un début de surcharge.



Load Average durant les 6 premières heures

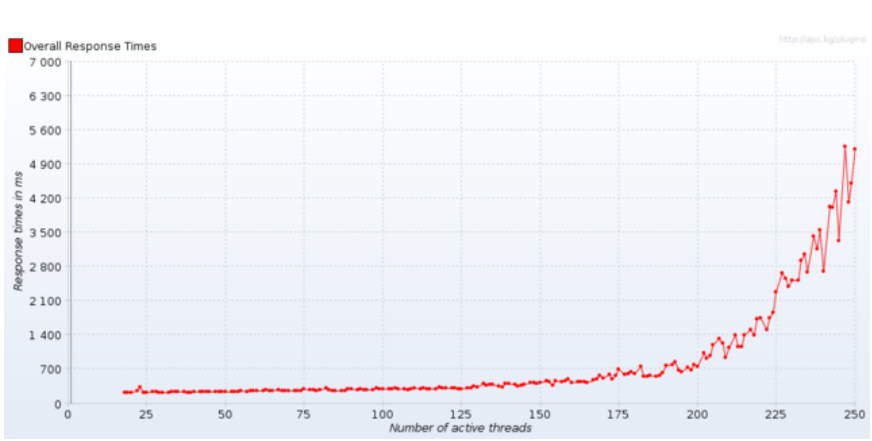
Ainsi, pour déterminer une approximation du nombre de visiteurs simultanés que peut accueillir notre plateforme, il suffit de regarder sur la figure 7 le nombre de threads actifs entre 22h40 et 23h40 :



Nombre de thread actifs en fonction du temps

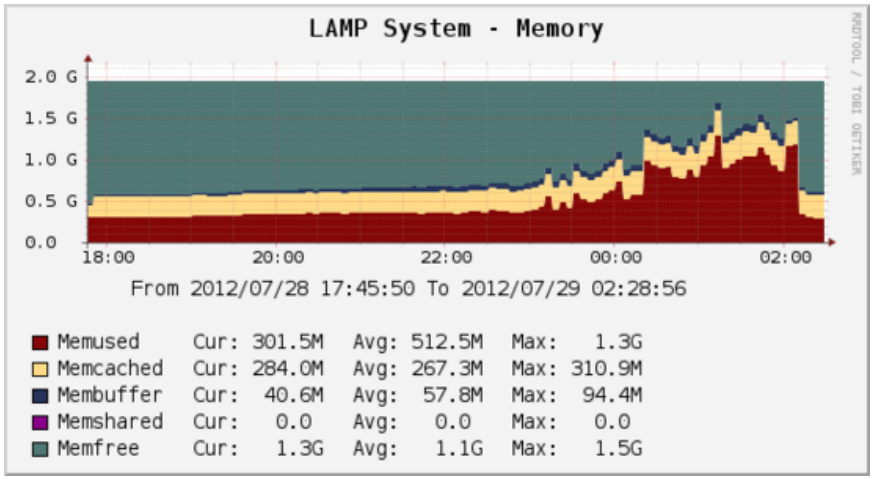
Donc pour conclure, notre serveur pourrait accueillir entre 170 et 200 internautes simultanément sans qu'ils ne subissent de lenteur de chargement sur le blog.

En effet, d'après le graphe des temps de réponse en fonction du nombre de threads, on peut voir que pour la tranche 170-200 threads, le temps de réponse moyen ne dépasse pas 1 seconde.

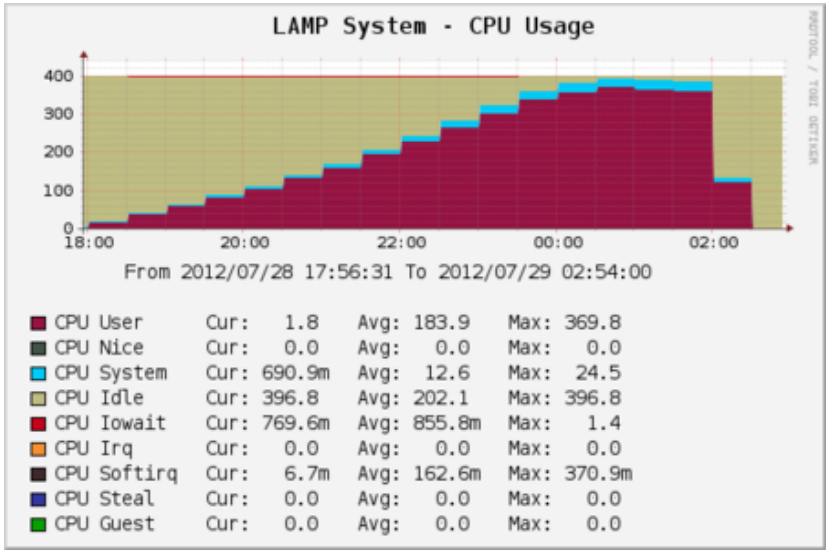


Temps de réponse en fonction du nombre de threads

On peut également conclure que c'est la ressource CPU qui est saturée et qui empêche d'obtenir des résultats plus élevés puisque les CPU ont atteint les 100 % d'utilisation et le load average a été élevé pendant le test alors que la RAM n'est pas entièrement consommée.



Graphe de consommation de la RAM



Consommation CPU

Conclusion

Nous venons donc de voir un test de charge qui, bien qu'il soit simple, constitue une bonne base pour vos prochains tests.

On aurait pu étendre l'utilisation de Jmeter à des tests de composants logiciels. En effet, on aurait pu s'amuser à tester notre application sur différents serveurs web en refaisant le match Apache vs Nginx vs LightHTTPd pour voir lequel est le plus performant.

On aurait également pu ajouter un reverse proxy afin de voir combien de requêtes par seconde supplémentaire notre serveur pourrait supporter.

Les testeurs de charge sont donc les outils idéaux pour justifier le choix d'un composant par rapport à un autre si la performance est l'un des critères de décision.

Articles qui pourraient vous intéresser...

Mise en place de l'intranet

Linux Pratique HS n° 48
septembre 2020

ParPelisse Romain

Réseau

Le précédent chapitre a présenté, de manière sommaire et limitée aux besoins de ce dossier, les fonctionnalités

Covid-19, télétravail : mise en œuvre d'accès distants sécurisés pour se rapprocher du SI

MISC n° 111
septembre 2020

ParSarr Amadou
Ladent Étienne

Sécurité Réseau

Les mesures de confinement prises par le gouvernement mi-

Premiers pas et prérequis

Linux Pratique HS n° 48
septembre 2020

ParPelisse Romain

Réseau Système

Dans ce premier chapitre, nous allons aborder de nombreux sujets, allant du respect des conditions

mars 2020 pour contrer la propagation du Covid-

Comment mettre en place un serveur NIS

Linux Pratique n° 121
septembre 2020

ParLaïchaoui Malik

Amélioration de l'infrastructure

Linux Pratique HS n° 48
septembre 2020

ParPelisse Romain

Réseau

Tout administrateur réseau qui se respecte se doit de mettre en place un système qui permettra aux

Réseau

Notre serveur est en place et notre intranet est entièrement fonctionnel ! Il est temps, maintenant, d'aller plus

DOMAINES

AUDIO/VIDÉO
BRÈVES
BUREAUTIQUE
CODE
DOMOTIQUE
DROIT
ÉLECTRONIQUE
EMBARQUÉ
GRAPHISME
HACKS
HUMEUR ET CRITIQUE
IA
JEUX
MOBILITÉ

NOS PUBLICATIONS

GNU/LINUX MAGAZINE

MISC

LINUX PRATIQUE

HACKABLE

AIDE & CONTACT

ABONNEMENTS

POUR LES PROFESSIONNELS

POUR LES PARTICULIERS

À PROPOS

QUI SOMMES-NOUS ?

EN SAVOIR PLUS SUR CONNECT

RADIO ET WIRELESS
RÉSEAU
RÉTRO
ROBOTIQUE
SÉCURITÉ
SOCIÉTÉ
SYSTÈME
TÉMOIGNAGE
TESTS ET PRISE EN MAIN
WEB

FOIRE AUX QUESTIONS

DEVENIR AUTEUR

DEVENIR ANNONCEUR

NOUS CONTACTER

LES AUTEURS

INFORMATIONS LÉGALES

Tél. : 03.67.10.00.20