

Technologies du Web

Master COMASIC

Technologies côté serveur

Antoine Amarilli¹

3 octobre 2013

1. Matériel de cours inspiré de notes par Pierre Senellart. Merci à Pierre Senellart pour sa relecture.

Serveurs

Traiter la requête (cas simples) ou la rediriger à un autre programme (cas complexes).

Apache. Logiciel libre et gratuit. Lancé en 1995 !

IIS. Fourni avec Windows, propriétaire.

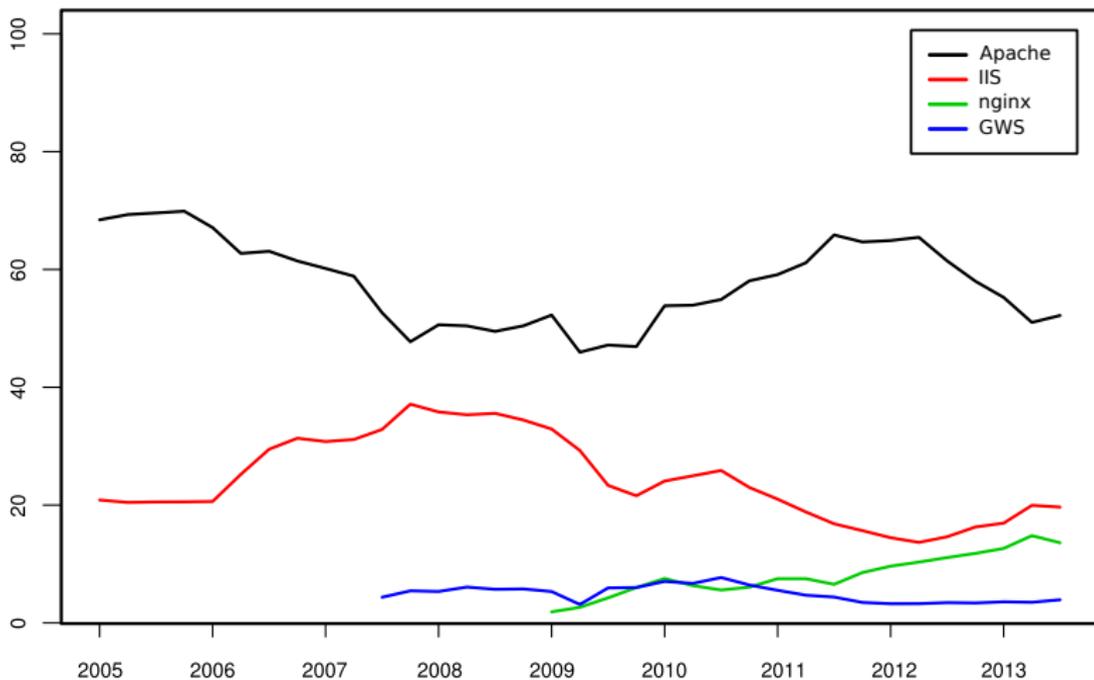
nginx. Haute performance, libre et gratuit, lancé en 2002.
(Cependant, en août 2013, lancement de Nginx Plus, payant.)

GWS. Google Web Server, interne. (Trafic de Google !)

lighttpd Alternative légère à Apache.

Autres Rares, expérimentaux, embarqués...

Parts de marché



Source : Netcraft.

Logs

- Souvent, le serveur Web loggue **toutes** les requêtes traitées !
- NCSA Common log format :
 - Adresse IP du client
 - Identité **ident** (obsolète)
 - Identité **authentification HTTP** (rarement utile)
 - **Date** et **heure** de fin de traitement
 - Première ligne de la **requête HTTP**
 - **Code d'état** retourné au client
 - **Taille** (octets) de la réponse
- Ajouts fréquents :
 - **User-Agent** pour identifier le client
 - **Referer** pour connaître la page précédente.

```
208.115.113.88 - - [22/Jan/2012:06:27:00 +0100]  
"GET /robots.txt HTTP/1.1" 404 266 "-"  
"Mozilla/5.0 (compatible; Ezooms/1.0; ezooms.bot@gmail.com)"
```

Utilisation des logs

- Pages les **plus visitées**.
- Chemin sur le site, temps passé sur les pages, entrées, sorties (mais avec les onglets ?)
- Distinguer **humains et robots** (pas de garanties !)
- Emplacement **géographique** des visiteurs
- **Mots-clés** saisis dans les moteurs de recherche
- **Liens** vers son site
- Nombre d'abonnés à un flux RSS dans User-Agent.
- **Spam** dans Referer
- Et plus ?
 - Logiciels pour exploiter directement les logs du serveur.
 - Tracking en PHP.
 - Tracking tiers en JavaScript (Google Analytics).
 - Tracking possible du pointeur de la souris en JavaScript, etc.

Sites Web statiques simples

- Les différentes ressources sont stockées dans des **fichiers** :
 - ⇒ /var/www/page.html, /var/www/style.css...
- Les pages sont organisées en une **arborescence** de répertoires.
- Les chemins demandés **correspondent** à l'arborescence :
 - ⇒ GET /a/b.html correspond à /var/www/a/b.html.
- Si un **répertoire** est demandé :
 - Servir **index.html** s'il existe.
 - Sinon, produire une liste des fichiers du répertoire.

⇒ **Pérennité** des URLs ?

Sites Web statiques simples

- Les différentes ressources sont stockées dans des **fichiers** :
⇒ /var/www/page.html, /var/www/style.css...
- Les pages sont organisées en une **arborescence** de répertoires.
- Les chemins demandés **correspondent** à l'arborescence :
⇒ GET /a/b.html correspond à /var/www/a/b.html.
- Si un **répertoire** est demandé :
 - Servir **index.html** s'il existe.
 - Sinon, produire une liste des fichiers du répertoire.

⇒ **Pérennité** des URLs ?

(Démonstration : naviguer dans une arborescence.)

Extensions Apache

- Les **fichiers .htaccess** permettent de paramétrer Apache :
 - `deny from all` pour interdire un répertoire.
 - **Authentication HTTP**
- **URL rewriting** :
 - `RewriteRule (*.png) /images/$1`
- **Server Side Includes** :
 - `<!--#include virtual="/footer.html" -->`

Table des matières

- 1 Serveurs Web
- 2 Langages côté serveur**
- 3 Bases de données
- 4 Frameworks
- 5 Aspects pratiques

CGI

- Moyen historique pour un **serveur Web** d'invoquer un **programme externe** (n'importe quel langage).
- Exécute le **programme** sur les paramètres de la requête.
- Le résultat de la requête est ce que **renvoie** par le programme.
- **Inconvénient** : créer un processus par requête est trop lourd.
 - ⇒ **FastCGI** et autres mécanismes.
 - ⇒ **Intégrer** le langage au serveur (par exemple PHP).

PHP

- Lancé en 1995 ; des centaines de millions de sites l'utilisent ².
- Langage de programmation complet.
- S'ajoute aux pages HTML. Exemple :

```
<ul>
  <?php
    $from = intval($_POST['from']);
    $to = intval($_POST['to']);
    for ($i = $from; $i < $to; $i++) {
      echo "<li>$i</li>";
    }
  ?>
</ul>
```

2. <http://www.php.net/usage.php>

Inconvénients de PHP

- Pas prévu comme un langage complet à l'origine.
 - ⇒ À l'origine, Personal Home Page.
 - ⇒ Difficile d'assurer la **compatibilité**...
- Promeut de mauvaises pratiques de **sécurité**.
 - ⇒ Historiquement, `$_POST['from']` accessible comme `$from!`
 - ⇒ Facile d'oublier de (re)valider les données utilisateur...
- Mauvaises **performances**?
 - ⇒ À l'origine, langage **interprété**.
 - ⇒ Cependant, compilation vers C++ (**HipHop**, pour Facebook).

Autres langages côté serveur

ASP Microsoft, add-on à IIS, propriétaire.

ASP.NET Microsoft, successeur d'ASP pour .NET.

ColdFusion Adobe, commercial, propriétaire.

JSP Intégration entre Java et un serveur Web.

Servlet classe Java suivant une API

Web container serveur pour gérer les servlets
(par exemple Apache Tomcat)

node.js Moteur JavaScript V8 (de Chrome) et serveur Web.

⇒ Même langage pour le client et le serveur.

⇒ Orienté événements, et VM performante.

Python Frameworks Web : **Django**, CherryPy, Flask.

Ruby Frameworks Web : **Ruby on Rails** (influent !), Sinatra.

Table des matières

- 1 Serveurs Web
- 2 Langages côté serveur
- 3 Bases de données**
- 4 Frameworks
- 5 Aspects pratiques

Bases de données

- **SGBD** : Système de Gestion de Bases de Données
- **Abstraction** de la tâche de **stocker** de l'information, la **mettre à jour**, et la récupérer suivant des **requêtes**.
- Modèle historique et le plus courant : **modèle relationnel**, **SQL**.
- **Avantages** :
 - Gérer **simplement** les données sans s'occuper du stockage
 - Remplacer plutôt **facilement** un SGBD par un autre
 - Facilement déléguer la tâche à des machines **séparées**
 - **Optimiser** les SGBD plutôt que les usages ad hoc
- Déviations du modèle relationnel : **NoSQL**.

Modèle relationnel : structure

- **Relations** (tables), par exemple Clients, Produits...
- **Attributs** (colonnes), par exemple Prénom, Identifiant...
- **Enregistrements** (lignes).
- **Exemple** : table clients :

id	prenom	nom	cp	ville	conseiller
1	Jean	Dupont	75013	Paris	42
2	François	Pignon	75005	Paris	42

- Les attributs ont un **type** (chaîne de caractères, nombre entier, décimal, date, blob binaire...)

SQL

- Structured Query Language
- Sémantique et modèle propre, liens avec la **logique**.
- Standardisé (1986, mis à jour en **2011**) mais extensions...
- **Sélection** :

```
SELECT prenom, nom, COUNT(*)  
FROM clients  
WHERE conseiller=42  
GROUP BY ville;
```

- **Mise à jour** :

```
UPDATE clients  
SET conseiller=43  
WHERE conseiller=42 AND cp=75013;
```

SGBDs

Oracle Commercial et propriétaire. Lancé en 1979.

IBM DB2 IBM, commercial et propriétaire. Lancé en 1983.

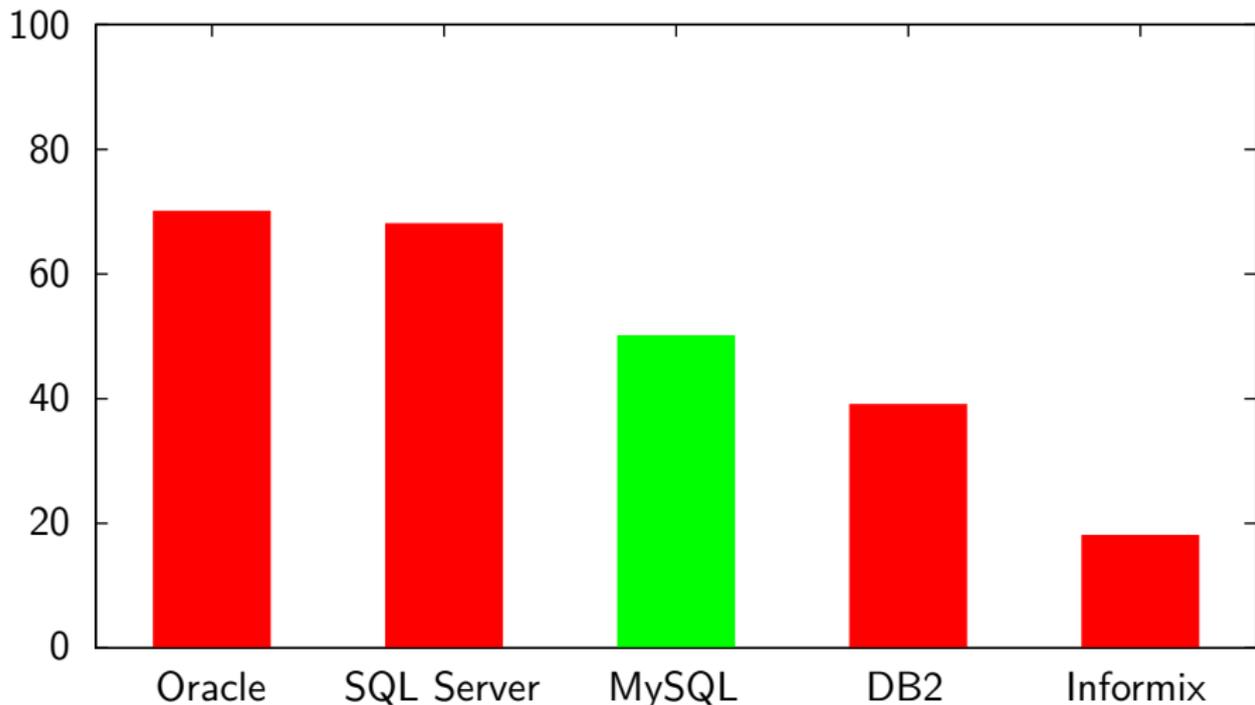
IBM Informix Autre système IBM. Lancé en 1981 (rachat).

SQL Server Microsoft, commercial et propriétaire. Lancé en 1989.

MySQL Libre (mais éditions commerciales), le plus courant sur le Web. Lancé en 1995.

PostgreSQL Libre, concurrent de MySQL. Lancé en 1995.

SGBDs, parts de marché en entreprise (cumulatives)



Source : étude Gartner de 2008 citée dans
<https://www.mysql.com/why-mysql/marketshare/>

Modèle relationnel : idées importantes

- **Contraintes d'intégrité** :
 - Clés **primaires** : identifiant unique, détermine l'enregistrement.
 - Clés **étrangères** : valeur qui fait référence à une clé primaire.
- **Formes normales** : éviter la répétition des données.
- **Procédures stockées** : code pour des requêtes complexes
- **Vues** : table virtuelle définie par une requête. (Matérialisée ?)
- **Triggers** : répondre à un événement
- **Transactions** : atomicité pour éviter les inconsistances.
- **Distribution** : répartir entre plusieurs machines
- **Concurrence** : traiter des requêtes simultanées
- **Performance** : index, caches mémoire, etc.
- **Utilisateurs et droits d'accès** (toujours relégué à PHP ou au framework Web en pratique).

NoSQL

- Beaucoup d'**expressivité** avec le modèle relationnel.
 - Structure très **contrainte** et peu modifiable.
 - Exigences fortes de **cohérence** sur du relationnel distribué.
- ⇒ **NoSQL** : renoncer au modèle relationnel pour de meilleures performances et un meilleur passage à l'échelle.
- Quelques exemples :
 - Stockage de **documents** entiers peu structurés.
 - Stockage de **graphes**.
 - Stockage de couples **clé-valeur**.
 - Stockage de **triplets** pour le Web sémantique.
 - Système le plus populaire³ : **MongoDB**, utilisé par Craigslist, le CERN, Shutterfly, Foursquare, etc.

3. <http://db-engines.com/en/ranking>

Injections SQL

- Une application serveur fabrique souvent des requêtes à partir de **données utilisateur** :

```
SELECT nom, telephone FROM employes WHERE nom='$nom' ;
```

- Si l'utilisateur soumet toto, on veut exécuter la requête :

```
SELECT nom, telephone FROM employes WHERE nom='toto' ;
```

- Maintenant, imaginons qu'un utilisateur soumette :

```
toto' UNION ALL SELECT nom, mdp FROM employes; --
```

- La base de données va recevoir et exécuter :

```
SELECT nom, telephone FROM employes WHERE nom='toto'  
UNION ALL SELECT nom, mdp FROM employes ;  
-- ' ;
```

⇒ Il faut toujours **échapper** ou **retirer** les caractères dangereux.

Table des matières

- 1 Serveurs Web
- 2 Langages côté serveur
- 3 Bases de données
- 4 Frameworks**
- 5 Aspects pratiques

Frameworks

- Ensemble de **fonctions** et d'**outils**, organisé autour d'un **langage**, pour les applications Web.
- Intégration AJAX, production de code JavaScript...
- **MVC** :
 - Modèle** La **structure** des données de l'application et les fonctions pour les manipuler.
 - Vue** La **présentation** des données destinée au client.
 - Contrôleur** Le **contrôle** de l'interaction avec les données du modèle à travers la vue.

ORM

- Object Relational Mapping.
- Les frameworks Web utilisent souvent la programmation **objet**.
- **Persistence** des objets dans une base de données relationnelle.
- Ne concerne pas les **méthodes** !
- Historiquement : tentatives de bases de donnée objet. Peu de succès.

Exemple d'un ORM (Django)

Définir une **classe** (table) avec des **attributs** :

```
from django.db import models
class Blog(models.Model):
    name = models.CharField(max_length=100)
    tagline = models.TextField()
```

Créer un objet (insérer un enregistrement) :

```
b = Blog(name='Beatles', tagline='Latest Beatles news.')
b.save()
```

Récupérer un objet (faire une requête) :

```
b = Blog.objects.filter(name='Beatles')
```

Templates

- Modèles avec **champs** de Pages HTML.
- **Exemple** (avec Jinja2) :

```
<h1>Résultats pour "{{ recherche }}"</h1>
<ul>
  {% for objet in resultats %}
    <li>
      <a href="details/{{ objet.id }}">
        {{ objet.nom }}
      </a>
    </li>
  {% endfor %}
</ul>
```

ROUTAGE DES URLS

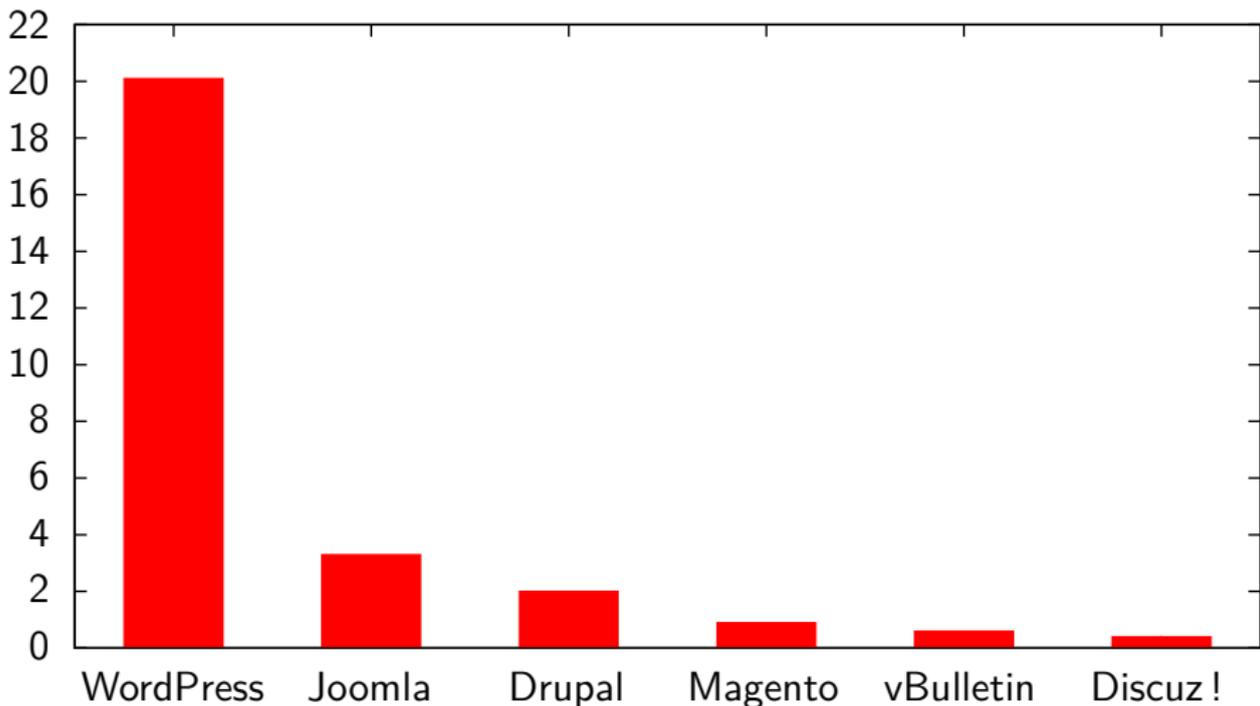
- Router suivant le **chemin** et la **méthode**.
- **Exemple** (avec Flask) :

```
@app.route('/')  
def index():  
    pass # Préparer la page d'accueil  
  
@app.route('/message/<int:message_id>')  
def message(message_id):  
    pass # Préparer l'affichage du message <message_id>  
  
@app.route('/upload', methods=['POST'])  
def upload():  
    pass # Traiter un upload
```

CMS

- Content Management System
- Faire un site **sans programmation** :
 - Édition de page avec un texte riche, ou langages simplifiés (Markdown, Textile, BBCode...).
 - Hébergement d'images, vidéos, etc.
 - Gestion d'utilisateurs.
 - Choix du thème graphique.
- Différents **types** :
 - Wikis** (contrôle de version, massivement éditable) : MediaWiki, MoinMoin, PmWiki...
 - Forums** phpBB, PunBB, Phorum, vBulletin...
 - Blogs** WordPress, Movable Type, Drupal, Blogger...
 - QA** (comme StackOverflow) : Shapado, OSQA (abandonné), AskBot.
 - Commerce** Magento, PrestaShop...

Parts de marché



Proportions des sites utilisant chaque CMS. (Tous en PHP !)

Source : http://w3techs.com/technologies/overview/content_management/all

Table des matières

- 1 Serveurs Web
- 2 Langages côté serveur
- 3 Bases de données
- 4 Frameworks
- 5 Aspects pratiques**

Infrastructure d'hébergement

- **Serveur** : machine qui reste toujours allumée pour répondre aux requêtes.
- **Datacenter** : local pour serveurs avec une bonne connexion, alimentation électrique, climatisation, sécurité physique...
- **Serveurs virtuels** : machine virtualisée qui imite une machine physique
- **Cloud** : location simple à grande échelle de machines
 - ⇒ Possibilité d'ajuster le nombre de machines selon la charge.
- **Content delivery network** : services de proxy intermédiaire.
- **Répartition de charge** entre plusieurs machines.
 - ⇒ Au niveau DNS : géographique et round-robin
 - ⇒ Au niveau logiciel

Comment on se fait héberger un site Web

- Les fournisseurs d'accès proposent souvent un hébergement.
- Souvent, support **PHP** et **MySQL** (et contenu statique).
- Espace disque limité.
- Généralement, on utilise un **client FTP** pour envoyer les pages PHP et le contenu statique aux serveurs du FAI.
- PHP souvent limité pour éviter les abus : pas de requêtes, envoi de courriels limité, temps de calcul et mémoire limités...
- En pratique, solution **peu pérenne** pour les services atypiques ou excessivement gourmands.

Comment on héberge un site Web soi-même

- Louer un **nom de domaine** (environ 10 euros par an).
- Obtenir une **machine**. Plusieurs choix :
 - Machine personnelle : de préférence IP fixe et bon upload.
 - ⇒ Abonnement Internet personnel
 - ⇒ Consommation électrique : 100 watts = 10 euros par mois
 - Virtual private server
 - ⇒ quelques euros par mois.
 - Dédié...
 - ⇒ quelques euros par mois aussi...
 - Machine personnelle mais en louant bande passante et espace dans une baie chez un datacenter.
- Configurer **SSH** pour se connecter à la machine, l'administrer et y transférer des fichiers.
- Installer un **serveur Web**.
- Installer éventuellement un **CMS**...

Identifier les technologies serveur

Sur le client, le code HTML, JavaScript, CSS est **accessible** (modulo minification et obfuscation). Sur le serveur, on n'a rien !

- **whois** : base de données fournissant (souvent) des infos sur le propriétaire d'un nom de domaine.
- Localisation **géographique** des IP des serveurs.
- **traceroute**, pour connaître le **chemin réseau** vers un hôte.
- Outils de scan (**nmap**) pour localiser les machines et identifier le système d'exploitation (fingerprinting).
- Header **Server**, possiblement faux.
- Forme des **identifiants de session**, cookies...
- Chemins d'accès et **extensions** : **.php**, **.asp**, ...
- **Commentaires** dans le code HTML.

⇒ Utilisé par les **pirates** pour identifier des services vulnérables.

Un exemple concret : Wikimedia

- wikipedia.org, **6e site** le plus visité au monde.⁴
- Organisation **caritative**, **142 employés** en septembre 2012.
- **75 000 utilisateurs** actifs,⁵ **10 millions** d'éditations par mois.
- **25 millions** de dollars de revenu en 2011 (surtout des dons).
- Coûts techniques en 2011 : quelques **millions** de dollars.
- À peu près **un millier** de serveurs au total.
- À peu près **20 milliards** de pages vues par mois :
 - **3 milliards** sur mobile
 - **10 milliards** seulement pour en.wikipedia.org
 - **1 milliard** pour fr.wikipedia.org
 - **8 000** par seconde en moyenne mais pointes à **50 000**.⁶
- **500 millions** de visiteurs uniques par mois.

4. <http://www.alexa.com/topsites>

5. <http://reportcard.wmflabs.org/>

6. <http://arstechnica.com/information-technology/2008/10/wikipedia-adopts-ubuntu-for-its-server-infrastructure/>

Infrastructure générale

- **Data centers** :
 - ⇒ Site principal : **Ashburn**, Virginia (Equinix).
 - ⇒ Pour l'Europe (load balancing DNS), **Amsterdam** (EvoSwitch).
 - ⇒ Anciens caches : **Séoul**, **Paris**.
 - ⇒ Failover sur l'ancien datacenter : **Tampa**, Floride (Hostway).
(Fermeture prévue.)
- Serveurs **Dell** sous **Ubuntu**.
- **puppet** pour gérer la configuration des serveurs.
- Logiciels de monitoring : Ganglia, Icinga, Nagios.
 - ⇒ `ganglia.wikimedia.org`
 - ⇒ `icinga.wikimedia.org`

Tâches principales

- Logiciel de gestion du wiki : **MediaWiki**, en PHP.
- Serveur **Apache**, HipHop envisagé pour PHP.
 - ⇒ 192 machines (à Ashburn)
- Base de données : **MariaDB** (fork de MySQL).
 - ⇒ 54 machines pour la base de données
 - ⇒ 10 machines de stockage avec 12 disques de 2 TB en RAID10
- Stockage de fichiers distribué : **Ceph** (anciennement **Swift**)
 - ⇒ 12 serveurs.
- Serveurs pour les travaux asynchrones (base de données NoSQL **Redis**)
 - ⇒ 16 serveurs

Caches

- Squid

 - ⇒ 8 machines pour le multimédia

 - ⇒ 32 machines pour le texte

- Varnish

 - ⇒ 8 machines

- Invalidation du cache avec MediaWiki

- Memcached entre MediaWiki et la base de données

 - ⇒ 16 machines

⇒ 90% du trafic n'utilise **que le cache** et non Apache.⁷

7. <https://blog.wikimedia.org/2013/01/19/wikimedia-sites-move-to-primary-data-center-in-ashburn-virginia/>

Autres services

- Proxies de terminaison SSL avec **nginx** :
⇒ 9 machines
- **Load balancing** avec LVS (Linux Virtual Server) :
⇒ 6 serveurs.
- **Indexation** pour la fonction de recherche :
 - Lucene 25 serveurs
 - Solr 3 serveurs
- **Redimensionnement** de fichiers multimédia :
 - Images 8 serveurs
 - Vidéos 2 serveurs
- **Statistiques** : 27 serveurs.
- **Traitement des paiements en ligne** : 4 serveurs.
- Serveurs DNS, services divers, snapshots, etc.

Exemple de rack

