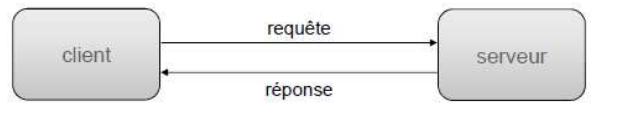
**Réseaux client / serveur**

Client, serveur, …

L’environnement client/serveur désigne une communication à travers un réseau entre deux parties applicatives (logicielles) : typiquement, d’un côté le client envoie des requêtes vers le serveur, de l’autre côté, le serveur attend les requêtes et y répond. Un serveur peut généralement répondre à un grand nombre de clients.

**Cas typique : Appel de procédure distante (RPC) :**

* **Requête** : spécification du service requis, paramètres d’appel
* **Réponse** : résultats, indicateurs éventuels d’exécution ou d’erreurs

**Avantages :**

L’ensemble des services et des données étant centralisées sur une même machine, cela facilite l’administration, la mise à jour des programmes et des données, les contrôles de sécurité, etc.

Cette centralisation peut avoir d’autres avantages : par exemple favoriser l’indexation et la recherche d’informations sur le Web (moteurs de recherches)

**Inconvénients** :

Si le serveur n’est plus disponible, les clients ne fonctionnent plus (intérêt de la redondance de services, de données …)

Le serveur supporte toute la charge de travail ; il peut saturer si trop de clients se connectent en même temps

Les clients ne communiquent pas entre eux (asymétrie de l’information au profit du serveur)

**Remarque** : le Peer To Peer est un cas particulier de client/serveur puisque chaque machine joue à la fois le rôle de client et de serveur (décentralisation des données, charge de travail répartie …)

Serveurs Web, de messagerie, ...

**Serveur de messagerie :**

Communique avec un client de messagerie (MUA : Mail User Agent) ou un autre serveur de messagerie (MTA : Mail Transport Agent) via le protocole SMTP (on parle ainsi de serveur SMTP). Classiquement, le MUA envoie un courrier à son MTA qui le transfert vers le MTA du destinataire. Le MTA du destinataire délivre alors le courrier au serveur de courrier électronique entrant (MDA : Mail Delivery Agent), qui stocke alors le courrier en attendant que l'utilisateur vienne le relever. Selon le protocole utilisé, on parle alors de serveur POP ou de serveur IMAP.

**Serveur de noms (annuaire des services)** :

permet de résoudre les correspondances adresses IP nom de domaine. Système de serveurs distribués. Organisation hiérarchique. Utilisation du protocole DNS.

**Serveur d’applications** :

Logiciel proposant un environnement d’exécution pour des composants applicatifs (EJB, JSP, Servlets …, composants ASP.NET …).

Exemples : WebSphere/JBoss/Glasfish (J2EE), Microsoft .Net …. Ils intègrent généralement un serveur HTTP.

Architectures client/serveur

**Architecture à 2 niveaux :** architecture client/serveur classique ; le client émet des requêtes vers un serveur qui renvoie des réponses ou des ressources.

Exemple : navigateur / serveur Web

**Architecture à 3 niveaux (trois tiers)** : le client est chargé de la présentation (interface graphique) ; le serveur d’application (middleware) est chargé des traitements et fait appel à un serveur de données ; le serveur de données gère les données (stockage, restitution, contrôle).

Exemple : navigateur / serveur Web / SGBD

**Architecture à N niveaux :** on peut généraliser l’architecture 3-tiers à N niveaux en spécialisant les serveurs dans des tâches de plus en plus précises : un serveur peut utiliser les services d’un ou plusieurs autres serveurs.

Exemple : le Webmail, un client de messagerie qui s’exécute sur un serveur Web

**Architecture en Peer to Peer :** chaque ordinateur du réseau est susceptible de jouer tour à tour le rôle de client et de serveur.

Exemple : les outils d’échange de ressources distribués sur internet (eMule, BitTorrent …)

Répartition de la charge

Un ***client lourd*** est un client qui doit effectuer en local une partie importante des traitements.

**Avantages** :

* Décharger le serveur d’autant de traitements sur l’ensemble des clients
* Applications théoriquement moins consommatrices de réseau

**Inconvénients** :

* Solutions plus coûteuses au niveau du déploiement, de la maintenance, de l’évolution de l’application et de la formation des utilisateurs

Un ***client léger*** est un client qui ne fait que formuler les demandes et les présenter à l’utilisateur, les traitements étant entièrement effectués par le serveur.

**Avantage :**

* L’évolution concerne le serveur uniquement, on évite ainsi N mises à jour sur les clients

**Inconvénients** :

* On n’exploite pas le parallélisme potentiel client/serveur : le serveur supporte à lui seul toute la charge de travail
* Encombrement réseau généralement plus important
* Interaction homme/machine limitée par nature

Un ***client riche*** est généralement utilisé pour évoquer un client Web dont les caractéristiques deviennent similaires aux logiciels traditionnels (interactivité, vitesse d’exécution).

Un client riche cherche à combiner les avantages du client lourd et du client léger.

Schéma du Gartner Group

Le Gartner Group propose une classification selon la manière de répartir les fonctions de présentation (IHM), de traitement (fonctions, calculs) et de gestion des données (stockage, restitution, contrôle) entre le client et le serveur

* **Client/serveur de présentation :** le client est dédié à la présentation des données et à l’interaction avec l’utilisateur
  + **Avantage :** séparation nette entre la logique de présentation et l’IHM proprement dite
  + **Inconvénient :** trafic réseau important généré par le protocole X
* **Client/serveur de traitement** : les traitements sont distribués entre le client et le serveur
  + **Avantages :**
    - Permet de bien répartir la charge de travail entre client et serveur
    - Généralement, trafic réseau moins important
  + **Inconvénient :** il n’offre pas la même modularité que le client/serveur de données (code généralement plus difficile à maintenir)
* **Client/serveur de données :** le client contient les traitements tandis que le serveur (SGBD) est dédié à la gestion et au contrôle de l’intégrité des données.
  + **Avantage :** indépendance entre traitements et données
  + **Inconvénient :** grande charge de travail pour le client
* **Client/serveur de données distribuées :** une partie des données est contenue localement, une autre à distance (décentralisation des données).
  + **Avantage :** augmentation de la capacité de stockage
  + **Inconvénient :** disponibilité et cohérence des données sur N sites difficile à gérer (1 site en panne peut éventuellement rendre le tout indisponible)

Problématique des serveurs

Problèmes majeurs d’un serveur : la montée en charge, la tolérance aux pannes et la sécurisation des accès

* **Montée en charge**

DNS : le serveur DNS peut servir de répartiteur de charge : sur le serveur DNS on associe un nom de domaine à plusieurs adresses IP puis un processus démon nommé ‘bind’ utilise la méthode du Round Robin pour choisir le destinataire de la requête

* **Tolérance aux pannes (haute disponibilité)**

**Protection des données** : codes détecteurs et correcteurs d’erreurs : bit de parité, code de Hamming, Contrôle de Redondance Cyclique …

**Surveillance applicative et matérielle :**

* Pannes applicatives : Lidirectord (associé à LVS) : surveillance applicative de chaque serveur (envoi de requêtes et attente de réponses) et modification en temps réel des règles de redirection.
* Pannes de machines : Heartbeat : un serveur maître et un serveur esclave. Battements de cœur du maître reçus à intervalle régulier. Si le maître tombe en panne, l’esclave prend le relai (basculement de l’adresse IP).

**La technologie RAID**

RAID (Redundant Array of Independent Disks) : Groupement redondant de disques Indépendants

RAID permet de constituer une unité de stockage à partir de plusieurs disques durs (grappe de disques)

**Niveaux RAID**

* RAID 0
  + But : Faire travailler N disque durs en parallèle
  + Comment : Ecriture d’un fichier répartie sur N disques
  + Avantages/Inconvénients :
    - Plus rapide : écriture simultanée
    - Peu fiable : pas de redondance d’informations, ni de mécanismes de correction d’erreurs
  + Bonne capacité : Celle du plus petit élément de la grappe x N
* RAID 1
  + But : Utiliser N disque de manière redondante (avec N>=2)
  + Comment : Les mêmes données sont écrites N fois (miroir)
  + Avantages/Inconvénients
    - Bonne fiabilité : accepte une défaillance de N-1 éléments
    - Faible capacité : celle du plus petit élément de la grappe
    - Lenteur due à l’écriture redondante
* RAID 3 & 4
  + But : Parallélisme et correction d’erreurs
  + Comment : Utilisation de N-1 disques pour répartir les données (comme RAID 0) et d’1 disque pour la parité
  + Avantages/Inconvénients
    - Fiabilité moyenne
      * - Si un disque de données tombe en panne, on utilise les autres disques de données + le disque de parité
      * - Si le disque de parité tombe en panne, il peut lui même être reconstruit à partir des autres disques
      * - Si plus d’un disque tombe en panne, les données sont perdues
    - Capacité, fiabilité et rapidité acceptable
* RAID 5
  + Utilisation de N disques pour répartir à la fois les données et la parité. C’est une variante de RAID 3 (qui évite le goulot d’étranglement sur un disque dû à l’écriture de la parité), de même fiabilité (pas plus d’un disque en panne)

Combinaison des RAID : RAID 10, 15, 50 … Le premier chiffre indique le niveau de RAID des grappes et le deuxième le niveau de RAID global

**La sécurisation des accès**

* **Firewall**

Un Firewall filtre le trafic en autorisant uniquement les accès permis par l’administrateur : filtrage des ports, des protocoles, des adresses IP. Exemple de mise à jour du fichier /sbin/iptables (Firewall offert par le noyau Linux )

* **Outils de détection d’intrusions (IDS) et de prévention d’intrusions (IPS)**

Les Firewalls permettent de bannir certains utilisateurs, de bloquer certains ports, mais qu’en est-il des ports ouverts, des attaques en cours ? Il existe des outils de détection et de prévention d’intrusions.