Mesures de performance

> Sometimes the only choices you have are bad ones. But you still have to choose.
>
> Le douzième docteur

Les tests de performance serveur deviennent obligatoires dès qu'une application doit être accessible à tout moment. Ces tests permettent d'évaluer la charge que le serveur est capable de gérer, c'est-à-dire le nombre de connexions simultanées acceptées.

1 Comment planifier une batterie de tests ?

Il faut tout d'abord connaître les exigences que l'on a envers notre installation. Les exigences de trafic ne sont bien entendu pas les mêmes entre une petite association gérant un forum, quelques fils de news et celle d'une grosse organisation devant être disponible tout le temps, avec par exemple des serveurs relayant des mises à jour de sécurité ou des contenus sensibles (commerce, banque, etc.).

Quelques questions à se poser :

- quel est notre nombre moyen d'utilisateurs prévus (en charge normale) ?
- quel est notre nombre maximal d'utilisateurs prévu / à supporter ?
- à quel moment faire des tests de charge de notre serveur ? (c-à-d. plutôt aux heures creuses ou pendant le week-end), sachant que cela peut très bien casser un ou plusieurs de nos serveurs (d'où des procédures de reprise sur panne)
- on ne teste pas seulement un serveur web, mais souvent aussi des modules qui marchent de pair (*PHP*, *FastCGI*, etc.), ainsi que le côté applicatif : mesurer l'impact de l'utilisation (authentification, utilisation de l'application…) est souvent nécessaire.

Une progression classique dans la succession des tests réalisables sur un serveur est la suivante :

- tester raisonnablement avec un faible volume de requêtes (pouvons-nous réaliser des mesures sur notre serveur ?) ;
- benchmarker en simulant cette fois-ci le nombre moyen d'utilisateurs ;
- réaliser un test de charge en visant cette fois le nombre maximum d'utilisateurs ;
- enfin, rechercher la limite absolue de votre installation en cherchant le nombre d'utilisateurs nécessaires pour l'empêcher de fonctionner (déni de service).

Dans tous les cas, il est nécessaire de s'assurer que l'on éprouve bien son serveur, et non les installations intermédiaires telles que les proxys et autres routeurs qui séparent la machine lançant les tests de celle que l'on souhaite éprouver (vérifiez avec la commande `htop` sur celui-ci).

Le processus d'analyse peut ensuite s'affiner. L'analyse boîte-noire, c-à-d. ne nécessitant pas de connaissances a priori sur le fonctionnement des applications hébergées est un point de départ.

Cela ne suffira pour réaliser une étude approfondie des capacités de votre système : il est nécessaire de passer ensuite à une approche de type boîte-blanche, c'est-à-dire à un protocole de test prenant en compte les spécificités de vos applications. Il n'est pas rare de découvrir des défauts au niveau applicatif pendant ce processus : rien ne sert par exemple de posséder un excellent moteur de recherche s'il n'est pas capable de résister à une charge élevée.

2 Les différents outils disponibles

Le premier logiciel à citer est `ab`[1] (pour Apache Benchmark). Proposé en standard avec Apache, il permet de spécifier le nombre de requêtes à lancer, le nombre de connexions simultanées, la durée du test, etc.

1. https://httpd.apache.org/docs/2.4/programs/ab.html

D'autres outils, en GUI ou en ligne de commande existent :

— JMeter [2]
— Taurus, un outil qui se base sur JMeter [3]
— Siege [4]
— Locust [5]
— etc. [6]

Il est préférable d'utiliser des outils permettant d'utiliser un fichier de configuration du test afin de pouvoir tester différents scénarios en même temps (page d'accueil, login, soumission d'un formulaire, etc).

3 Analyser les résultats

Utiliser des programmes de tests n'est pas tout : la phase la plus importante reste à faire. Les longs listings produits par les outils de tests ne sont généralement pas lisibles directement [7]. Il est nécessaire de réaliser une synthèse de toutes ces données chiffrées.

Locust permet de récupérer les données au format CSV, format idéal pour traiter les données de toutes les façons possibles : insertion en base de données, génération de graphiques, etc.

Configurer son serveur pour qu'il génère des logs spécifiques durant la phase de test est aussi un bon moyen pour pouvoir analyser ce qui s'est produit. On peut ainsi avoir un autre point de vue, côté serveur.

On dispose d'ailleurs d'outils d'analyse de journaux qui peuvent aussi servir pour l'analyse du fonctionnement quotidien du serveur. Analog [8] ou Webalizer [9] sont des utilitaires permettant d'extraire les informations utiles des logs.

Citons aussi GoAccess [10], qui permet d'obtenir des informations en temps réel depuis les logs de son serveur web (quelle IP revient le plus souvent, quelle est la page la plus consultée, etc).

4 Amélioration des performances

Une fois les mesures effectuées, il peut se révéler nécessaire de rendre votre serveur plus efficace.

Pour cela, plusieurs choix s'offrent à vous.

4.1 Configuration matérielle

Un des moyens les plus simples pour augmenter les performances est souvent d'augmenter la mémoire sur votre serveur web.

Bien sûr, disposer d'une machine plus performante est toujours un plus. Il faut privilégier en particulier la partie disques durs :

— le `SCSI` permet de dégager le processeur d'une partie des traitements liés aux entrées-sorties ;
— Le `RAID` permet de jumeler plusieurs disques pour obtenir de meilleures performances ;
— Les disques `SSD` permettent des accès disques fulgurants mais leur coût est plus élevé que pour les disques traditionnels.

2. https://jmeter.apache.org/
3. https://gettaurus.org/
4. https://www.joedog.org/siege-home/
5. http://locust.io/
6. Une petite recherche sur https://alternativeto.net vous permettra de trouver d'autres alternatives à `ab`
7. enfin, si, parfois, mais on ne peut pas en faire de jolis tableaux comparatifs, avec de beaux graphiques, toussa
8. http://www.analog.cx — le site est mort mais le logiciel est toujours disponible dans les dépôts Debian
9. http://www.webalizer.org
10. https://goaccess.io

Attention cependant à n'utiliser en production que des systèmes `RAID` secourus, ou sinon le risque de perte de données devient trop important [11].

Vous utilisez du stockage distant comme NFS, CephFS, du stockage object S3 ou Swift ? Vérifiez que le facteur limitant de votre site web ne se situe pas à ce niveau-là (performance du réseau entre le serveur web et le serveur de stockage, performances du serveur de stockage...)

Avec la virtualisation que l'on rencontre de plus en plus, l'ajout d'un ou deux cœurs se fait en un clic. Pensez-y aussi.

Ces trois paramètres (mémoire, disques et processeurs) nécessitent d'analyser le comportement de votre serveur en charge pour découvrir quel est le facteur limitant. Une fois celui-ci identifié et réglé, un des autres paramètres prendra sa place en tant que goulot d'étranglement, et ainsi de suite jusqu'à atteindre le niveau de service désiré. Il n'y a pas de solution miracle, et la modification de ces paramètres doit être décidée en toute connaissance de cause.

4.2 Réglages du serveur

Au niveau Apache

Chargez le nombre minimal de modules nécessaires au fonctionnement de votre serveur. Cela réduit d'autant l'occupation mémoire d'Apache et de chacun de ses processus fils / threads.

Pensez à faire des tests en faisant varier le nombre de processus fils / threads, le nombre maximum de requêtes gérées par un processus fils / thread, ou encore le temps d'attente pour le `KeepAlive`.

La mise en place d'un cache peut aussi augmenter la résistance de votre serveur, mais attention, cela ne peut pas s'appliquer partout (comme par exemple, des pages très dynamiques comme le cours de la bourse).

Au niveau Nginx

De la même façon qu'on essaye d'avoir un Apache possédant les seuls modules nécessaires, on prendra soin de n'activer que les modules nécessaires.

Tout comme Apache, on manipulera le nombre de threads de Nginx (voir https://nginx.org/en/docs/ngx_core_module.html, les directives `worker_`).

Au niveau des modules applicatifs

Il existe de nombreux systèmes de cache, qui permettent de faire baisser la charge processeur.

Par exemple OPcache [12] pour des applications PHP.

Le fichier `php.ini` permet en outre de régler la taille de la RAM utilisée (paramètre `memory_limit`) entre autres paramètres.

Au niveau applicatif

Les optimisations les plus importantes sont de toutes façon celles obtenues par une bonne conception de votre application. Par exemple, les pages les plus fréquentées doivent rester aussi légères que possibles, et leur contenu au maximum statique.

Les technologies telles les requêtes AJAX permettent l'envoi d'une application composée de code javascript à la première connection, d'une taille non négligeable mais dont le transfert s'optimise bien car ne

11. le `RAID 0`, c'est mal !
12. https://www.php.net/manual/fr/book.opcache.php

nécessitant pas de traitement dynamique côté serveur (les caches sont alors très efficaces), avec ensuite des échanges légers correspondant au contenu dynamique généré par le serveur.

Votre application utilise souvent d'autres services, tels une base de donnée ou un annuaire : étudier les performances de ces autres composants est aussi indispensable.