IBD : Intergiciels et Bases de Données

Introduction aux systèmes distribués

Vania Marangozova-Martin
Université de Grenoble, LIG

Vania.Marangozova-Martin@imag.fr

Site web : http://ibd.forge.imag.fr

Objectifs

- Appréhender la complexité des systèmes distribués
- Introduction aux intergiciels
  - Pourquoi en a-t-on besoin?
  - Qu'est-ce qu'on y met dedans?
- Considérer des aspects pratiques
  - UE avec une forte composante technique
  - Travailler avec des technologies actuelles : serveurs web, bases de données noSQL, cloud…

Planning

Semaine	Mercredi 13:30 - 18:30
5	Cours Introduction (Vania M-M) 13:30-15:00 TP Client-serveur communication Java par sockets 15:15-16:45
6	Cours RMI (Vania M-M) 13:30-15:00 Tutoriel : Getting started with RMI 15:15-16:45 TP RMI Chat 17:00 - 18:30
7	TP RMI Chat 13:30 - 15:00 Démos TP RMI
8	interruption HIVER
9	Cours Applis Multi-tiers et servlets 13:30 - 15:00 TP Servlets 15:15 - 16:45
10	Cours Hibernate (Cyril L) 13:30 - 15:00 Titoriel : Getting Started with Hibernate 15:15 - 16:45 TP Hibernate 17:00 - 18:30
11	Cours noSQL (Cyril L) 13:30 - 15:00 Tutoriel : Getting started with noSQL and MongoDB 15:15 - 16:45 TP MongoDB 17:00 - 18:30
12	Cours Cloud (Vania M-M) 13:30 - 15:00 Tutoriel : Getting started with Google App Engine 15:15 - 16:45
13	Projet cloud lancement
14	Projet cloud support
15	Soutenances projet cloud
16	Interruption PRINTEMPS

Détails d'organisation

- Evaluation
  - Note de travail pratique
    - Application de chat (RMI) : démo
    - Réseau social (Cloud) : démo
  - Examen final
- Page web
  - ibd.forge.imag.fr

Contact

- Systèmes distribués et intergiciels
  - Vania Marangozova-Martin (@imag.fr)
    Maître de Conférences, Université de Grenoble
    Laboratoire LIG, équipe Nanosim

- Bases de données
  - Cyril Labbé (@imag.fr)
    Maître de Conférences, Université de Grenoble
    Laboratoire LIG, équipe SIGMA

- TPs
  - Vania ou Cyril + Thomas Calmant (@imag.fr)

Plan

1. Qu'est-ce qu'un système distribué
   - Mécanismes de communication
   - Services et interfaces
   - Architecture client/serveur
2. Qu'est-ce qu'un intergiciel (*middleware*)
3. Références

Qu'est-ce un système distribué?

- "*A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable*."

  Leslie Lamport, 1987.

Pourquoi la distribution?

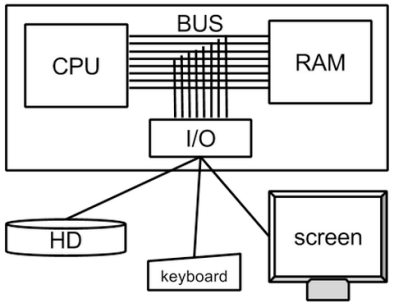
La répartition est un état de fait pour un nombre important de systèmes

- Besoins propres des systèmes
  - Intégration de parties existantes initialement séparées
  - Intégration massive de ressources
    - Grilles de calcul, cloud, data centres
  - Nouveaux domaines d'application de l'informatique
    - Intégration d'objets du monde réel (informatique omniprésente, *ubiquitous computing*)
      - Surveillance et commande d'installations
- Possibilités techniques
  - Coût et performances des machines et des communications
  - Interconnexion généralisée
    - Informatique+télécom+TV
    - Réseaux de capteurs

Caractéristiques des systèmes distribués

- Définition d'un système distribué
  - Ensemble composé d'éléments reliés par un système de communication; les éléments ont des fonctions de traitement (processeurs); de stockage (mémoire), de relation avec le monde extérieur (capteurs, actionneurs)
  - Les différents éléments du système ne fonctionnent pas indépendamment mais collaborent à une ou plusieurs tâches communes.

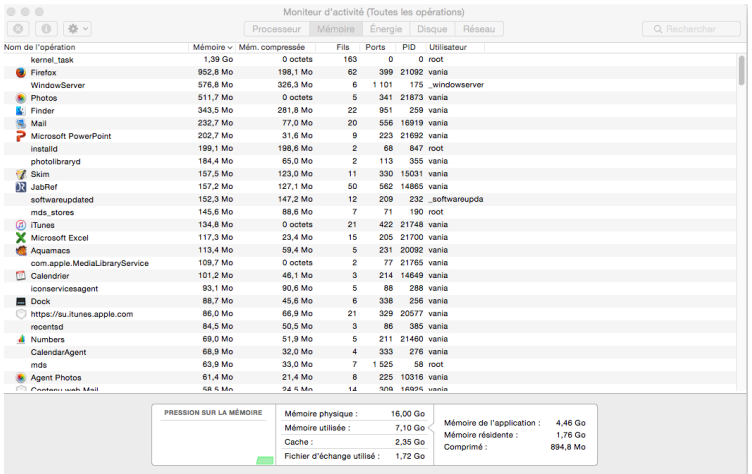
En quoi un système distribué diffère d'un système centralisé? 1/7



système centralisé

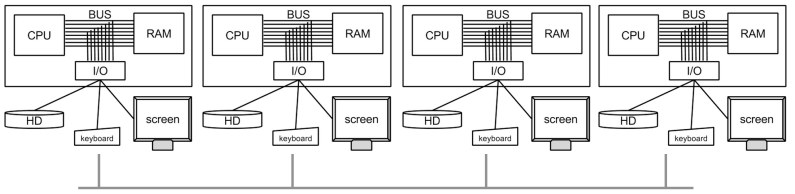
- Tout est sur la même machine et accessible
  - La mémoire
  - Les données sur disque
  - Les I/O
- Le programme s'exécute en local sur la machine
  - On peut le surveiller
    - Etat mémoire
    - Etat processus
    - …

En quoi un système distribué diffère d'un système centralisé? 2/7



En quoi un système distribué diffère d'un système centralisé? 3/7

- Qu'est-ce qu'on distribue?
  - Les ressources sont distribuées
    - CPU
    - Mémoire (vive, disques) distribuée et non partagée



En quoi un système distribué diffère d'un système centralisé? 4/7

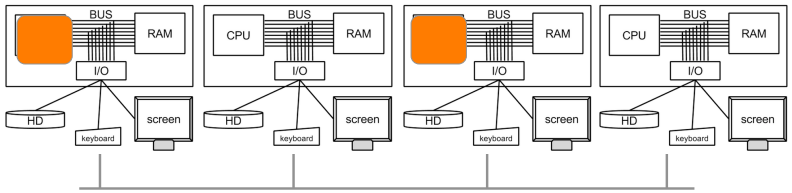
- Qu'est-ce qu'on distribue?
  - Le code

une variable ici

n'est pas accessible ici

est différent de ce qu'on fait là

ce qu'on fait ici



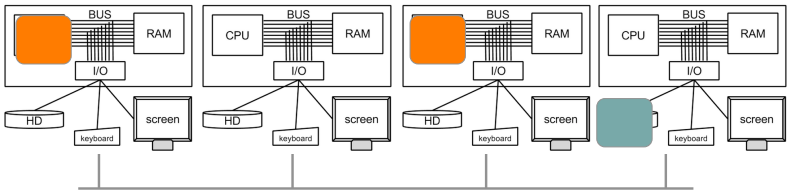
En quoi un système distribué diffère d'un système centralisé? 5/7

- Qu'est-ce qu'on distribue?
  - Les données

les données ne sont pas là

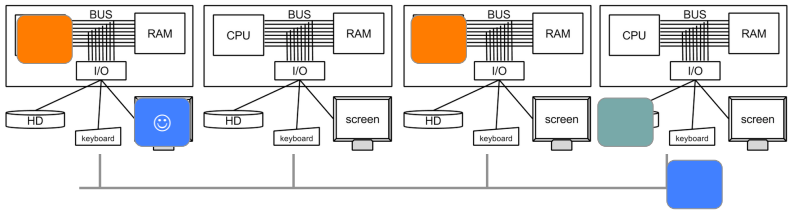
les données ne sont pas là

les données ne sont pas là



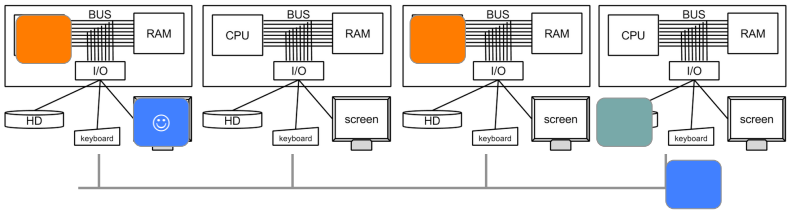
En quoi un système distribué diffère d'un système centralisé? 6/7

- Qu'est-ce qu'on distribue?
  - Les intéractions



En quoi un système distribué diffère d'un système centralisé? 7/7

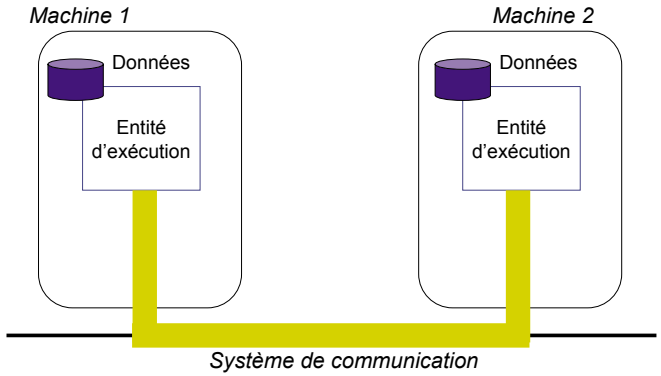
- Pas de temps global
- Pas d'état global
- … et il faut quand même que cela marche ☺



Propriétés souhaitées des systèmes distribués

- Le système doit pouvoir fonctionner (au moins de façon dégradée) même en cas de défaillance de certains de ses éléments
- Le systèmes doit pouvoir résister à des perturbations du système de communication (perte de messages, déconnexion, performances dégradées)
- Le système doit pouvoir résister à des attaques contre sa sécurité (violation de confidentialité, de l'intégrité, usage indu de ressources, déni de service)

Schéma d'un système réparti

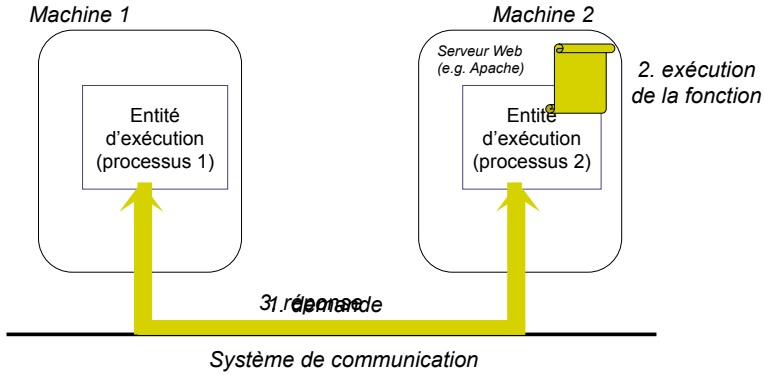


Mécanismes de communication

- Communication directe (synchrone)
  - Programme à programme
    - Ex. Appel de fonction à distance
  - Programme à base de données
    - E.g. Transactions distribuées (banque)
- Communication indirecte (asynchrone)
  - Communication par messages

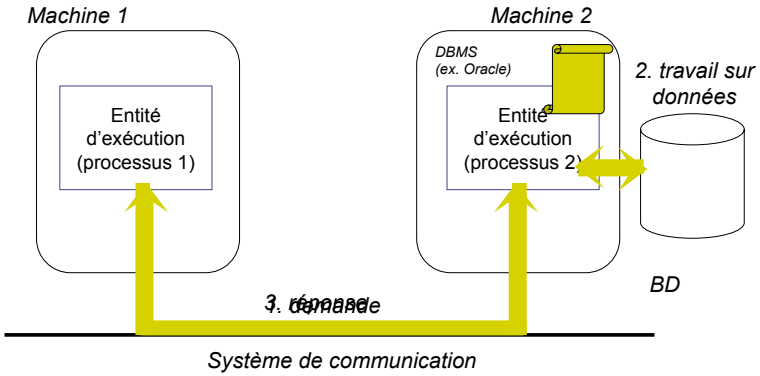
Mécanismes de communication (2)

- Appel de fonction à distance (ex. Application web)



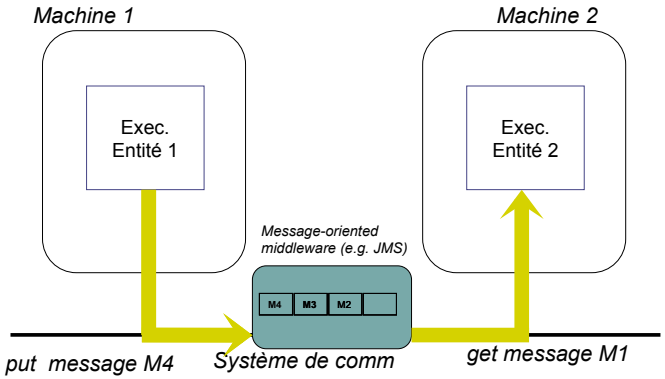
Mécanismes de communication (3)

- Transactions distribuées (ex. serveur de base de données)



Mécanismes de communication (4)

- Communication par messages (e.g. a chat system)



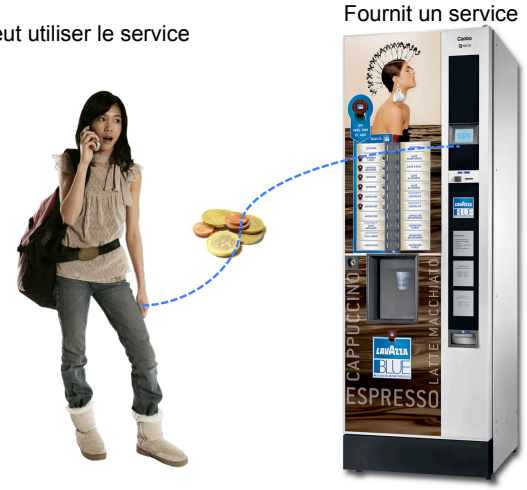
Plan

1. Qu'est-ce qu'un système distribué
   - Mécanismes de communication
   - **Services et interfaces**
   - Architecture client/serveur
2. Qu'est-ce qu'un intergiciel (*middleware*)
3. Références

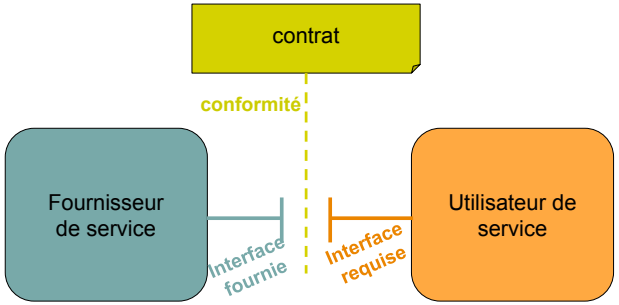
Services et interfaces

- Service
  - Un ensemble de fonctionnalités réutilisables qui définissent un comportement donné et que l'on utilise d'une manière pré-définie
    - Chaque composant logiciel ou matériel a un comportement bien défini et fournit donc un service
    - Un service peut être implémenté de différentes manières
    - "A service is a contractually defined behavior that can be implemented and provided by any component for use by another component, based solely on the contract",
      Bieber et. al., Service oriented programming, http://www.openwings.org/
- Interface
  - Un service est accessible via une ou plusieurs interfaces
  - Une interface définit les interactions possibles entre un fournisseur de service et un utilisateur de service

Un exemple du monde réel



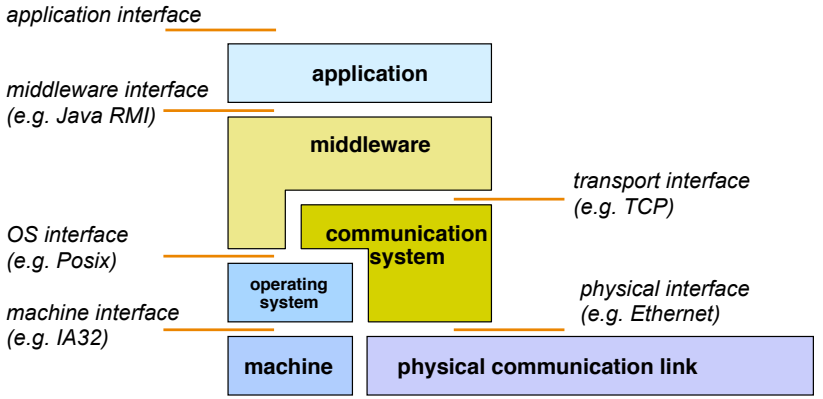
Interfaces (1/2)



Interfaces (2/2)

- Un service dépend de deux interfaces
  - Une interface requise (point de vue client)
  - Interface fournie (point de vue serveur)
- Le contrat
  - Le contrat spécifie la conformité entre les deux interfaces
  - Le client et le serveurs sont considérés être des boîtes noires i.e. leurs fonctionnements/implémentations ne sont pas connus.
    - Ils peuvent donc évoluer
    - tant que le contrat est respecté
- Le contrat peut également spécifier des aspects qui ne sont pas liés aux interfaces
  - Propriétés extra-fonctionnelles qui sont liées à la QoS

Qq exemples d'interfaces importantes dans les systèmes informatiques

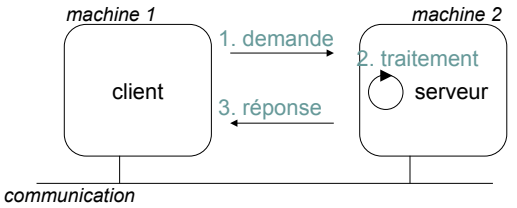


Plan

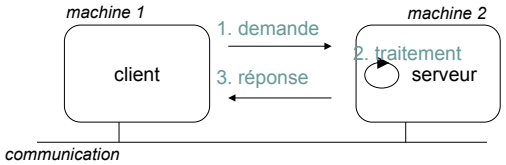
1. Qu'est-ce qu'un système distribué
   - Mécanismes de communication
   - Services et interfaces
   - **Architecture client/serveur**
2. Qu'est-ce qu'un intergiciel (*middleware*)
3. Références

Architecture client/serveur (1)

- Définitions
  - L'architecture client/serveur est un modèle d'interaction
  - Le serveur fourni un service
  - Le client demande le service
  - Dans la plupart des cas le client et le serveur sont déployées sur deux machines différentes (ce n'est toutefois pas obligatoire)
  - Exemples: RPC, Java RMI, Web Services, etc.



Architecture client/serveur (2)



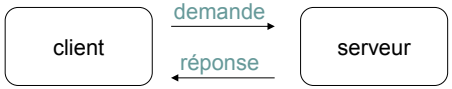
- Message de demande
  - Envoyé par le client au serveur
  - Spécifie le service demandé (le serveur peut fournir plusieurs services)
  - Contient les paramètres pour le service
- Message de retour
  - Envoyé par le serveur au client
  - Contient le résultat d'exécution ou une erreur
- La communication est synchrone
  - Le client se bloque en attendant la réponse du serveur

Architecture client/serveur (3)

- Avantages
  - Structuration
    - Séparation entre la spécification d'un service et son implémentation
    - Par conséquent, les implémentations peuvent changer tant que la même interface est utilisée
  - Protection/sécurité
    - Le client et le serveur s'exécutent dans des domaines différents (domaine = espace mémoire, droits, protection…)
  - Gestion de ressources
    - Un serveur peut (et souvent est) être partagé par plusieurs clients

Architecture client/serveur (4)

- Serveur partagé
  - Point de vue client



  - Point de vue serveur
    - Choisir parmi les demandes client
    - Gestion de requêtes (séquentielle ou parallèle)

Architecture client/serveur (5)

- Choix de la requête client = ordonnancement
  - Choix d'une requête en attente
  - Traitement
  - Retour du résultat
- Stratégies d'ordonnancement
  - First-In First-Out (FIFO)
  - Le plus court d'abord
  - Priorités

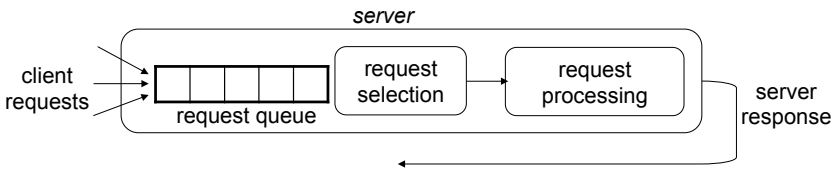
Architecture client/serveur (6)

- Traitement des requêtes
  - Le client et le serveur sont deux exécutions indépendantes
  - Pendant que le client est bloqué en attendant la réponse du serveur
  - Le serveur peut traiter plusieurs requêtes en parallèle
    - vrai parallélisme (e.g. multiprocessors, I/O)
    - pseudo-parallélisme
  - Implémentation avec
    - Plusieurs processus ou
    - plusieurs threads

Architecture client/serveur (7)

- Gestion de ressources chez le serveur avec processus unique

```
while (true) {
      receive(client_id,message);
      extract(message, service_id, params);
      results = do_service(service_id, params);
      send(client_id, results);
}
```

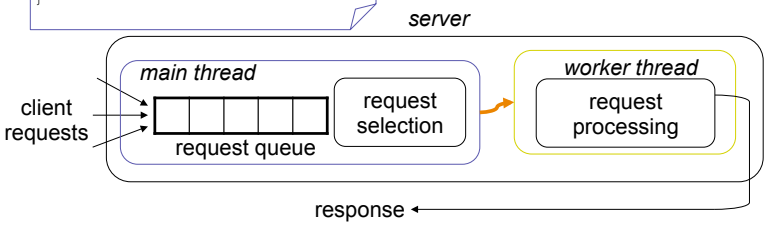


Architecture client/serveur (8)

- Gestion de ressources chez le serveur avec plusieurs processus

```
while (true) {
   receive(client_id,message);
   extract(message, service_id,
      params);
   thr = create_thread(client_id,
      service_id,params);
}
```

```
Program executed by thread thr:
   results = do_service(
      service_id, params);
   send(client_id, results);
   exit
```



Architecture client/serveur (9)

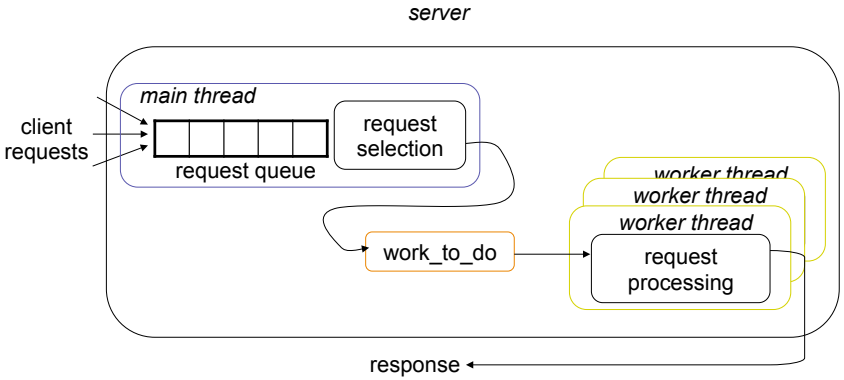
- Gestion de ressources chez le serveur avec un pool de processus

```
while (true) {
   receive(client_id,message);
   extract(message, service_id,
      params);
   work_to_do.put(client_id,
      service_id,params);
}
```

```
Pool of processes:
while (true) {
   work_to_do.get(
      client_id, service_id,
      params);
   results = do_service(
      service_id, params);
   send(client_id, results);
}
```


Architecture client/serveur (10)

- Gestion de ressources chez le serveur avec un pool de processus



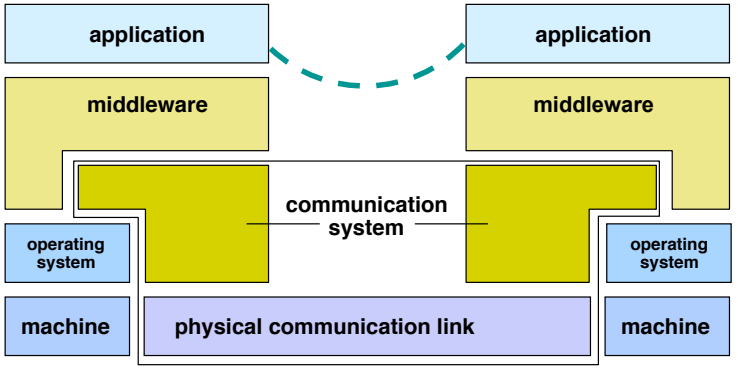
Architecture client/serveur (11)

- Utilisation de l'architecture client/serveur
  - Avec des opération de bas niveau
    - Se basant sur des fonctions du système de communication
    - Exemple: Sockets
      - TCP, connected mode
      - UDP, unconnected mode
  - Avec des opérations de haut niveau
    - En utilisant un intergiciel
    - Exemple: RMI (Remote Method Invocation) dans OO

Plan

1. Qu'est-ce qu'un système distribué
   - Mécanismes de communication
   - Services et interfaces
   - Architecture client/serveur
2. **Qu'est-ce qu'un intergiciel (*middleware*)**
3. Références

Intergiciel



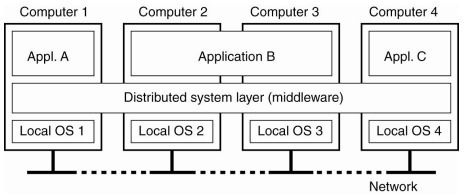
Caractéristiques principales d'un intergiciel

- Fonctions
  - Cacher la distribution
  - Cacher l'hétérogénéité des ressources matérielles et logicielles
  - Fournir des services communs utiles (réutilisables)
  - Fournir une interface (API) de haut niveau pour la programmation d'applications
- Objectifs
  - Implémentation, évolution and reutilisation de code applicatif
  - Portabilité entre plate-formes
  - Interoperabilité entre applications/plates-formes hétérogènes

Systèmes distribués de calcul

- Objectif
  - Calcul distribué à bonne (haute) performances
- Calcul sur grappe (*cluster computing*)
  - Plusieurs machines interconnectées par LAN
  - Homogènes = même OS, hardware
  - Noeud de gestion centralisé
- Calcul sur grille (*grid*)
  - Hétérogénéité
  - Echelle
  - Dispersion géographique
- Nuage (cloud)
- Applications
  - Streaming
  - Services Web
  - Calcul scientifique

MPI
OpenPBS
Globus
gLite
OpenStack
STORM
…

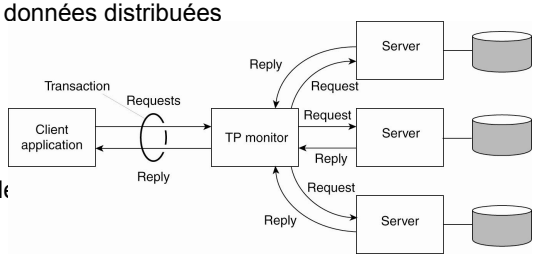


Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e,
(c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

Systèmes d'information

Oracle
MongoDB
pH1
Google Datastore
…

- Objectif
  - Fournir l'accès aux données distribuées
- Mise en place
  - Transactions
  - ACID propriétés
  - Traitement de grosses quantités de données (noSQL)
- Applications
  - Streaming
  - E-Commerce
  - …

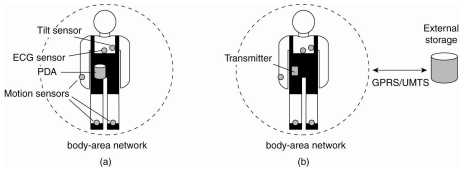


M. van Steen, Lecture on Distributed Systems, Chapter 1, http://www.cs.vu.nl/~steen/

Systèmes omniprésents

LSM
e-health
…

- Objectif
  - Accès à des données partagées depuis n'importe où
  - En préservant la confidentialité des données
- Mise en place
  - Support des changements de contexte, de la mobilité
  - Architecture ad-hoc



- Applications
  - Domotique
  - Santé (suivi médical automatisé)

Références

- Chris Britton, Peter Bye. *IT Architectures and Middleware: Strategies for Building Large, Integrated Systems (2nd Edition)*. Addison-Wesley, 2004.
- George Coulouris, Jean Dollimore, Tim Kindberg. *Distributed Systems: Concepts and Design (4th Edition)*. Addison Wesley, 2005.
- Arno Puder, Kay Römer, Frank Pilhofer. *Distributed Systems Architecture: A Middleware Approach*. Morgan Kaufmann, 2005.
- Andrew S. Tanenbaum, Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice Hall, 2006.
- This lecture is partly based on lectures given by Sacha Krakowiak, http://proton.inrialpes.fr/people/krakowia/