République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaïa
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Option : Fiabilité et Évaluation des Performances des Réseaux

---

Mémoire de Master

en

Recherche Opérationnelle

Thème :

Évaluation de performance des systèmes de serveurs web distribués : cas de l'équilibrage de charge

Présenté par :

M^{elle} Faiza CHOUBANE

Soutenu devant le jury composé de :

Président :	M^{elle} N. Bousba	Doctorante	Université de Béjaia
Examinateurs :	M^{r} M.Yazid	MAB	Universit de Béjaia
	M^{r} M.Athmani	Doctorant	Universit de Béjaia
Rapporteur :	M^{r} F.Semchedine	MCB	Universit de Béjaia

Juin, 2012.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

La croissance de l'Internet et du World Wide Web a sensiblement augmentée la quantité des informations en ligne et des services disponibles pour toute la population.
L'explosion des demandes d'accès à ces loisirs a mis une sérieuse pression sur l'infrastructure de l'Internet causée par le besoin en terme de systèmes Web très évolués, capables de servir des millions d'utilisateurs. Pour construire de tels systèmes, les développeurs ont, de plus en plus, basculé vers les conceptions distribuées à cause de leurs possibilités d'extension et leurs coûts réduits à la place d'une grosse machine comme les Mainframes. [1]
Parmi ces systèmes distribués, citons : les serveurs Web distribués, les serveurs de base de données distribués, et les grilles de calcul.

Dans les systèmes de serveurs Web distribués, les requêtes arrivent, doivent être affectées à l'une des machines hôtes pour le traitement. La Qualité de Service (QoS) dans ces systèmes se concentre sur la manière d'affectation de ces requêtes afin de minimiser le temps de réponse et d'améliorer ainsi les performances . Cet aspect a fait l'objet de beaucoup de travaux .

Les auteurs ont montré, dans [1], [2], [3] que la distribution de la taille des tâches Web, est (Heavy-Tailed) où $\Pr\{X > x\} \sim x^{-\alpha}, 0 < \alpha < 2$, et dont une petite fraction (environ 1%) des grandes tâches Web, génère plus de 50% de la charge totale du système. Cette propriété provoque des problèmes dont : i) ces grandes tâches Web sont difficiles à être affectées aux différents serveurs, et ii) les petites tâches Web, lorsqu'elles sont avec les grandes tâches Web dans la même file d'attente, seront retardées lors de leur traitement.

Ces problèmes, suscités par la nature de la distribution des tâches Web, ont fait l'objet de plusieurs travaux dans la littérature. Des techniques naïves, comme Random et Round Robin avaient été proposées. Ces techniques, connues comme étant des techniques statiques, ne prennent pas en charge des informations lors de l'affectation des tâches Web et elles ont des difficultés à s'adapter au trafic Web comme le montrent Harchol-Balter et al [4]. C'est pourquoi, des techniques comme LLF (Least Loaded First), considérées comme des techniques dynamiques, ont été proposées. Elles considèrent la charge du serveur et d'autres informations au moment de l'affectation. Ces techniques, comme Shortest-Queue et Least-Work-Remaining, ont de meilleures performances que les techniques traditionnelles, lorsque la distribution est exponentielle,[5]. Cependant ces performances se dégradent sous un trafic et une variabilité élevés [4], [6]. Pour surmonter cette caractéristique du trafic Web, des techniques basées sur la taille des tâches Web ont été proposées (comme SITA-U [6], SITA-E [4], SITA-V, EQUILOAD [7]).

1. Ordinateurs centraux de grandes puissance de traitement de données.

Ces techniques utilisent des intervalles de taille où chaque serveur a l'intervalle approprié et le répartiteur affecte les tâches Web arrivant au serveur en se basant sur l'intervalle de taille. Quoique ces techniques aient montré une amélioration dans les performances sous le trafic Web par rapport aux techniques rappelées précédemment, leurs limitation est la possibilité de tomber dans les situations où les ressources des serveurs ne seront pas complètement utilisées. Récemment, des variantes d'une nouvelle technique LFF (Least Flow time First) [8], [9],[10] ont été proposées. Contrairement à LLF et Global Class Policy [11] (laquelle affecte la tâche Web au serveur libre), cette technique affecte les tâches Web en choisissant le serveur qui a le plus petit temps de séjour. LFF a montré une amélioration des performances, mais sa limitation se situe dans l'utilisation des files d'attente multi-section pour chaque serveur, surtout dans le cas des systèmes qui proposent plusieurs services à la fois (ont les appelent systèmes multiservices).

D'autres techniques, comme TAGS et TAPTF [12] ont été proposées. Ces techniques attribuent un quantum [2] de temps pour chaque serveur, et le répartiteur affecte les tâches Web à un serveur qui commence son traitement jusqu'à la fin du quantum. Si la tâche Web n'a pas fini son traitement, elle sera arrêtée et recommencera son exécution au serveur suivant. Le processus Arrêter/Recommencer compte un inconvénient pour ces techniques. En effet, ces techniques assument que la taille de la tâche Web n'est pas connue à priori, ce qui n'est pas le cas pour notre étude.

Dans ce mémoire, nous allons modèliser une technique proposée prenant en charge ces inconvénients. L'idée principale de cette technique est de diviser les tâches Web de grande taille en fragmants et de les exécuter en parallèle afin de faire converger le temps de service des grandes tâches Web avec celui des fragments et ainsi d'optimiser les mesures de performances : le temps d'attente, le temps de séjour,... etc. Par ailleurs, cette technique sépare les tâches Web de petite taille des grandes tâches Web afin d'éviter qu'elles soient retardées lors de leur exécution.

Le mémoire est organisé comme suit : dans le chapitre1, nous avons présenté un état de l'art des systèmes de serveurs Web distribués, nous avons commencé par citer les différentes entités qui participent dans la construction des systèmes de serveurs Web distribués, les différentes architectures possible pour l'extension de ces systèmes et enfin, nous avons cité les différents types d'approches pour l'affectation des tâches Web qui arrivent au système. Dans le chapitre2, nous exposons les différents travaux de la littérature relatifs à l'affectation des tâches Web en montrant les limites des techniques proposées. Une description détaillée d'une nouvelle technique est donnée dans le chapitre3.

2. Mot latin signifiant " combien " et qui s'écrit "quanta" au pluriel, représente la plus petite mesure indivisible, que ce soit celle de l'énergie, de la quantité de mouvement on de la masse.

1

LES SYSTÈMES DE SERVEURS WEB DISTRIBUÉS

1.1 Introduction

La nécessité des systèmes de serveurs Web distribués est devenue le but le plus important dans la conception de nouvelles architectures de serveurs Web pour les différentes organisations qui désirent surpasser la croissance explosive du trafic dans le World Wide Web, tout en partageant cette charge sur les différents serveurs Web plutôt que d'envoyer tout le trafic à un seul serveur qui risque ainsi de devenir surchargé.

Dans de telles architectures, quelques entités (le serveur Web, le DNS : Domain Name System, le protocole HTTP et le Répartiteur) contribuent toutes pour garantir le meilleur service exigé par le client, en terme de qualité de service (i.e. temps de réponse optimal, un délai minimal,..., etc.)de sécurité,..., etc.

Ce chapitre représente un état de l'art sur les systèmes de serveurs Web distribués. D'abord, de la section 1.1 à la section 1.4, nous introduisons les différentes entités qui sont utiles pour la création des systèmes de serveur Web. Nous parlerons de l'entité Serveur Web qui a pour but de fournir le contenu (pages html, applet java,...,etc.) aux clients, le protocole HTTP qui est conçu comme une couche intermédiaire entre le serveur Web et les clients, et le DNS, qui assure la résolution et l'envoie de l'adresse IP du serveur Web vers le client. Dans la section 1.5, nous montrerons comment une transaction Web est réalisée entre les différentes entités lors de l'accès aux objets d'une page Web. Ensuite, dans la section 1.6, nous citerons les différentes options pour l'extension des systèmes de serveurs Web, en mentionnant les diverses manières pour la mise à niveau des architectures d'un seul serveur ou un serveur moins puissant vers des serveurs distribués ou plus

puissants. Dans la section 1.7, nous porterons une attention sur les différentes approches pour l'affectation des tâches Web aux serveurs Web en considérant la classification des entités qui accomplissent l'action d'affectation.

1.2 Le Serveur Web

Le serveur Web est l'entité la plus importante dans les systèmes de serveurs Web distribués. C'est une collection de matériels, systèmes d'exploitation, logiciels réseaux et serveur HTTP (e.x. NCSA, Apache) qui coopèrent tous pour assurer la mission principale qui consiste dans la satisfaction des demandes de requêtes des clients.

Le serveur Web écoute sur le port HTTP (typiquement le port 80) pour les requêtes HTTP entrantes. Lorsqu'il y a une requête qui arrive, il établit une connexion avec le client, lit la requête HTTP à partir du client, envoie les objets demandés (fichier texte, pdf, vidéo,..., etc.) et retourne à son état d'écoute.

Plusieurs serveurs Web, comme Zeus Web Server [13] et Flash Web Server sont disponibles et utilisent diveres options d'optimisation de performances (dépendant ainsi de la plateforme du système d'exploitation sur lequel les serveurs s'exécutent), d'où ils ont des caractéristiques de performance différentes. Apache [14], est le serveur Web le plus disponible et le plus libre. Il est actuellement le serveur Web le plus déployé sur Internet. D'après des estimations récentes, il est utilisé par plus de 60% des sites Web [15].

1.3 Le protocole HTTP(HyperText Transfer Protocol)

Le protocole HTTP (HyperText Transfer Protocol) est un protocole de la couche application du modèle OSI, développé pour le World Wide Web. Ce protocole est un ensemble de règles qui définissent comment le client et le serveur Web interagissent et transfèrent les fichiers (Texte, Images, Audio, Vidéo,...,etc.).

Comme le montre la figure 1.1, une interaction HTTP consiste en un ensemble de requêtes envoyées du client vers le serveur Web suivies de réponses du serveur au client. Le client ouvre donc d'abord une connexion TCP qui résulte de l'échange de paquets SYN comme part des étapes du protocole TCP. i.e, lorsque le client et le serveur terminent les étapes de connexion TCP et se synchronisent, le client envoie une requête HTTP, et le serveur doit à son tour répondre en envoyant le fichier demandé qui sera analysé et affiché par le client.

Une requête HTTP,est composée de quelques éléments comme : les méthodes (methods : GET, POST,..., etc.), une URL : Uniform Resource Locator, les entêtes HTTP

(HTTP headers),..., etc. Lorsque le client envoie une requête, le serveur l'analyse et, se base sur le type de la méthode, il envoie la réponse avec quelques informations de paramétrage (e.g. le status code : pour indiquer si la requête est acceptée ou non, les entêtes,...,etc.).

Plusieurs versions de ce protocole avaient été proposées, de la version HTTP/0.9 à la version HTTP/1.1. La dernière version du protocole HTTP est HTTP/1.1, qui est une mise à jour de l'ancienne version HTTP/1.0 . Les auteurs, dans , ont introduit quelques nouvelles caractéristiques comme, par exemple, la connexion persistante (persistent connection) dans laquelle le client peut envoyer plusieurs requêtes HTTP durant la même connexion TCP, contrairement à la version HTTP/1.0 où le client doit ouvrir une nouvelle connexion TCP à chaque nouvelle requête. Une autre caractéristique est le parallélisme (pipelining). Le pipelining permet au client d'envoyer plusieurs requêtes de manière parallèle durant la même connexion TCP sans avoir à attendre les réponses des anciennes.

Les nouvelles caractéristiques du protocole HTTP/1.1 (persistent connection et pipelining), avaient fourni la preuve de réduire significativement la latence du client et le trafic du réseau [16], [17], [18], [19], [20], et aidé les chercheurs et les développeurs à trouver de nouvelles solutions dans différents axes de recherche, dont le protocole HTTP fait partie, [17], [16].

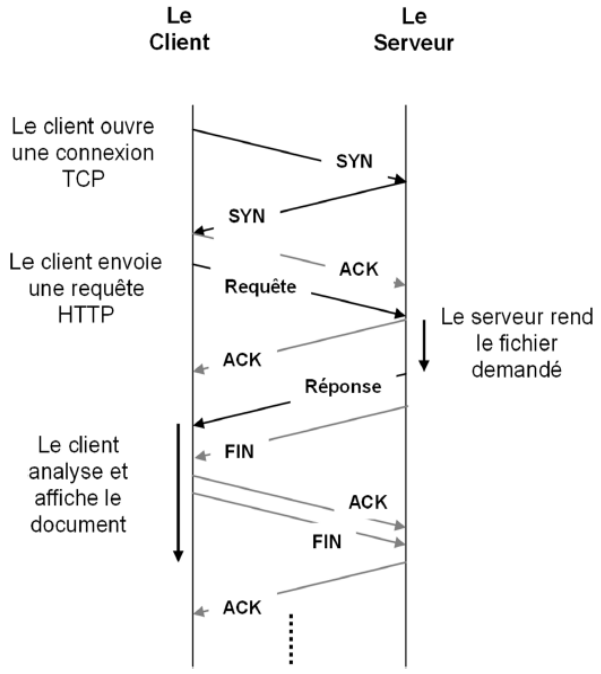

FIGURE 1.1 – Le protocole HTTP.

1.4 Le DNS (Domain Name System)

Les machines connectées aux réseaux IP, comme Internet, ont toutes une adresse IP. A cause de leur forme numérique, ces adresses IP sont difficiles à retenir par les utilisateurs. Le DNS (Domain Name System) a été proposé pour anticiper ce problème tout en assurant la correspondance entre l'adresse IP (e.g. 192.168.16.1) de la machine concernée en un nom de domaine (Domain Name) (e.x. www.Resyd2.com).

Le DNS est composé de trois éléments majeurs :
Domain Name Space : qui est une spécification de l'arborescence des domaines et des sous domaines. La figure 1.2, montre un exemple d'une arborescence de domaine. Dans cet exemple, le domaine racine (.) a trois sous-domaines immédiats : MIL, EDU, et COM. Toutes les feuilles sont également des domaines.

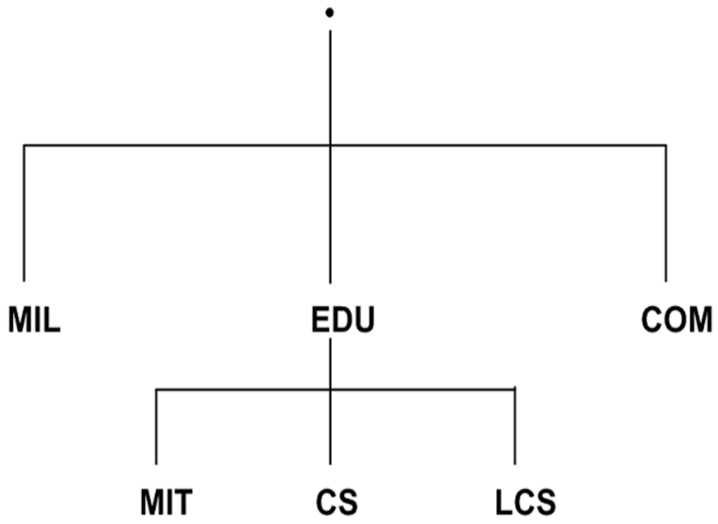

FIGURE 1.2 – Exemple d'une arborescence de domaines.

Name Servers :le Name Server est un ensemble de programmes qui conserve des informations nécessaire pour le fonctionnement du DNS, comme, par exemple, les RR " Resource Records " (i.e. Une structure dans laquelle est maintenue la correspondance entre les noms des serveurs et les adresses IP et qui contient d'autres informations utiles pour la coordination entre les différents serveurs de noms " Name Servers "). En pratique, le serveur de nom sera autoritaire pour tout ou une partie du domaine connue sous le nom ZONE's. Par exemple, un serveur de nom peut être autoritaire pour les domaines dont le suffixe est " .com ", qui peut déléguer un autre serveur de noms pour qu'il soit autoritaire du sous-domaine " Resyd2.com ". Cette hiérarchie permet une gestion efficace des domaines et des sous-domaines, du fait que chaque serveur de noms est responsable de la " ZONE " qu'il gère.

Resolver :ce composant est un programme qui se charge de l'interaction directe avec le client, tout en répondant aux requêtes de ce dernier par l'extraction du contenu des noms de serveurs. Les étapes de fonctionnement du DNS peuvent être vues dans la figure 1.3 comme suit : lorsque le client envoie une requête pour le nom de serveur désiré (e.g. www.Resyd2.com), le résolvant (Resolver) sera appelé, et il envoie, en premier, une requête récursive à son serveur de nom local (étape 1). Dans le cas où l'adresse est cachée au niveau du serveur de nom local, elle sera envoyée directement au client, sinon, le serveur de nom local envoie une requête au serveur de nom racine (Root Name Server : connu sous (.) Servers) (étape 2). Celui-ci retourne une référence du serveur

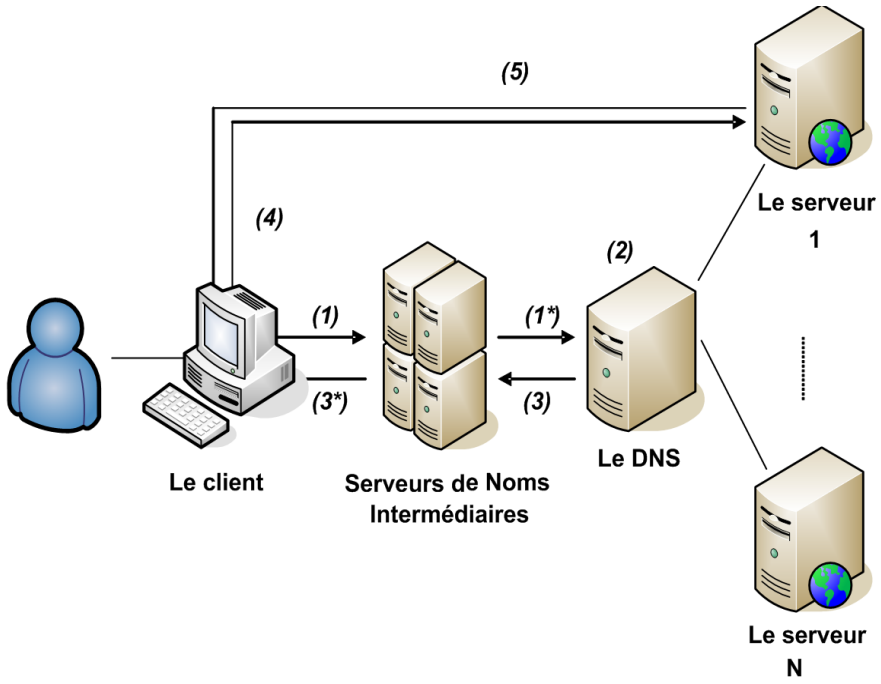

FIGURE 1.3 – Le fonctionnement du DNS.

de nom du niveau supérieur de l'arborescence des noms de domaines (.com dans notre cas) (étape 3). Le domaine de niveau supérieur répond par l'adresse IP du serveur de noms autoritaire du cluster (ns.Resyd2.com) dans lequel le serveur www.Resyd2.com est localisé (étape 4),(étape 5). En interrogeant ce serveur de nom autoritaire, le serveur de nom local obtient l'adresse IP de www.Resyd2.com et sauvegarde le tout dans la "Resource Record " correspondante avec une valeur TTL (Time To Live) qui représente la période de validité de la correspondance (nom de serveur, adresse IP) dans le cache du serveur de nom local (étape 6),(étape 7). Enfin, le client reçoit, du serveur de nom, l'adresse IP du serveur Web sélectionné (étape 8).

La phase de recherche DNS pour l'adresse IP du serveur Web a constitué un point de discussion dans plusieurs travaux et cela à cause du coût qu'elle provoque pour récupérer l'objet Web du serveur, surtout à la croissance des systèmes de serveurs Web géographiquement distribués et les services de distributions du contenu (CDS : Content Distribution Services) qui utilisent le mécanisme DNS pour rediriger les requêtes des clients au serveur le plus proche [21], [22]. Le temps de la phase de recherche DNS est une fraction significative du temps de latence global lorsque le couple (nom serveur, adresse IP) n'est pas caché, notamment dans le cas où le serveur de nom autoritaire est loin du serveur de nom local. Pour remédier à ce problème, beaucoup de travaux de recherche ont visé le cache de l'adresse IP, au niveau du serveur de nom local pendant un TTL, comme solution, afin de réduire la fraction de temps de la phase de recherche DNS [23], [24], [25], [26], [27], [28], [29].

Le choix de la valeur du TTL assignée par le serveur de nom autoritaire était difficile. Un TTL de grande valeur réduit la charge sur les serveurs de noms aussi bien que le trafic réseau, et augmente le nombre de requêtes de recherche DNS pour l'adresse IP, au niveau du serveur de nom local. Ce facteur améliore les performances du mécanisme DNS en termes de temps de réponse, mais augmente la possibilité de péremption des informations des RR (Resource Records) cachées dans les serveurs de nom autoritaire. Par exemple, dans le cas où une machine est déplacée d'une région à une autre (ainsi on lui assigne une nouvelle adresse IP), le RR correspondant pour cette machine change dans le serveur de nom autoritaire.

Un TTL d'une petite valeur permet des changements rapides dans la base de données du DNS (ce qui évite le problème des informations périmées), mais d'un autre côté, elle augmente le trafic réseau où plusieurs requêtes seront envoyées au serveur de nom autoritaire. Une valeur typique de TTL reste un point à discuter, car ce choix est lié à d'autres paramètres comme la charge du système, la distribution des clients,..., etc [23], [29].

1.5 La transaction Web

Dans le contenu Web, qui est typiquement structuré en utilisant le langage HTML (HyperText Markup Language) [30], une page Web est une collection d'objets Web. Cet objet est simplement un fichier d'un format spécifique (e.x. un fichier HTML, une image JPEG, une applet Java, un clip vidéo,...,etc.), qui est accessible par une seule URL (e.x. http ://www.Resyd2.com/index.html). Une URL a deux principales composantes : le " nom hôte " du serveur qui héberge les objets (e.x. www.Resyd2.com) et le chemin d'accès des objets (e.x. index.html).

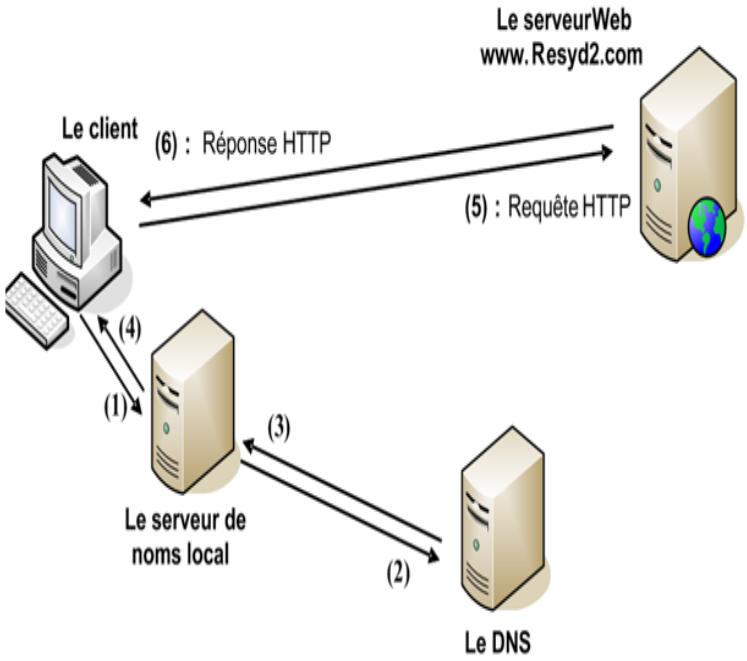

FIGURE 1.4 – Principe d'une transaction Web.

Lors d'une transaction Web, qui est une interaction entre le client et le serveur Web, pour récupérer tous les objets (qui sont les fichiers demandés par le client : Texte, Image, Applet Java,... etc.) qui constitue la page Web, et comme montré dans la figure 1.4, lorsque le client accède au site Web en utilisant l'URL (e.x. www.Resyd2.com), une requête est envoyée au serveur de nom local qui à son tour, répond par l'adresse IP du serveur Web tout en contactant le DNS (c.f. la section 1.3) (étape 1),(étape 2),(étape 3) (étape 4).

Lorsque le client reçoit l'adresse IP du serveur Web, il ouvre une connexion TCP basée sur le protocole HTTP et envoie une requête HTTP (c.f. la section 1.2) au serveur Web pour extraire les objets de la page Web (étape 5),(étape 6). Le client peut mener à une session dans laquelle il émet une séquence de requêtes pour la page Web durant une seule visite au site Web, afin d'afficher tous les objets de cette page.

1.6 Extension des systèmes de serveurs Web

Les administrateurs de sites Web ont toujours opté pour l'augmentation de la capacité des serveurs Web, afin de supporter un grand nombre d'accès aux services fournis par ces serveurs, ainsi que les ressources, tout en fournissant des performances adéquates. La première option telle qu'elle a été classée par Cardellini et al [31], et utilisée pour l'extension de ces systèmes de serveurs Web consiste en la mise à niveau du serveur Web vers une machine puissante et rapide. Cette stratégie, connue sous le nom hardware scale-up, consiste simplement à développer le système en ajoutant plus de ressources (e.x. CPU's, Disques, matériels réseaux,..., etc.) aux machines existantes. Bien que la technique de hardware scale-up soulage momentanément la pression de la charge sur le système, elle reste une solution non efficace en terme de coût à long terme, surtout vis-à-vis de la croissance des demandes de services des clients qui caractérise actuellement le domaine du Web.

Une deuxième solution consiste à améliorer les performances des systèmes de serveurs Web au niveau logiciel, nommée software scale-up. Cela inclut : l'amélioration des systèmes d'exploitation des serveurs (e.x. [32], [33], [34]), la construction d'un serveur Web efficace et portable (e.x. le serveur Web Flash, le serveur Web Zeus [13]), et l'implémentation de différentes techniques d'affectation des requêtes (e.x. [35],). Augmenter la puissance d'une seule machine ne résout pas le problème d'extension des serveurs Web, d'où une autre solution a été proposée, pour remédier à la charge de la croissance des requêtes et fournir un système de serveur Web extensible. Le principe de cette solution est de déployer un système de serveurs Web distribués composé de plusieurs serveurs. La charge de demande de service qui atteint ce système doit être partagée sur tous les serveurs participants, afin d'améliorer les performances. Cette approche, dans laquelle les capacités du système sont augmentées en ajoutant des serveurs, est appelée scale-out . Dans cette approche, nous distinguons deux types d'architecture : une architecture local scale-out où les serveurs se situent dans la même zone réseau (i.e. localement distribués), et l'architecture global scale-out, dans laquelle les serveurs Web sont géographiquement distribués dans différentes régions. La figure ci-dessous, résume les différentes architectures possibles pour l'extension des systèmes de serveurs Web.

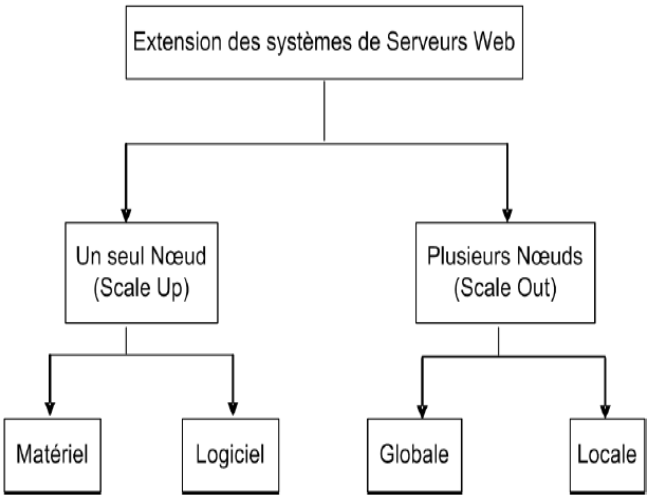

FIGURE 1.5 – Les différentes architectures pour l'extension des systèmes de serveurs Web.

1.7 Les approches d'affectation des requêtes HTTP

Dans cette section, nous allons citer les différentes approches d'affectation des requêtes HTTP, comme la classifié Cardellini et al [36], en se basant sur l'entité qui distribue les requêtes, entrantes au système, sur les serveurs. Pour cela, il existe quatre classes de techniques :

- Approches basées sur le client ;
- Approches basées sur le DNS ;
- Approches basées sur le Répartiteur ;
- Approches basées sur le Serveur.

1.7.1 Approches basées client

Cette approche d'affectation des requêtes utilise le client comme une entité pour partager la charge sur les différents serveurs du système. Par conséquent, elle peut être appliquée à n'importe quelle architecture de serveur Web. Il existe deux approches principales qui mettent le mécanisme de sélection de serveur Web au niveau du client : celles qui se basent sur le Client Web (i.e. le navigateur) et celles qui se basent sur le Proxy côté client.

1.7.1.1 Client Web

Cette entité joue un rôle très important pour l'affectation des requêtes, surtout dans le cas où le client Web connaît l'emplacement des serveurs répliqués du système des serveurs Web. La figure 1.6 montre les étapes durant lesquelles le client affecte les requêtes aux serveurs. Lorsque le Client Web reçoit la requête de

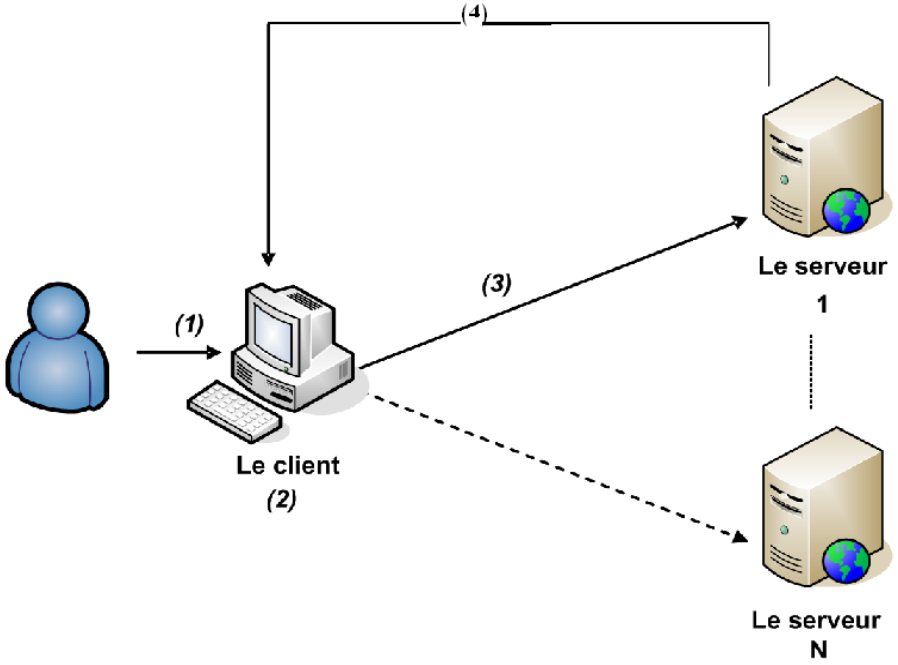

FIGURE 1.6 – Approche basée client.

l'utilisateur (étape 1), il sélectionne un noeud du cluster grâce aux informations qu'il possède sur l'emplacement des serveurs (étape 2). Après la résolution de l'adresse IP, le client envoie la requête au serveur sélectionné (étape 3). Ce serveur Web se charge de répondre au client tout en fournissant les objets demandés dans la requête (étape 4).

Lorsque des requêtes successives arrivent au même client, elles peuvent être affectées à d'autres serveurs choisies par ce dernier. Exemple de cette approche : le navigateur Web Netscape [37], [38] et les clients intelligents (où des fonctionnalités de serveur sont migrées au client).

1.7.1.2 Proxy côté client

Le proxy est une autre entité importante qui peut intervenir lors de la répartition des requêtes sur les serveurs Web. Cette entité est presque similaire à l'entité citée précédemment (le Client Web). Son rôle se voit davantage, grâce au système de cache dont il peut garder trace des URL des serveurs répliqués et router les requêtes client au serveur approprié [39].

L'application de ce type d'approches (i.e. basées clients Web et basées Proxy) dans les systèmes de serveurs Web distribués nécessite quelques modifications au niveau de l'entité cliente. Ces modifications ne sont pas contrôlés par les compagnies qui gèrent ces systèmes.

1.7.2 Approches basées DNS

Dans ce type d'approche, l'entité principale qui assure le partitionnement de la charge sur les différents noeuds du système de serveurs Web distribués est le : DNS, qui est les serveur DNS autoritaire pour le cluster des serveurs. Lors du processus de translation de l'adresse URL vers une adresse IP, le DNS peut sélectionner n'importe quel noeud du cluster de serveurs Web, en implémentant des techniques de gestion efficace afin d'assurer le choix du serveur approprié. La figure 1.7 montre les différentes étapes du fonctionnement de l'approche basée DNS. Deux cas sont possibles dans cette approche : soit que la résolution de l'adresse IP est cachée dans les serveurs de noms intermédiaires, et donc l'adresse du serveur sera directement livrée au client sans passer par d'autres serveurs de noms (étape 1),(étape 3*). Soit que, la requête atteint le DNS (étape 1),(étape 1*), lequel sélectionne l'adresse IP d'un serveur Web avec une valeur TTL (étape 2). Après quoi, le couple (adresse IP, TTL) est envoyé au client (étape 3*), tout en passant par tous les serveurs de noms intermédiaires qui gardent une copie dans leur cache (étape 3). A la fin, le client envoie sa requête au serveur (étape 4) et reçoit une réponse de ce dernier (étape 5).

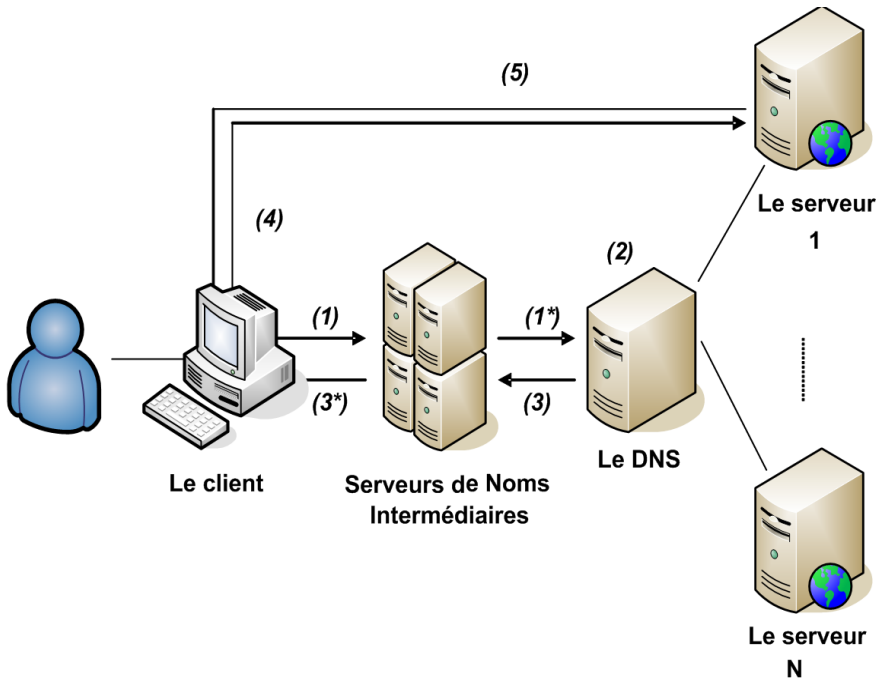

FIGURE 1.7 – Approche basée DNS.

La présence des serveurs de noms intermédiaires et de la valeur TTL, fournit moins de contrôle pour le DNS sur les requêtes des clients. D'un côté, ce contrôle limité évite que le DNS devient un goulot d'étranglement, mais d'un autre côté, le DNS ne contrôle pas la charge affectée aux différents noeuds du cluster, ce qui a comme risque de surcharger l'un des serveurs. Il existe deux classes de cette approche : les approches qui utilisent la même valeur TTL pour toutes les résolutions des adresses IP connu sous le nom : les algorithmes avec un TTL constant (Constant TTL algorithms), implémentés dans National Center for Supercomputing Applications (NCSA) [40], SunSCALR framework, Ibmnamed [37], et Cisco DistributedDirector [41], et les algorithmes qui adaptent la valeur du TTL en se basant sur des informations dynamiques que se soit de la part des serveurs et/ou des clients, connus comme des algorithmes avec un TTL dynamique (Dynamic TTL algorithms). Un exemple de cette classe d'algorithmes a été proposé dans [42].

1.7.3 Approches basées Répartiteur

Une approche basée répartiteur est une alternative de l'approche basée DNS, dont le but est d'assurer un contrôle total sur les requêtes clients et de masquer la technique d'affectation sur les différents serveurs. Cela est réalisé en étendant la virtualisation de l'adresse IP du niveau URL (cas du DNS) au niveau IP. En fait, cette approche alloue une seule adresse IP virtuelle (VIP) au cluster de serveurs Web, qui est l'adresse affectée au répartiteur. Ce dernier se chargera de l'affectation des requêtes et du choix du serveur approprié (i.e. chaque serveur a une adresse IP privée que le répartiteur utilise pour l'identifier) en se basant sur l'algorithme implémenté à ce niveau.

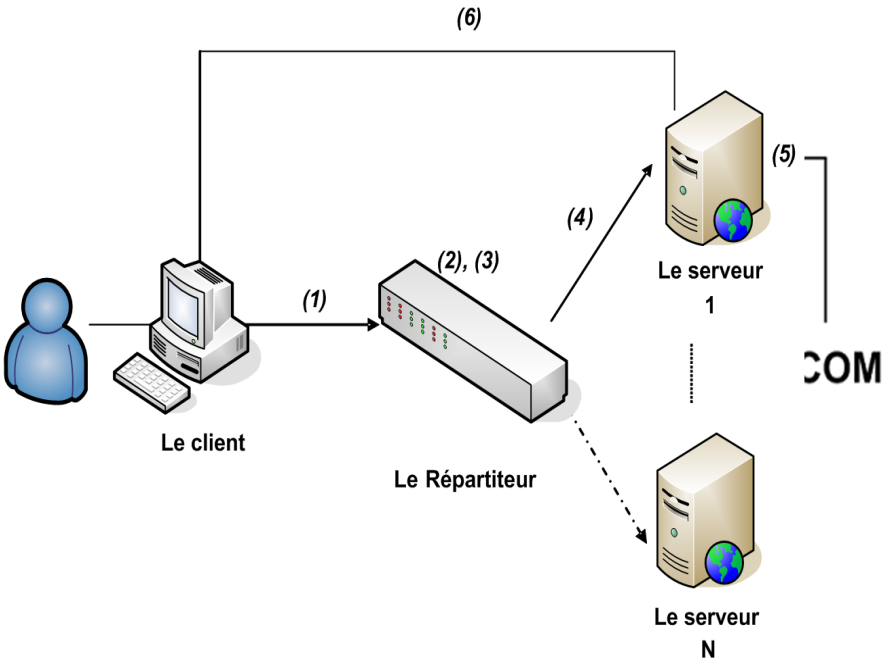

FIGURE 1.8 – Approche basée Répartieur.

La figure 1.8 montre un exemple du mécanisme de l'approche. Les requêtes clients arrivent touces au répartiteur car c'est lui qui a l'adresse IP publique (étape 1). A ce moment, et après l'étape de choix du serveur reposant sur l'algorithme d'affectation (étape 2), le répartiteur réécrit les adresses IP de destination de chaque paquet (en modifiant ainsi les checksum IP et TCP, car les deux sont liés à l'adresse destination) et remplace son adresse IP virtuelle par l'adresse IP du serveur sélectionné (étape 3). Bien qu'une requête consiste en un ensemble de paquets, le répartiteur affecte toujours les paquets qui appartiennent à la même connexion au même serveur Web, tout en analysant l'adresse source de chaque paquet (étape 4). Le serveur Web, à son tour, modifie son adresse IP par celle du répartiteur (étape 5), et envoie la réponse au client (étape 6).

Magicrouter, LocalDirector [43], IBM Network Dispatcher, Distributed Server Groups approach et Cisco DistributedDirector [41] illustrent cette approche.

1.7.4 Approches basées Serveur

L'approche basée serveur utilise deux étapes d'affectation : les requêtes clients sont d'abord affectées par le DNS à un serveur Web du cluster , puis chaque serveur réaffecte la requête reçue à n'importe quel autre serveur du cluster. A la différence des autres approches (i.e. basée DNS et basée Répartiteur), cette dernière permet aux serveurs de participer à l'affectation des requêtes aux différents serveurs Web du cluster. Un cas de cette approche, dans lequel elle utilise le mécanisme de redirection HTTP, est montré dans la figure 1.9. En premier lieu, les requêtes clients qui arrivent au système sont affectées par le DNS à l'un des serveurs participants (étape 1),(étape 1*),(étape 2) (étape 3),(étape 3*). Dans le cas où le serveur prend la décision de réaffecter la requête à un autre serveur (e.x. à cause de sa charge élevée en terme de requêtes qui existent dans sa file), il utilise le mécanisme de redirection du protocole HTTP. Dans l'exemple, le serveur 1 reçoit la requête (étape 4) et décide de la rediriger vers le serveur 2 (étape 5) (étape 6). Le client, donc, envoie sa requête vers le serveur 2 (étape 7) et reçoit une réponse (étape 8).

Le Scalable server World Wide Web (SWEB) system [44], des algorithmes de redirection (DR, CR-RR, ACR-RR, ACR-RR-Alarm,... etc.) proposés dans [45], et l'algorithme Distributed Packet Rewriting [46] sont des exemples qui illustrent cette approche.

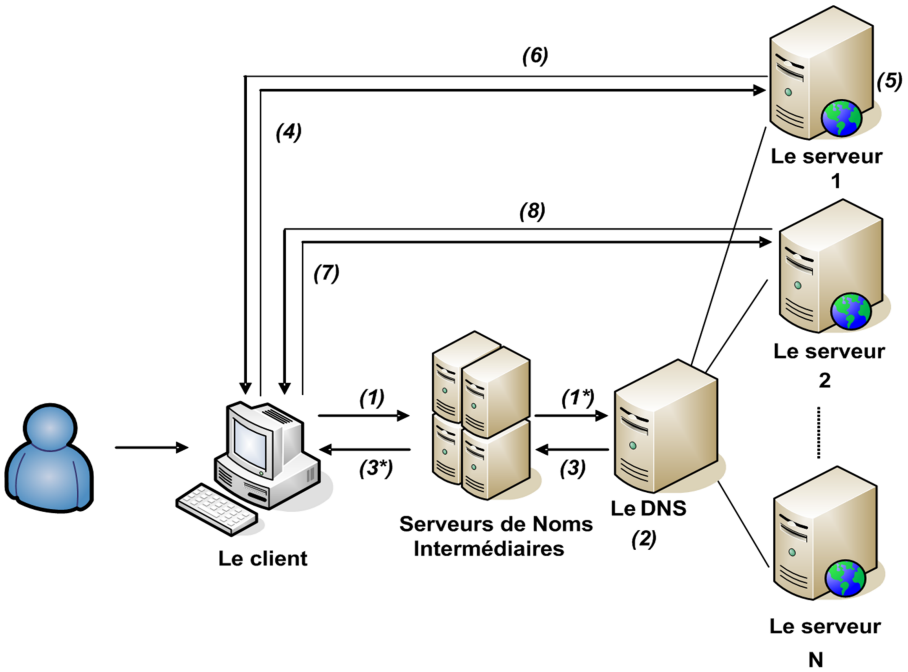

FIGURE 1.9 – Approche basée Serveur.

1.8 Conclusion

Dans ce chapitre, nous avons présenté les différents composants et architectures des systèmes de serveurs Web distribués. Cet état de l'art a donné une vue générale sur la plate-forme sur laquelle se base les différentes approches qui assurent le partage de la charge des requêtes HTTP des clients sur les différents serveurs du cluster. Dans les deux chapitres suivants, nous citerons les différents travaux de recherche concernant le problème de la qualité de service en terme de partage optimal des requêtes. D'abord, dans le Chapitre 2 nous fournissons une définition de la distribution qui a caractérisé le domaine du Web (Heavy-Tailed), ainsi que les métriques de performance pour l'évaluation des techniques d'affectation de requêtes.

2

TRAVEAUX ANTÉRIEURS ET CRITIQUES

2.1 Introduction

Dans plusieurs travaux de recherche, comme [47], [5], des techniques avaient été développées en se basant sur la supposition que la distribution des tailles des tâches Web est exponentielle. L'apparition de quelques travaux de recherche, comme [48], [3], [2], [49], lesquels ont motivé la distribution Heavy-Tailed, avait orienté l'axe de recherche et forcé les chercheurs à se rediriger pour adapter les anciennes techniques à cette distribution, ou bien proposer d'autres nouvelles qui répondent à sa nature.

En effet, beaucoup de questions restent ouvertes pour le problème d'affectation des tâches Web dans les systèmes de serveurs Web distribués, afin d'assurer une qualité de service optimale. La nature de la distribution Heavy-Tailed a suscité plusieurs travaux de recherche, dont le but commun est de surpasser les problèmes liés à cette nature.

Dans ce chapitre, nous allons exposer les différents travaux de recherche qui existent dans la littérature pour les stratégies d'affectation des tâches Web dans les systèmes de serveurs Web distribués basés répartiteur comme c'est classifiés dans [50]. Nous allons voir les techniques qui n'assument aucune connaissance de la charge des serveurs connues comme des techniques statiques, et leurs opposées, les techniques dynamiques. Nous citons ainsi celles qui supposent que la taille des tâches Web est connue à priori de celles qui n'utilisent pas cette sorte d'information.

Avant d'entamer ces techniques, nous présenterons d'abord la distribution de Heavy-Tailed.

2.2 Distribution Heavy-Tailed

La distribution Heavy-Tailed (connue aussi sous le nom de distribution Power-Low) a été observée dans plusieurs phénomènes naturels, y compris les deux phénomènes physique et sociologique. Un exemple des phénomènes physiques est la distribution des dégâts d'ouragan. Beaucoup d'ouragans provoquent très peu de dégâts, alors qu'un petit pourcentage d'ouragans en cause beaucoup. Un autre exemple, pour les phénomènes sociologiques, est la distribution du peuple autour du monde, où la plupart des endroits sont vides ou peu peuplés, tandis qu'une petite partie des terres est occupée par d'énormes populations.

Récemment, la distribution Heavy-Tailed, avait été observée dans les systèmes informatiques. En particulier, les tailles des tâches ont été découvertes exhiber une distribution Heavy-Tailed. Par exemple, si nous considérons le besoin, en terme de temps CPU, des tâches, la plupart des tâches exigent une petite fraction de temps CPU par rapport à un petit nombre de tâches dont ils exigent un temps CPU élevé.

La figure 2.1 montre la distribution des besoins en terme de temps CPU des processus UNIX . Cette figure montre les tâches qui exigent au moins 1 seconde de temps CPU et la distribution s'adapte étroitement à la courbe :

$$Pr\{BesoindesProcessusCPU > t\} \sim \frac{1}{t}$$

La distribution vue ci-dessous est un exemple de la distribution Heavy-Tailed. En général, la distribution est dite Heavy-Tailed si :

$$Pr\{T > t\} \sim t^{-\alpha}$$

Où $0 < \alpha < 2$. (Dans les mesures plus haut, $\alpha = 1$). La distribution la plus simple de Heavy-Tailed est la distribution Pareto, dont la fonction densité est définie par :

$$f(t) = \alpha k^{\alpha} t^{-\alpha-1}, \alpha, k > 0, t \geq k$$

Sa fonction de répartition :

$$F(t) = Pr\{T > t\} = 1 - (\frac{k}{t})^{\alpha}$$

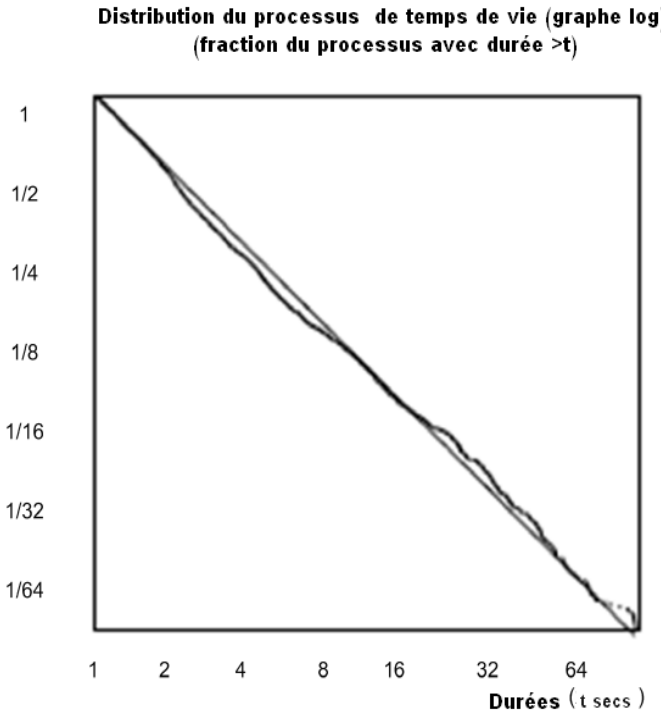

FIGURE 2.1 – Distribution des temps CPU des processus UNIX.

Toutefois, la courbe de cette distribution est loin de celle d'une distribution exponentielle. Cette différence peut être vue dans la figure 2.2. En d'autre termes, la variable aléatoire qui suit une distribution Heavy-Tailed peut être d'une grande valeur, avec une probabilité non négligeable, contrairement à la distribution exponentielle où :
$Pr\{X > x\} \sim 0, x \longrightarrow \infty$.

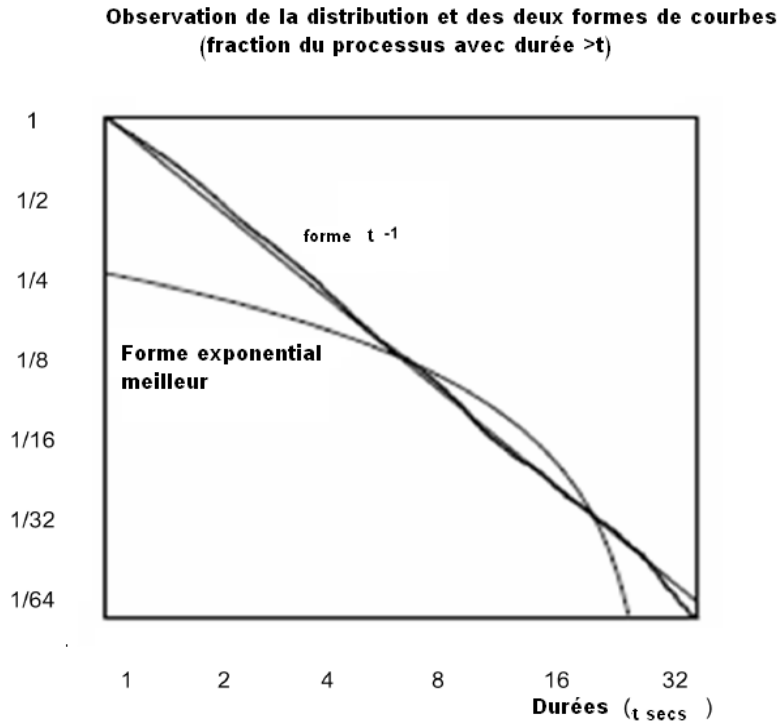

FIGURE 2.2 – La courbe de la distribution exponentielle pour les données mesurées des temps CPU .

La distribution Heavy-Tailed a des propriétés qui la caractérise :

◇ **taux de défaillance décroissant** : en particulier, plus une tâche a fonctionné longtemps, plus elle est prévue de continuer son fonctionnement. En fait, pour la distribution vue dans la figure 2.1, une tâche d'âge CPU t (i.e. une tâche qui a utilisé t seconde de temps CPU) a une probabilité $\frac{1}{2}$ d'utiliser d'autres t secondes.

◇ **variance infinie** (et si $\alpha \geq 1$, une moyenne infinie). En réalité, n'importe quel ensemble de mesures ou de traces finies a une variance finie. La caractéristique c'est que la variance est très grande lorsque la charge de travail (i.e. workload) est Heavy-Tailed.

◇ la propriété qu'une petite fraction ($< 1\%$) des tâches de grande taille participe à une grande fraction (la moitié) de la charge de travail . Contrairement à la distribution exponentielle où 1% des tâches de grande taille contribue seulement à 5% de la charge de travail . Pour comprendre cette propriété, considérons l'exemple suivant : dans la vie, la majorité des gens ont très peu d'argent , peu de gens ont beaucoup d'argent, et 1% des gens riches ont plus d'argent que tous les autres gens.

Plus α est petit, plus la dernière propriété est confirmée, i.e. plus est petite la fraction de grandes tâches qui contribue à la moitié de la charge totale du système.

Cette distribution est connue dans plusieurs mesures récentes dans les systèmes informatiques. Cela inclut, par exemple :

- ◇ temps CPU nécessaire des processus UNIX mesuré à Bellcore :$1 \leq \alpha \leq 1.25$ [51].
- ◇ temps CPU nécessaire des processus UNIX mesuré à UC Berkeley :$1 \sim 1$.
- ◇ les tailles des fichiers stockés dans les sitesWeb et les tailles des fichiers accessibles : $1.1 \leq \alpha \geq 1.3$ [2],[3]. par les requêtes Web
- ◇ les tailles des fichiers stockés dans les systèmes de fichiers UNIX [49].
- ◇ temps $\frac{I}{O}$.
- ◇ tailles des transferts FTP sur Internet $0.9 \leq \alpha \geq 1.1$ [52].

Le temps SlowDown

Le temps SlowDown d'une tâche Web est son temps d'attente divisé par sa taille (temps d'exécution). C'est le temps d'attente de la tâche Web normalisé par sa taille. Minimiser le temps SlowDown est souvent important, parce que cela a pour résultat une minimisation du temps d'attente des petites tâches Web par rapport aux grandes tâches Web. Cela est dû à la séparation des petites tâches Web de celles de grande taille dans la file d'attente. Le SlowDown d'une tâche i est défini comme :

$$TempsSlowDown_i = \frac{TempsAttente_i}{TempsService_i}$$

D'où le temps slowdown moyen de n tâches dans la file est :

$$TempsSlowDownMoyen = \sum_{i=1}^{n} \frac{TempsSlowDown_i}{n}.$$

2.2.0.1 Le temps de séjour

Une autre métrique qui influe sur les performances du système est le temps de séjour. Le temps de séjour est défini comme la somme du temps d'attente additionnée au temps de service, connu aussi comme le temps depuis l'arrivée de la tâche Web jusqu'à son départ. Le temps de séjour d'une tâche i est défini comme :

$$TempsSejour_i = TempsAttente_i + TempsService_i.$$

Donc, le temps de séjour moyen est :

$$TempsSejourMoyen = \sum_{i=1}^{n} = \frac{TempsSejour_i}{n}.$$

Réduire cette métrique a de meilleurs gains en performances, car plus le temps de séjour est minimisé, plus le client sera satisfait par le temps de réponse du système.

2.2.0.2 Le temps d'attente

Le temps d'attente d'une tâche Web est définie comme le temps depuis lequel la tâche Web arrive au système jusqu'à ce qu'elle quitte le système, moins son temps de service. Nous calculons, pour l'évaluation, le temps d'attente moyen, i.e. le temps d'attente moyen de la tâche Web par rapport aux autres tâches. En d'autres termes, le temps d'attente d'une tâche Web est la somme des temps de service de toutes les tâches qui la précèdent dans la file d'attente, donc le temps d'attente d'une tâche n, est :

$$TempsAttente_n = \sum_{i=1}^{n-1} TempsService_i.$$

Où le temps de service moyen pour un serveur k est :

$$TempsServiceMoyen_n = \sum_{i=1}^{n_k} \frac{TailleTache_i}{(n_k * CapaciteTraitemen_k)}.$$

Minimiser cette métrique a comme résultat une amélioration de la latence vue par le client.

2.3 Techniques assumant une connaissance de la taille des tâches Web

On distingue deux classes de techniques qui assument que la taille des tâches Web est connue à priori : les techniques statiques et les techniques dynamiques.

2.3.1 Techniques statiques

Ce type de techniques n'utilise aucune sorte d'information lors du processus d'affectation des tâches Web. Les techniques qui existent sont :

2.3.1.1 Random(ou uniform Random

Dans cette technique naïve, les tâches Web sont envoyées aux serveurs aléatoirement. Le serveur i reçoit une tâche Web avec une probabilité $\frac{1}{h}$ où h est le nombre de serveurs.

2.3.1.2 Round-Robin

Dans Round-Robin, les tâches Web sont affectées aux serveurs dans une manière cyclique, où la tâche Web i sera affectée au serveur i mod h. Cette technique essaye d'égaliser le nombre de tâches Web dans chaque serveur.

L'avantage de ces deux techniques est qu'elles sont simple à implémenter, mais les auteurs dans [4], ont montré que les performances du système sous Random et Round-Robin, sont similaires et non optimales.

2.3.1.3 SITA-E (Size Interval Task Assignment with Equal load)

SITA-E [4] est une technique statique qui assume que la taille de la tâche Web est connue à priori. L'idée principale de SITA-E est d'assurer que la charge affectée pour chaque serveur est la même. Donc, chaque serveur a un intervalle de taille, et accepte seulement les tâches Web dont la taille appartient à cet intervalle. Ces intervalles sont choisis de manière à ce que tous les serveurs reçoivent la même charge. SITA-E, en utilisant les intervalles de tailles, s'est avéré efficace pour la réduction de la variance dans les tailles de tâches Web. Cela est dû au fait que les tâches Web de petite taille seront mises dans la même file que les autres petites tâches Web qui appartiennent au même intervalle de taille. Donc, les tâches Web de grande taille ne seront jamais mises dans la même file que les petites tâches Web, et ces dernières ne seront jamais retardées lors de son exécution.

La technique SITA-E est simple à implémenter. A la différence des techniques dynamiques, elle ne considère aucune information sur la charge des serveurs lors de sa tâche d'affectation.

Une des limites de cette technique est que les expressions mathématiques des intervalles de taille sont statiques, et cela influe sur les performances lorsque la distribution des tailles des tâches Web change. Dans [4], les résultats de simulation ont montré que SITA-E se comporte mal quand la variabilité augmente (i.e.α diminue), à cause de la taille des tâches Web qui appartiennent toutes à l'intervalle du serveur 1, et donc les performances du système convergent vers ceux du serveur 1.

SITA-V (Size Interval Task Assignment with Variable load) La technique

SITA-V, est basée sur la propriété de la distribution Heavy-Tailed dans laquelle il y a un grand nombre de tâches Web de petite taille et un petit nombre de tâches Web de

grande taille. Donc, lorsque les tâches Web arrivent, le répartiteur inspecte leur taille et affecte toutes les tâches Web de petite taille aux serveurs qui sont chargés au-dessous de la charge moyenne du système. Pour les grandes tâches Web, il les affecte aux serveurs qui sont chargés au-dessus de la charge moyenne du système.

L'idée mère dans SITA-V est de réduire le SlowDown moyen en mettant toutes les tâches Web de petite taille (qui sont très nombreuses) pour qu'elles soient exécutées, par les serveurs qui ont une basse charge. Donc le SlowDown moyen sera réduit (car le SlowDown des tâches Web de petite taille est petit). Nous voyons que cette idée ne convient pas si la distribution est exponentielle, parceque les tâches Web ont presque la même taille, ainsi, les serveurs qui ont plusieurs tâches Web seront surchargés. Par contre, ceux qui ont peu de tâches Web serons moins chargés.

Dans la littérature, les résultats ont montré que SITA-V a de mauvaises performances lorsque $0.9 < \alpha < 1$, parceque la taille des petite tâches Web augmente, donc SITA-V souffre lors de la décision d'affectation. SITA-V a montré aussi de mauvaises performances quand la charge du système est élevée, car elle a moins de flexibilité pour basculer à travers les tâches Web lors du choix des petites et grandes tâches Web. Une autre limite de SITA-V, qui est une conséquence de sa nature, est qu'elle n'utilise pas complètement la capacité de traitement des serveurs avant que les tâches Web de grande taille arrivent au système. Dans la littérature, la technique SITA-V est attractive dans les systèmes de serveurs Web distribués basés DNS, grâce à son implémentation simple qui n'exige aucune communication du serveur au répartiteur.

2.3.1.4 SITA-U (Size Interval Task Assignment with Unbalanced load)

Comme les techniques citées précédemment (i.e. SITA-E et SITA-V), SITA-U [6] assume que la taille de la tâche Web est connue à priori. SITA-U utilise des intervalles de tailles dont le répartiteur se base pour affecter les tâches Web aux différents serveurs. Ces intervalles de tailles sont choisis de façon à minimiser le SlowDown.

Dans [6], les résultats des tests de performance du SlowDown moyen ont montré une amélioration importante des performances de SITA-U par rapport à SITA-E. Lorsque la charge du système est élevée , la technique SITA-U provoque de mauvaises performances, car toute la fraction des tâches Web de petite taille s'oriente vers le serveur 1, d'où la convergence des performances du système vers ceux du serveur 1.

2.3.1.5 EQUILOAD

La technique EQUILOAD [7] est inspirée de quelques anciens travaux sur les techniques basées sur la taille et qui s'appuient sur le principe de l'intervalle de taille pour allouer les tâches Web aux serveurs [4]. La technique EQUILOAD partitionne les tailles des tâches Web en N intervalles, $[s_0 = 0, s_1), [s_1, s_2)...[s_{N-1}, s_N = \infty)$. Le serveur i sera responsable pour le traitement des tâches Web dont leur taille se situe entre s_{i-1} et s_i. La caractéristique de la technique EQUILOAD est qu'elle utilise une double technique d'allocation. En premier, le répartiteur assigne chaque tâche Web rapidement à un des serveurs participant en utilisant une technique simple comme Random ou Round-Robin. Après quoi, lorsque le serveur i reçoit la tâche Web du répartiteur, il vérifie sa taille 's '. Si $s_{i-1} < s < s_i$, le serveur met la tâche Web dans la file, autrement, il réalloue cette tâche Web au serveur j qui satisfait la condition $s_{j-1} < s < s_j$.

2.3.1.6 ADAPTLOAD

La technique ADAPTLOAD [**?**] est une version on-line de EQUILOAD qui a été développée pour traiter le problème de partitionnement statique des intervalles de tailles. ADAPTLOAD fournit un mécanisme online pour adapter les intervalles de tailles au changement de la charge de travail. L'idée de base de cette technique est que les bornes des intervalles changent en s'appuyant sur les K dernières tâches Web arrivées au système. Donc, si la charge de travail change, les intervalles de tailles changent aussi.

La limite de la majorité des techniques basées sur la taille, par utilisation des intervalles de tailles, est que son comportement conduit à des situations dans lesquelles les ressources des serveurs ne seront pas complètement utilisées. Principalement, dans le cas où plusieurs tâches Web de petite taille arrivent successivement sans avoir une tâche Web de grande taille, où les serveurs qui ont un intervalle pour les tâches Web de grande taille restent libres.

2.3.1.7 Techniques dynamiques

Ces techniques prennent en charge des informations relatives à la charge des serveurs et la capacité de traitement lors de la décision d'affectation des tâches Web.

2.3.1.8 CQ (Central Queue)

Dans cette technique, le répartiteur est considéré comme possédant une file d'attente FCFS (First Come First Served) et les tâches seront gardées jusqu'à ce que l'un des serveurs devient libre. Bien que cette technique montre de meilleures performances sous une distribution exponentielle quand il y a une variabilité basse, les auteurs dans [4], [6], ont montré qu'elle provient de mauvaises performances sous une charge de travail réel .

2.3.1.9 CS-CQ (Cycle Stealing with Central Queue)

La technique CS-CQ [53] est une variante de Central Queue Policy. L'idée principale de Cycle Stealing with Central Queue est que les serveurs sont divisés en deux groupes distincts : un groupe désigné comme " Short Servers ", lequel est dédié pour l'exécution des tâches Web de petite taille, et un autre groupe nommé " Long Servers " pour le traitement des tâches Web de grande taille. Comme pour le cas de la technique Central Queue, les tâches Web sont gardées dans la file d'attente du répartiteur. Lorsque l'un des serveurs " Short Servers " devient libre, il prend la première tâche Web, de petite taille, de la file pour le traitement. Egalement, pour un des serveurs " Long Servers ", lorsqu'il devient libre, il prend la première tâche Web de grande taille de la file. Néanmoins, s'il n'y a pas de tâche Web de grande taille, le serveur des " Long Servers " exécute une tâche Web de petite taille. Dans un cas similaire, et si les " Short Servers " deviennent libres, les " Long Servers " deviennent " Short Servers " et vice versa.

2.3.1.10 SRPT (Shortest Remaining Processing Time)

Dans SRPT, la taille de la tâche Web est assumée connue à priori. L'idée mère de cette technique est de donner la priorité aux tâches Web qui sont rapides ou n'exigent pas un temps d'exécution grand. Lorsque le serveur exécute une tâche Web de grande taille, et une autre de petite taille arrive, la tâche Web de grande taille sera arrêtée et le serveur bascule pour exécuter celle de petite taille.

Ce comportement présente quelques limites pour cette technique. La première est que les tâches Web de grande taille seront toujours arrêtées lorsque celles de petite taille arrivent, ajoutant ainsi quelques retards additionnels par le processus de basculement. Une autre limite, qui est une conséquence de la nature de la technique, est que la priorité est toujours donnée aux tâches Web de petite taille, ce qui influe vraiment sur le traitement des tâches Web de grande taille lorsque le nombre de petite tâches Web est énorme.

Des tests de performance ont été réalisés, dans [35], en se basant sur la supposition que toutes les tâches Web sont pour des fichiers de petite taille (i.e. des tâches Web de petite taille). Cette technique a prouvé de meilleures performances (en se basant sur cette supposition), mais ce n'est pas toujours le cas dans les systèmes de serveurs Web distribués

où il y a des fichiers Web de petite taille et d'autres de grande taille. et vice versa.

2.3.1.11 LLF (Least Loaded First)

A la différence des techniques statiques, les techniques dynamiques essayent d'affecter les tâches Web basées sur quelques informations sur les serveurs. Quelques techniques, comme Shortest-Queue, considèrent le contenu de la file pour l'affectation des tâches Web. Dans cette technique, le répartiteur envoie les tâches Web, qui arrivent, au serveur qui a le minimum des nombres de tâches Web dans la file. Une autre technique, comme Least-Work-Remaining, utilise le travail restant au serveur comme critère pour affecter les tâches Web en sélectionnant le serveur qui a le minimum de ce critère.

Dans [4], les résultats ont montré que, prendre en charge juste la charge aux serveurs, n'améliore pas les performances, à cause de la variabilité de la charge de travail qui augmente par le fait que la distribution est Heavy-Tailed et que le plus important c'est de minimiser cette variabilité. Harchol et al [4], ont découvert que la technique Least-Loaded-First est similaire à la technique Central Queue.

2.3.1.12 GC (Global Class)

Awerbuch et al [11] ont proposé une technique pour l'affectation des tâches Web. L'idée de base de cette technique est de créer des classes de tâches Web. L'algorithme de cette technique crée ces classes en s'appuyant sur le temps d'exécution restant. Donc, si ce dernier est dans l'intervalle $[2_k, 2_{k+1})$, la tâche Web sera de classe k , quand la tâche Web arrive, l'algorithme cherche le serveur qui est libre ou dont la classe de cette tâche Web est inférieure à celle de la tâche Web que le serveur est entrain d'exécuter. Ainsi, cette nouvelle tâche est insérée dans la file du serveur, et ce dernier commence son traitement. Sinon, l'algorithme met la tâche Web dans son pool. Chaque fois que le serveur termine l'exécution d'une tâche Web, l'algorithme compare la classe de la tâche Web contenue dans la file du serveur avec celle de la tâche Web contenue dans son pool. La tâche qui a la classe minimale sera exécutée.

Une des limites de cette technique est le problème de famine des tâches Web de grande taille à cause de l'affectation qui est basée sur les classes de tailles entre les petites tâches Web et les grandes tâches Web, d'où les petites tâches Web auront toujours la classe inférieure par rapport aux grandes tâches Web.

2.3.1.13 TG (Task Grouping)

L'approche Task Grouping [54] désire atteindre le même temps de séjour dans chaque file d'attente des serveurs, pour réduire le temps de réponse du système. L'idée principale de cette technique est : lorsque le répartiteur a un ensemble de tâches Web, il prend un sous-ensemble des serveurs et essaye d'assigner ces tâches Web à un serveur basé sur sa charge de traitement. Donc, l'algorithme calcule la charge de chaque serveur proportionnellement à sa capacité de traitement en considérant la charge de traitement restante des autres serveurs de l'ensemble. Pour chaque serveur, l'algorithme trouve le sous-ensemble de tâches Web qui appartient à sa charge calculée et les assigne à ce serveur.

Les résultats de test dans [54], ont montré que cette technique a de mauvaises performances lorsque la variabilité augmente (α diminue). Cela est dû au fait que l'algorithme regroupe un grand nombre de tâches Web de petite taille dans la même file, ce qui a comme conséquence une augmentation sur le temps d'attente et le temps SlowDown. Une autre limite concerne le répartiteur, qui peut devenir un goulot d'étranglement (Bottleneck), car chaque fois qu'une tâche Web arrive, il construit l'ensemble de tâches.

2.3.1.14 LFF (Least Flow time First)

Les auteurs dans [8], [9], [10] avaient proposé des variantes d'une nouvelle technique pour surpasser les deux limites de LLF qui sont :

1) LLF ne considère pas l'ordre d'exécution des tâches Web.
2) lorsqu'elle assigne une tâche Web, elle ne considère pas la capacité de traitement du serveur. LFF affecte dynamiquement la tâche au serveur qui est le moins chargé et qui a une grande capacité de traitement. Cette technique utilise une file multi section pour réduire le retard causé par la variation en taille.

Les résultats de tests de performances ,dans [8], [9], [10],ont montré une amélioration de LFF par rapport à d'autres techniques comme Random, LLF et SITA-E. Une des limites de cette technique est la partition statique des intervalles de tailles dans les files multisection. Cela influe dans le cas des systèmes qui proposent différents service appelés : systèmes multi services, où chaque type de service se comporte selon la distribution Heavy-Tailed.

Les techniques dynamiques doivent exiger un échange d'informations entre les serveurs et le répartiteur. Cet échange dégrade les performance du système surtout dans le cas où la distance entre le répartiteur et les serveurs est importante, à travers le réseau Internet.

2.4 Techniques n'assumant aucune connaissance de la taille des tâches web

L'autre type de techniques, c'est les techniques qui assument que la taille des tâches Web n'est pas connue à priori. Dans la littérature, il existe deux techniques de ce genre : TAGS et TAPTF.

2.4.1 TAGS (Task Assignment by Guessing Size)

A la différence des techniques précédentes, où la taille des tâches Web est connue à priori , La technique TAGS, n'assume aucune connaissance de la taille des tâches Web. TAGS associe des intervalles de temps pour chaque serveur. Quand la tâche Web arrive, le répartiteur l'assigne au premier serveur, qui commence et poursuit son exécution jusqu'à la fin du temps affecté au serveur. Si la tâche Web n'a pas fini son exécution, elle sera détruite et recommencera son exécution au serveur suivant. TAGS a la caractéristique de réduire la variance des tailles des tâches Web en définissant l'intervalle de temps dans chaque serveur, et les tâches Web de petite taille s'exécutent dans les serveurs de niveau plus haut et celles de grande taille, dans les serveurs de niveau le plus bas.

Dans la littérature, il a été montré que les performances de TAGS sont mauvaises, lors du processus d'affectation des tâches Web, sous une charge élevée du système. Une des limites de cette technique est qu'elle met quelques pénalisations en détruisant et en relançant les tâches Web, spécialement pour les tâches Web de grande taille, vu qu'elles sont celles qui seront souvent relancées. Une autre limite de cette technique est l'excès créé lors de la phase de re-exécution.

2.4.2 TAPTF (Task Assignment based on Prioritising Traffic Flows)

Comme TAGS, TAPTF [12] assume que la taille des tâches Web n'est pas connue à priori. TAPTF est proposée pour traiter deux limites de TAGS :

1) réduire la variance des tâches Web qui partagent la même file d'attente pour re-grouper ensemble les tâches Web qui ont presque la même taille .

2) réduire l'excès dans les serveurs, causé par les tâches Web, dont le temps de traitement n'appartient pas à l'intervalle de temps limite d'un serveur, et qui seront re-exécutées aux serveurs suivants.

Une comparaison analytique entre TAPTF, TAGS et Random a été réalisée, dans [12]. Les résultats ont montré que TAPTF a de meilleures performances par rapport à TAGS, dans le cas d'une variabilité basse et une charge de système élevée. Une des limites de cette technique est que le problème de migration des tâches entre les serveurs est diminué mais n'est pas éliminé complètement. Un autre problème est celui des techniques qui ont un intervalle de taille, où dans une variabilité élevée, toutes les tâches Web seront affectées au serveur 1.

2.5 Conclusion

Dans ce chapitre, nous avons présenté tous les travaux de recherche qui existent dans la littérature concernant le problème d'affectation des tâches Web dans les systèmes de serveurs Web distribués. Les avantages et les inconvénients, par rapport à la nature du trafic Web et le besoin en terme de qualité de service, de chacune des techniques ont fait l'objet d'une motivation pour la proposition d'une nouvelle technique. Premièrement, dans le Chapitre 3, nous allons détailler le principe de la nouvelle technique tout en poursuivant le modèle des systèmes de serveurs Web sur lequel elle se base, et l'algorithme qui caractérise son fonctionnement. Pour valider et prouver l'efficacité de cette technique, une étude analytique a été réalisée par la technique de la modélisation par file d'attente. Les résultats obtenus sont présentés dans le chapitre .

3

APPROCHE BASÉE RÉPARTITEUR POUR L'AFFECTATION DES TÂCHES WEB

3.1 Introduction

Dans ce chapitre, nous modélisons une nouvelle technique pour l'affectation des tâches Web dans un système de serveurs Web distribués. Cette technique, comme d'autres existantes, fonctionnent avec un volume important de tâches Web qui arrivent au système dont le répartiteur essaye de les affecter aux serveurs participants et vise à traiter quelques limites liées aux différentes techniques qui existent dans la littérature, en particulier les problèmes posées par les tâches Web de grande taille qui déséquilibre la charge du système et a de l'influence sur ses performances (i.e. temps de réponse).

Le chapitre est organisé comme suit : dans un premier temps, dans la section 3.1, nous motivons l'idée principale de la technique proposée. Nous montrons ensuite, dans la section 3.2, le modèle des serveurs Web sur lequel est basée la technique, et les différentes hypothèses pour la distribution et la taille de la tâche Web. Dans la section 3.3, nous envisageons l'algorithme de la nouvelle technique et ses caractéristiques en terme d'amélioration autant pour les tâches Web de grande taille que pour celles de petites taille, en suite nous modélisons la nouvelle technique proposée par une file d'attente M/Bp/1 FCFS.

3.2 Problèmes rencontrés dans l'affectation des tâches web :

le problème des tâches Web de grande taille reste le problème principal qui confirme la non convenance de ces dernières à cause de la propriété de la distribution des tâches Web (i.e. Heavy-Tailed) dont moins de 3% des tâches Web de grande taille participe à plus de 50% dans la charge totale du système. Ces tâches Web ont de mauvaises propriétés :

- elles sont difficiles à être affectées convenablement aux différents serveurs à cause de leurs tailles énormes qui impliquent un temps d'exécution important, un temps d'attente élevé,..,etc.

- lorsqu'elles sont dans la même file d'attente avec les tâches Web de petite taille, ces dernières seront retardées lors de leur exécution d'où l'influence en sens négatif sur leur temps d'exécution, temps d'attente,..., etc.

Ces problèmes ont constitué le premier facteur qui a motivé la proposition de cette technique et qui a comme but de :

- traiter le problème de la taille énorme des tâches Web jugées de grande taille.

- traiter le problème du retard pour les tâches Web de petite taille causé par celles de grande taille lorsque elles sont ensemble sur la même file d'attente.

3.3 Le Modèle

Nous allons présenté, dans cette section, le modèle sur lequel est basée la technique et qui comporte le modèle de serveurs Web distribués, la taille des tâches Web et la connaissance ou non de la taille des tâches Web.

3.3.1 Le modèle de serveurs Web distribués

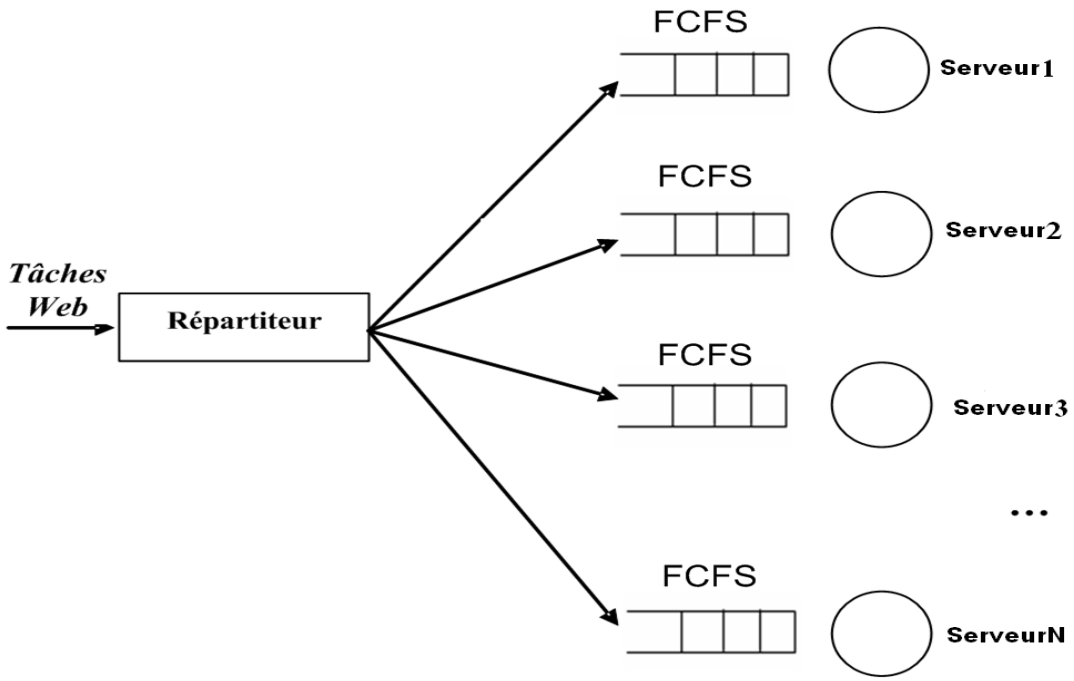

FIGURE 3.1 – Modèle de Serveurs Web Distribués.

Le modèle de serveurs distribués qui constitue la plate-forme de la technique proposée assume, comme montré dans la figure 3.1, N serveurs Web qui forme un cluster. Chaque serveur a des files d'attente avec une discipline FCFS (First Come First Served : Premier Arrivé Premier Servie) pour exécuter les différentes tâches Web qu'ils reçoivent. Les tâches Web arrivent en premier lieu au répartiteur suivant une loi de poisson de taux λ [55] et ce dernier les affecte aux différents serveurs qui participent dans le cluster selon l'algorithme de la technique qui se charge de cette tâche afin de réduire le temps de réponse et améliorer les performances du système.

3.3.2 Taille des tâches Web

Comme cela a été précisé dans le chapitre 2, la taille des tâches Web suit une distribution Heavy-Tailed. Dans la pratique, les serveurs web ont des fichiers de petite taille et d'autres de grande taille, d'où la distribution la plus convenable pour représenter

ces fichiers dans les serveurs web est la distribution Bounded Pareto B (α, k,p) [4] . Cette distribution est définie par sa fonction densité qui a la forme :

$$\text{f(t)}=\frac{\alpha k^{\alpha}}{1-(\frac{k}{p})^{\alpha}} t^{-\alpha-1}; k \leq t \leq p$$

Où :

α : représente la variation de taille des fichiers Web ;

k : la plus petite taille observable de fichiers Web ;

p : la plus grande taille observable de fichiers Web.

3.3.3 Connaissance de la taille des tâches Web

La taille des tâches Web est assumée, dans cette partie, connue à priori. Cette considération est réalisable par le fait que le flot des requêtes vers les sites web est dominé par les requêtes statiques, [56]. La confirmation de ces études a été révélée par les journaux d'événement pour les serveurs Proxy en 2004 où 67%-73% des requêtes sont pour le contenu statique [57]. Cette propriété permet au répartiteur d'inspecter facilement le temps de service de la tâche Web (i.e. la taille) comme c'est le cas dans [43].

Une autre solution permettant d'estimer la taille de la tâche Web est basée sur une caractéristique du protocole HTTP qui permet d'envoyer une requête "GET" initiale de petite taille connue sous le nom " HEAD Request " ; la réponse pour cette requête contient la taille du fichier concerné et le répartiteur peut facilement la récupérer ou l'extraire.

Les auteurs ont trouvé, dans la littérature un autre mécanisme pour estimer la taille de la tâche Web en se basant sur le protocole TCP. L'idée principale est que le répartiteur essaye de terminer le processus de connexion avec le client d'abord avant de décider lequel des serveurs sera la cible pour l'exécution de la requête. Cette technique permet au répartiteur d'examiner la requête et inspecter ainsi la taille de la tâche Web.

3.4 Algorithme

Comme montré dans la figure 3.1, notre modèle est composé d'un répartiteur qui assure la distribution des tâches Web entre les différents serveurs du système. Avec N serveurs Web, chaque serveur ou serveur a une ou plusieures files d'attente associées aux différentes tâches Web qui arrivent au système. Ces files d'attente sont séparées selon la taille de la tâche Web : des files d'attente (S : pour "Small") pour les tâches Web de petite et des files d'attente (L : pour "Large") conçu pour les tâches Web de grande taille ou plus précisément les fragments de ces dernières. Pour les tâches Web de petite taille, chaque serveur accepte seulement les tâches Web dont la taille fait partie de l'intervalle des tailles correspondant. Cet intervalle est conçu d'une façon à ce que la charge totale des tâches Web de petite taille soit diviser équitablement sur les M serveurs où $M = \frac{N}{2}$. Spécifiquement, si on a f(t) la fonction densité,k la taille minimale des tâches Web, les bornes t_1, t_2 des intervalles des tâches Web de petite taille peut être définit comme suit :

$$ChargeAffecteePour(M)Serveurs = \int_k^{t_1} tf(t)d_t = \int_{t_1}^{t_2} tf(t)d_t = \frac{\int_k^{t_2} tf(t)d_t}{2}$$

Dans ce cas, les tâches Web de petite taille seront exécuter sur tous les serveurs tant qu'il n'y a pas de tâches Web de grande taille afin d'assurer une utilisation maximale des ressources des serveurs disponibles.

Lorsqu'une tâche Web de grande taille arrive, le répartiteur la divise en fragments et affecte ces fragments aux (N - h) serveurs (où h est le nombre de serveurs qui participent à satisfaire l'exécution des tâches Web de petite taille dans le cas où il y a une tâche Web de grande taille) pour les exécuter en les mettant dans la file d'attente désignée (L). Les fragments sont des nombres d'octets dans le fichier qui est désigné par la requête, ces nombres d'octets peuvent être calculés en utilisant une caractéristique du protocole HTTP "byte-range header". Cette caractéristique permet de spécifier la taille des octets à récupérer comme paramètre dans la requête HTTP, par exemple : les octets d'un fichier de 100 à 200 sont un fragment de 100 Bytes. L'idée principale pour le traitement des tâches Web de grande taille est d'assurer une exécution parallèle des fragments afin que le temps d'exécution des fragments soit le même sur tous les serveurs (i.e. N -h), d'où l'avantage que les métriques de performance pour une tâche Web de grande taille convergent vers ceux d'un des fragments en choisissant celui qui a le minimum de temps d'attente, le minimum de temps de séjour,...,etc. Si la grande taille de la tâche Web est T_L, la taille d'un fragment F_j qui sera affecté au serveur j est :

$$F_j = \frac{T_l + \sum_{k=h+1}^{N} ChargeRestante_k}{N-h} - ChargeRestante_j$$

Où

$$\sum_{k=h+1}^{N} ChargeRestante_k$$

est la somme totale des charges restantes de tâchesWeb non encore traitées pour les N - h serveurs qui participent au traitement des tâches Web de grande taille, et $ChargeRestante_j$ la charge restante de tâche Web du serveur j. Par exemple, si la taille de la tâche Web est de 10000, avec N - h = 2 serveurs, la taille du fragment pour le premier serveur et le deuxième, si la charge restante est de 2000 et 4000 respectivement, est de $\frac{10000+2000+4000}{2}$ - 2000 = 6000 et $\frac{10000+2000+4000}{2}$ - 4000 = 4000 respectivement.

Dans ce cas, les tâches Web de petite taille seront affectées aux h serveurs restants jusqu'à la terminaison de l'exécution des fragments.

Les étapes de l'algorithme de la technique SWLT (Split Large Web Task) sont récapitulées comme suit :

1 lorsque les tâches Web arrivent au répartiteur, ce dernier les inspectes en récupérant leurs taille .

2 si la tâche Web est de petite taille, le répartiteur l'affecte aux serveurs en se basant sur la taille pour le choix du serveur parmi les serveurs qui participent au traitement des tâches Web de petite taille. Soient les serveurs (2i + 1, i = 0, 1, 2, . . .) qui ont l'intervalle de taille [k, x_1] et les serveurs (2i, i = 1, 2, . . .) qui ont l'intervalle de taille de]x_1, x_2], l'affectation des différentes tâches Web se fait par taille pour les deux types de serveurs (i.e. (2i + 1, i = 0, 1, 2, . . .) et (2i, i = 1, 2, . . .)) et en utilisant le temps d'attente comme critère d'affectation pour les serveurs qui ont le même intervalle de taille. Dans le cas où il y a apparition d'une tâche Web de grande taille, le répartiteur affecte ceux de petite taille aux h serveurs (où h est le nombre de serveurs qui participent à satisfaire l'exécution des tâches Web de petite taille) jusqu'à la fin du traitement des tâches Web de grande taille .

3 si la tâche Web est de grande taille, le répartiteur divise cette tâche Web en fragments en utilisant la caractéristique du protocole HTTP : " byte-ranges ", à condition que ces fragments soient lancés d'une façon parallèle et terminent l'exécution en même temps. Cette condition conduit à définir la taille du fragment qui doit être calculée

comme suit :

$$F_j = \frac{T_l + \sum_{k=h+1}^{N} ChargeRestante_k}{N-h} - ChargeRestante_j$$

Où F_j représente la taille du fragment pour un serveur j, T_L la grande taille de la tâche Web,

$$\sum_{k=h+1}^{N} ChargeRestante_k$$

est la somme totale des charges restantes de tâches Web non encore traitées des N - h serveurs qui participent au traitement des tâches Web de grande taille et $ChargeRestante_j$ la charge restante des tâches Web du serveur j. Après quoi, le répartiteur affecte les fragments et, lorsque les serveurs terminent l'exécution, les résultats des fragments (i.e. les parties du fichier sollicité) seront regroupés pour former le fichier originale.

4 lorsque les serveurs terminent l'exécution des fragments de la tâcheWeb de grande taille et, s'il n'y pas de tâche Web de grande taille dans la file, le répartiteur sollicite tous les serveurs pour le traitement des tâchesWeb de petite taille jusqu'à l'arrivée d'une tâche Web de grande taille.

3.4.1 Caractéristiques de l'algorithme

L'analyse de l'algorithme de la nouvelle technique SWLT permet de tirer quelques caractéristiques innovantes par rapport aux techniques existantes telles que :

- *Réduire la variation des tâches Web* : cela est dû à l'utilisation des files d'attente séparées, ce qui a comme avantage d'empêcher que les tâches Web de petite taille soient dans la même file d'attente que celles de grande taille.

- *L'affectation optimale et l'exécution parallèle des fragments des tâches Web de grande taille* : l'exécution parallèle des fragments d'une tâche Web de grande taille converge les métriques de performance de cette dernière vers ceux du fragment qui a les valeurs optimales au sens du temps d'attente, temps séjour,... etc, ce qui a comme but d'améliorer au maximum les métriques de performance des tâches Web de grande taille, qui ont été toujours le problème des techniques déjà proposées pour

les systèmes de serveurs web distribués.

- *L'adéquation aux systèmes de serveurs Web multi-services* : le problème des serveursWeb multi-services est qu'ils proposent différents types de service où chaque type suit la distribution Heavy-Tailed. Le partitionnment des tâches Web de grande taille couvre la variation de tâches Web et permet de traiter les différents types de service de la même façon.

3.5 La modélisation de la technique SWLT

SWLT est une nouvelle technique parmi les autres, rédigée au paravent, qui nous permet de nous donner une politique, assez performante, afin d'assigner des tâches web aux serveurs web. Le but de modéliser cette technique SWLT est d'étudier les problèmes rencontrés dans l'affectation des tâches web cités dans la section 3.2.

3.5.1 Le modéle de la technique SWLT

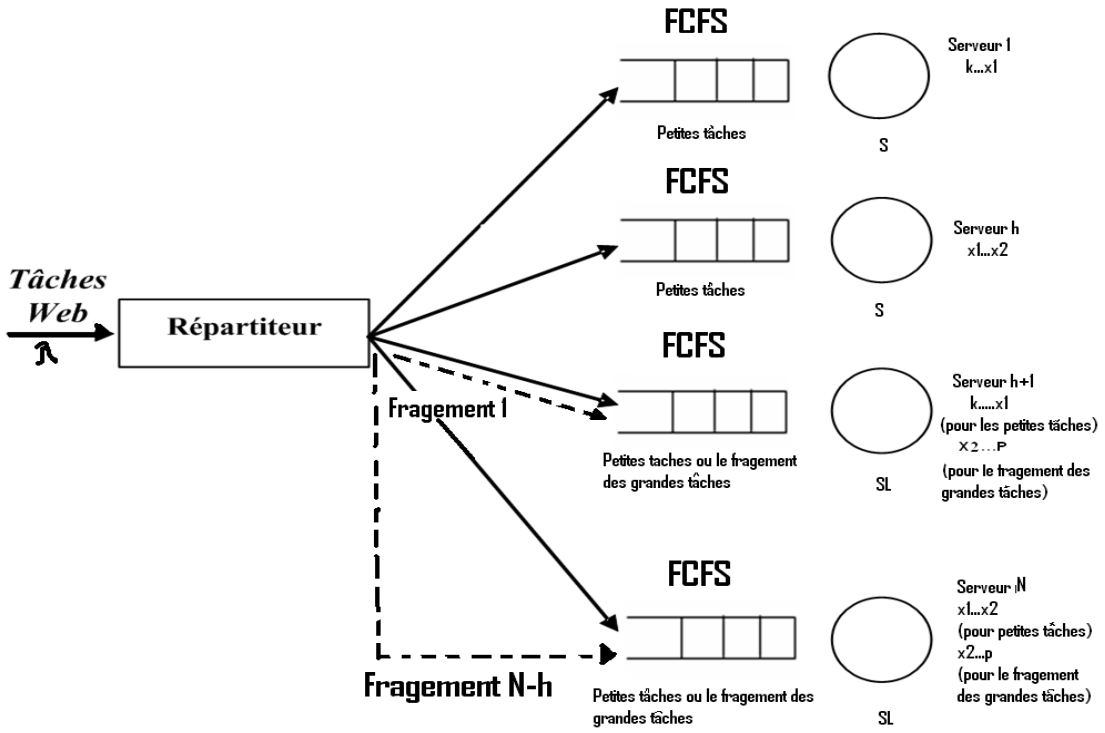

FIGURE 3.2 – Modèle de Serveurs Web Distribués utilisant la technique SWLT.

3.5.2 La modélisation de la technique SWLT par le système de file d'attente de M/Bp/1 FCFS

La figure 3.2, se modélise, par une file d'attente de type M/Bp/1 FCFS, puisque le systèmes de serveurs web distribués de cette technique, possède des files d'attentes séparées,tel que :

M :représente le temps des arrivés des clients web (i.e les tâtches) de même paramètre de taux λ qui suit une loi de poisson P(λ).

Bp :représente la taille des tâches web arrivées afin d'être servie par le serveur web au niveau du service qui suit la loi de Bounded Pareto B(p,k,α).

1 :représente le nombre de serveur web au niveau du service, qui est 1.

La taille de la file d'attente, òu les tâches web arrivent et s'installent à son niveau et patientent afin d'être servie par le serveur web est infinie.

La population source des arrivées des tâches web est infinie.

Le temps d'attente des tâches web arrivées au niveau du répartiteur ou le dispatcher(en anglais) afin qu'elles soient traitées par lui, est négligeable.

Le temps de service du dispatcher par apport aux tâches traitées est négligeable.

La discipline du service est FCFS (le premier arrivé, premier servie).

Soit $\{x(t), t \geq 0\}$, $x(t)$:v.a qui représente le nombre de tâches web dans le système de serveur web distribué à l'instant t.

À cet effet, nous considérons le processus x(t) aux instant, t_1, t_2, t_3,..., t_n òu des tâches web, terminent leur service et quittent, le système de serveur web distribué définit ainsi un processus stochastique, à temps discret : x_n=x(t_n), n=1,2,..., òu t_n, est l'instant du départ de la nième tâches web. On considère le nombre A_n de tâches web, qui entrent dans le système de serveur web distribué pendant que la nième tâche web est servie, les variables A_n sont indépendant, leur distribution est : $P(A_n = w) = a_w = \int_{w=0}^{\infty} \exp^{-\lambda t} \frac{(\lambda t)^w}{w!} b(t)dt$, qui représente le taux d'arrivé des clients web mutipliés par leur tailles .

tel que, $b(t)$ représente la fonction de densité de la loi générale qui est pour ce cas : la loi de Bounded pareto, qui est aussi la fonction de densité de la taille du client web, tel que : $b(t) = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-\alpha-1}$, et w, représente la valeur du nombre de tâches web, qui entrent dans le système de serveur web distribué donc, on a :

$$P(A_n = w) = a_w = \int_{w=0}^{\infty} \exp^{-\lambda t} \frac{(\lambda t)^w}{w!} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-\alpha-1} dt.$$

$$P(A_n = w) = a_w = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha) w!} \int_{w=0}^{\infty} \exp^{-\lambda t} (\lambda t)^w t^{-\alpha-1} dt.$$

$$P(A_n = w) = a_w = \frac{\alpha k^\alpha \lambda^w}{(1-(\frac{k}{p})^\alpha) w!} \int_{w=0}^{\infty} \exp^{-\lambda t} t^{(w-\alpha)-1} dt, \text{ d'où on pose } F(t) = t^{(w-\alpha)-1}$$

$$P(A_n = w) = a_w = \frac{\alpha k^\alpha \lambda^w}{(1-(\frac{k}{p})^\alpha) w!} \int_{w=0}^{\infty} \exp^{-\lambda t} F(t)dt$$

$$P(A_n = w) = a_w = \frac{\alpha k^\alpha \lambda^w \overline{b}(\lambda)}{(1-(\frac{k}{p})^\alpha) w!} \text{ et } a_w \geq 0,\ w \geq 0,\ \alpha > 0,\ \lambda \geq 0,\ k \leq t \leq p\ ,\overline{b}(\lambda) :$$

est la fonction de la transformée de la place du temps de services t.

Alors $X_{n+1} = X_n - 1 + A_{n+1} si X_n \geq 1$, et $X_{n+1} = A_{n+1} si X_n = 0$, n=1,2..), relation qui s'écrit également $X_{n+1} = X_n - \sigma_n + A_{n+1}$, avec $\sigma_n = 1 si X_n > 0$ et $\sigma_n = 0 si X_n = 0$. Telque

X_{n+1} ne dépend donc que de X_n, et A_{n+1} et non pas des valeurs prises par $X_{n-1}, X_{n-2}, ...$ La suite de variables aléatoires $\{X_n : n \geq 1\}$, s'appelle chaine de Markov induite du processus $\{X_t : n \geq 1\}$. Ses probabilités de transition $P_{ij} = P(X_{n+1} = j/X_n = i)$ se calculent par :
$P_{0j} = a_j si ij \geq 0$
$P_{ij} = a_{j-i+1} si 1 \leq i \leq j+1$
$P_{ij} = 0$ sinons.
La matrice des probabilé de transition prend la forme :

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3... \\ a_0 & a_1 & a_2 & a_3... \\ 0 & a_0 & a_1 & a_2... \\ 0 & 0 & a_0 & a_1... \\ . & . & . & ..... \end{pmatrix}$$

Le comportement des tâches web envers leur arrivée à la file d'attente de type M/Bp/1 FCFS, nous fait comprendre qu'on peut se déplacer de n'importe qu'elle état ver n'importe qu'elle état d'après le comportement de la matrice des probabilités transitoires P, associé à la file d'attente M/Bp/1 donc, $\{x(t), t \geq 0\}$ est une chaine de Marcove, irréductible qui converge ver une distribution limite $\rho = \lambda E\{t\} < 1$, qui est l'instant du trafic ou bien, le coefficient d'utilisateur, c'est aussi le nombre moyen d'arrivée des tâches web par duré moyenne de service.

3.5.3 Distribution stationnaire de la file d'attente M/Bp/1 FCFS pour le système de serveur web distribué qui utilise la technique SWLT

Supposons, maintenant que $\rho < 1$,et soit $\Pi = (\Pi_0, \Pi_1, \Pi_2, ...)$,la distribution stationnaire de la chaine de Marcove induite, il ne sera pas généralement possible de trouver la distribution Π, elle-même, mais nous pouvons calculer la fonction génératrice qui est :
$\Pi(z) = \frac{(1-\rho)A(z)(z-1)}{z-A(z)}$, où $A(z) = \sum_{w=0}^{\infty} a_w z^w = \overline{b}(\lambda - \lambda z)$,qui la transformé de la place de la densité de probabilité du temps de service t, qui suit la loi de Bounded pareto(traitement des tâches web par le serveur web).
Le résultat de la distribution stationnaire $\Pi(z)$ est calculé de la façon suivante :
supposons que $\rho = \lambda E\{t\} < 1$ et soit $\Pi = (\Pi_1, \Pi_2, ..., \Pi_n), \sum_i^n \Pi_i = 1$, on calcule la fonction génératrice $\Pi(z)$ de la façon suivant telque on a $\Pi = \Pi P$,òu $\Pi_j = \sum_{i=1}^{\infty} \Pi_i P_{ij}, j = 0, 1, 2....$
$\Pi_j = a_j \Pi_0 + \sum_{i=1}^{j+1} \Pi_i a_{j-i+1}$.
$\Pi_j = a_j \Pi_0 + \sum_{i=0}^{j+1} \Pi_i a_{j-i+1} - a_{j+1}\Pi_0$.
Si on multiple cette équation par z^j, et si l'on somme par j on aura que :
$\sum_{j=0}^{\infty} \Pi_j z^j = \Pi_0 \sum_{j=0}^{\infty} a_j z^j + \frac{1}{z} \sum_{j=0}^{\infty} c_{j+1} z^{j+1} - \frac{\Pi_0}{2} \sum_{j=0}^{\infty} a_{j+1} z^{j+1} (*)$
Telque $c_{j+1} = \sum_{i=0}^{j+1} a_{j-i+1} \Pi_i$, qui représente le produit de convolution des distribution A et Π, et on introduisant, la fonction génératrice , on trouve que :
$\Pi(z) = \sum_{i=0}^{\infty} \Pi_i z^i$. $A(z) = \sum_{i=0}^{\infty} a_i z^i$. $C(z) = \sum_{j=0}^{\infty} = a_j z^j = \Pi(z) A(z)$, on obtient :
$\Pi(z) = \Pi_0 A(z) + \frac{1}{z}[C(z) - c_0] - \frac{\Pi_0}{z}[A(z) - a_0]$

$\Pi(z) = \frac{\Pi_0 A(z)(z-1)}{z-A(z)}$.
De la relation :
$X_{n+1} = X_n - \sigma_n + A_{n+1}$.
On titre, puisqu'on se trouve en régime stationnaire,$E(A_{n+1}) = E(\sigma_n) = P(\sigma_n > 0) = P(X_n) = 1 - P(X_n = 0)$ d'où $\Pi_0 = 1 - \rho$. Ce qui conserne le calcule de A(z), il est calculé de la façon suivante :
$A(z) = \sum_{w=0}^{\infty} a_w z^w$.
$A(z) = \sum_{w=0}^{\infty} z^w \int_0^{\infty} \exp^{-\lambda t} \frac{(\lambda t)^w}{w!} b(t)dt$.
$A(z) = \int_0^{\infty} \exp^{-\lambda t} \sum_{w=0}^{\infty} \frac{(\lambda zt)^w}{w!} b(t)dt$.
$A(z) = \int_0^{\infty} \exp^{-\lambda t} \exp^{-\lambda zt} b(t)dt$.
$A(z) = \int_0^{\infty} \exp^{-(\lambda z - \lambda)t} b(t)dt$.
$A(z) = \overline{b}(\lambda - \lambda z)$.

3.5.4 Nombre moyen du nombre de tâches web entrant dans le système de serveur web distribué durant le service de la nième tâche web

Le nombre moyen du nombre de tâches web entrant dans le système de serveur web distribué durant le service de la nième tâche web, est donné par la formule suivante :

$E(A_n) = \sum_{w=0}^{\infty} wP(A_n = w)$;

$E(A_n) = \int_{w=0}^{\infty} \sum_{w=0}^{\infty} \exp^{-\lambda t} \frac{(\lambda t)^w}{w!} b(t)dt$;

$E(A_n) = \int_{w=0}^{\infty} \sum_{w=1}^{\infty} \exp^{-\lambda t} \frac{(\lambda t)^{w-1}}{(w-1)!} (\lambda t) b(t)dt$;

$E(A_n) = \int_{w=0}^{\infty} \exp^{-\lambda t} (\lambda t) b(t)dt$;

Car $\sum_{w=1}^{\infty} \exp^{-\lambda t} \frac{(\lambda t)^{w-1}}{(w-1)!} = 1$, d'aprés le résultat de la formule du binôme du newton.

$E(A_n) = \frac{\alpha k^{\alpha}}{(1-(\frac{k}{p})^{\alpha})} \int_{w=0}^{\infty} \exp^{-\lambda t} t^{(-\alpha)-1} (\lambda t)dt$;

$E(A_n) = \frac{\lambda \alpha k^{\alpha}}{(1-(\frac{k}{p})^{\alpha})} \int_{w=0}^{\infty} \exp^{-\lambda t} t^{(-\alpha)} dt$, on pose $F(t) = t^{(-\alpha)}$, d'où :

$E(A_n) = \frac{\alpha k^{\alpha}}{(1-(\frac{k}{p})^{\alpha})} \int_{w=0}^{\infty} \exp^{-\lambda t} F(t)dt$;

$E(A_n) = \frac{\alpha k^{\alpha}}{(1-(\frac{k}{p})^{\alpha})} \overline{b}(\lambda)$.

3.5.5 Nombre moyen de tâches web dans le système de serveur web distribué

Soit x :nombre de tâches dans le système de serveur web distribué.

Soit x_q :nombre de tâches dans la file d'attente du système de serveur web distribué.

Soit $E\{x\}$:nombre moyen de tâches dans le système de serveur web distribué.
Tel que leur résultas est donnés comme suite :
Pour calculer le nombre moyen de tâche web dans le système de file d'attente en régime stationnaire, on pourait s'appyer sur la relation d : $E(x) = \lim \Pi^{'}(z), quandz \longrightarrow 1$.
Ce calcul s'avérant plutôt pénible, nous exposons ci-aprés une méthode plus élémentaire.
Reprenons la relations fondamentale : $X_{n+1} = X_n - \sigma_n + A_{n+1}$. En l'élevant au carré et observant que : $\sigma_n^2 = \sigma_n$, et $\sigma_n X_n = X_n$, on trouve :$X_{n+1}^2 = X_n^2 + \sigma_n + A_{n+1}^2 - 2X_n - 2\sigma_n A_{n+1}^2 + 2X_n^2 A_{n+1}$.
Pour passer à l'espérance mathématique, on rappelle que :

- A_{n+1} est indépendante de X_n et de σ_n.
- $E(X_{n+1}^2) = E(X_n^2)$.
- $E(A_n) = E(\sigma_n) = \rho$.

aprés le calcul on obtient :

$E\{X_n\} = \frac{\rho - 2\rho^2 + E(A_n^2)}{2(1-\rho)}$, on réalité $E\{X_n\} = E\{x\}$, car le nombre de tâche web à l'instant du départ de la n'ième tâche dans le système de file d'attente=nombre de tâche web dans le système de file d'attente . tel que : $E(A_n^2) = \rho + \lambda^2 Var(t) + \rho^2$.
en remplaçant la valeur de $E(A_n^2)$, dans $E\{x\}$, on trouve :$E\{x\} = \rho + \frac{\rho^2 + \lambda^2 var(t)}{2(1-\rho)}$ Telque $Var(t)$:est la varriance du temps de service de t, ou bien la moyenne des moyenne.
d'óu :$Var(t) = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}[\frac{(p^{2-\alpha} - k^{2-\alpha})}{(2-\alpha)} - [\frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}]\frac{(p^{1-\alpha} - k^{1-\alpha})}{(1-\alpha)^2}]$, ce résultat est trouvé par $Var(t) = E\{t^2\} - (E\{t\})^2$, telque $E\{t\} = \int_k^p t \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-\alpha-1} dt \Longrightarrow E\{t\} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_k^p t^{-\alpha} dt \Longrightarrow E\{t\} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha * (1-\alpha)} [t^{-\alpha+1}]_k^p \Longrightarrow E\{t\} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha * (1-\alpha)} (p^{1-\alpha} - k^{1-\alpha})$ et $E\{t^2\} = \int_k^p t^2 \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-\alpha-1} dt \Longrightarrow E\{t^2\} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_k^p t^{-\alpha+1} dt \Longrightarrow E\{t^2\} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha * (2-\alpha)} [t^{-\alpha+2}]_k^p \Longrightarrow E\{t^2\} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha * (2-\alpha)} (p^{2-\alpha} - k^{2-\alpha})$.

3.5.6 Nombre moyen de tâches web dans la file d'attente du système de serveur web distribué

Soit $E\{x_q\}$:nombre moyen de tâches dans la file d'attente du système de serveur web distribué.
D'après la formule de little, on a : $E\{x\} = E\{x_q\} + \rho \Longrightarrow E\{x_q\} = E\{x\} - \rho$, qui résulte : $E\{x_q\} = \frac{\rho^2 + \lambda^2 Var(t)}{2(1-\rho)}$, telque $\rho = \lambda E\{t\}$, en remplaçant la valeur de ρ, dans $E\{x_q\}$, on

trouve :$E\{x_q\} = \frac{(\lambda E\{t\})^2+\lambda^2 Var(t)}{2(1-\lambda E\{t\})} \Longrightarrow E\{x_q\} = \frac{\lambda^2 E\{x\}^2+\lambda^2 var(t)}{2(1-\lambda E\{x\})}$
$\Longrightarrow E\{x_q\} = \frac{\lambda^2 E\{t\}^2+\lambda^2(E\{t^2\}-E\{t\}^2)}{2(1-\lambda E\{t\})} \Longrightarrow E\{x_q\} = \frac{\lambda^2 E\{t^2\}}{2(1-\lambda E\{t\})}$.

3.5.7 Temps d'attente moyen d'une tâches web dans le système de serveur web distribué

Soit w_q :Temps d'attente moyen d'une tâches web dans le système de serveur web distribué.
D'après la formule de little, on a : $E\{x_q\} = w_q * \lambda_e \Longrightarrow w_q = \frac{E\{x_q\}}{\lambda_e} \Longrightarrow w_q = \frac{\lambda E\{t^2\}}{2(1-\lambda E\{t\})}$, telque λ_e :le taux d'entrée des tâches web dans le système, et $\lambda_e = \lambda$, car La taille de la file d'attente, òu les tâches web arrivent et s'installent à son niveau et patientent afin d'être servie par le serveur web est infinie .

3.5.8 Temps de séjour moyen d'une tâches web dans le système de serveur web distribué

Soit w_1 :Temps de séjour moyen d'une tâches web dans le système de serveur web distribué.
D'après le Théorème1 de la tachnique SITA-E[4], on a : $w_1 = w_q\lambda_e \Longrightarrow w_1 = \frac{\lambda^2 E\{t^2\}}{2(1-\lambda E\{t\})}$.

3.5.9 Temps de SlowDown d'une tâche web

Soit S :représente la v.a du temps de SlowDown de la tâche web dans le système M/Bp/1FCFS Le temps de SlowDown moyen d'une tâche web est donnée par la formule suivante :
$E\{S\} = \frac{w_q}{E\{t\}} \Longrightarrow E\{S\} = \frac{\frac{\lambda E\{t^2\}}{2(1-\lambda E\{t\})}}{E\{t\}} \Longrightarrow E\{S\} = \frac{\lambda E\{t^2\}}{2(1-\lambda E\{t\})} * E\{t^{-1}\}$ telque :$E\{t^{-1}\} = \int \frac{1}{t} b(t) \Longrightarrow E\{t^{-1}\} = \int \frac{1}{t} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-\alpha-1} \Longrightarrow E\{t^{-1}\} = \int t^{-\alpha-2} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \Longrightarrow E\{t^{-1}\} = \frac{t^{-\alpha-1}}{(-\alpha-1)} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \Longrightarrow E\{S\} = \frac{\lambda E\{t^2\}}{2(1-\lambda E\{t\})} * \frac{t^{-\alpha-1}}{(-\alpha-1)} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}$.
Jusqu'a ici nous avons modélisé que le système de file d'attente M/Bp/1FCFS associé à la technique SWLT, ce qui ce suivra sera sur la généralisation de la modélisation de la technique SWLT sur N serveurs.

3.6 La généralisation de la modélisation de la technique SWLT par rapport à N serveurs

La politique de la technique SWLT, d'assigement des tâches web peut, être appliquée pour une certaine, distribution de la taille de la tâche web suit une distribution de la loi de bounded parto, $B(T_i, T_{i-1}, \alpha)$, où on le simbolise ,par $t_i \curvearrowright B(T_i, T_{i-1}, \alpha)$,d'où sont moment ou son esperence est $E\{t_i\} = \int_{T_{i-1}}^{T_i} tf(t)dt$, et dans cette technique nous avons deux cas :
le cas ou il existe des tâches , de petites tailles et de grandes tailles, les tâches de petites tailles, seront éxécutées par h servers et le reste des tâches grandes sera éxécuté par (N-h), serveurs restant tout, en supposons qu'on a N serveurs dans le système, si on a l'arrivés des grandes tâches sinon on divisera le système sur $M = \frac{N}{2}$. Pour cella :
T_i :représente la plus grande taille distribuée pour le nième serveur.
T_{i-1} :représente la plus petite taille distribuée pour le nième-1 serveur.
Pour cela on a :$E\{t_i\} = \int_{T_0=k}^{T_1} tf(t)dt = \int_{T_1}^{T_2} tf(t)dt = \int_{k}^{T_2} \frac{tf(t)}{2}dt = \frac{E\{t\}_1}{2} = E\{t_i\}_2, i = 1....., N$,qui représente la charge totale des tâches Web de petite taille soit diviser équitablement sur les M serveurs où $M = \frac{N}{2}$. $i = 1...., N$,maintenant dans le cas où il existe des tâches de grandes tailles, alors on :
$E\{t_i\} = \int_{T_2}^{p} tf(t)dt = E\{t\}_2 = E\{t_i\}_{N-h}, i = h....., N-1$,qui représente la charge totale des tâches Web de grande taille soit diviser en fragments sur N-h serveurs restants, (où h est le nombre de serveurs qui participent à satisfaire l'exécution des tâches Web de petite taille dans le cas où il y a une tâche Web de grande taille) , pour $i = h...., N-1$.
Le calcule de $E\{t_i\}_2, i = 1....., N$, est trouvé de la façon suivante :

$E\{t_i\}_2 = \int_{T_{i-1}}^{T_i} tf(t)dt = \int_{T_{i-1}}^{T_i} t\frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-1-\alpha}dt.$
$E\{t_i\}_2 = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-\alpha}dt.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t^{-\alpha}dt.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [\frac{t^{1-\alpha}}{(1-\alpha)}]_{T_{i-1}}^{T_i}.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [\frac{t_i^{1-\alpha} - t_{i-1}^{1-\alpha}}{(1-\alpha)}]_{T_{i-1}}^{T_i}.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)} [t_i^{1-\alpha} - t_{i-1}^{1-\alpha}].....(*).$

Telque : $T_i = (\frac{2-i}{2}k^{1-\alpha} + \frac{i}{2}T_2^{1-\alpha})$, et $\alpha \neq 1$, pour $i = 1..., 2$, d'aprés le théoreme3 de la tachnique SITA-E[4], donc on remplace les, valeur T_i, et T_{i-1} déduite de T_i, dans $(*)$, on a pour $T_{i-1} = (\frac{2-i+1}{2}k^{1-\alpha} + \frac{i-1}{2}T_2^{1-\alpha}), i = 1..., 2.$

$E\{t_i\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)} [[(\frac{2-i}{2}k^{1-\alpha} + \frac{i}{h}T_2^{1-\alpha})^{\frac{1}{1-\alpha}}]^{1-\alpha} - [(\frac{2-i+1}{2}k^{1-\alpha} + \frac{i-1}{2}T_2^{1-\alpha})^{\frac{1}{1-\alpha}}]^{1-\alpha}].$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)} [\frac{T_2^{1-\alpha} - k^{1-\alpha}}{2}].$
$E\{t_i\}_2 = \frac{E\{t\}_1}{2}...(1).$

Le calcule de $E\{t_i\}_{N-h}, i = h.....,N-1$, est trouvé de la façon suivante :
$E\{t_i\}_N = \frac{E\{t\}_1}{2} + E\{t_i\}_{N-h}$

$E\{t_i\}_N = \frac{E\{t\}_1}{2} + F_i...(3)$,telque : $F_i = \frac{E\{t_i\}_{N-h}=\frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)}[\frac{p^{1-\alpha}-T_2^{1-\alpha}}{2}]+\sum_{i=h+1}^{N}\rho_i}{N-h} - \rho_i$,et $E\{t_i\}_{N-h}$ déduit de (1), F_i :représente la fragmentation des tâches vis à vis aux i serveur, qui représente la charge affectée aux serveurs vis à vis au système l'orsqu'il s'agit des tâches de grandes et petites tailles,pour i=h...,N-1,ρ_i : représente la charge restante de la tâche web du serveur i, et$\sum_{i=h+1}^{N}\rho_i$: représente la somme totale des charges restantes de tâches web non encore traitées pour les N-h serveurs qui participent au traitemant des tâches web de grandes taille, tous ça est valable pour tous ceux qui suivera comme performence dans la modélisation, et $\alpha \neq 1$. Tout ce qu'on nous venons de faire sur le calcule de $E\{t_i\}_N$, est pour $\alpha \neq 1$, maintenant si $\alpha = 1$ on a :$T_i = k(\frac{T_2}{k})^{\frac{i}{2}}$, pour i=1...,N, d'aprés le théoreme3[40], d'où on peut déduire : $T_{i-1} = k(\frac{T_2}{k})^{\frac{i-1}{2}}$ et $E\{t_i\}_h$, est calculé de la façon suivante :

$E\{t_i\}_2 = \int_{T_{i-1}}^{T_i} tf(t)dt = \int_{T_{i-1}}^{T_i} t\frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}t^{-1-\alpha}dt.$
$E\{t_i\}_2 = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}t^{-\alpha}dt.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}\int_{T_{i-1}}^{T_i} t^{-\alpha}dt.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}\int_{T_{i-1}}^{T_i} \frac{1}{t}dt.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}[\lg(t)]_{T_{i-1}}^{T_i}.$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}[\lg(T_i) - \lg(T_{i-1})].$
$E\{t_i\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}[\lg(k(\frac{T_2}{k})^{\frac{i}{2}}) - \lg(k(\frac{T_2}{k})^{\frac{i-1}{2}})].$
$E\{t_i\}_2 = \frac{\frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}[\lg(T_2)-\lg(k)]}{2}...(1).$
Et $E\{t_i\}_{N-h}$, est calculé de la façon suivante :

$T_i = k(\frac{T_2}{k})^{\frac{i}{N-h}}.$
$T_{i-1} = k(\frac{T_2}{k})^{\frac{i-1}{N-h}}.$
$E\{t_i\}_{N-h} = F_i$, telque :
$E\{t_i\}_N = E\{t_i\}_2 + F_i$qui représente la charge affecté aux serveurs vis à vis au système l'orsqu'il s'agit des tâches de grandes et petites tailles,pour $i = h..., N-1$, et $\alpha = 1$.
Telque $F_i = \frac{E\{t_i\}_{N-h}=E\{t_i\}_2=\frac{\frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}[\lg(p)-\lg(T_2)]}{2}+\sum_{i=h+1}^{N}\rho_i}{N-h} - \rho_i$,et $E\{t_i\}_{N-h}$ déduit de (1).
D'aprés le théoreme4 de la technique SITA-E[4] :
$p_i = \int_{T_{i-1}}^{T_i} \frac{f(t)}{2}dt$=la valeur de la taille de la tâche, observable pour chaque serveur i, pour$i = 1..., N$.

$p_i = \int_{T_{i-1}}^{T_i} \frac{f(t)}{2}dt = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{2(1-(\frac{k}{p})^\alpha)(1-\alpha)}t^{-1-\alpha}dt$
$p_i = \int_{T_{i-1}}^{T_i} \frac{f(t)}{2}dt = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)}[-t^{-\alpha}]_{T_{i-1}}^{T_i}.$
$p_i = \int_{T_{i-1}}^{T_i} \frac{f(t)}{2}dt = \frac{\alpha k^\alpha}{2(1-(\frac{k}{p})^\alpha)(1-\alpha)}[-t_i^{-\alpha} + t_{i-1}^{-\alpha}]$,d'où $T_{i-1} = (\frac{N-h-i+1}{h}k^{1-\alpha} + \frac{i-1}{N-h}T_2^{1-\alpha}) +$

$(\frac{i-1}{2}k^{1-\alpha}+\frac{2-i+1}{2}T_2^{1-\alpha})^{\frac{1}{1-\alpha}}$, $T_i=(\frac{i}{2}k^{1-\alpha}+\frac{i-1}{2}T_2^{1-\alpha})+(\frac{i}{N-h}k^{1-\alpha}+\frac{N-h-i}{N-h}T_2^{1-\alpha})^{\frac{1}{1-\alpha}}$. valable pour les serveurs qui traitent les petites tâches, et pour ceux qui traitent les grandes tâches, on aura, $\acute{p_i}=p_i=\int_{T_{i-1}}^{T_i}f(t)dt=\int_{T_{i-1}}^{T_i}\frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)}t^{-1-\alpha}dt$

$p_i=\int_{T_{i-1}}^{T_i}f(t)dt=\frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)}[-t^{-\alpha}]_{T_{i-1}}^{T_i}$.

$p_i=\int_{T_{i-1}}^{T_i}f(t)dt=\frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)}[-t_i^{-\alpha}+t_{i-1}^{-\alpha}]$,où $T_{i-1}=(\frac{N-h-i+1}{h}T_2^{1-\alpha}+\frac{i-1}{N-h}p^{1-\alpha})+(\frac{i-1}{2}T_2^{1-\alpha}+\frac{2-i+1}{2}p^{1-\alpha})^{\frac{1}{1-\alpha}}$, $T_i=(\frac{i}{2}T_2^{1-\alpha}+\frac{i-1}{2}p^{1-\alpha})+(\frac{i}{N-h}T_2^{1-\alpha}+\frac{N-h-i}{N-h}p^{1-\alpha})^{\frac{1}{1-\alpha}}$.

$\frac{\acute{p_i}+\sum_{i=h+1}^{N}\rho_i}{N-2}-\rho_i=F_i$,donc la plus grande tâche dans chaque serveur i est :$p_i^{''}=\acute{p_i}+F_i$ ça si $\alpha\neq 1$, sinon si $\alpha=1$, on aura pour :

$T_i=k(\frac{T_2}{k})^{\frac{i}{2}}+k(\frac{T_2}{k})^{\frac{i}{N-h}}$.

$T_{i-1}=k(\frac{T_2}{k})^{\frac{i-1}{2}}+k(\frac{T_2}{k})^{\frac{i-1}{N-h}}$.d'où

$p_i=\int_{T_{i-1}}^{T_i}\frac{f(t)}{2}dt=\frac{\alpha k^\alpha}{2(1-(\frac{k}{p})^\alpha)(1-\alpha)}[-(k(\frac{T_2}{k})^{\frac{i}{2}}+k(\frac{T_2}{k})^{\frac{i}{N-h}})^{-\alpha}+(k(\frac{T_2}{k})^{\frac{i-1}{2}}+k(\frac{T_2}{k})^{\frac{i-1}{N-h}})^{-\alpha}]$

qui représente les tâches les plus grandes pour pour les serveur i qui traitent les petites tâches, et pour ceux qui traitent les plus grandes tâches, on a :

$F_i=\frac{p_i^{'}=\frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(1-\alpha)}[-(T_2(\frac{p}{T_2})^{\frac{i}{2}}+k(\frac{p}{T_2})^{\frac{i}{N-h}})^{-\alpha}+(T_2(\frac{p}{T_2})^{\frac{i-1}{2}}+k(\frac{p}{T_2})^{\frac{i-1}{N-h}})^{-\alpha}]+\sum_{i=h+1}^{N}\rho_i}{N-h}-\rho_i$, maintenant ,pour avoir les plus grandes tâches pour chaque serveur i dans le système n, on a : $p_i^{''}=p_i+F_i$. D'aprés le théoreme3 de la technique SITA-E[4], on aura que le taux des arrivées des, tâche web, est λp_i, pour i serveur, d'aprés le théoreme1[4], pour le système de N serveurs reste α, et que la charge cité au paravant est calculé de la façon suivante $\rho=\lambda E\{t_i\}_N$, puisque nous somme des systèmes, alors $\rho_i=\lambda_i p_i E\{t_i\}_N$,telque $E\{t_i\}$, représente le temps de service moyen des taches web , dans chaque système de serveur i, possédant une file d'attente M/Bp/1 FCFS pour $i=h...,N-1$, et puique dans la technique, SWLT est basée sur le partage des tâches petites et grandes,on aura :

$\rho_i=\lambda_i p_i E\{t_i\}_N$

$\rho_i=\lambda_i p_i\int_{T_{i-1}}^{T_i}tf(t)dt$

$\rho_i=\lambda_i p_i(E\{t_i\}_{N-h}+E\{t_i\}_h)\Longrightarrow\sum_{i=1}^{N}\rho_i=\rho=\lambda E\{t_i\}_N$, pour le système.

D'aprés la modélisation faite pour un seule file d'attente de type M/Bp/1 FCFS, alors on a trouvé que le temps moyen d'attente de tâches web pour une file d'attente était :$w_q=\frac{\lambda E\{t^2\}}{2(1-\lambda E\{t\})}$, si on le généralise par apport au système, on trouve que $w_{q_i}=\frac{\lambda_i E\{t_i^2\}_N}{2(1-\lambda_i E\{t_i\}_N)}$ qui représente le temps moyen de tâches web d'attente pour chaque serveur i , et $E\{t_i^2\}_N=\int_{T_{i-1}}^{T_i}t^2f(t)dt$, pour le temps moyen d'attente de tâches web pour le système de N serveurs on a :

$w_{q_N}=\sum_{i=1}^{N}p_i w_{q_i}$, telque : w_{q_N}, représente le temps d'attente de tâches web pour le système de N serveurs, d'où on déduit le temps de séjour moyen de tâches web pour chaque serveur i, qui est :

$w_{1_i}=\lambda_i p_i w_{q_i}$, car le temps de séjour moyen de tâches web dans une file d'attente était $w_1=w_q\lambda$, pour le temps de séjour moyen de tâches web dans le système de N serveurs, on a :

$w_{1_N}=\sum_{i=1}^{N}p_i w_{1_i}$,telque : w_N, représente le temps de séjour de tâches web pour le système de N serveurs, pour le temps moyen de SlowDown de tâches web pour chaque serveur i,

on a que : $E\{S_i\} = w_i * E\{t_i^{-1}\}$, telque :$E\{t_i^{-1}\} = \int_{T_{i-1}}^{T_i} t^{-1} f(t)dt$, d'oùle temps moyen de SlowDown de tâches web pour le système de N serveurs est :$E\{S\} = \sum_{i=1}^{N} p_i E\{S_i\}$ telque :

$E\{t_i^2\}_2 = \int_{T_{i-1}}^{T_i} t^2 f(t)dt = \int_{T_{i-1}}^{T_i} t^2 \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-1-\alpha} dt.$

$E\{t_i^2\}_2 = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{1-\alpha} dt.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t^{1-\alpha} dt.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [\frac{t^{2-\alpha}}{(2-\alpha)}]_{T_{i-1}}^{T_i}.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [\frac{t_i^{2-\alpha} - t_{i-1}^{2-\alpha}}{(2-\alpha)}]_{T_{i-1}}^{T_i}.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(2-\alpha)} [t_i^{2-\alpha} - t_{i-1}^{2-\alpha}].....(*).$

Telque : $T_i = (\frac{2-i}{2} k^{1-\alpha} + \frac{i}{2} T_2^{1-\alpha})$, et $\alpha \neq 1$, pour $i = 1..., N$, d'aprés le théoreme3 de la technique SITA-E[4], donc on remplace les, valeur T_i, et T_{i-1} déduite de T_i, dans $(*)$, on a pour $T_{i-1} = (\frac{2-i+1}{2} k^{1-\alpha} + \frac{i-1}{2} T_2^{1-\alpha}), i = 1..., N.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(2-\alpha)} [[(\frac{h-i}{2} k^{1-\alpha} + \frac{i}{2} T_2^{1-\alpha})^{\frac{1}{1-\alpha}}]^{2-\alpha} - [(\frac{2-i+1}{2} k^{1-\alpha} + \frac{i-1}{2} T_2^{1-\alpha})^{\frac{1}{1-\alpha}}]^{2-\alpha}].$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(2-\alpha)} [\frac{T_2^{1-\alpha} - k^{1-\alpha}}{2}].$

$E\{t_i^2\}_2 = \frac{E\{t^2\}_1}{2} ...(1.1).$

Le calcule de $E\{t_i^2\}_{N-h}, i = h....., N-1$, est trouvé de la façon suivante :

$E\{t_i^2\}_{N-h} = F_i$, $E\{t_i^2\}_N = E\{t_i^2\}_{N-h} + F_i, ...(3.1)$,et

$F_i = \frac{E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(2-\alpha)} [\frac{p^{1-\alpha} - T_2^{1-\alpha}}{N-h}] + \sum_{i=h+1}^{N} \rho_i}{N-h} - \rho_i$,$E\{t_i^2\}_{N-h}$ déduit par (1.1), qui représente la charge affectée aux serveurs vis à vis au système l'orsqu'il s'agit des tâches de grandes et petites tailles, pour i=h...,N-1, et $\alpha \neq 1$. Tout ce qu'on nous venons de faire sur le calcule de $E\{t_i^2\}_N$, est pour $\alpha \neq 1$, maintenant si $\alpha = 1$ on a :$T_i = k(\frac{T_2}{k})^{\frac{i}{2}}$, pour i=1...,N, d'aprés le théoreme3de la technique SITA-E[4], d'où on peut déduire : $T_{i-1} = k(\frac{T_2}{k})^{\frac{i-1}{2}}$ et $E\{t_i\}_2$, est calculé de la façon suivante :

$E\{t_i^2\}_2 = \int_{T_{i-1}}^{T_i} t^2 f(t)dt = \int_{T_{i-1}}^{T_i} t^2 \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-1-\alpha} dt.$

$E\{t_i^2\}_2 = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{2-\alpha} dt.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t^{2-\alpha} dt.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t dt.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} \frac{t^2}{2} dt.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)2} [t^2]_{T_{i-1}}^{T_i}.$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [(T_i)^2 - (T_{i-1})^2].$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [(k(\frac{T_2}{k})^{\frac{2i}{2}}) - (k(\frac{T_2}{k})^{\frac{2i-2}{2}})].$

$E\{t_i^2\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} (k(\frac{T_2}{k})^{\frac{2i}{2}})[1 - (\frac{k}{T_2})^{\frac{2}{2}})]...(1.1)$. Et $E\{t_i^2\}_{N-h}$, est calculé de la façon suivante :$E\{t_i^2\}_{N-h} = F_i$,donc $E\{t_i^2\}_N = E\{t_i^2\}_2 + F_i$,...,(3.1),telque :

$F_i = \frac{E\{t_i\}_{N-h}=\frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}(T_2(\frac{p}{T_2})^{\frac{2i}{h}})[1-(\frac{T_2}{p})^{\frac{2}{h}})]+\sum_{i=h+1}^{N}\rho_i}{N-h} - \rho_i$, $E\{t_i\}_{N-h}$ déduit de (1.1), qui représente la charge affectée aux serveurs vis à vis au système l'orsqu'il s'agit des tâches de grandes et petites tailles,pour $i = h..., N-1$, et $\alpha = 1$.

$E\{t_i^{-1}\}_2 = \int_{T_{i-1}}^{T_i} \frac{1}{t} f(t)dt = \int_{T_{i-1}}^{T_i} \frac{1}{t} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-1-\alpha} dt.$

$E\{t_i^{-1}\}_2 = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-2-\alpha} dt.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t^{-2-\alpha} dt.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [\frac{t^{-1-\alpha}}{(-1-\alpha)}]_{T_{i-1}}^{T_i}.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [\frac{t_i^{-1-\alpha}-t_{i-1}^{-1-\alpha}}{(-1-\alpha)}]_{T_{i-1}}^{T_i}.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(-1-\alpha)} [t_i^{-1-\alpha} - t_{i-1}^{-1-\alpha}].....(*).$

Telque : $T_i = (\frac{h-i}{2} k^{1-\alpha} + \frac{i}{2} T_2^{1-\alpha})$, et $\alpha \neq 1$, pour $i = 1..., N$, d'aprés le théoreme3de la technique SITA-E[4], donc on remplace les, valeur T_i, et T_{i-1} déduite de T_i, dans $(*)$, on a pour $T_{i-1} = (\frac{2-i+1}{2} k^{1-\alpha} + \frac{i-1}{2} T_2^{1-\alpha}), i = 1..., N.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(-1-\alpha)} [[(\frac{2-i}{2} k^{1-\alpha} + \frac{i}{2} T_2^{1-\alpha})^{\frac{1}{1-\alpha}}]^{-1-\alpha} - [(\frac{2-i+1}{2} k^{1-\alpha} + \frac{i-1}{h} T_2^{1-\alpha})^{\frac{1}{1-\alpha}}]^{-1-\alpha}]...(1.2).$

$E\{t_i^{-1}\}_2 = \frac{E\{t^{-1}\}_1}{2}...(1.2)$

Le calcule de $E\{t_i^{-1}\}_{N-h}, i = h....., N-1$, est trouvé de la façon suivante : $E\{t_i^{-1}\}_N = E\{t_i^{-1}\}_2 + F_i...(3.2)$,telque :

$F_i = \frac{E\{t_i^{-1}\}_{N-h}=\frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)(-1-\alpha)}[[(\frac{N-h-i}{N-h} T_2^{1-\alpha} + \frac{i}{N-h} p^{1-\alpha})^{\frac{1}{1-\alpha}}]^{-1-\alpha} - [(\frac{N-h-i+1}{N-h} T_2^{1-\alpha} + \frac{i-1}{N-h} p^{1-\alpha})^{\frac{1}{1-\alpha}}]^{-1-\alpha}]}{N-h} +$

$\frac{\sum_{i=h+1}^{N} \rho_i}{N-h} - \rho_i$, $E\{t_i^{-1}\}_{N-h}$ déduit de (3.2), qui représente la charge affectée aux serveurs vis à vis au système l'orsqu'il s'agit des tâches de grandes et petites tailles,pour i=h...,N-1, et $\alpha \neq 1$. Tout ce qu'on nous venons de faire sur le calcule de $E\{t_i^{-1}\}_N$, est pour $\alpha \neq 1$, maintenant si $\alpha = 1$ on a :$T_i = k(\frac{T_2}{k})^{\frac{i}{2}}$, pour i=1...,N, d'aprés le théoreme3[4], d'où on peut déduire : $T_{i-1} = k(\frac{T_2}{k})^{\frac{i-1}{2}}$ et $E\{t_i^{-1}\}_2$, est calculé de la façon suivante :

$E\{t_i^{-1}\}_2 = \int_{T_{i-1}}^{T_i} \frac{1}{t} f(t)dt = \int_{T_{i-1}}^{T_i} \frac{1}{t} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-1-\alpha} dt.$

$E\{t_i^{-1}\}_2 = \int_{T_{i-1}}^{T_i} \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} t^{-2-\alpha} dt.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t^{2-\alpha} dt.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} \int_{T_{i-1}}^{T_i} t^{-3} dt.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)-2} [t^{-2}]_{T_{i-1}}^{T_i}.$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{(1-(\frac{k}{p})^\alpha)-2} [(T_i)^{-2} - (T_{i-1})^{-2}].$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} [(k(\frac{T_2}{k})^{\frac{-2i}{2}}) - (k(\frac{T_2}{k})^{\frac{-2i+2}{2}})].$

$E\{t_i^{-1}\}_2 = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} (k(\frac{T_2}{k})^{\frac{-2i}{2}})[1 - (k(\frac{T_2}{k})^{\frac{2}{2}})]....(1.2).$

Et $E\{t_i^{-1}\}_{N-h}$, est calculé de la façon suivante :
$E\{t_i^{-1}\}_{N-h} = F_i$, donc $E\{t_i^{-1}\}_N = E\{t_i^{-1}\}_2 + F_i ...(3.3)$, telque :
$F_i = \frac{E\{t_i^{-1}\}_{N-h} = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha}(k(\frac{p}{T_2})^{\frac{-2i}{N-h}})[1-(T_2(\frac{P}{T_2})^{\frac{2}{N-h}})]+\sum_{i=h+1}^{N}\rho_i}{N-h} - \rho_i, E\{t_i^{-1}\}_{N-h}$ déduit de (3.3),qui représente la charge affectée aux serveurs vis à vis au système l'orsqu'il s'agit des tâches de grandes et petites tailles,pour $i = h..., N-1$, et $\alpha = 1$.

Conclusion Générale

Les systèmes de serveurs Web distribués offrent à leurs utilisateurs un accès rapide à l'information tout en assurant une possibilité d'extension et un coût réduit par rapport aux systèmes à une seule machine dont le risque de surchargement est non négligeable. Ces systèmes de serveurs Web distribués doivent assurer une qualité de service en terme de temps de réponse lors des accès aux données, services et applications. Les tâches Web affectées à ces systèmes ont montré qu'elles suivent une distribution Heavy-Tailed. Cette distribution a la propriété qu'une petite fraction de tâches Web de grande taille participe dans la moitié de la charge totale du système.

Dans le chapitre 1, nous avons présenté un état de l'art des systèmes de serveurs Web distribués , nous avons commencé par citer les différentes entités qui participent dans la construction des systèmes de serveurs Web distribués, les différentes architectures possible pour l'extension de ces systèmes et enfin, nous avons cité les différents types d'approches pour l'affectation des tâches Web qui arrivent au système.

Dans le chapitre 2, nous avons analysé les travaux de recherche qui existent dans des articles pour le problème d'affectation des tâches Web dans les systèmes de serveurs Web distribués. Nous avons parlé des techniques qui assument que la taille des tâches Web est connu à priori, contenant les techniques statiques (i.e. Random, Round Robin, SITA, etc) qui n'utilisent aucune sorte d'information lors du processus d'affectation des tâches Web, et celles jugés dynamiques (i.e. Central Queue, Least Work Remaining, etc) basées sur la charge du serveur et sa capacité de traitement comme critère de choix du serveur (qui se chargera du traitement de la tâche Web). Nous avons, ainsi, cité d'autres sortes de techniques, qui assument que la taille des tâches Web n'est pas connu à priori (i.e. TAGS, TAPTF).

Le chapitre 3, a fait l'objet de la modélisation d'une nouvelle technique appelée SLWT (Split Large Web Task) pour l'affectation des tâches Web dans les systèmes de serveurs Web distribués. Dans ce chapitre, nous avons motivé cette proposition et introduit le modèle sur lequel elle est basée tout en citant les hypothèses pour la taille et la connaissance ou non des tâches Web. Nous avons, ainsi, introduit l'algorithme détaillé pour le principe de fonctionnement de SLWT qui se base, d'un côté, sur le fait de partitionner les tâches Web de grandes tailles en fragments et les exécuter en parallèle et, d'un autre, les séparer des tâches Web de petites tailles pour éviter le retard de traitement de ces dernières, poursuivit par sa modèlisation par la file d'attente M/Bp/1 FCFS.

Comme perspectives, nous proposons une amélioration de l'algorithme de la technique au niveau de l'affectation des tâches Web de petite taille qui est basée sur les intervalles de taille comme critère. Ce critère d'affectation parait limité dans le sens où lorsqu'il y aura des tâches Web de petite taille qui arrivent au système et dont leurs taille n'appartient pas à l'intervalle de taille d'un serveur dédié, le serveur dédié va être libre. Ceci signifie que l'on n'assure pas une utilisation maximale des ressources. La solution pour cet inconvénient est que tous les serveurs doivent participer à l'exécution des tâches Web de petite taille sans l'utilisation des intervalles de taille. En d'autres termes, chaque serveur a la possibilité d'exécuter une tâche Web de petite taille tout en créant des files d'attente avec priorité de taille pour chaque serveur afin d'améliorer les performances.

CLOSSAIRE

CPU :Unité centrale, centre de contrôle et de calcul d'un ordinateur, qui interprète et exécute les instructions. Cette unité centrale de traitement, communément appelée CPU (Central Processing Unit), constitue par définition le cerveau de l'ordinateur.

UNIX :Système d'exploitation multi-utilisateur et multitâche dont dérivent des dizaines d'autres systèmes d'exploitation.Utilisable sur de nombreux types de plates-formes, UNIX est apprécié dans le monde de l'informatique pour sa portabilité, son niveau de sécurité élevé, sa stabilité et sa puissance depuis le début des années 1970.

Cyclique :Désigne un intervalle de temps qui correspond plus ou moins exactement au successifs d'un même phénomène, ou bien d'une transformation d'un système qui revient à sont état initiale .

Pool :Désigne un ensemble de ressource réutilisable géré de façon commune pour un emsenble d'usage telque : prosessus informatique,ordinateur .

Étranglement : Point d'un système limitant les performances globales, sur le temps de traitement des tâches, web.

Le protocole HTTP (HyperText Transfer Protocol) : est un protocole de la couche application du modèle OSI, développé pour le World Wide Web. Ce protocole est un ensemble de règles qui définissent comment le client et le serveur Web interagissent et transfèrent les fichiers (Texte, Images, Audio, Vidéo, etc.).

TCP (Transmission Control Protocol) : TCP se situ au niveau de la couche transport du Modèle TCP/IP. Le protocole TCP est responsable de la création des paquets et de leur Réassemblages et s'assurent que les données sont correctement délivrées.

IP(Internet Protocole) :Protocole de la couche intereréseau du modèle TCP/IP responsable de l'adressage et l'envoie des paquets TCP sur le réseau.

Modèle OSI(Open Systems Interconnection) :Norme définie par l'ISO pour mettre l'interconnexion des systèmes hétérogènes. Il s'agit du modèle le plus connu et le plus utilisé pour décrire les environnements réseau.

Uniform Resource Locator [URL], ou adresse de ressources unifiée : est un système d'adressage permettant l'accès à diverses ressources hébergées dans différents serveurs via le Web (World Wide Web). Cette adresse permet un échange universel, comme une adresse postale ou un numéro de téléphone.

Scalable : mot trés utilisé, pour indiquer à quel point un système hardware ou logiciel parvient à répondre à une demande grandissante de la part des utilisateurs (de plus de requêtes). il s'agit d'une capacité de montée en change.

Checksum : représente, la somme de tous les bits d'un message, dans le but de détecter des erreurs de transmission. Si la somme calculée aprés réception, est différente, il ya une erreur de transfert.

Adresse IP virtuelle (Virtual IP address, 'VIP' or 'VIPA') : est une adresse IP non connectée à un ordinateur ou une carte réseau. Les paquets entrants sont envoyés à l'adresse IP virtuelle, mais en réalité ils circulent tous via des interfaces réseau réelles. Les VIPs sont surtout utilisées pour la redondance de connexion : si un ordinateur ou une carte réseau tombe en panne, un autre ordinateur ou une autre carte réseau peut alors reprendre la connexion.

Cluster : est un grappe de serveurs constitué de deux serveurs au minimum (appelé aussi noeuds) et partageant une baie de disques commune, pour assurer une continuité de service et/ou repartir la charge de calcul et/ou la charge réseau.

Résumé

Le Web permet aux utilisateurs l'accès à des services, à des centres de données et à des applications. La croissance des demandes d'accès (tâches Web) incite les développeurs à basculer vers la distribution des systèmes de serveurs Web (clusters) en raison de leur possibilité d'extension et de leur coût réduit plutôt qu'un seul serveur puissant (comme les grosses machines). Mais ces systèmes de serveurs Web distribués, engendrent des problèmes, durant leur fonctionnement tels que : la difficulté de partage des grandes tâches Web entre les serveurs et, le retard de traitement des petites tâches Web causé par le traitement des grandes tâches Web. La qualité de Service dans ces systèmes se concentre sur le choix de la manière d'affectation de ces tâches Web afin de minimiser le temps de réponse et d'améliorer ainsi les performances, qui n'est pas le cas vu aux problèmes rencontrés, durant leur fonctionnement. Ces tâches Web suivent généralement la distribution "Heavy-Tailed".Nous avons, dans ce mémoire, proposé un algorithme prenant en charge ces problèmes. Le principe de la technique est de partitionner les tâches Web de grandes tailles afin de minimiser leurs métriques de performance et les séparer des tâches Web de petites tailles pour éviter le retard de traitement de ces dernières.

Mots clés : requête Http, Tâches Web, la distribution Heavy-Tailed, performance, Théorie des files d'attente, Partage de charge, QoS.

Abstract

The Web allows peoples the access for services,data centers, and applications. The growing demands of this information motivates the developers to turning to the distributed designs of server systems (clusters) because of their scalability and cost-effectiveness instead of one server with high performance like mainframes. However, these distributed Web server systems generate some problems :large Web tasks make the load difficult to balance among servers, and small Web tasks will be delayed by large ones when they are in the same queue. The Quality of Service (QoS), in these systems, focus on the choice of the manner that affect these Web tasks in order to minimize the response time and thus, improve the performances, that is not case seen the problems met during their functioning . These Web tasks, generally, follow the "Heavy-Tailed" distribution. We have, in this thesis, proposed a new policy that deal to these problems.The main idea of this policy is to split the large Web tasks in order to minimize their performance metrics and separate theme from small Web tasks to avoid the delay of small Web tasks.

Keywords :Http Requests, Web Task, Heavy-Tailed distribution, performance, Queueing Theory,Load Sharing, QoS.

Bibliographie

[1] M. F. Arlitt and C. L. Williamson. Web server workload characterization : The search for invariants. *Proceedings of the 1996 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 126–137, 1996.

[2] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic : Evidence and possible causes. *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems.*

[3] M. S. Taqqu M. E. Crovella and A. Bestavros. Heavy-tailed probability distributions in the world wide web.

[4] M. Crovella M. Harchol-Balter and C. Murta. On choosing a task assignment policy for a distributed server system. *10th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Lecture Notes in Computer Science*, (No. 1469) :pp. 1–13, September 1998.

[5] W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probability,14*, pages 181– 189, 1977.

[6] B. Schroeder and M. Harchol-Balter. Evaluation of task assignment policies for supercomputing servers : The case for load unbalancing and fairness.

[7] A. Riska G. Ciardo and E. Smirni. Equiload : a load balancing policy forclustered web servers. *Performance Evaluation*, (46) :101–124, 2001.

[8] B. Fu and Z. Tari. A dynamic load distribution strategy for systems under high task variation and heavy traffic. *Proceedings of ACM Symposium on Applied Computing*, pages pp. 1031–1037, 2002.

[9] J. Broberg B. Fu and Z. Tari. Task assignement strategy for overloaded systems. *Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)*, pages pp.1346 –1530, 2003.

[10] Z. Tari R. Baldoni, J. Broberg and A. Y. Zomaya. A least flow-time first load sharing approach for distributed server farm.

[11] S. Leonardi B. Awerbuch, Y. Azar and O. Regev. Minimizing the flow time without migration. *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 198–205, 1999.

[12] Z. Tari J. Broberg and P. Zeephongsekul. Task assignment based on prioritising traffic flows. *International Conference on Principles of Distributed Systems (OPODIS'04).*

[13] Zeus technolgies ltd. *Zeus Web Server. http ://www.zeustechnologies.com.*

[14] Apache Server Foundation. Apache http server project. *http ://www.apache.org.*

[15] Netcrafe,netcrafe web server survey. *http ://www.netcraft.com/survey/*.

[16] A. Kirpal P. Rodriguez and E. Biersack. Parallel-access for mirror sites in the internet. *Proceedings of IEEE INFOCOM '00*, pages pp. 864 – 873, March 2000.

[17] K. Obraczka J. Heidemann and J. Touch. Modeling the performance of http over several transport protocols. *IEEE/ACM Tans. on Networking*, (5(5)).

[18] C. E. Wills B. Krishnamurthy. Analysing factors that influence end-to-end web performance. *Computer Networks*, (33(1-6)) :17–32, 2000.

[19] J. C. Mogul B. Krishnamurthy and D. M. Kristol. Key differences between http/1.0 and http/1.1. *Computer Networks*, (31(11-16)) :1737–1751, 1999.

[20] A. Baird-Smith E.Prud'hommeaux H. W. Lie H. F. Nielsen, J. Gettys and C.Lilley. Network performance effects of http/1.1, css1, and png. *Proceedings of ACM Sigcomm 1997*, pages pp. 155–166, Cannes, France, Sept. 1997.

[21] R. Tewari A. Shaikh and M. Agrawal. On the effectiveness of dns-based serverselectio. *Proceedings of IEEE Trans. on Computers*, (34(3)) :204–217, Mar. 1985.

[22] E. Cohen and H. Kaplan. Proactive caching of dns records : Addressing a performance bottleneck. *Proceedings of Symp. on Applications and the Internet,*.

[23] M. Colajanni and P. S. Yu. Adaptive ttl schemes for load balancing of distributed web servers. *SIGMETRICS Perform. Eval. Rev,Vol. 25*, (No. 2) :pp.36–42, 1997.

[24] M. Colajanni V. Cardellini and P. S. Yu. Dns dispatching algorithms with state estimators for scalable web-server clusters. *World Wide Web, Vol. 2*, (No. 3) :101–113, 1999.

[25] P. S. Yu M.Colajanni and D. M. Dias. Analysis of task assignment policies in scalable distributed web-server systems. *IEEE Trans. Parallel Distrib.Syst,Vol. 9*, (No.6) :pp.585–600, 1998.

[26] B. Smith H. Zhu and T. Yang. Hierarchical resource management for web server clusters with dynamic content. *SIGMETRICS 99 : Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages pp. 198–199, 1999.

[27] M. Yuen C. Cheung and A. C. H. Yip. Dynamic dns for load balancing. *ICDCSW '03 : Proceedings of the 23rd International Conference on Distributed Computing Systems*, page pp. 962, 2003.

[28] R. Huang Z. Xu and L. N. Bhuyan. Load balancing of dns-based distributed web server systems with page caching. *ICPADS '04 : Proceedings of the Parallel and Distributed Systems, Tenth International Conference on (ICPADS'04)*, page pp.587, 2004.

[29] D. Chatterjee and A. Zomaya. A task-based adaptive ttl approach for web server load balancing. *ISCC '05 : Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05)*, pages pp. 877–884, 2005.

[30] World wide web consortium. *HyperText Markup Language,http ://www.w3.org/MarkUp.*

[31] M. Colajanni V. Cardellini, E. Casalicchio and P. Yu. The state of the art in locally distributed web server systems. *ACM Computing Surveys*, ((3)3) :263–311, June 2002.

[32] P. Druschel G. Banga and J. C. Mogul. Better operating system features for faster network servers. *ACM Performance Evaluation Review*, (26(3)) :23–30, Dec.1998.

[33] T. Barzilai E. M. Nahum and D. D. Kandlur. Performance issues in www servers. *IEEE/ACM Trans. Networking*, (10(2)) :2–11, Feb. 2002.

[34] P. Druschel V. S. Pai and W. Zwaenepoel. Io-lite : A unified i/o buffering and caching system. *ACM Trans. Comput. Syst*, (18(1)) :37–66, Feb. 2000.

[35] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling : Investigating unfairness. *Proceedings of the 2001 ACMIFIP Joint International Conference on Measurement and Modeling of Computer Systems (Cambridge, MA, June)*, pages 279–290, New York, June 2001.

[36] M. Colajanni V. Cardellini and P.S. Yu. Dynamic load balancing on web-server systems. ieee internet computing.

[37] S. L. Garfinkel. The wizard of netscape.webserver magazine. pages pp. 58–64, July-Aug,1996.

[38] W. Foss D. Mosedale and R. McCool. Lesson learned administering netscape's internet site. *IEEE Internet Computing, vol. 1*, (no. 2) :pp. 28–35, Mar.-Apr. 1997.

[39] L. Baum M. Baentsch and G. Molter. Enhancing the web's infrastructure : From caching to replication. *IEEE Internet Computing, vol. 1, no. 2*, pages 18–27, Mar.-Apr. 1997.

[40] R. E. McGrath T. T. Kwan and D. A. Reed. Ncsa's world wide web server :design and performance. *IEEE Computer*, (no. 11) :pp. 68–74, Nov. 1995.

[41] Cisco's DistributedDirector. Available online at. http ://www.cisco.com/warp/public/cc/cisco/mkt/scale/distr/index.shtml.

[42] P. S. Yu M. Colajanni and V. Cardellini. Dynamic load balancing in geographically distributed heterogeneous web-servers. *Proceedings Of 18th IEEE Int'l.Conf. on Distributed Computing Systems (ICDCS'98)*, pages pp. 295–302, Amesterdam,The Netherlands, May 1998.

[43] Cisco's LocalDirector. Available online at http : //www.cisco.com/warp/public/cc/cisco/mkt/scale/locald/index.shtml.

[44] V. Holmedahl D. Andresen, T. Yang and O. H. Ibarra. Sweb : Toward a scalable world wideweb-server on multicomputers. *Proceedings of 10th IEEE Int'l. Symp,On Parallel Processing, Honolulu*, pages 850–856, April 1996.

[45] M. Colajanni V. Cardellini and P. S. Yu. Redirection algorithms for load sharing in distributed web-server systems. *Proceedings. Of 19th IEEE Int'l. Conf. on Distributed Computing Systems (ICDCS'99), Austin, TX.*

[46] J. Liu A. Bastavors, M. E. Crovella and D. Martin. Distributed packet rewriting and its application to scalable web server architectures. *Proceedings Of 6th IEEE Int'l. Conf. on Network Protocols (ICNP'98), Austin, TX*, pages 1–26, Oct. 1998.

[47] P. Varaiya A. Ephremides and J. Walrand. A simple dynamic routing problem. *IEEE Transactions on Automatic Control*, (AC-25(4)) :690–693, 1980.

[48] G.T. Kim K. Park and M.E. Crovella. On the relationship between file sizes,transport protocols, and self-similar network traffic. *Proceedings of the Fourth International Conference on Network Protocols (ICNP)*, pages pp. 171–180, October 1996.

[49] Gordon Irlam. Unix file size survey 1993. *Available at http ://www.base.com/gordoni/ufs93.html*, September 1994.

[50] F. SEMCHEDINE. Qos dans les systèmes de serveurs web distribués. pages pp. 29–38, 2005-2006.

[51] W. E. Leland and T. J. Ott. Load-balancing heuristics and process behavior. *Proceedings of Performance and ACM Sigmetrics*, pages pp. 54–69, 1986.

[52] V. Paxson and S. Floyd. Wide-area traffic : The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, pages pp. 226–244, June 1995.

[53] T. Osogami-A. Scheller-Wolf M. Harchol-Balter, C. Li and M.S. Squillante. Analysis of task assignment with cycle stealing under central queue. *Proceedings of 23rd International Conference on Distributed Computing Systems (ICDCS '03)*, pages pp. 628–637, 2003.

[54] L. Tan and Z. Tari. Dynamic task assignment in server farms : Better performance by task grouping. *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, page p.175, July 01-04, 2002.

[55] S.A. Nozaki. and S.M. Ress. Approximations in finite-capacity multiserver queues with poisson arrivals. *Applied Probability,15*, pages pp. 826–834, 1978.

[56] A. Feldmann. Web performance characteristics. *http ://www.research.att.com/anja/feldmann/papers.html*, IETF plenary Nov.'99.

[57] IRCache Home. The trace files. *http ://www.ircache.net/Traces/, 2004.*

Résumé

Le Web permet aux utilisateurs l'accès à des services, à des centres de données et à des applications. La croissance des demandes d'accès (tâches Web) incite les développeurs à basculer vers la distribution des systèmes de serveurs Web (clusters) en raison de leur possibilité d'extension et de leur coût réduit plutôt qu'un seul serveur puissant (comme les grosses machines). Mais ces systèmes de serveurs Web distribués, engendrent des problèmes, durant leur fonctionnement tels que : la difficulté de partage des grandes tâches Web entre les serveurs et, le retard de traitement des petites tâches Web causé par le traitement des grandes tâches Web. La qualité de Service dans ces systèmes se concentre sur le choix de la manière d'affectation de ces tâches Web afin de minimiser le temps de réponse et d'améliorer ainsi les performances, qui n'est pas le cas vu aux problèmes rencontrés, durant leur fonctionnement. Ces tâches Web suivent généralement la distribution "Heavy-Tailed".Nous avons, dans ce mémoire, proposé un algorithme prenant en charge ces problèmes. Le principe de la technique est de partitionner les tâches Web de grandes tailles afin de minimiser leurs métriques de performance et les séparer des tâches Web de petites tailles pour éviter le retard de traitement de ces dernières.

Mots clés : requête Http, Tâches Web, la distribution Heavy-Tailed, performance, Théorie des files d'attente, Partage de charge, QoS.

Abstract

The Web allows peoples the access for services,data centers, and applications. The growing demands of this information motivates the developers to turning to the distributed designs of server systems (clusters) because of their scalability and cost-effectiveness instead of one server with high performance like mainframes. However, these distributed Web server systems generate some problems :large Web tasks make the load difficult to balance among servers, and small Web tasks will be delayed by large ones when they are in the same queue. The Quality of Service (QoS), in these systems, focus on the choice of the manner that affect these Web tasks in order to minimize the response time and thus, improve the performances, that is not case seen the problems met during their functioning . These Web tasks, generally, follow the "Heavy-Tailed" distribution. We have, in this thesis, proposed a new policy that deal to these problems.The main idea of this policy is to split the large Web tasks in order to minimize their performance metrics and separate theme from small Web tasks to avoid the delay of small Web tasks.

Keywords :Http Requests, Web Task, Heavy-Tailed distribution, performance, Queueing Theory,Load Sharing, QoS.