Chapitre 5 :

Introduction aux Systèmes Répartis

Systèmes répartis vs. ?

- Système distribué en opposition à système centralisé
- Système centralisé : tout est localisé sur la même machine et accessible par le programme
  - Système logiciel s'exécutant sur une seule machine
  - Accédant localement aux ressources nécessaires (données, code, périphériques, mémoire ...)
  - Possibilité de dater de manière non ambiguë les évènements

Définition

- Une définition parmi d'autres (Andrew Tanenbaum):

  Une collection d'ordinateurs indépendants qui apparaît à un utilisateur comme un système unique et cohérent.

  - Les machines sont autonomes
  - Elle sont reliées par un réseau de communication
  - Les utilisateurs ont l'impression d'utiliser un seul système

Visions d'un système distribué

- Vision matérielle d'un système distribué : architecture matérielle
  - Machine multi-processeurs avec mémoire partagée, CPU multicore
  - Cluster d'ordinateurs dédiés au calcul/traitement massif parallèle
  - Ordinateurs standards connectés en réseau
- Vision logicielle d'un système distribué
  - Système logiciel composé de plusieurs entités logicielles s'exécutant indépendamment et en parallèle
    - Réalisation d'une certaine tâche globale
- Dans ce cours :
  - Conception logicielle des systèmes distribués
  - Par défaut sur une architecture matérielle de type ordinateurs connectés en réseau

Pourquoi des systèmes répartis ?

- Adaptation de la structure d'un système à celle des applications
- Plus de capacités de stockage
- Plus de capacités de traitement
- Besoin de communication et de partage d'information
- Partage de ressources (programmes, données, services)
- Aspects économiques (rapport prix/performance)
- Réalisation de systèmes à grande capacité d'évolution (Extensibilité)
- Tolérance aux fautes (disponibilité, fiabilité…)

Systèmes distribués

- Les systèmes distribués sont dans un cadre différent par rapport aux systèmes centralisés
  - Concurrence
    - Les éléments formant le système s'exécutent en parallèle et de manière autonome
    - Pas d'état ou d'horloge globale commune
    - Non déterminisme : deux exécutions donnent des résultats différents
  - Communication est un point crucial
    - Potentiellement non fiable
    - Temps de communication non négligeables
  - Points de problèmes de fiabilité en nombre accru
    - Problème matériel d'une machine
    - Problème de communication via le réseau
    - Problème logiciel sur un des éléments du système

Exemples de systèmes répartis

- Serveur de fichiers
  - Physiquement : fichiers se trouvent uniquement sur le serveur
  - Virtuellement : accès à ces fichiers à partir de n'importe quelle machine cliente en faisant « croire » que ces fichiers sont stockés localement
- Intérêts
  - Accès aux fichiers à partir de n'importe quelle machine
  - Système de sauvegarde associé à ce serveur
  - Transparent pour l'utilisateur
- Inconvénients
  - Si réseau ou le serveur plante : plus d'accès aux fichiers pour personne

Exemples de systèmes répartis

- Autre exemple de système distribué : Web
  - Un serveur web auquel se connecte un nombre quelconque de navigateurs web (clients)
  - Accès à distance à de l'information
    - Accès simple
      - Serveur renvoie une page HTML statique qu'il stocke localement
    - Traitement plus complexe
      - Serveur interroge une base de données pour générer dynamiquement le contenu de la page
    - Transparent pour l'utilisateur : les informations s'affichent dans son navigateur quelque soit la façon dont le serveur les génère

Exemples de systèmes répartis

- Calculs scientifiques
  - Plusieurs architectures matérielles généralement utilisées
    - Ensemble de machines identiques reliées entre elles par un réseau dédié et très rapide (cluster)
    - Ensemble de machines hétérogènes connectées dans un réseau local ou bien encore par Internet (grille)
  - Principe général
    - Un (ou des) serveur distribue des calculs aux machines clients
    - Un client exécute son calcul puis renvoie le résultat au serveur
  - Avantage
    - Utilisation d'un maximum de ressources de calcul
  - Inconvénient
    - Si réseau ou serveur plante, arrête le système

Fonctions d'un système réparti

- Transparence
  - d'accès aux informations
  - de localisation des objets
  - …
- Gestion de la concurrence d'accès aux ressources
- Fiabilité
  - tolérance aux fautes
  - évitement de fautes
  - détection et recouvrement de fautes
- Hétérogénéité (du matériel et logiciel)
- Sécurité
- Flexibilité : facilité d'utilisation, de maintenance, de mise à jour...
- Performance (optimisation de la gestion des ressources)

Transparence

- Transparence
  - Fait pour une fonctionnalité, un élément d'être invisible ou caché à l'utilisateur ou à un autre élément formant le système distribué
- But est de cacher l'architecture, le fonctionnement de l'application ou du système distribué pour apparaître à l'utilisateur comme une application unique cohérente
- L'ISO définit plusieurs transparences (norme RM-ODP)
  - Accès, localisation, concurrence, réplication, mobilité, panne, performance, échelle

Transparences

- Transparence d'accès
  - 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
- 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

Transparences

- Transparence de mobilité
  - Possibilité de déplacer des éléments/ressources
- Transparence de panne
  - Doit supporter qu'un ou plusieurs éléments tombe 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 ...)
- Un système donné va offrir un certain nombre de transparences
  - Souvent au minimum transparences de localisation, d'accès et de concurrence

Concurrence d'accès aux ressources

- Système distribué = éclaté
  - Connaissance des éléments formant le système : besoin d'identification et de localisation
  - Gestion du déploiement et de la présence d'éléments essentiels
- Communication à distance est centrale
  - Techniques et protocoles de communication
  - Contraintes du réseau : fiabilité (perte de données) et temps de propagation (dépendant du type de réseau et de sa charge)
- Naturellement concurrent et parallèle
  - Chaque élément sur chaque machine est autonome
  - Besoin de synchronisation, coordination entre éléments distants et pour l'accès aux ressources (exclusion mutuelle ...)

Fiabilité

- Nombreux points de pannes ou de problèmes potentiels
  - Réseau
    - Une partie du réseau peut-être inaccessible
    - Les temps de communication peuvent varier considérablement selon la charge du réseau
    - Le réseau peut perdre des données transmises
  - Machine
    - Une ou plusieurs machines peut planter, engendrant une paralysie partielle ou totale du système
- Peut augmenter la fiabilité par redondance, duplication de certains éléments
  - Mais rend plus complexe la gestion du système
- Tolérance aux fautes
  - Capacité d'un système à gérer et résister à un ensemble de problèmes

Hétérogénéité

- Des machines utilisées (puissance, architecture matérielle...)
- Des systèmes d'exploitation tournant sur ces machines
- Des langages de programmation des éléments logiciels formant le système
- Des réseaux utilisés : impact sur performances, débit, disponibilité ...
  - Réseau local rapide
  - Internet
  - Réseaux sans fil

Hétérogénéité

- Exemple hétérogénéité des données : codage des entiers
  - Entier sur 32 bits (4 octets)
  - A partir de l'adresse de l'entier en mémoire, les 4 octets ne sont pas toujours placés dans le même ordre
- Exemple hétérogénéité des données : codage des chaînes de caractères
  - Principe courant : un tableau de char (octet) avec information sur la taille ou la fin de chaîne
    - En C : code ASCII de valeur 0 pour marquer la fin de la chaîne |B|o|n|j|o|u|r|\0|
    - En Pascal : le premier caractère est un nombre (codé via un code ASCII) précisant la longueur de la chaîne |\7|B|o|n|j|o|u|r|

Sécurité

- Nature d'un système distribué fait qu'il est beaucoup plus sujet à des attaques
  - Communication à travers le réseau peuvent être interceptées
  - On ne connaît pas toujours bien l'élément distant avec qui on communique
- Solutions
  - Connexion sécurisée par authentification avec les éléments distants
  - Cryptage des messages circulant sur le réseau

Algorithmique distribuée

- Développement d'algorithmes dédiés aux systèmes distribués en prenant en compte les spécificités de ces systèmes (Diapositive 6)
- On y retrouve notamment des adaptations de problèmes classiques en parallélisme
  - Exclusion mutuelle, élection ...
- Mais aussi des problèmes typiques des systèmes distribués
  - Horloge globale, état global, diffusion causale, consensus ...
- S'intéresse principalement à deux grandes familles de problèmes
  - Synchronisation et coordination entre processus distants
  - Entente sur valeurs communes et cohérence globale dans un contexte non fiable (crash de processus, perte de messages ...)

Algorithmique distribuée

- Les algorithmes distribués s'appuie sur des caractéristiques du système
  - Caractéristiques des éléments formant le système
    - Fiable, pouvant se planter, pouvant envoyer des messages erronés ...
  - Caractéristiques de la communication
    - Fiable ou non fiable, temps de propagation borné ou pas...
- Un algorithme distribué se base donc sur un modèle de système distribué qui caractérise
  - Les processus (éléments logiciels formant le système)
  - La communication
  - Les pannes

Différences entre modèles de systèmes en algorithmique parallèle et distribuée

- Parallèle : sur une même machine
  - Les processus ont accès à une mémoire locale commune
  - Les processus s'exécutent par rapport à la même horloge
- Distribué : sur un ensemble de machines
  - Pas de mémoire partagée commune
  - Pas d'horloge commune
  - Temps de propagation des messages entre processus distants est non nul

Processus

- Élément logiciel effectuant une tâche, un calcul
  - Exécution d'un ensemble d'instructions
  - Une instruction correspond à un événement local au processus
    - Dont les événements d'émission et de réception de messages
  - Les instructions sont généralement considérées comme atomiques
- Il possède une mémoire locale
- Il possède un état local
  - Ensemble de ses données et des valeurs de ses variables locales
- Il possède un identifiant qu'il connaît
  - Et connaît en général les identifiants des/d'autres processus
- Pas de connaissance sur l'état des autres processus
- Les processus d'un système s'exécutent en parallèle

Canal de communication

- Canal logique de communication point à point
  - Pour communication entre 2 processus
  - Transit de messages sur un canal
- Caractéristiques d'un canal
  - Unidirectionnel ou bidirectionnel
  - Fiable ou non : perd/modifie ou pas des messages
  - Ordre de réception par rapport à l'émission
    - Ex. : FIFO = les messages sont reçus dans l'ordre où ils sont émis
  - Synchrone ou asynchrone
    - Synchrone : l'émetteur et le récepteur se synchronisent pour réaliser l'émission et/ou la réception
    - Asynchrone : pas de synchronisation entre émetteur et récepteur
  - Taille des tampons de message cotés émetteur et récepteur
    - Limitée ou illimitée

Exemples de modèles de canaux

- Modèle généralement utilisé
  - Fiable, FIFO, tampon de taille illimitée, asynchrone (en émission et réception) et bidirectionnel
  - Variante courante avec réception synchrone
- Modèle des sockets TCP
  - Fiable
  - FIFO
  - Bidirectionnel
  - Synchrone pour la réception
    - On reçoit quand l'émetteur émet
    - Sauf si données non lues dans le tampon coté récepteur
  - Asynchrone en émission
    - Emetteur n'est pas bloqué quand il émet quoique fasse le récepteur

Système synchrone / asynchrone

- Un modèle synchrone est un modèle où les contraintes temporelles sont bornées
  - Un processus évoluera dans un temps borné
  - Un message arrivera en un certain délai
  - On connaît la limite de dérive des horloges locales
- Un modèle asynchrone n'offre aucune borne temporelle
  - Modèle bien plus contraignant et rendant impossible ou difficile la réalisation de certains algorithmes distribués
  - Exemple : on ne sait pas différencier en asynchrone
    - Le fait qu'un processus est lent ou est planté
    - Du fait qu'un message est long à transiter ou est perdu

Modèle de fautes

- Processus correct
  - Processus non planté, qui ne fait pas de fautes
- Trois grands types de fautes ou pannes
  - Franches : le processus ne fait plus rien
    - Fail-stop : les autres processus peuvent détecter que ce processus est planté
    - Crash : les autres processus ne peuvent pas détecter que le processus est planté
  - Par omission : le processus ou canal ne réalise pas une action
    - des messages sont perdus ou non délivrés
  - Arbitraires ou byzantines : fautes quelconques (calcul erroné ou non fait , message «créé» , message modifié, dupliqué voire supprimé)
    - Cas de fautes les plus complexes à gérer
    - Les autres fautes peuvent être considérées comme des cas particuliers des fautes byzantines
    - 2 catégories
      - Malicieuses : erreurs volontaires (virus, attaques ...)
      - Naturelles : problème non voulu, non déclenché