

I403C Systèmes d'exploitation avancé

Séance 6

Système d'exploitation distribué

Sébastien Combéfis

2018–2019

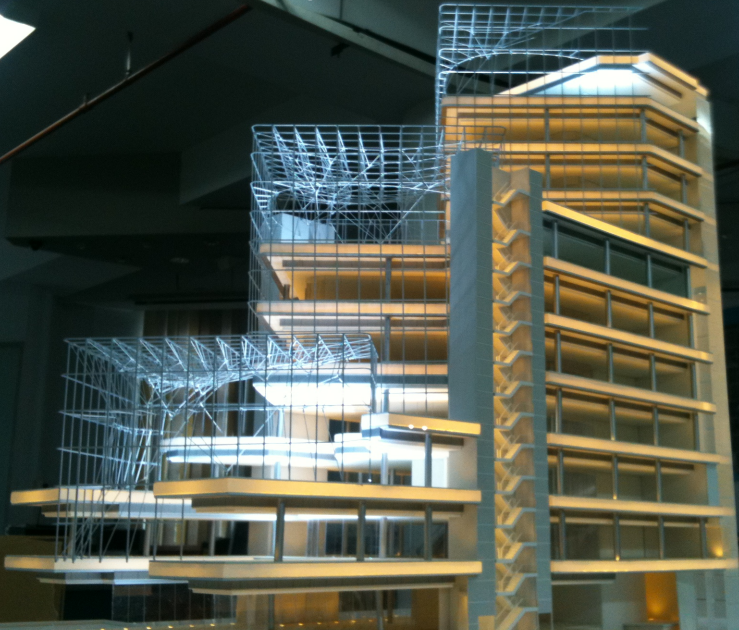


Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

Objectifs

- Comprendre ce qu'est un **OS distribué**
 - Système distribué et comparaison OS réseau/OS distribué
 - Principaux modèles de systèmes, problèmes et objectifs
- **Éléments de design** d'un OS distribué
 - Primitives de communication par `send/receive` ou RPC
 - Détails des points d'attention pour designer un OS distribué

Modèle de systèmes



OS distribué (1)

- **OS distribué** exécuté sur plusieurs machines
 - Permet de contrôler les ressources du système distribué
 - Fournit une interface d'utilisation facile aux utilisateurs
- Apparaît comme un **OS centralisé** pour ses utilisateurs

Mais tourne effectivement sur plusieurs processeurs indépendants
- Nécessité d'avoir une **transparence totale**

L'utilisateur voit un uniprocasseur virtuel

OS distribué (2)

- Plusieurs systèmes multimachines pas des **systèmes distribués**
 - L'ARPANET contient plusieurs ordinateurs, mais distinguables
 - Un réseau local n'est non plus pas un système distribué
- **Le software** détermine si un système est distribué ou non
 - User ne doit pas savoir (et s'en fout) sur quelle machine il est
 - Où son software est exécuté, où sont ses fichiers...

OS réseau (1)

- Collection d'ordinateurs individuels **reliés dans un réseau**

Partage de ressources comme imprimante, ou serveur de fichiers

- Se **distingue d'un OS distribué** sur plusieurs points
 - Chaque PC a son OS privé pas étendu au système distribué
 - Chaque utilisateur travaille normalement sur sa machine privée
 - Localisation des fichiers bien connue par chaque utilisateur
 - Le système n'est pas très tolérant aux pannes

Problème et objectif (1)

- Changement majeur dans la **technologie des microprocesseurs**
 - Beaucoup plus puissants qu'avant et beaucoup moins chers
 - Abandon mainframe pour système avec **des** petits processeurs
- Plusieurs **autres avantages** aux systèmes distribués
 - Simplicité du software avec fonction spécifique par processeur
 - Croissance incrémentale du système par ajout de processeurs
 - Meilleure fiabilité et disponibilité du système

Problème et objectif (2)

- Quelques éléments négatifs à la distribution
 - Risque d'avoir un overhead de communication si mal réparti
 - Machines passent leur temps sur protocole de communication
- Difficulté de maintenir un état global du système
 - Collecter une telle information peut s'avérer coûteux
 - Maintenir cette information à jour est encore pire
 - Difficulté d'optimiser l'exploitation du système sans cela

Modèle de système (1)

- Trois principales catégories de **modèles de distribution**

Modèles mini-ordinateur, station de travail et pool de processeurs

- **Mini-ordinateur** connecte une douzaine de petites machines
 - Chacune des machines possède plusieurs utilisateurs
 - User loggué sur machine précise, mais accès distant à toutes
 - Évolution de la machine centrale en time-sharing

Modèle de système (2)

- Modèle **station de travail** connecte machines puissantes
 - CPU puissant, mémoire et affichage, pas toujours disque
 - Majorité du travail fait sur station de travail
 - Système de fichiers unique et global à tout le monde
- **Processor pool** inverse ratio CPU par user loggué
 - < 1 pour mini-ordinateur, 1 pour station de travail et > 1 ici
 - Allocation de CPUs à un utilisateur lorsqu'il en a besoin

OS réseau (2)

- Plusieurs idées proviennent des **OS réseau**
 - Connexion de machines via la ligne téléphonique
 - Permettre des logins à distance et transferts de fichiers
- **Trois éléments** visibles font différence avec OS distribué
 - Systèmes de fichiers individuels des machines connectées
 - Protection affecte identifiant unique pour user sur une machine
 - Localisation d'un programme qui est exécuté

Vers l'OS distribué

- **Unification du système de fichiers** pour partage entre tous
 - Un seul répertoire `bin`, un fichier de mots de passe...
 - Répartition de la charge sur disque pour le seul espace de noms
- **UID unique** pour chaque utilisateur sur tout le système

Même UID valide sur toutes les machines, sans mapping local
- Pouvoir **créer un processus** sans spécifier la localisation
 - Appel système `create_process` choisit où exécuter
 - Exécution n'importe où, mais connexion préalable à la machine
 - Autoriser processus à accéder à toutes ressources du système

Implémentation OS distribué

- OS réseau construit par ajout d'une couche software
 - Ajout de software ou librairie exécuté sur OS local
 - Capture des appels systèmes et exécution locale ou distante
- Difficultés avec certains types d'appels systèmes
 - Communication et signaux interprocessus
 - Caractères d'interruption spéciale depuis clavier
 - Un nouveau kernel doit être écrit pour les OS distribués

Primitive de communication



Communication

- Pas de partage de **mémoire principale** entre les machines

Pas de communication avec mémoire partagée, sémaphore...

- Communication essentiellement par **passage de messages**
 - Simple utilisation des primitives `send` et `receive`
 - Appel de procédure à distance comme extension haut niveau

Framework de communication

- Communication utilisant le **modèle ISO OSI**

Définition de sept couches : physical, datalink, network, transport, session, presentation et application

- **Avantages et inconvénients** d'utiliser ISO OSI

- Connexion de machines très différentes facilement
- Lourdeur de communication et échanges inter-couches
- Ressources et temps calcul pour les protocoles

Passage de messages

- Modèle classique le plus répandu est le **client-serveur**
 - Processus client a besoin d'un service et contacte serveur
 - Serveur reçoit requêtes et répond aux clients
- **Deux primitives** de base assurent la communication
 - `send(dst, msg)` envoie un message à une destination
 - `recv(src)` reçoit des messages de sources
 - Pas besoin de setup, coordination ou connexion préalable

Aspect bloquant

- Primitive **fiable et non bloquante** ou bloquantes
 - Garantie de livraison d'un `send`, retransmission
 - Acknowledgement avant que `send` ne rende la main
- Avantages et inconvénients de **primitives non bloquantes**
 - Flexibilité en permettant E/S message parallèles
 - Programmation et tests difficiles car pas reproductible
 - Protection nécessaire des buffers pour accès concurrent

Aspect buffer

- Protocole *rendezvous* pour gérer échange

Le send bloque jusqu'à ce qu'un receive soit fait, et transmission

- Possibilité de *stocker des messages* dans des buffers
 - Permet d'avoir plusieurs appels send sans tout bloquer
 - File d'attente tant que le destinataire n'a rien receive
 - Plus complexe, nécessite protection, interruption, nettoyage...
- *Communication structurée* en distinguant requête et réponse
 - Client fait SEND_GET pour envoyer requête et attendre réponse
 - Server fait GET_REQUEST puis traite et SEND_REPLY

Remote Procedure Call (RPC) (1)

- Similitude entre `send/receive` et **appel d'une procédure**
 - Sémantique communication entre machines = appel procédure
 - Appel $p(x, y)$ côté client, exécution transparente sur serveur
- Ajout **stub dans code client** pour encapsuler communication
 - Généré par compilateur ou dynamiquement lié à l'exécution
 - Empaqueter paramètre, demande d'appel, dépaqueter retour
 - Stub serveur complètement symétrique

Remote Procedure Call (RPC) (2)

- Quelques **difficultés** avec les RPC
 - Passer variables par référence nécessite pointeur global
 - Représentation des données pas uniformes selon architectures
- **Mécanisme de RPC** encapsulé dans langage plus haut niveau
 - Java RMI se base sur spécification machine virtuelle (JVM)*



Design d'OS distribué

Nom et protection (1)

- Un OS doit maintenir des **noms pour les objets** manipulés
 - Fichier, boîte aux lettres, processus, périphérique E/S...
 - Un processus présente le nom à l'OS pour accéder à l'objet

- Noms des objets gérés comme des **mappings**

Fichier : Nom \leftrightarrow i-node UNIX, Mémoire : VA \leftrightarrow PA...

- Utilisation d'un **serveur de noms** unique
 - Maintient la table de mapping entre les noms et les ressources
 - Pas applicable pour gros systèmes sinon bottleneck

Nom et protection (2)

- Partition du système en **plusieurs domaines**
 - Un serveur de noms est dédié à chacun des domaines
 - Utilisation d'un arbre de nommage global
- Chaque machine peut être responsable de **ses propres noms**

Broadcast à tous d'une requête, et réponse par l'intéressé

Gestion des ressources

- Connaitre l'**état des ressources** dans système centralisé
 - Très facile à avoir car stocké dans plusieurs tables locales
 - Information également maintenue à jour facilement
- Difficile de maintenir un **état global** d'un système distribué
 - Niveau de granularité bas difficile à avoir (processeurs)
 - Un serveur avec une table centrale devient bottleneck

Allocation des processeurs

- Organisation des processeurs en **hiérarchie indépendante**
 - Comme à l'armée, dans l'académie, l'industrie...
 - Il y a des processeurs travailleurs, managers...
- Un **manager responsable** pour k travailleurs
 - Garde la liste de qui est occupé et qui est disponible
 - Structure hiérarchique avec doyen responsable de k managers
 - Limite la quantité d'information de coordination échangée
- **Promotion d'un subalterne** lors du crash d'un responsable

Avoir un comité tout en haut au lieu d'une seule racine

Ordonnancement

- Ordonnancement facile dans des conditions simplifiée
 - Processeur exécute 0 ou 1 processus (pas multiprogrammation)
 - Les processus sont complètement indépendants
- Processus coopératifs échangent beaucoup de messages
 - Intéressant de les faire exécuter ensemble et proches
 - Exécution simultanée de processus communiquant préférable
 - Information sur le niveau de communication pas obtainable

Répartition de charge

- Deux optiques pour répartir les processus sur processeurs
 - Processus coopératifs à séparer pour augmenter parallélisme
 - Ou sur même machine pour diminuer couts de communication
- Nécessité d'ajouter du load balancing en plus
 - Ne pas avoir des processeurs inoccupés ou surchargés
 - Objectifs parfois contradictoires ordonnanceur/load balancer
 - Nécessité de récolter de l'information sur la charge effective

Détection de deadlock

- Deux principaux types de **deadlocks** peuvent survenir
 - Accès concurrent à un ensemble commun de ressources
 - Attentes mutuelles de messages lors de communications
- Deadlocks **pour les ressources** sont les traditionnels
 - Même problème que dans les systèmes centralisés
 - Beaucoup plus difficile à détecter car pas d'état global
- Deadlocks **lors de la communication** détectables

Un processus peut envoyer un message à qui il attend

Tolérance aux pannes (1)

- Système distribué **réputé plus fiable** que système centralisé
 - Pas de corruption ou de pertes de données si plus fiable
 - Disponibilité si capable de répondre aux requêtes
- **Duplication** des ressources et des instances pour robustesse
 - Système tolérant aux pannes continue de fonctionner
 - Possibilité d'un mode dégradé et moins performant

Tolérance aux pannes (2)

- Duplication des processus critiques
 - Toujours prévoir un processus backup sur un autre processeur
 - Exécution miroir complet du processus backup
 - Ça coute des ressources et il faut des processeurs similaires
- Mise à jour concurrente de données par plusieurs utilisateurs
 - Chaos potentiel lorsque une ou des machines crashent
 - Un master tape et plusieurs update tapes appliqués en une fois

- Offre de services par des **processus serveurs** niveau user

Permet d'avoir un plus petit kernel, et un OS plus modulaire

- **Plusieurs services** possibles

- Service fichiers offre services disque, fichier plat ou répertoire
- Service d'impression reçoit fichier et les imprime
- Service processus pour en créer des nouveaux
- Service terminal, e-mail, temps, démarrage, gateway...

Crédits

- <https://www.flickr.com/photos/hdaparis/14481507547>
- <https://www.flickr.com/photos/0742/5263773126>
- <https://www.flickr.com/photos/lennykphotography/24360710603>