Propriétés et architectures des systèmes repartis

Université Virtuelle de Côte d'Ivoire

Table des matières

Objectifs

À la fin de cette leçon, l'apprenant sera capable de :

- Décrire les propriétés des systèmes repartis;
- Identifier les architectures de systèmes repartis

Les propriétés des systèmes repartis

1. Tolérance aux pannes

Un système réparti doit pouvoir tolérer la panne des machines :

- Une machine tombe en panne.
- Une machine envoie des informations erronées.
- Une machine n'est plus atteignable (problème réseau)

2. Hétérogénéité

Comment s'affranchir des différences matérielles et logicielles des ressources du système ?

L'utilisateur n'a pas à se soucier des différences matérielles ou logicielles des ressources qu'il utilise. Notion d'interopérabilité.

Sources d'hétérogénéité :

- machines (architecture matérielle)
- systèmes d'exploitation
- langages de programmation
- protocoles de communications

3. Sécurité

- Confidentialité
- Intégrité : Droits d'accès, Pare-Feu
- Authentification
    - Identification des applications partenaires
    - Non-répudiation (s'assurer qu'un contrat signé via internet, ne peut être remis en cause par l'une des parties) Messages authentifiés.
- Combien de personnes utilisent l'application, qui sont-ils ?
    - Nécessité de se protéger contre les intrusions

- Nécessité de stocker les accès des clients dans des fichiers journaux

4. Disponibilité

- Prêt à l'utilisation et toujours disponible.
- Continuer un service globalement même dégradé

5. La flexibilité

La structure du système doit faciliter son évolution. Pour cela, le noyau doit être le plus petit possible et assurer quatre services minimum :

- les mécanismes de communication interprocessus
- la gestion de la mémoire
- la gestion et l'ordonnancement des processus
- la gestion des entrées / sorties de bas niveau

Le noyau ne fournit ni système de fichiers, ni les répertoires. Les services que le noyau ne fournit pas sont implantés par des serveurs de niveau utilisateurs. Pour obtenir un service, un processus envoie un message au serveur approprié. Il existe une interface bien définie pour chaque service. On peut implémenter et installer des nouveaux services sans avoir à arrêter le système ou le réinitialiser avec un nouveau noyau. Le système est donc très souple.

6. La Scalability (passage à l'échelle réelle)

Le passage à l'échelle réelle ou scalability concernant au moins trois aspects :

- Nombre d'utilisateurs et/ou de processus (taille) ;
- Distance maximale entre les nœuds (géographique) ;
- Nombre de domaines administratifs (administratifs)

7. Transparence

Propriété fondamentale : Tout cacher à l'utilisateur :

- La répartition doit être non perceptible : une ressource distante est accédée comme une ressource locale
- Cacher l'architecture et le fonctionnement du système réparti

L'ISO définit plusieurs transparences : `accès, localisation, concurrence, réplication, mobilité, panne, performance, échelle` :

- *Transparence d'accès*
  - L'utilisateur accède à une ressource locale ou distante de façon identique
  - Accès à des ressources distantes aussi facilement que localement
  - Accès aux données indépendamment de leur format de représentation

- *Transparence de localisation* : Accès aux éléments/ressources indépendamment de leur localisation. L'utilisateur ignore la situation géographique des ressources. Transparence à la migration
- *Transparence de concurrence* : Exécution possible de plusieurs processus en parallèle avec utilisation de ressources partagées
- *Transparence de réplication* : Possibilité de dupliquer certains éléments/ressources pour augmenter la fiabilité
- *Transparence de mobilité* : Possibilité de déplacer des éléments/ressources
- *Transparence de panne* : Doit supporter qu'un ou plusieurs éléments tombent en panne
- *Transparence de performance* : Possibilité de reconfigurer le système pour en augmenter les performances
- *Transparence d'échelle* : Doit supporter l'augmentation de la taille du système (nombre d'éléments, de ressources ...)

Exercice : Activité d'auto-évaluation No 1

II

[solution n°1 p.18*]*

Exercice

Les propriétés des systèmes repartis sont :

- ☐ L'amabilité
- ☐ La jointure
- ☐ La transparence
- ☐ La continuité
- ☐ La flexibilité
- ☐ La scalabilité
- ☐ La vélocité
- ☐ La disponibilité

Exercice

La transparence d'un système reparti consiste à :

- ☐ Faire en sorte que l'utilisateur voit le fonctionnement du système par le biais de son architecture
- ☐ Cacher l'architecture et le fonctionnement du système aux utilisateurs
- ☐ Faire en sorte que l'utilisateur voit l'architecture du système par le biais de son fonctionnement
- ☐

Faire en sorte que l'utilisateur accède à une ressource distante comme s'il accédait à une ressource locale

- ☐ Laisser voir le fonctionnement et l'architecture du système aux utilisateurs

Exercice

Les différentes formes de transparence sont :

- ☐ la jointure
- ☐ l'accès
- ☐ la réplication
- ☐ la flexibilité
- ☐ la concurrence
- ☐ l'ouverture
- ☐ la disponibilité
- ☐ la localisation

Exercice

L'hétérogénéité d'un système consiste à :

- ○ augmenter la charge applicative du système
- ○ mettre en place un mécanisme permettant la transparence et la sécurité du système
- ○

s'affranchir des différences matérielles ou logicielles des ressources du système (rendre le système interopérable)

- ○ s'affranchir des pannes techniques du système

Architectures des systèmes repartis

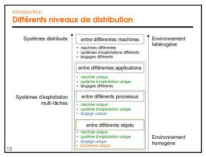
III

Les architectures des systèmes distribués varient en fonction de la forme ou du niveau de distribution.

>>les formes de distributions:

- *Distribution physique :*
  - fonctionnement en réseaux (LANs, WANs)
    - Machines parallèles homogènes : noeuds de calcul montés dans des « racks » reliés par un réseau d'interconnexion unique et très haute performance. *Ex.* : super-calculateurs, clusters ;
    - Machines parallèles hétérogènes, ordinateurs variés en termes de processeurs, de mémoire, de bande passante, etc. Généralement associés à une couche logicielle type middleware
  - architectures multi-processeurs
    - mémoire partagée avec accès direct pour tous les CPUs
    - mémoire cache pour assurer un accès rapide aux données et éviter les goulots d'étranglement entre les calculs
- *Distribution structurelle :*
  - programmation structurée
  - programmation objet
  - programmation par composants
  - programmation par aspects
- *Distribution fonctionnelle :*
  - décomposition en services indépendants (affichage, calcul,
  - stockage de données, etc.)
  - services Web

>>Les niveaux de distribution



>>*Modèle d'architecture logique*:

- L'architecture d'un système est sa structure en termes de composants spécifiquement séparés, conçue dans le but d'assurer un bon fonctionnement en fonction des demandes présentes mais aussi futures
- Le système doit être fiable, gérable, adaptable et tout cela à un coût réaliste. Un tel modèle est décrit par :
    - le placement des entités sur le réseau (distribution des données, charge effective)
    - les relations existantes entre ces entités (rôles fonctionnels, communications)
- Principales architectures
    - *client/serveur*
        - architectures 2-tiers
        - architectures 3-tiers
        - architectures n-tiers
    - *métacomputing*
        - P2P
        - clusters
        - grillles
        - agents autonome

Dans le cadre de cette section, nous allons nous focaliser sur les architectures logiques.

1. Le modèle client-serveur

Analogie : le modèle interviewer / interviewé

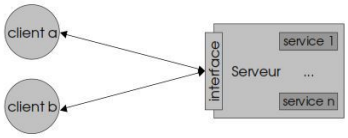
- communication entre 2 entités : toute communication concernant les éléments d'un groupe peut être décomposée en un ensemble d'échanges entre deux entités
- une entité a l'initiative du dialogue (l'interviewer) et l'autre est dans l'attente d'une requête (l'interviewé)
- l'entité interviewée est programmée pour répondre à un ensemble très précis de requêtes :
    - la liste des requêtes autorisées doit être parfaitement définie
    - cette liste est l'interface de l'entité interviewée
    - à chaque requête correspond un service

Le modèle client/serveur

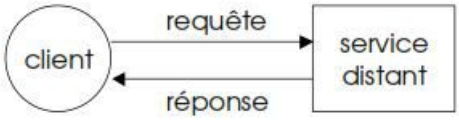
- l'interviewé fournit des services ? c'est le serveur
- l'interviewer requiert des services ? c'est le client

>>Fonctionnement

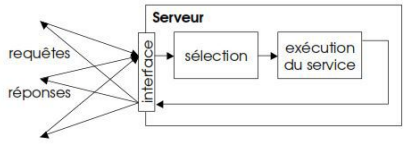
- le client émet un message contenant une requête à destination d'un serveur
- le serveur exécute le service associé à la requête émise par
- le client le serveur retourne au client un message contenant le résultat du service effectué



- *Du point de vue du client*



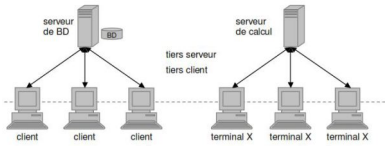
- *Du point de vue du serveur*
  - gestion des requêtes (priorité)
  - exécution du service (séquentiel, concurrent)
  - mémorisation ou non de l'état du client



Remarque : Le modèle client serveur

- la communication est toujours initiée par le client : le serveur est en mode réactif
- mode d'exécution synchrone
- chaque entité peut jouer les deux rôles (client et serveur)
- on ne se préoccupe pas de la manière dont la communication est réalisée :
  - on suppose seulement qu'il existe un moyen d'échanger des messages
  - la plupart des protocoles classiques (FTP, HTTP, etc.) sont basés sur ce principe
- l'interface du serveur est un élément essentiel de la communication : définit les requêtes autorisées
- concept simple mais base théorique pour des architectures beaucoup plus compliquée

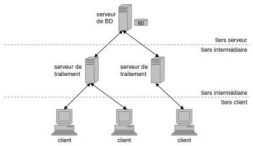
Architecture 2-tiers



Autres exemples de serveurs

- serveur de fichiers
- serveur de noms
- serveur Web
- serveur d'impression
- etc.

Architecture 3-tiers



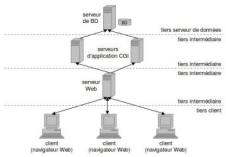
Apport des architecture 3-tiers

- *Performance* : traitement métier proche du tiers de données, quelque soit la localisation du tiers client
- *Scalabilité* : possibilité d'ajouter des tiers métiers pour satisfaire la demande de nouveaux clients
- *Modularité* : tiers client peut être développé indépendamment du tiers de données
- *cohérence et évolutivité*:
    - centralisation de la logique métier dans le tiers intermédiaire
    - évite les incohérences liées à des implantations différentes de la logique métier dans les différents clients
    - permet la prise en compte facile de changements de logique métier

Dans les architectures 3-tiers :

- tiers client = présentation de l'application
- tiers serveur = gestion des données
- tiers intermédiaire = ensemble de services fournis au client pour consulter/modifier/etc. les données
- peut être considéré comme un ensemble de tiers, chacun correspondant à un service particulier

Exemple d'architecture n-tiers : application Web



2. Modèles du meta-computing

Schéma opératoire

- connecter des ressources hétérogènes de manière à former une très grande capacité de calcul
- ces ressources peuvent être distribuées le long de très grands réseaux, voire sur l©Internet tout entier

Caractéristiques	P2P	Clusters	Grilles
Type de noeuds	PCs de bureau	PCs dédiés, racks	PCs de bureau, stations de travail, serveurs
Participants	Un, plusieurs	Plusieurs	Plusieurs
Gestion des utilisateurs	Centralisée, décentralisée	Centralisée	Décentralisée
Gestion des ressources	Distribuée	Centralisée	Distribuée
Allocation / Prévision	Décentralisée	Centralisée	Décentralisée
Taille	Millions	100s	Milliers à millions
Capacité en calcul	Haute, mais peut varier	Garantie	Haute, mais peut varier
Bande passante	Basse	Très haute	Basse

P2P et « Internet computing »

- *Peer to Peer (P2P)*
  - Peer = entité disposant de capacités similaires aux autres entités d'un système
  - outils de stockage et d'échange de données
- *Principes*
  - éviter les vulnérabilités du réseau par la répartition
  - découvrir les ressources par un procédé de diffusion : méta-description qui permettent de les retrouver par des moteurs de recherche
- *P2P « purs » et « hybrides »*
  - P2P « purs » = tous les noeuds participants sont des Peers et aucun serveur central n'est utilisé pour contrôler, coordonner ou gérer les échanges entre Peers
- P2P « hybrides » = serveur central qui effectue un certain nombre de fonctions essentielles

>>*Avantages du P2P* :

- réduction du coût global par répartition
- accroissement de la scalabilité et de la sûreté de fonctionnement : pas d'autorité centrale
- agrégation des ressources et interopérabilité des systèmes
- accroissement de l'autonomie : les utilisateurs ne dépendent pas d'un seul fournisseur centralisé
- anonymat et respect de la vie privée très grande dynamité
  - les ressources entrent et sortent du système de manière totalement continue

- la communication ne s'en trouve généralement pas affectée.

Internet Computing

- applications P2P parallélisables : une même tâche est effectuée sur chaque Peer avec différents paramètres
- utiliser les ressources inutilisées des noeuds
- si un noeud détecte une inactivité, il informe un client maître
- activité complètement transparente pour l'utilisateur

Exemples d'implantations

- P2P pur : KaZaa, FreeHeaven, JXTA, Gnutella, Freenet
- P2P hybride : Napster, bittorent
- Internet Computing : SETI@Home, RSA-135

Clusters

Ensemble d'ordinateurs complets (processeur, mémoire, périphériques d'E/S) en général interconnectés par l'intermédiaire d'un LAN

- utilisé comme une seule ressource de calcul unifiée
- les composants d©un cluster sont typiquement des stations de travail ou des PCs et sont généralement homogènes
- premier cluster « Beowulf » créé en 1994 pour la NASA

Architecture

- tiers d'accès = fournit les services d'accès et d'authentification
- tiers de gestion = responsable des fonctionnalités basiques du cluster (service de fichiers, gestion des sauvegardes, ...)
- tiers de calcul = fournit la puissance de calcul du cluster : travaux exécutés sur un ou plusieurs noeuds
- deux services critiques
    - Batch Queing System = reçoit les demandes de travaux
    - Job Scheduler = détermine l'ordre d'exécution des travaux en fonction de la charge processeur moyenne, la charge mémoire moyenne, ...

Grilles

partage de ressources et de puissance de calcul dans de très larges organisations multi-institutionnelles

- infrastructure capable d'unifier diverses ressources
- met en oeuvre un grand nombre de ressources hétérogènes

- peuvent s'accroître de quelques ressources à plusieurs millions
- probabilité de ressources défaillantes élevée

Architecture

- une fabrique de grille : toutes les ressources (PCs, SANs, clusters,...) distribués géographiquement
- un middleware de grille : offre les services de base (gestion des processus distants, co-allocation des ressources, accès au stockage, ...)
- un environnement de développement de grille : offre des services de très haut niveau permettant aux développeurs de développer des applications et des brokers agissant de manière globale
- des applications de grille et des portails de grille
- Ex. : Globus, Legion, CERN Data Grid ou Unicore

Agents autonomes (Agents mobiles)

- *Définition*
  Un agent autonome ou agent mobile est une entité matérielle ou logicielle dotée des propriétés suivantes :
  - *autonomie* : les agents agissent sans l'intervention directe d'humains ou d'autres agents, et ont le contrôle de leurs actions et de leur état interne ;
  - *capacité sociale* : les agents interagissent avec d'autres agents (et éventuellement des humains) grâce à des langages de communication agent ;
  - *réactivité :* les agents perçoivent leur environnement (le monde physique, un utilisateur via une interface graphique, un ensemble d'autres agents, Internet, une combinaison de tout cela), et répondent en permanence aux changements qui s'y produisent ;
  - *pro-activité*: les agents n'agissent pas seulement en réponse aux sollicitations de l'environnement mais peuvent exhiber un comportement orienté par un but en prenant l'initiative d'agir.
- *Caractéristiques*
  - à la frontière des systèmes distribués et de l'IA
  - inspiration sociologique

Exercice : Activité d'Autoévaluation No 2

IV

[solution n°2 p.19*]*

Exercice

Les architectures des systèmes distribués varient en fonction de :

- O du volume du trafique
- O de la densité du réseau
- O la forme ou du niveau de distribution
- O des règles de gestion

Exercice

Les formes de distribution sont :

- ☐ distribution structurelle
- ☐ distribution fonctionnelle
- ☐ distribution gaussienne
- ☐ distribution physique
- ☐ distribution uniforme

Exercice

Les niveaux de distribution sont :

- ☐ distribution WAN
- ☐ distribution entre machine
- ☐ distribution LAN
- ☐ distribution entre application
- ☐ distribution uniforme

- ☐ distribution entre processus
- ☐ distribution transversale
- ☐ distribution entre objet

Exercice

Les différentes architectures client/serveur sont :

- ☐ Architecture multi-serveurs
- ☐ Architecture 2-tiers
- ☐ architecture mono-serveur
- ☐ architecture 3-tiers
- ☐ architecture 1-tiers
- ☐ architecture n-tiers

Exercice

Les différentes architecture de systèmes repartis métacomputing sont :

- ☐ Backbone Network
- ☐ multi-processing
- ☐ P2P
- ☐ Agents Autonomes
- ☐ Agents invariants
- ☐ Grille
- ☐ n-tiers
- ☐ Cluster

Solutions des exercices

> **Solution** n°1

Exercice p. 7

Exercice

Les propriétés des systèmes repartis sont :

- ☐ L'amabilité
- ☐ La jointure
- ☑ La transparence
- ☐ La continuité
- ☑ La flexibilité
- ☑ La scalabilité
- ☐ La vélocité
- ☑ La disponibilité

Exercice

La transparence d'un système reparti consiste à :

- ☐ Faire en sorte que l'utilisateur voit le fonctionnement du système par le biais de son architecture
- ☑ Cacher l'architecture et le fonctionnement du système aux utilisateurs
- ☐ Faire en sorte que l'utilisateur voit l'architecture du système par le biais de son fonctionnement
- ☑

Faire en sorte que l'utilisateur accède à une ressource distante comme s'il accédait à une ressource locale

- ☐ Laisser voir le fonctionnement et l'architecture du système aux utilisateurs

Exercice

Les différentes formes de transparence sont :

- ☐ la jointure
- ☑ l'accès
- ☑ la réplication
- ☐ la flexibilité
- ☑ la concurrence
- ☐ l'ouverture
- ☐ la disponibilité
- ☑ la localisation

Exercice

L'hétérogénéité d'un système consiste à :

- ○ augmenter la charge applicative du système
- ○ mettre en place un mécanisme permettant la transparence et la sécurité du système
- ◉ s'affranchir des différences matérielles ou logicielles des ressources du système (rendre le système interopérable)
- ○ s'affranchir des pannes techniques du système

> **Solution** n°2

Exercice p. 16

Exercice

Les architectures des systèmes distribués varient en fonction de :

- ○ du volume du trafique
- ○ de la densité du réseau
- ◉ la forme ou du niveau de distribution
- ○ des règles de gestion

Exercice

Les formes de distribution sont :

- ☑ distribution structurelle

- [x] distribution fonctionnelle
- [ ] distribution gaussienne
- [x] distribution physique
- [ ] distribution uniforme

Exercice

Les niveaux de distribution sont :

- [ ] distribution WAN
- [x] distribution entre machine
- [ ] distribution LAN
- [x] distribution entre application
- [ ] distribution uniforme
- [x] distribution entre processus
- [ ] distribution transversale
- [x] distribution entre objet

Exercice

Les différentes architectures client/serveur sont :

- [ ] Architecture multi-serveurs
- [x] Architecture 2-tiers
- [ ] architecture mono-serveur
- [x] architecture 3-tiers
- [ ] architecture 1-tiers
- [x] architecture n-tiers

Exercice

Les différentes architecture de systèmes repartis métacomputing sont :

- [ ] Backbone Network
- [ ] multi-processing

- ☑ P2P
- ☑ Agents Autonomes
- ☐ Agents invariants
- ☑ Grille
- ☐ n-tiers
- ☑ Cluster