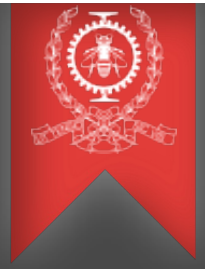


Architecture des clients pour l'infonuagique

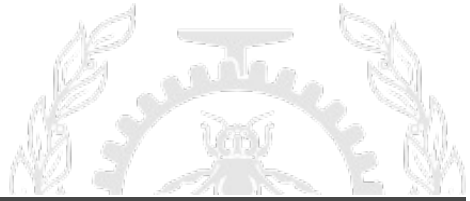
Module 2

INF8480 Systèmes répartis et infonuagique

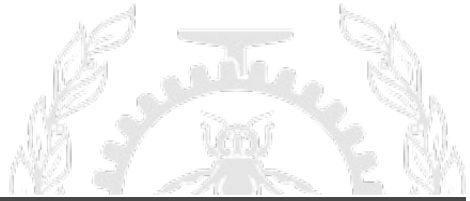
Michel Dagenais

École Polytechnique de Montréal
Département de génie informatique et génie logiciel

Sommaire



Architecture des clients pour l'infonuagique



Répartition du travail

- Services offerts de manière transparente par une multitude serveurs anonymes répartis à travers le monde.
- Le client fait l'affichage graphique simple.
- Le client fait l'affichage graphique spécialisé (décodage vidéo…).
- Le client peut exécuter certaines parties des applications offertes par le serveur.
- Le client peut stocker temporairement certaines données et applications pour des fins de performance et d'autonomie (pour opérer en mode déconnecté), sous une forme de cache.
- Le client est un ordinateur complet avec des applications et des données installées localement et mises à jour ou synchronisées seulement à la demande.

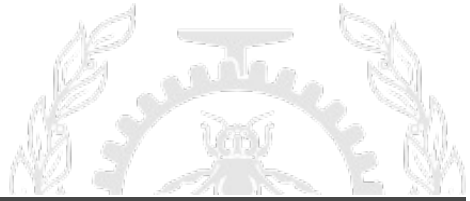
Les terminaux alphanumériques

- Affichage de caractères avec position adressable.

Terminaux graphiques

- Caractères alphanumériques
- Dessins vectoriels ou par matrice de pixels.

Architecture des clients pour l'infonuagique

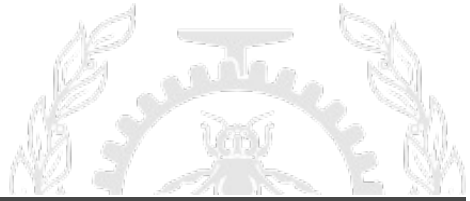


Les clients légers

- Ordinateur de faible puissance qui exécute une application d'affichage graphique soit en mémoire morte, soit téléchargée à l'initialisation.
- Aucune gestion des ordinateurs clients qui sont tous identiques, seulement une gestion des comptes usagers.
- Terminaux X11 sous POSIX (Linux, Unix, Solaris...).
- Citrix Independant Computing Architecture (ICA).
- Microsoft Remote Desktop Protocol (RDP) et Remote Desktop Services (RDS) / Terminal Server.
- Virtual Network Computing (VNC).
- Possibilité de créer une session graphique sur le serveur, de s'y connecter avec un client, se déconnecter sans terminer la session et s'y reconnecter avec un autre client!

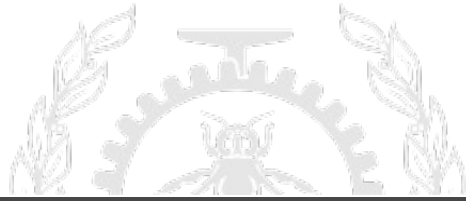
Les serveurs pour les applications lourdes

- Ordinateur client autonome pour les applications usuelles.
- Applications spécifiques pour lesquelles le traitement est envoyé à un serveur.
  - Calcul scientifique.
  - Rendu graphique par lancer de rayon pour un film.
  - Certains jeux 3D.
  - Reconnaissance de la voix Siri sur iPhone.
  - Calcul de chemin Google Maps sur Android.



Les ordinateurs ou applications sans stockage

- Ordinateur sans disque avec chargement du système d'exploitation par réseau (Preboot eXecution Environment - PXE Boot) et utilisation de serveurs de fichiers.
- Ordinateur avec disque local SSD et copie temporaire des applications. Mises à jour automatiques et stockage des données sur un serveur de fichiers, par exemple le ChromeBook.
- Fureteur avec toutes les préférences sauvées sur un serveur, par exemple Chrome.



Les applications téléchargées

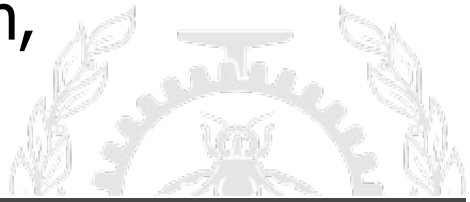
- Code binaire qui doit être pour la bonne architecture et la bonne gamme de versions du système d'exploitation et des librairies.
- Code indépendant de l'architecture (e.g. bytecode Java) mais qui doit être pour la bonne gamme de versions de la machine virtuelle et des librairies.
- Applications signées par une autorité digne de confiance. Exécutées comme une application ordinaire avec les privilèges usuels de l'usager.
- Machine virtuelle avec langage sécuritaire (e.g. Java). L'application ne peut utiliser que les API fournis.
- Bac à sable fourni par le système d'exploitation. L'application ne peut faire que des appels systèmes qui sont vérifiés pour les droits d'accès avant d'être exécutés.

La sécurité dans une machine virtuelle comme la JVM

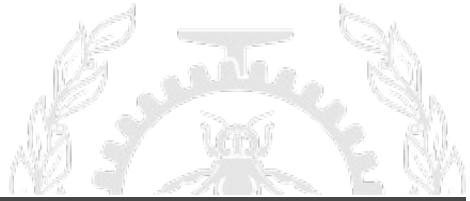
- Langage sécuritaire sans possibilité de corruption. Pas d'arithmétique de pointeur, de pointeur non initialisé ou avec une valeur incorrecte, de débordement de vecteur...
- Interfaces de programmation restreintes avec accès vérifiés pour toutes les opérations plus dangereuses (entrées-sorties).
- Eviter la lecture de données sensibles et leur envoi à l'extérieur. Restreindre l'affichage pour éviter que l'application puisse prendre le contrôle de l'écran et simuler une autre application (e.g. accès bancaire) et tromper l'usager l'incitant à fournir des informations sensibles.
- La machine virtuelle Java fait appel à de nombreuses librairies natives et plusieurs trous de sécurité exploitables à partir du Java ont été trouvés dans ces librairies au fil du temps.

La sécurité dans un bac à sable avec Linux

- Applications compilées qui peuvent effectuer n'importe quelle instruction non privilégiée.
- Le système d'exploitation restreint les opérations permises à une application en fonction de son code d'usager, ses groupes et ses paramètres de capacités.
- Le système impose des quotas d'utilisation des ressources du système pour éviter qu'une application puisse monopoliser les ressources.
- Avec le module Security Enhanced Linux (SELinux), ou avec AppArmor, un contrôle beaucoup plus fin des permissions d'accès est possible.
- Plusieurs démons offrent des services aux applications et vérifient les droits d'accès (notification, localisation, téléphonie...).

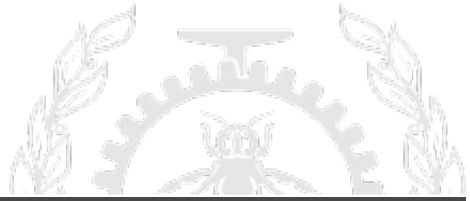


Architecture des clients pour l'infonuagique



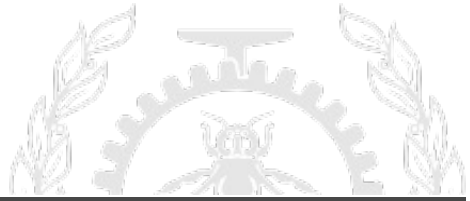
Le fureteur Web comme plate-forme pour les applications

- Courriel, agenda, cartographie, édition de document, visionnement de vidéo, potins assistés par ordinateurs (PAO, par exemple Facebook).
- HTML.
- Javascript.
- Applet Java.
- Modules d'extension non normalisés comme Microsoft Silverlight, Flash, Active-X, ...



HTML5

- Langage déclaratif, basé sur XML, pour décrire le contenu d'une page Web.
- Inclut <video>, <audio>, MathML et SVG (Scalable Vector Graphics).
- Langage JavaScript avec interfaces de programmation (API) pour modifier le document (Document Object Model, DOM) et pour informer le fureteur de l'état de la page (qui peut alors être inclus dans un marque-page).



Javascript

- Langage de script qui ressemble au C et Java mais s'inspire plus sémantiquement de langages comme scheme et self.
- API pour modifier l'affichage via des changements au document avec le Document Object Model.
- Lecture des entrées (clavier et souris) pour la validation des entrées dans les formulaires, interagir avec un jeu ou d'autres applications.
- Interagir avec le fureteur (communiquer l'état pour les marque-pages).
- Communiquer en réseau pour les applications réparties, le suivi des usagers, les statistiques...
- Aucun accès à la machine locale, accès en réseau seulement au serveur d'origine de la page.
- La performance des fureteurs pour exécuter du Javascript est devenue un critère important.

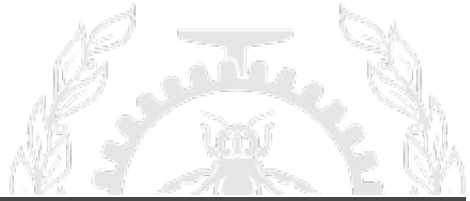
Applet Java dans le fureteur

- Le code Java arrive sous forme de bytecode prêt à s'exécuter sur le client dans la machine virtuelle Java associée au fureteur.
- Le fureteur exécute la machine virtuelle dans un processus séparé avec peu ou pas d'accès à la machine locale.
- Accès au réseau seulement vers le serveur d'origine.
- Possibilité d'application signée qui demande un accès étendu (caméra, micro, fichiers...).
- Affichage graphique avec la librairie Swing dans un cadre du fureteur ou dans une nouvelle fenêtre.
- Permet d'exécuter efficacement des applications relativement complexes et gourmandes.
- Est-ce que Java est un standard ouvert avec implémentation de référence libre?

AJAX (Asynchronous JavaScript and XML)

- Pages avec Javascript et API comme DOM.
- Requêtes au serveur par HTTP.
- Encodage XML ou JSON.
- Permet des applications très dynamiques et évite d'attendre de manière synchrone après les requêtes au serveur.
- Exemple: afficher rapidement le début de la page et aller chercher la suite seulement si l'usager fait défiler la page vers la fin (Google images, Amazon...).
- Il est plus difficile de mettre au point ces applications.
- L'indexation est plus difficile.
- Les signets basés seulement sur le URL n'ont pas l'information sur l'état courant.

Architecture des clients pour l'infonuagique



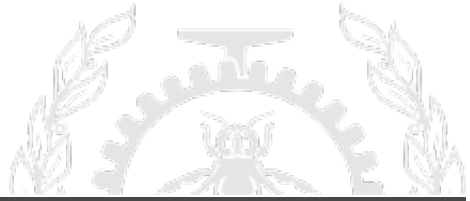
Applications Android

- Les applications Android peuvent être écrites en Java ou directement en code binaire (C/C++ et assembleur compilés).
- Doit être pour la bonne architecture si en binaire.
- Doit être pour la bonne version du système (compatibilité vers l'avant).
- Sécurité basée sur l'exécution sous le système d'exploitation Linux avec un numéro d'usager différent pour chaque application.
- Tous les accès vers les ressources passent par un démon qui vérifie les permissions données par l'usager.
- Certaines applications malicieuses peuvent fonctionner si l'usager donne les permissions demandées au moment de l'installation.

Applications IOS

- Les applications IOS peuvent être écrites en Objective C ou Swift et sont compilées.
- Toutes les applications doivent être signées.
- Les applications s'exécutent sur le système d'exploitation en tant que l'usager mobile et ont un répertoire maison assigné aléatoirement.
- Tout accès à de l'information ou des ressources se fait via des services (démons) qui vérifient si l'accès doit être autorisé.
- Le code est en mode lecture seulement et la pile en mode non exécutable. Les adresses de départ des différentes régions sont randomisées.
- L'exécution en arrière-plan ne se fait que via des fonctions de rappel bien contrôlées.
- Toutes les informations en mémoire permanente sont encryptées.

Architecture des clients pour l'infonuagique



Choix du type de client Ex. 2.1.

Une grande entreprise planifie son organisation informatique et doit décider comment répartir la charge entre les clients et les serveurs. Quatre modèles de clients de puissance de calcul différente sont disponibles. Pour chaque modèle, le prix et la puissance sont comme suit : i) $200 2000 MIPS, ii) $400 80000 MIPS, iii) $1000 100000 MIPS, iv) $2000 175000 MIPS. Pour les serveurs, le coût est estimé à $2000 pour 100000 MIPS car il y a un surcoût important pour toute l'infrastructure qui les entoure (espace, réseautique...).

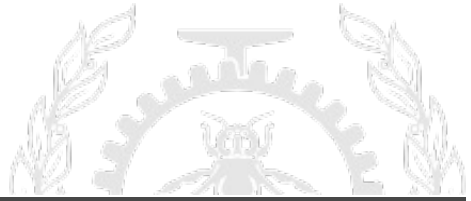
Pour un client donné, les applications suivantes sont exécutées pour une certaine fraction des heures de travail et avec une puissance consommée associée : a) 1500 MIPS 15%, b) 50000 MIPS 25%, c) 90000 MIPS 10%, d) 150000 MIPS 10%. Un client ne travaille qu'avec une seule application à la fois. Lorsqu'une application peut s'exécuter sur le client, car il a une performance suffisante, cela ne coûte rien pour le serveur. Autrement, l'application s'exécutera sur un serveur et demandera une certaine puissance pour une certaine fraction du temps, ce `a quoi un prix peut être associé.
Lequel client sera-t-il le plus rentable d'acheter pour cette compagnie ?

Choix du type de client – Solution Ex. 2.1.

Avec le client i, il faudra dépenser $200 pour le client et 50000 MIPS x .25 + 90000 MIPS x .10 + 150000 MIPS * .10 = 36500 MIPS pour exécuter les applications trop grosses, pour un coût total de $200 + 36500 MIPS * $2000 / 100000 MIPS = $930.

Avec le client ii, il faudra dépenser $400 pour le client et 90000 MIPS x .10 + 150000 MIPS * .10 = 24000 MIPS pour exécuter les applications trop grosses, pour un coût total de $400 + 24000 MIPS * $2000 / 100000 MIPS = $880.

Avec les clients iii et iv, il faudra dépenser $1000 et $2000 respectivement pour le client, et la solution sera nécessairement moins avantageuse que la ii qui ne coûte que $880 (e.g. pour le client iii on a 150000 MIPS * .10 = 15000 MIPS pour exécuter les applications trop grosses, pour un coût total de $1000 + 15000 MIPS * $2000 / 100000 MIPS = $1300). La solution ii est donc la plus avantageuse.



Communication – Ex. 2.2.

Un groupe de 45 processus, sur autant de nœuds connectés sur le même réseau local, communiquent à l'aide de messages de groupe. Un processus du groupe doit envoyer un message de groupe de manière atomique aux autres processus du groupe.

Comment cela peut-il être réalisé?

Combien de messages seront envoyés sur le réseau par les différents processus, pour un message de groupe atomique, si la multi-diffusion est disponible?

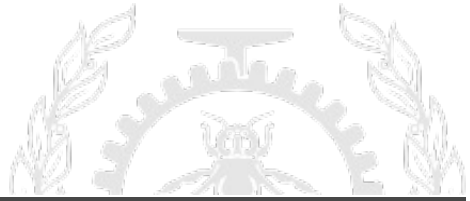
Si la multi-diffusion n'est pas disponible?

Communication – Solution - Ex. 2.2.

Il faut procéder en deux phases. Lors de la première phase, on envoie le message en question aux autres membres du groupe en demandant un accusé de réception et en spécifiant d'attendre la confirmation avant de livrer le message à l'application. Si tous accusent réception du message, la confirmation est ensuite envoyée. Si la multi-diffusion est disponible, il faut l'envoi initial, 1 message en multi-diffusion, n-1 soit 44 accusés de réception venant des autres membres du groupe, et un message de confirmation en multi-diffusion, soit un total de 46 messages. Sans la multi-diffusion, il faut 44 messages pour le message initial, 44 pour les accusés de réception et 44 pour la confirmation, soit un total de 132 messages.

Magasiner les applications - Ex. 2.3.

Les magasins d'applications peuvent contenir des millions d'applications différentes. Dans la mesure où les applications seront exécutées sur les téléphones des individus, comment s'assure-t-on d'un bon niveau de sécurité? Est-ce que les applications ont été vérifiées par le magasin qui les offre?



Magasiner les applications – Solution - Ex. 2.3.

Il existe un grand nombre d'applications malicieuses qui espionnent les utilisateurs, ou tentent de dérober des informations personnelles (e.g. bancaires). Les sociétés comme Google ou Apple n'ont pas le code source des applications ni le temps requis pour les analyser en détail. Certains tests, dont la nature reste secrète, sont effectués afin de détecter certains cas d'applications malveillantes qui sont alors retirées du magasin. De plus, chaque application n'a accès qu'à un certain nombre de permissions qui doivent lui être octroyées explicitement par l'utilisateur. Toutefois, la granularité des permissions n'est pas assez fine et ne permet souvent pas de détecter les excès. Ainsi, une application peut avoir une raison valable pour justifier de demander le droit d'accéder à la caméra, au micro, au réseau et au GPS (e.g. reconnaître un oiseau en fonction de l'image, du son, de la localisation et de l'interrogation d'un serveur) mais en abuser pour espionner les moindres faits et gestes de l'utilisateur.

Conclusion

- Beaucoup de temps était perdu à une certaine époque pour installer adéquatement un ordinateur client. Toute mise à jour ou remplacement du matériel était un cauchemar.
- Langage de haut niveau, indépendant du matériel, et librairies avec compatibilité vers le haut, permettent une grande portabilité et même des applications mobiles.
- Où est le bon niveau de virtualisation : fureteur, Java, système d'exploitation, matériel?
- Quelles applications et données doivent-elles mieux résider localement?
- Sécurité du client, du réseau et des serveurs?

Résumé