Hébergement Web

Djelloul Ziadi

Université de Rouen - France

M2SSI'2017
DIR 2017/2018

Plan du cours

HTTP : HyperText Transfer Protocol

- Protocole Client/Serveur
- Une requête - Une réponse
- Client appelé : User-Agent soumis une requête pour des ressources
- Serveur : reçoit et traite les requêtes :
  - charge un fichier à partir du disque
  - redirige les requêtes vers d'autres serveurs web
  - interroge un serveur de bases de données
  - effectue des calculs
  - ...

Protocole HTTP/0.9

- Début des années 90 : HTTP/0.9

1. Le client ouvre la connexion au serveur
   - `telnet adresse-serveur 80`
2. Le client envoie sa requête
   - `GET /index.html`
3. Le serveur envoie la réponse
   - ```
     <html>
     <head>
     <title>Master 2 SSI</title>
     </head>
     <body>
     Bienvenue sur le site SSI.
     </body>
     </html>
     ```
4. Le serveur ferme la connexion
     ```

Protocole HTTP/1.0

- main 1996 : HTTP/1.0 (RFC 1945)
- Intègre : Gestion du cache, identification, hôtes virtuels
- Utilisation d'en-têtes spécifiés dans la (RFC 822)
  - `Date: Fri, 31 Dec 2016 23:59:59 GMT`
    `Server: Apache/1.0.0`
    `Content-Type: text/html`
    `Content-Length: 59`
    `Expires: Sat, 01 Jan 2017 00:59:59 GMT`
    `Last-modified: Fri, 09 Sep 2016 14:21:40 GMT`
- Structure d'une requête HTTP
  - `Ligne de commande (Commande, URL, Version de protocole)`
    `En-tête de requête`
    `[Ligne vide]`
    `Corps de la requête`
- Structure d'une réponse HTTP
  - `Ligne de statut (Version, Code-réponse, Texte-réponse)`
    `En-tête de réponse`
    `[Ligne vide]`
    `Corps de la réponse`

Terminologie

- URL
- Commande
- Code réponse
- En-têtes

URL

- URL Uniform Resource Locator
- identifiant qui spécifie comment protocole et où adresse accéder à une rssource.
- Syntaxe : protocole:adresse

protocole	signification	exemple
file	accède à un fichier local	file :///etc/resolv.conf
ftp	accède à un fichier distant via FTP	ftp ://ftp.ssi.edu/rapport.tar.gz
http	accède à un fichier distant via HTTP	http ://www.ssi.edu/index.html
https	accède à un fichier distant via HTTP/SSL	https ://www.ssi.edu/exam.html
ldap	accède au service d'annuaire LDAP	ldap ://ldap.ssi.edu :389/cn=ziadi
mailto	envoie un courriel	mailto :admin@ssi.edu
...	...	...

- URL web

```
protocole://[login:password@]hostname[:port][/path][?query][#anchor]
```

Commandes HTTP (méthodes)

- `GET` : demande l'envoi de la ressource située à l'URL spécifiée
- `HEAD` : demande l'envoi de l'en-tête de la ressource située à l'URL spécifiée
- `POST` : envoie les données au programme situé à l'URL spécifiée
- `PUT` : envoie les données à l'URL spécifiée
- `DELETE` : supprime la ressource située à l'URL spécifiée
- `CONNECT` : accès à un serveur web sécuritsé située à l'URL spécifiée
- `OPTIONS` : donne les méthodes supportées par le serveur pour URL spécifiée

Classes de codes de réponse HTTP

- `1xx` : Code Informationnel (ex. `100` : continuer l'envoi)
- `2xx` : Succès (ex. `200` : OK)
- `3xx` : Redirection (ex. `301` : Changement d'adresse définitif)
- `4xx` : Erreur du client web (ex. `404` : Ressource introuvable)
- `5xx` : Erreur du serveur web (ex. `500` :Erreur interne du serveur)

Quelques en-têtes HTTP

Nom	Sens	signification
`Host` : www.ssi.edu :8080	$\rightarrow$	nom de domaine et le port demandé
`Content-Type` : application/json	$\leftrightarrow$	type requis ou envoyé
`User-Length` :		
`Authorization` : Basic YWxhZGR...	$\rightarrow$	param. identification client
`Set-Cookie` : theme=bleu	$\leftarrow$	demande de stockage d'un cookie
`Cookie` : theme=bleu	$\rightarrow$	cookie envoyé par le client
`User-agent` :		
`Server` :		
`Upgrade` : HTTP/2.0		
`Last-Modified` : Wed, 21 Oct 2015 07 :28 :00 GMT	$\leftarrow$	date et l'heure de la dernière modif. de la ressource
`Expires` :		
`Cache-Control` :		
`Age` :		

Protocole HTTP/1.1

- HTTP 1.1 (RFC 2616)
- Une meilleure gestion du cache r/t HTTP 1.0
- L'en-tête `Host` devient obligatoire
- Utilisisation des connexions persistantes
  - une connexion n'est pas immédiatement fermée après une requête : `keep-alive`
  - envoi de plusieurs requêtes sur la même connexion sans attendre les réponses : `pipelining`
  - La gestion de la persistance d'une connexion est gérée par l'en-tête `Connection: keep-alive`
- HTTP 1.1 supporte la négociation de contenu `Accept-`
  - `Accept` `Accept: text/html`
  - `Accept-Charset` `Accept-Charset: ISO-8859-1,utf-8`
  - `Accept-Language` `Accept-Language: fr`
  - `Accept-...` ...

Protocole HTTP/2

- SPDY (speedy) conçue par Google pour augmenter les capacités du protocole HTTP
- SPDY utilise la compression, le multiplexage, et la priorisation.
- SPDY a été intégré par L'IETF dans HTTP/2
- HTTP/2 est un protocole binaire qui multiplexe de nombreux flux sur une seule connexion TCP (normalement cryptée par TLS).

Exemple de transaction

```
ziadi@ubuntu:~/WEB$ curl -svo /dev/null http://ww.google.fr
* Rebuilt URL to: http://ww.google.fr/
*   Trying 216.58.208.195...
* Connected to ww.google.fr (216.58.208.195) port 80 (#0)
> GET / HTTP/1.1
> Host: ww.google.fr
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Location: https://www.google.fr/
< Content-Type: text/html; charset=UTF-8
< X-Content-Type-Options: nosniff
< Date: Thu, 09 Nov 2017 09:42:21 GMT
< Expires: Sat, 09 Dec 2017 09:42:21 GMT
< Server: sffe
< Content-Length: 219
< X-XSS-Protection: 1; mode=block
< Cache-Control: public, max-age=2592000
< Age: 176696
<
{ [219 bytes data]
* Connection #0 to host ww.google.fr left intact
ziadi@ubuntu:~/WEB$
```

Exemple de transaction

```
ziadi@ubuntu:~/WEB$ curl -svo /dev/null http://www.univ-rouen.fr
* Rebuilt URL to: http://www.univ-rouen.fr/
*   Trying 193.52.152.28...
* Connected to www.univ-rouen.fr (193.52.152.28) port 80 (#0)
> GET / HTTP/1.1
> Host: www.univ-rouen.fr
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 11 Nov 2017 10:42:48 GMT
< Server: Apache
< Set-Cookie: JSESSIONID=E259E25AA70BD7E89395F018FE057BAD; Path=/
< Connection: close
< Transfer-Encoding: chunked
< Content-Type: text/html
<
{ [1251 bytes data]
* Closing connection 0
ziadi@ubuntu:~/WEB$
```

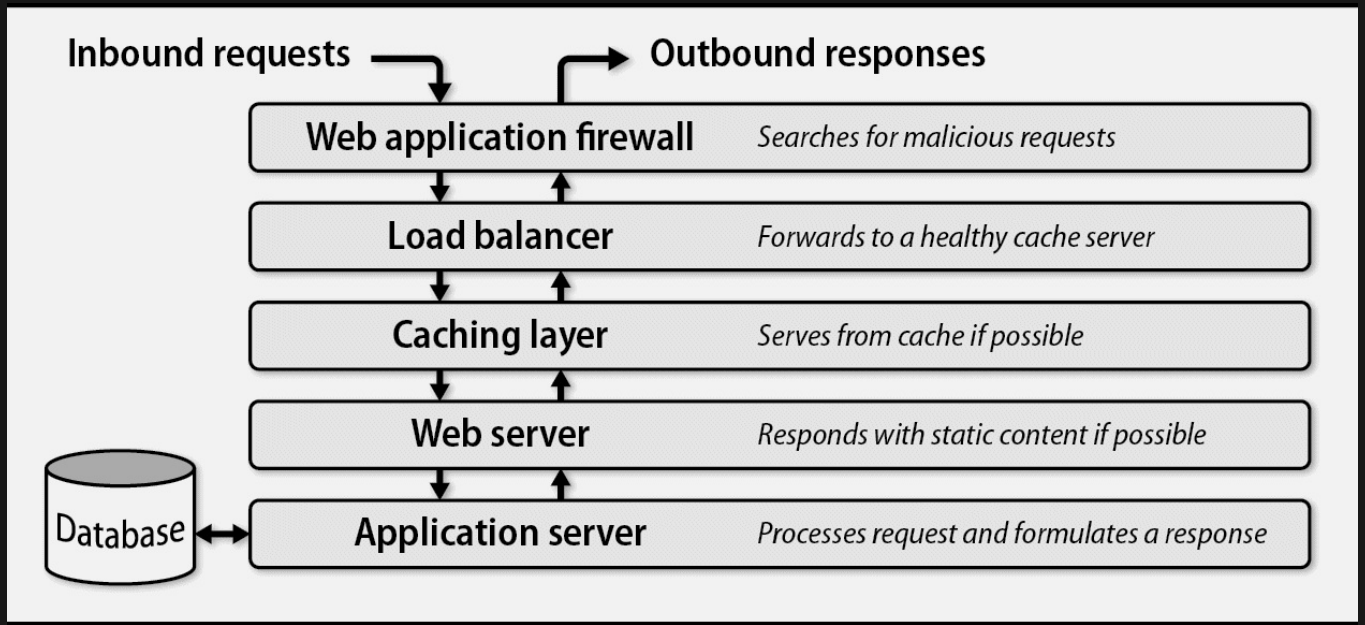
Types de serveurs HTTP

- Serveur de contenus : Envoie du contenu statique (Apache, Nginx)
- Cache : Accèlere l'accès au contenu (Squid, Varnish)
- Seveur d'applications : Le serveur web délègue la génération de réponses à d'autres serveurs (Unicorn, Tomcat)
- Pare-feu Web : Analyse le trafic HTTP (ModeSecurity)
- Répartiteurs de charge (Load Balancer) : Redirige le trafic vers différents serveurs (HAProxy)

Serveur Mandataire Web(proxy)

- Serveur intermédiare qui reçoit des rquêtes HTTP, effectue un traitement, puis envoie les requêtes vers les leurs destinations.
- Le proxy est transparent pour les clients
- Les répartiteurs de charge, les pare-feu web, et les serveurs cache sont des serveurs mandataires spécialisés
- Un serveur web qui redirige les reqêutes vers des serveurs d'applications peut être considéré comme serveur mandataire

Composants d'hébergement Web



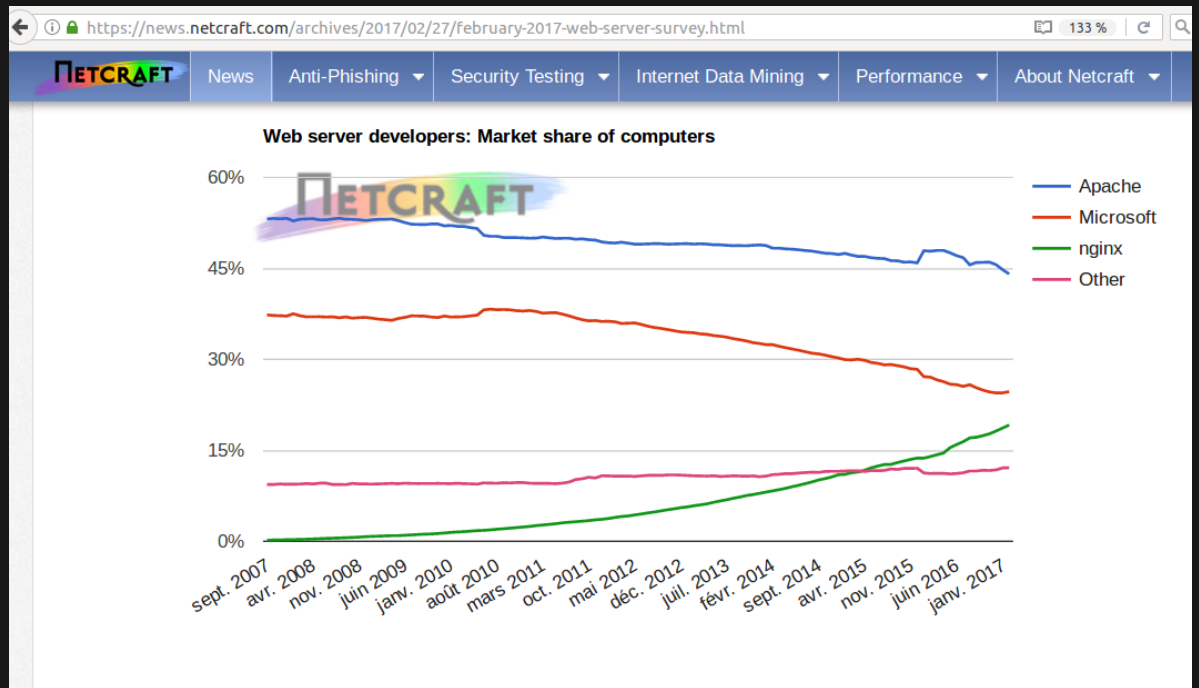
- Pour maximiser la disponibilité, chaque couche doit s'exécuter sur un ou plusieurs nodes
- Le serveur web Nginx avec le support de cache activé est plus efficace qu'un pile de serveurs s'exécutant sur machines virtuelles.

Serveurs Web : quelques fonctionnalités

- `Virtual hosts` : Hébergement de plusieurs site sur un seul serveur
- `Journalisation` : traces des requêtes et des réponses
- `Authentifiation` : permet de sécuriser l'accès aux ressources web
- `Routage` : Redirection en fonction de l'URL
- `Génération de contenu dynamique` : via serveurs d'applications
- `Gestion des connexion TLS`

Serveurs de Web : principaux serveurs

- 1995 : Apache HTTP Server (apache.org)
  La référence des implémentation de serveur HTTP
- 1995 : Microsoft IIS
- 2002 : Nginx (“engine X”) (nginx.org)
  NGINX est un serveur polyvalent conçu pour la rapidité et l’efficacité
- 2014 : H2O Server (h2o.examp1e.net)
  H2O tire pleinement parti des fonctionnalités HTTP/2



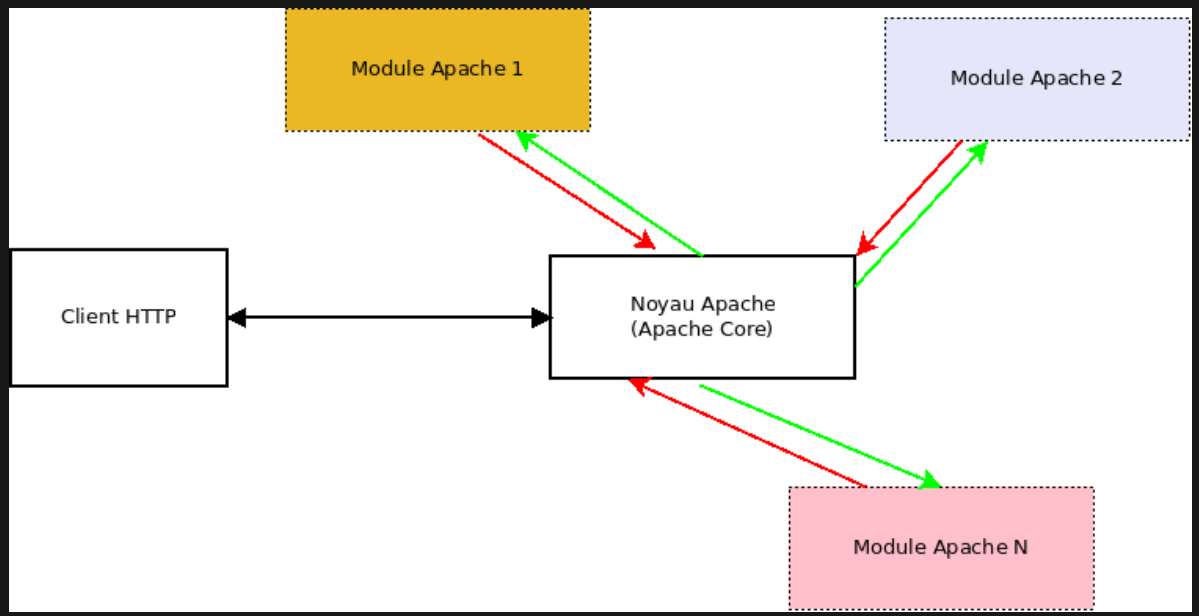
Serveur HTTP Apache

- 1993 NCSA HTTPd National Center for Supercomputing Applications (Robert McCool)
- 1994 Apache sous forme de patches à appliquer à NCSA HTTPd (Apache Group)
- 1995 nouvelle architecture modulaire Apache 1.0
- 1999 nombreuses améliorations Apache 1.3
- 2000 Apache 2.0 multi-processus et multi-threadé
  http://httpd.apache.org/docs/2.2/new_features_2_0.html
- 2005 Apache 2.2
  http://httpd.apache.org/docs/2.2/new_features_2_2.html
- 2012 Apache 2.4
  http://httpd.apache.org/docs/trunk/new_features_2_4.html

Structure du Serveur HTTP Apache

Apache est composé de deux blocs :

- `Apache Core` : Fonctionalités de base d'Apache
  Un petit noyau pour le traitement des requêtes/réponses HTTP et pour les `modules multi-traitement` (MPM)
  - https://httpd.apache.org/docs/trunk/fr/mod/core.html
- `Apache Modules` : Etendent les Fonctionalités de base
  - Authentification, Proxy, Réecriture d'URL ...
  - https://httpd.apache.org/docs/2.4/fr/mod/
  - `apt-cache search libapache2-mod`



Modules compilés dans Apache

```
root@ubuntu:~/SAVE# apache2 -l
Compiled in modules:
  core.c
  mod_so.c
  mod_watchdog.c
  http_core.c
  mod_log_config.c
  mod_logio.c
  mod_version.c
  mod_unixd.c
```

Utilitaire pour les extensions d'Apache

- APache eXtenSion tool
- Paquet `apache2-dev`
- man apxs
- apxs -c -i -a mod_hello.c

Gestion des requêtes client

- `multi-process` un nouveau processus pour chaque requête
- Chaque processus a son propre espace d'adressage
  - Apache en mode prefork
- `multi-thread` un nouveau thread (processus léger) pour chaque requête
- Les threads d'un même processus s'exécutent dans un espace mémoire partagé
  - IIS Windows
  - Apache en mode worker

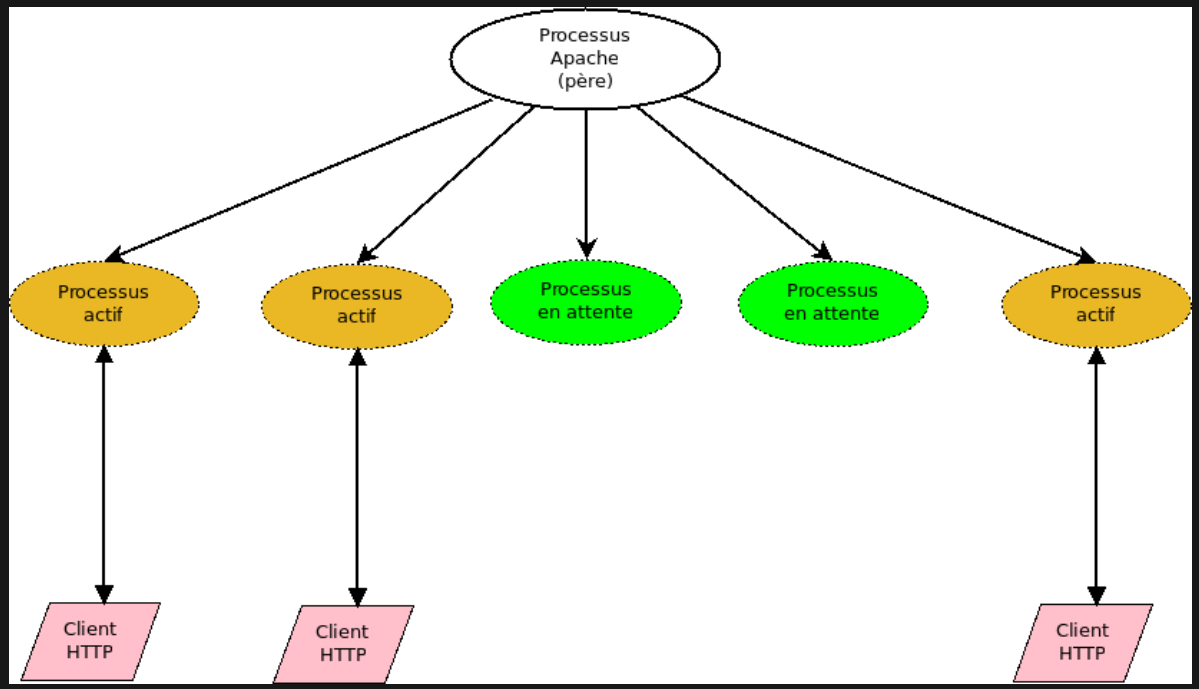
Modules multi-traitement

- BeOS : beos
- Netware : mpm_netware
- OS/2 : mpmt_os2
- *nix : `prefork`
- *nix : `worker`
- *nix : `event`
- Windows : mpm_winnt

MPM Prefork

- Multi-Processing
- Avantages
  - Stabilité
  - Isolation
- Inconvennient
  - Perte de performance : `fork` gourmand en processeur
- `apache-mpm-prefork` un serveur web avec démarrage anticipé de processus, sans thread
- Modèle traditionnel du serveur Httpd

Configuration mode prefork



Configuration mpm prefork

```
<IfModule mpm_prefork_module>
    StartServers                5
    MinSparseServers            5
    MaxSparseServers           10
    ServerLimit               150
    MaxClients                150
    MaxRequestsPerChild         0
</IfModule>
```

- `StartServers` : nbr de processus à lancer au démarrage d'Apache
- `MaxClients` : nbr. total de requêtes pouvant être exécutées simultanément sur le serveur
- `ServerLimit` : nbr. max de processus qu'Apache peut créer
- `MinSparseServers` : nbr. min de processus à conserver
- `MaxSparseServers` : nbr. max de processus à conserver
- `MaxConnectionsPerChild` : nbr. max de requêtes par processus

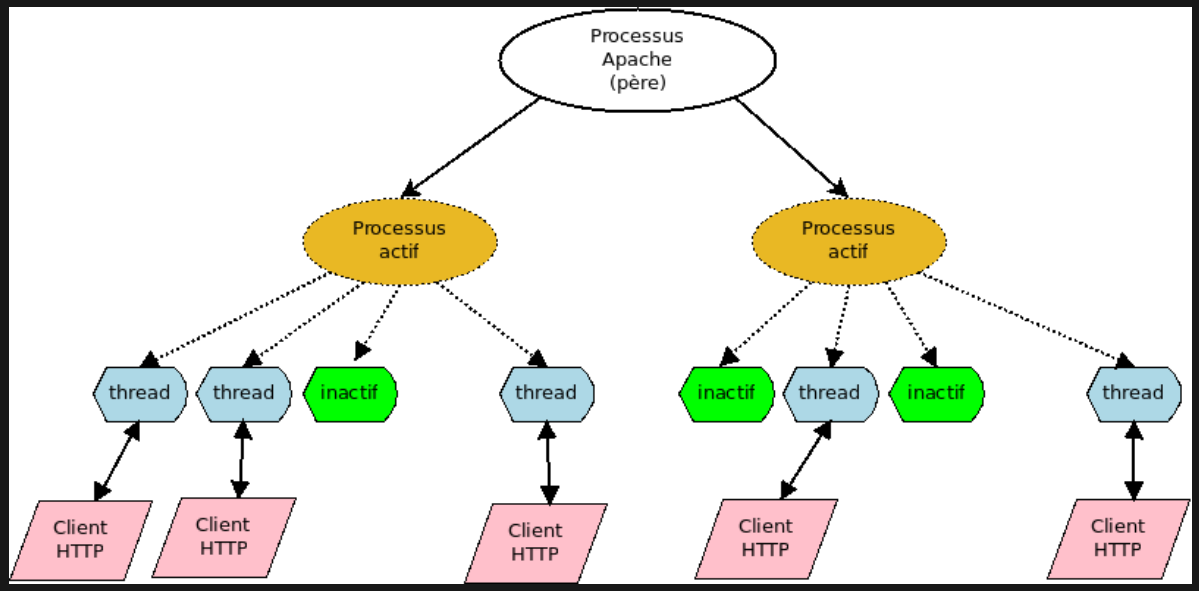
Les limites

- Nombre de coeurs `nproc, lscpu`
- Limite des descripteurs ouverts par un processus `ulimit`

MPM Worker

- Multi-Threading (nouveauté d'Apache 2.0)
- `apache-mpm-worker` un serveur web hybride multi-processus multi-thread
- traite un grand nombre de requêtes avec moins de ressources système qu'un serveur basé sur des processus
- Non compatible avec les bibliothèques "non-thread safe" (ex. mod_php)

Configuration mode worker



MPM worker

```
<IfModule mpm_event_module>
        StartServers                       2
        MinSpareThreads                   25
        MaxSpareThreads                   75
        ThreadLimit                       64
        ThreadsPerChild                   25
        MaxRequestWorkers                150
        MaxConnectionsPerChild             0
</IfModule>
```

- `StartServers` : nbr de processus à lancer dès le démarrage
- `MinSpareThreads` : nbr. min de threads de traitement à conserver
- `MaxSpareThreads` : nbr. max de threads de traitement à conserver
- `ThreadsPerChild` : nbr. de threads de traitement par processus
- `ThreadLimit` : nbr. max de threads de traitement que l'on peut définir par processus
- `MaxRequestWorkers` : nbr. de threads de traitement
- `MaxConnectionsPerChild` : nbr. max de requêtes par processus

MPM event

- Hybride multi-processus multi-thread
- Séparation de “suivi de connexion” `Keep-Alive` et réponse aux requêtes
- Le thread de traitement ne fait que répondre aux requêtes
- Le thread de gestion suit les connexions
- worker (synchrone) – event (asynchrone)

MPM compilé avec Apache

```
root@ubuntu:/etc/apache2# apache2 -V
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress th
is message
Server version: Apache/2.4.18 (Ubuntu)
Server built:   2017-09-18T15:09:02
Server's Module Magic Number: 20120211:52
Server loaded:  APR 1.5.2, APR-UTIL 1.5.4
Compiled using: APR 1.5.2, APR-UTIL 1.5.4
Architecture:   64-bit
Server MPM:     event
  threaded:     yes (fixed thread count)
    forked:     yes (variable process count)
Server compiled with....
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
```

MPM event dans Apache

```
root@ubuntu:~# cat /etc/apache2/mods-enabled/mpm_event.conf
<IfModule mpm_event_module>
        StartServers             2
        MinSpareThreads         25
        MaxSpareThreads         75
        ThreadLimit             64
        ThreadsPerChild         25
        MaxRequestWorkers      150
        MaxConnectionsPerChild   0
</IfModule>
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# pstree | grep apache2
        |-apache2---2*[apache2---26*[{apache2}]]
root@ubuntu:~#
```

Configuration d'Apache

```
root@ubuntu:~# ls /etc/apache2/
apache2.conf    envvars          mods-enabled     sites-enabled
conf-available  magic            ports.conf
conf-enabled    mods-available   sites-available
root@ubuntu:~#
```

Configuration répartie dans plusieurs fichiers

- `apache2.conf` : fichier de configuration principal (rarement modifié)
  Utilise `Include` pour inclure les autres fichiers de configuration
- `mods-enabled/*.load` : directives de chargement de modules (`LoadModule`)
- `mods-enabled/*.conf` : éléments de configurations propre à chaque module chargé
- `sites-enabled/*` : configuration spécifique des sites
- `mods-available` : contient l'ensemble des définitions des modules installés
- `sites-available` : contient les fichiers de configuration de tous les sites

Configuration d'Apache

- `conf-enabled/*` : configurations globales au serveur
- `ports.conf` : ports TCP en écoute (`Listen`) et (`NameVirtualHost`) pour la configuration des hôtes virtuels
- `envvars` : variables d'environment pour `apache2ctl`
- `magic` : pour le module (mod_mime_magic). Le type de contenu d'un document peut être déterminer en regardant les quelques premiers octets de ce contenu

Quels adresses écouter ?

- fichier `ports.conf`
- `Listen 10.0.0.1:8080`
  Accèpte les requêtes à destination de l'adresse 10.0.0.1 sur le port 8080
- `Listen *:80` ou `Listen 80`
  Apache écoute le port 80 sur toutes les adresses locales disponibles
-
```
<IfModule mod_ssl.c>
  Listen 443
</IfModule>
```

- `NameVirtualHost *:80`

Hôtes Virtuels

- Le principe des Hôtes Virtuels consiste à faire fonctionner un ou plusieurs sites Web sur une même machine.
  - `www.ssi.edu`
  - `www.gil.org`
  - `www.lrt.fr`
- quelques façons d'héberger plusieurs sites web
  - Par adresse IP (une adresses IP par site)
  - Par nom de machine (noms de sites différents et une adresse IP)
  - par machine virtuelles (une machine virtuelle par site)

Hôtes virtuels basés sur les adresses IP

- Nécessite une adresse IP pour chaque hôte virtuel hébergé
- Utilise intefaces virtuelles

```
ip addr add 10.0.0.1/8 dev eth0 label eth0:1
```

- Nombre d'alias IP limité

```
<VirtualHost 10.0.0.1:80>
 DocumentRoot /var/www/site1
 ...
</VirtualHost>


<VirtualHost 192.168.1.1:80>
 DocumentRoot /var/www/site2
 ...
</VirtualHost>


<VirtualHost 10.0.0.1:8080>
 DocumentRoot /var/www/site3
 ...
</VirtualHost>
```

Hôtes virtuels basés sur les noms

- ensemble de noms de machines partageant la même adresse IP
- l'entête HTTP `HOST` (renseignée par `User-Agent`) permet de différentier les sites.
- Directives `ServerName` et `ServerAlias`
- Directive `NameVirtualHost`

```
NameVirtualHost *:80


<VirtualHost *:80>
    ServerName www.ssi.edu
    ServerAlias ssi.edu
    DocumentRoot "/var/www/ssi"
</VirtualHost>


<VirtualHost *:80>
    ServerName www.lrt.edu
    DocumentRoot "/var/www/lrt"
</VirtualHost>
```

Sections de configuration

- Les directives peuvent s'appliquer au serveur dans son ensemble, ou à une section.
- sections de configuration
  - Paramètres Globaux du serveur (ensemble du serveur) : `KeepAlive On`, `PidFile`, ...
  - Paramètres globaux conditionnels (ensemble du serveur) : `<If>`, `<IfDefine>`, `<IfModule>` et `<IfVersion>`
  - Evaluées pour chaque requête (section de configuration) : <Directory> `<DirectoryMatch>`, `<Files>`, `<FilesMatch>`, `<Location>`, `<LocationMatch>`, `<Proxy>`, `<ProxyMatch>` et `<VirtualHost>`

Directives évaluées au démarrage

- `Redirect` exécutée si `Apache` a été lancé avec l'option `-D Ferme`

```
<IfDefine Ferme>
   Redirect  "/"  "http ://www.gil.org/"
</IfDefine>
```

- `Listen` exécutée si le module `ssl` a été chargé

```
<IfModule ssl_module>
   Listen  443
</IfModule>
```

- les directives ne s'appliquent que si la version est supérieure ou égale à 2.4.0.

```
<IfVersion >= 2.4>
   Require  all  denied
</IfVersion>
```

Conteneurs de système de fichiers

```
<<Directory "/var/www/ssi">
    Options  +Indexes

    <Files "examen.pdf">
        Require all denied
    </Files>
</Directory>

<DirectoryMatch "^/www/.*/[0-9]3">
    require   valid-user
</DirectoryMatch>

<FilesMatch " \.(gif|jpe?g|png)$">
    Require all denied
</FilesMatch>
```

Conteneurs de l'arborescence du site web

- A utiliser pour appliquer des directives à des contenus situés en dehors du système de fichiers
- Ne jamais utiliser <Location> pour restreindre l'accès à des objets du système de fichiers

```
<Location "/server-status">
    SetHandler server-status
</Location>
```

Interdit l'accès à `www.ssi.edu/private123` et `www.ssi.edu/private/doc/toto.html`

```
<LocationMatch "^/private">
    Require all denied
</LocationMatch>
```

Authentification et autorisation

- `Authentification` : procédé permettant de s'assurer qu'une entité est bien celle qui prétend être (identification)
- `Autorisation` : procédé permettant de vérifier si une entité authenifiée à le droit d'accomplir une action
- Différents types de modules d'authentification/autorisation
  - `auth_*` : modules frontaux chargés de la méthode communication des données. Ils n'effectuent pas les opérations d'authentification/Autorisation
  - `authn_*` : modules ne traitant que les opérations d'authentification
  - `authz_*` : modules ne traitant que les opérations d'autorisation
  - `authnz_*` : modules ne traitant les opérations d'authentification et d'autorisation

Authentification

- Deux types d'authentification
  - `Basic` : RFC 1945, 2617
    mot de passe circule en clair
    fonctionne avec tous les client HTTP
  - `Digest` : RFC 2617, 2069
    le mot de passe ne circule pas en clair (mot de passe hashé)
    plus d'échange entre le client et le serveur r/t à Authentification type Basic
  - `Form` : basé sur `Basic`.
    Il permet de restreindre l'accès en recherchant les utilisateurs dans les fournisseurs spécifiés à l'aide d'un `formulaire de connexion HTML`

```
root@ubuntu:/usr/lib/apache2/modules#  ls *auth_*
mod_auth_basic.so  mod_auth_digest.so  mod_auth_form.so
```

Authentification `Basic`

- utiliser pour l'authentification basique HTTP
- https://httpd.apache.org/docs/2.4/mod/mod_auth_basic.html
- Echange client/serveur
  1. Le client demande l'URL
  2. Le serveur répond par `401 Unauthorized` + l'entête `WWW-Authenticate: Basic realm="zone protege"`
  3. Le client demande le login et le mot de passe et envoie la réponse avec l'entête `Authorization: Basic 'toto' 'passworddetoto'`
  4. Le serveur vérifie les données de l'authentification et renvoie la ressource en cas de succès ou revient au point 2 en cas d'echec

Authentification Basic Apache

```
AuthType Basic
AuthName "Fichiers protégés"
AuthBasicProvider file
AuthUserFile "/etc/apache2/.passwords"
```

- `AuthType` : type d'authentification
- `AuthName` : correspond au `realm`
- `AuthBasicProvider` : indique le type de ("back-end") provider
- `AuthUserFile` : indique l'emplacement du fichier des mots de passe pour le provider `file`

Authentification Basic

```
AuthType Basic
AuthName "Zone protégée"
AuthBasicProvider file
AuthUserFile "/etc/apache2/.passwords"
AuthGroupFile "/etc/apache2/.groups"
Require group m2ssi
```

```
# cat  /etc/apache2/.groups
m2ssi: jean paul idriss wissal sarah
```

```
# htpasswd -c /etc/apache2/.passwords jean
# htpasswd  /etc/apache2/.passwords wissal
# htpasswd  /etc/apache2/.passwords idriss
```

Authentification `Digest`

- Utiliser dans l'authentification à base de condensés MD5
- https://httpd.apache.org/docs/2.4/mod/mod_auth_digest.html
- Echange client/serveur
  1. Le client demande l'URL
  2. Le serveur répond par `401 Unauthorized` + l'entête `WWW-Authenticate: Digest challenge` (`challeng` : niveau de protection, algo de chiffrement, domaine de protection, ...)
  3. Le client demande le login et le mot de passe, calcule une réponse en fonction du challenge puis envoie la réponse avec l'entête `Authorization: Digest condensésMD5`
  4. Le serveur vérifie les données de l'authentification et renvoie la ressource en cas de succès ou revient au point 2 en cas d'echec

Authentification Digest : exemple

```
AuthType Digest
AuthName "Zone protégée"
AuthDigestDomain "/" "http://www.ssi.edu/prive2/"
AuthDigestProvider file
AuthUserFile "/etc/apache2/.passwords"
Require valid-user
```

- `AuthType` : type d'authentification
- `AuthName` : correspond au `realm`
- `AuthDigestDomain` : liste des `URI` protégées
- `AuthDigestProvider` : indique le type de ("back-end") provider
- `AuthUserFile` : indique l'emplacement du fichier des mots de passe pour le provider `file`

Les providers d'authentification/Autorisation

- Authentification Directive `AuthBasicProvider` et `AuthDigestProvider`

```
# ls /usr/lib/apache2/modules$ ls mod_authn_*
mod_authn_core.so  mod_authn_dbd.so
mod_authn_dbm.so   mod_authn_file.so
mod_authn_socache.so
```

- Autorization Directive `Require`

```
# ls /usr/lib/apache2/modules$ ls mod_authn_*
mod_authz_core.so        mod_authz_host.so
mod_authz_dbd.so         mod_authz_owner.so
mod_authz_dbm.so         mod_authz_user.so
mod_authz_groupfile.so
```

Directive d'autoroization

- Contrôle d'accès en fonction de l'hôte du client
  - La directive `Require` permet d'accorder ou d'interdire l'accès à certaines ressources
  - Conjonction avec les directives `RequireAll`, `RequireAny`, et `RequireNone`
- Contrôle d'accès en fonction de variables arbitraires
- Syntaxe dépond du providers d'autorisation
  par exemple le provider `mod_authnz_ldap` fournit les directives `Require ldap-user`, `Require ldap-group`, `Require ldap-dn`, ...

Directive Require

```
- mod_authz_core
Require all granted
Require all denied
Require env OUVERT

- mod_authz_user  mod_authz_groupefile
Require user jean wissal
Require group m2ssi gil
Require valid-user

- mod_authz_host
Require ip 192.168.1 10 172.16.1.32/27
```

Combinaison des autorisations

- Une suite `Require`

```
Require valid-user
Require not ip 10.0.0.0/8
```

- `RequireAny`

```
<RequireAny>
        Require group m2ssi
        Require ip 192.168.1.0/24
</RequireAny>
```

- `RequireAny`

```
<RequireAll>
        Require all granted
        Require not ip 10.0.0.4
</RequireAll>
```

```
<Directory "/var/www/ssi/docs">
   <RequireAll>
         <RequireAny>
            Require user superadmin
            <RequireAll>
            Require group m2ssi
            Require group admin
               <RequireAny>
                Require group gil
                Require user idriss
                </RequireAny>
            </RequireAll>
         </RequireAny>
         <RequireNone>
            Require group invite
            Require user jean
         </RequireNone>
   </RequireAll>
</Directory>
```

Journalisation

- Importance des informations sur l'activité et les performances de votre serveur`CustomLog`
- Identification de disfonctionnement du serveur `ErrorLog`
- Ne pas tout journaliser (dégradation des performances)
- `mod_log_config` : Journalise les requêtes envoyées au serveur

  https://httpd.apache.org/docs/current/fr/mod/mod_log_config.html

```
LogLevel Emerg, Alert, Crit, Error, Warn, Notice, Info, Debug
LogFormat "%h %l '%r' %>s %b" log_ssi
CustomLog /var/log/ssi-access.log log_ssi


CustomLog /dev/null combined
```

Connexion TLS : Transport Layer Security

```
Génération de la clé privée
// openssl genrsa -des3 -out serv.key 1024
# openssl genrsa  -out serveur.key 1024

Génération du fichier csr (Cerficat Signing Request)
# openssl req  -new -key  serveur.key -out serveur.csr

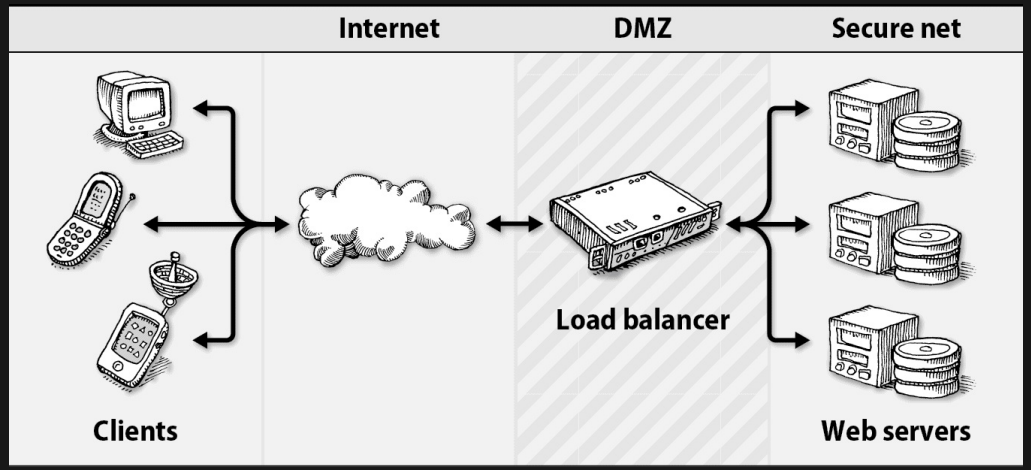
Génération du certifcat
# openssl x509 -req -days 365
  -in serveur.csr -signkey server.key -out serveur.crt

# Configuration du mode ssl
SSLCertificateFile      /etc/apache2/ssl/serveur.crt
SSLCertificateKeyFile   /etc/apache2/ssl/serveur.key
```

Serveur Mandataire

- Serveur Mandataire (Proxy) : serveur qui agit comme un intermédiaire pour les requêtes des clients vers d'autres serveurs
- Serveur Mandataire Inverse (Reverse Proxy) : permet à un client d'accéder à des serveurs internes (non visibles)
- Serveur intermédiaire entre les clients et les serveurs web. Il conserve des copies des réponses aux requêtes les plus fréquentes, transitant par son biais
  Un serveur cache répond au client s'il dispose de la ressource demandé (réduction de la bande passante)

Répartiteur de charge

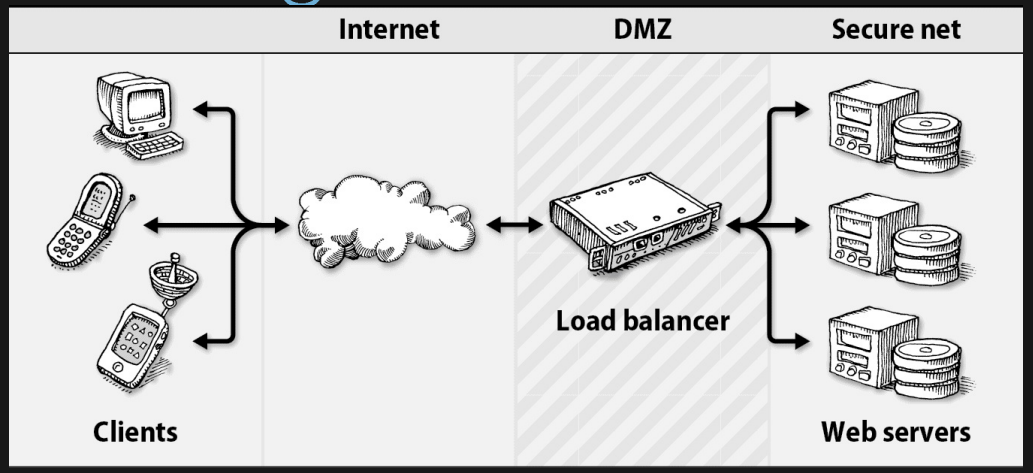


- Mandataire Inverse
- Répartition basée sur
  - Couche transport : adresse IP et port
  - Couche transport : URL, cookie, entêtes HTTP
- Les Répartiteurs de charge open source :
  - `NGINX`
  - `HAProxy`
  - `Apache proxy_balancer` (pas très utilisé)

Répartiteur de charge

- Serveur unique
  - “Point individuel de défaillance”
  - Arrêt du serveur en cas de maintenance
  - Risque d’attaque
  - Risque de dégradation de performance
- Plusieurs serveurs
  - un Répartiteur et plusieurs serveurs
    - ne traite pas les requêtes mais, les route vers d’autres serveurs
    - un serveur qui nécessite une maintenance peut être supprimé du pool de serveurs
    - distribution intelligente des multiples requêtes sur les différents serveurs
    - suit l’état des différents serveurs pour s’assurer de leur bon fonctionnement. Il peut supprimer un serveur du pool
  - Deux répartieurs pour éviter le “Point individuel de défaillance”

Répartiteur de charge



- Plusieurs serveurs
  - un Répartiteur et plusieurs serveurs
    - ne traite pas les requêtes mais, les route vers d'autres serveurs
    - un serveur qui nécessite une maintenance peut être supprimé du pool de serveurs
    - distribution intelligente des multiples requêtes sur les différents serveurs
    - suit l'état des différents serveurs pour s'assurer de leur bon fonctionnement. Il peut supprimer un serveur du pool
    - améliore la sécurité
  - Deux répartieurs pour éviter le "Point individuel de défaillance"