Architecture Distribuée

Cours n°6

A.Saval

Objectifs du cours “Architectures distribuées”

- Compréhension des motivations
- Compréhension de la logique de conception d'une architecture distribuée
- Maîtrise des principaux modèles
- Aperçu des problèmes posés
- Aperçu de quelques frameworks existants

Aperçu du cours

- Introduction
- Problème de conception d'architecture
- Architecture logique & matérielle
- Système distribué
- Modèles d'architecture
  - Client/serveur
  - 3-tiers
  - N-tiers
  - Virtualisation

MODELES D'ARCHITECTURE

Aperçu du cours

- Découverte de services
- Clusters
- Peer-to-Peer
- Cloud

Services distribués

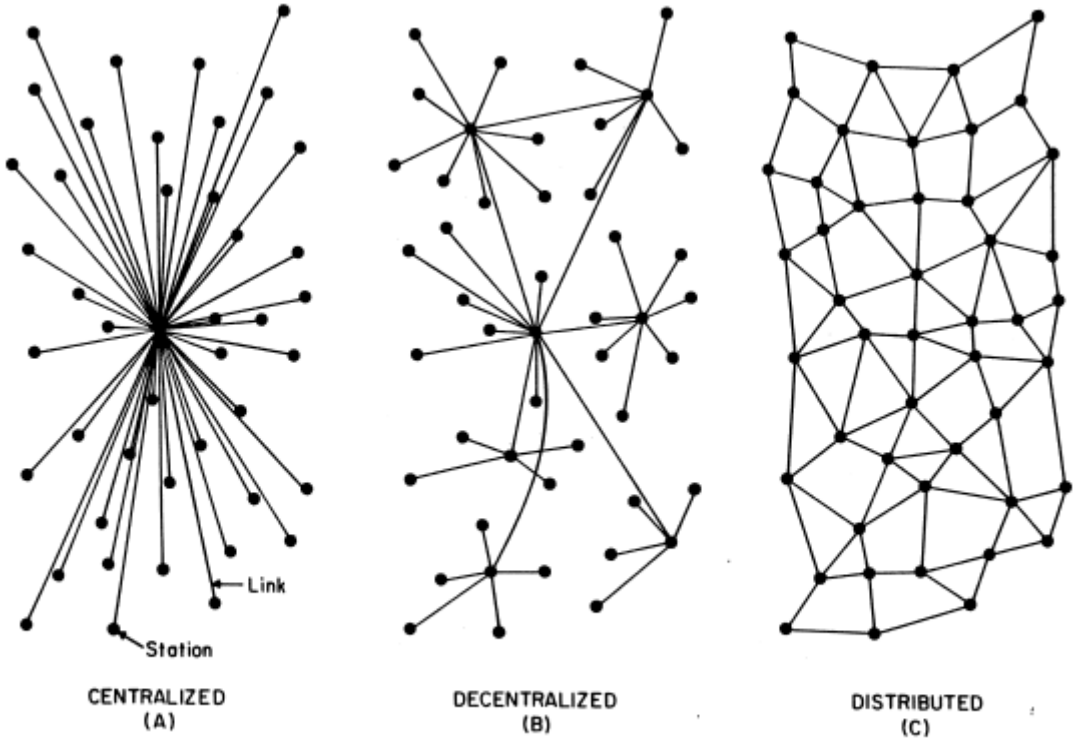
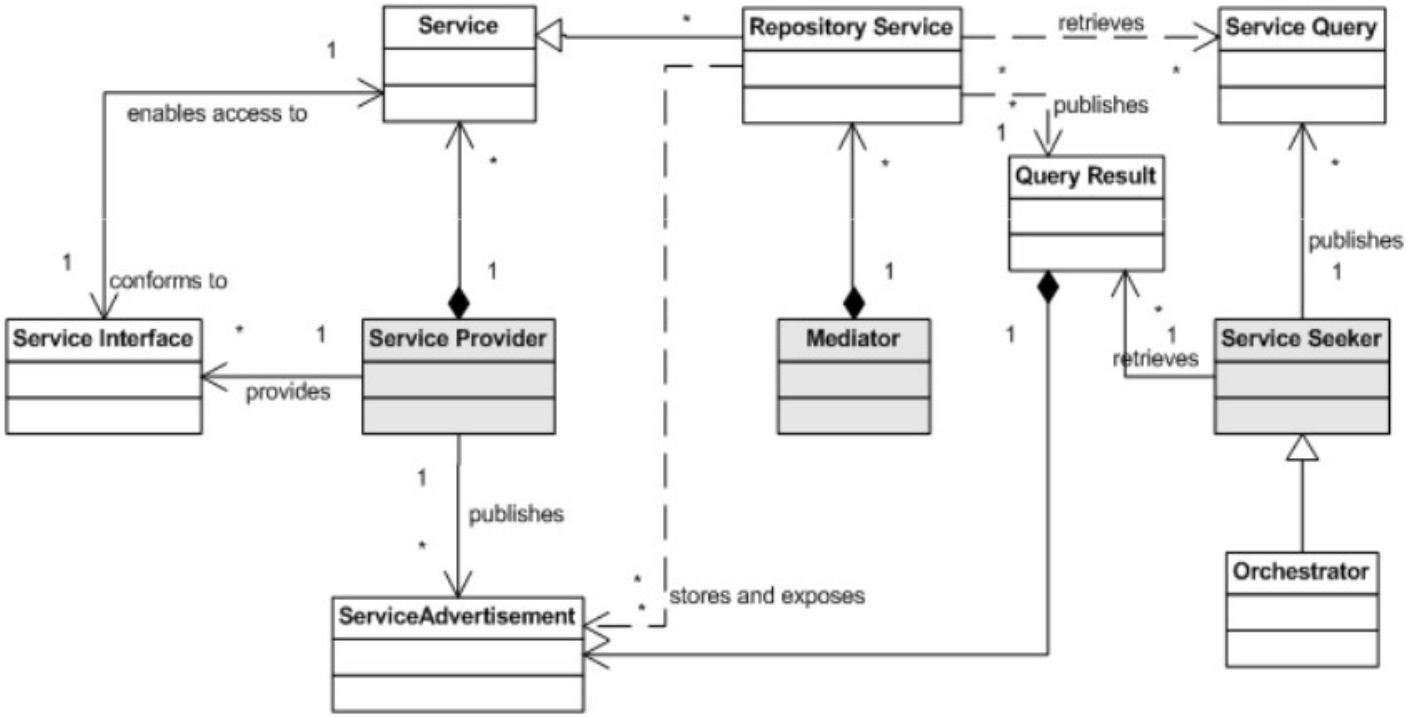


FIG. 1 — Centralized, Decentralized and Distributed Networks

Découverte de services

Problème:

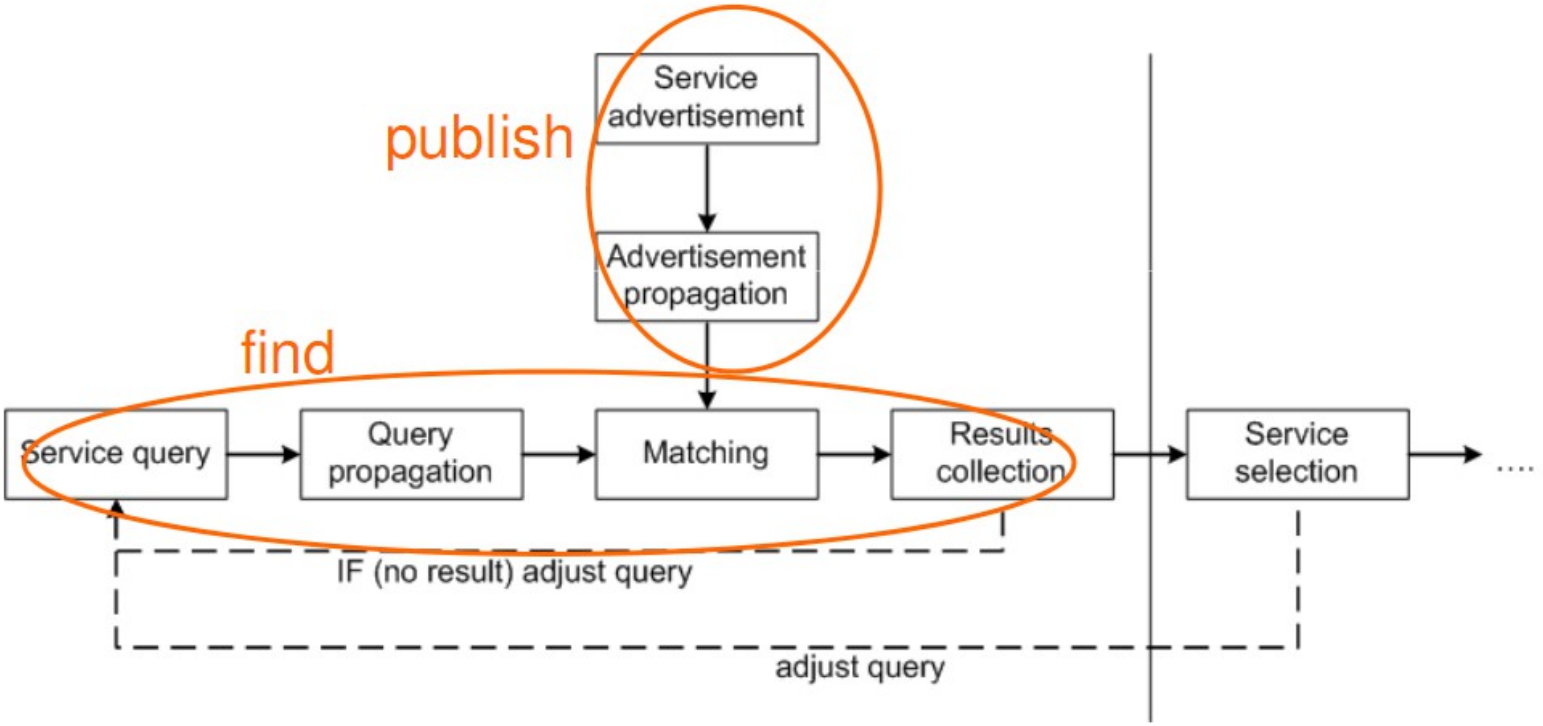
Comment peuvent se rencontrer deux éléments qui ne se connaissent pas?



Slides based on work by Melissa Tjiong, SAN, TU/e

Découverte de services

Déroulement:



Découverte de services

Méthodes existantes:

- Le client contacte le service par un accès connu préalablement
- Le service se déclare auprès d'un annuaire (le client contactera l'annuaire pour récupérer l'accès à ce service)
- Le service est un super-serveur capable de générer des serveur en fonction des requêtes reçues.

L'annuaire et le super-serveur sont des serveurs.

Découverte de services

Méthodes existantes:

- Broadcast (du client et/ou du serveur)

  Avantages:

  - Completement distribué
  - Pas d'état
  - Pas de configuration

  Désavantages:

  - Tous les éléments doivent implémenter le protocole
  - Passage à l'échelle
  - Broadcast

Découverte de services

Résultats de la découverte de services:

- Ensemble de services qui répondent à la requête

Exemples: UDDI

Universal Description Discovery and Integration (2000)

Extention de XML

Indépendant de la plateforme

Supporté: IBM, Microsoft, SAP

Standard OASIS et groupe de travail W3

Exemples: UDDI

Spécifications: plus de 420 pages

Extensions utiles propriétaires

Pas d'interopérabilité entre les annuaires

=> 2006 fermeture des premiers annuaires

=> 2010 UDDI est mort

Exemples

Autres Annuaires

- JAXR
- ebXML Registry
- S-RAMP

- HP Systinet
- IBM WSRR
- Software AG CentraSite

Modèle: Cluster (grappe)

Description:

- Ensemble de machines inter-connectées pour en simuler une seule.

- Utilisations:
  - Disponibilité
  - Gestion de la charge
  - Puissance de calcul

Modèle: Cluster

Avantages:

- Partage des tâches
- Partage des resources
- Isolation rapide des éléments
- Evolution par ajout de nouvelles machines

Désavantage:

- Division des resources entre plusieurs tâches

Exemple: PostgreSQL

Program	License	Maturity	Replication Method	Sync	Connection Pooling	Load Balancing	Query Partitioning
PGCluster	BSD	See version details on site	Master-Master	Synchronous	No	Yes	No
pgpool-I	BSD	Stable	Statement-Based Middleware	Synchronous	Yes	Yes	No
pgpool-II	BSD	Recent release	Statement-Based Middleware	Synchronous	Yes	Yes	Yes
slony-I	BSD	Stable	Master-Slave	Asynchronous	No	No	No
Bucardo	BSD	Stable	Master-Master, Master-Slave	Asynchronous	No	No	No
Londiste	BSD	Stable	Master-Slave	Asynchronous	No	No	No
Mammoth	BSD	Stable	Master-Slave	Asynchronous	No	No	No
rubyrep	MIT	Recent Release	Master-Master, Master-Slave	Asynchronous	No	No	No

From http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling#Comparison_matrix

NoSQL

Évolution des SGBD et des transactions ACID:

- Atomiques
- Cohérentes
- Isolées
- Durables

Théorème CAP:

Besoin de synchronisation entre 2 noeuds => perte de la cohérence

Modèle: Peer-to-Peer

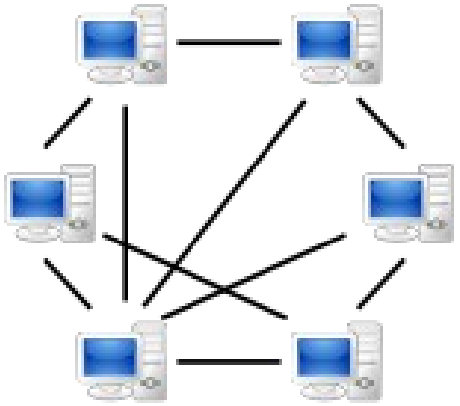
Objectifs:

- Partage de resources
- Coopération entre des communautés
- Rendre les éléments symétriques
- Augmenter la concurrence

Modèle: Peer-to-Peer

Peer:

- Fonctions et contibutions identiques
- Peut joindre n'importe quel Peer
- Passif ou actif



Super Peer:

- Propose des services particuliers

  (découverte de resources, statistiques ...)

Modèle: Peer-to-Peer

Pocessus habituel:

- Un *Peer* trouve un accès vers une communauté
- Il rejoind la communauté
- Il fournit des services aux autres *Peers*
- Il utilise les services des autres *Peers*
- L'ensemble des services fournis par les *Peers* de la communauté constitue un nouveau service.

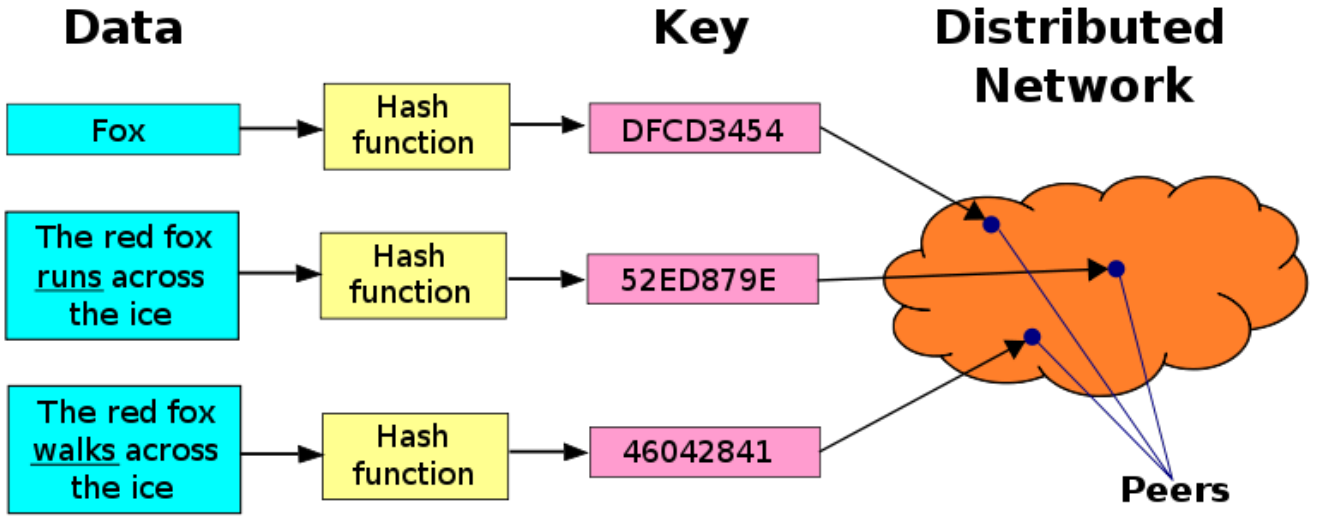
Distributed Hashtable (DHT)

Description:

- Système distribué qui fournit un service de recherche de paires

  (clé, valeur)

- Chaque noeud participe au stockage et peut efficacement récupérer n'importe quelle valeur.
- L'ajout et la suppression de noeud affecte peu l'accès aux données.



Distributed Hashtable (DHT)

Avantages:

- Décentralisation
- Tolérance aux problèmes
- Passage à l'échelle efficace

Complexité d'accès à un éléments dans une communauté à *n* participants est O(log n)

- Ajout de sécurité, anonymisation, gestiond e la charge, vérification des erreurs …

Exemple: DHT

Il existe de nombreuses implémentations:

- Apache Cassandra
- BitTorrent DHT
- CAN (Content Addressable Network)
- Chord
- Kademlia
- ...

Exemple: Chord

Présenté par Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, et Hari Balakrishnan, et conçu au MIT.

Description:

- Utilise des noeuds répartis en cercle contenant 2^m noeuds
- Peut gérer 2^m-1 paires (clé, valeur)
- Chaque clée est hashé avec SHA-1
- Chaque noeud à un noeud suivant et un précédent

Exemple: Chord

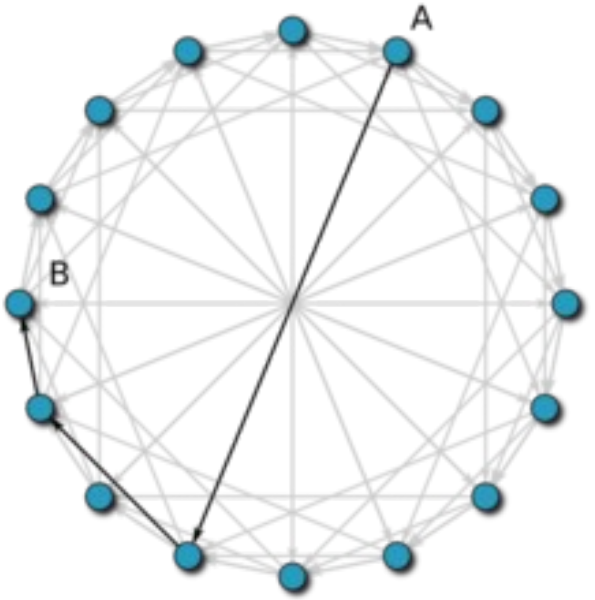
Le noeud suivant d'un noeud est celui qui le succède dans le sens horaire.

Des noeuds peuvent disparaître ou apparaître: Chaque noeud garde en mémoire les *r* noeuds qui le suivent et qui le précèdent.

Une clé *c* est liée au noeud qui suit l'indentifiant de *c*. Si on a *C* clés et *N* noeuds, chaque noeud contiendra *C/N* clées.

Puisque chaque noeud connait ses successeurs, une recherche linéraire permet de trouver une clé *c*.

Chord utilise une table supplémentaire par noeud qui permet de couper une partie de la recherche.



Modèle: Cloud

Description:

- Marketing !

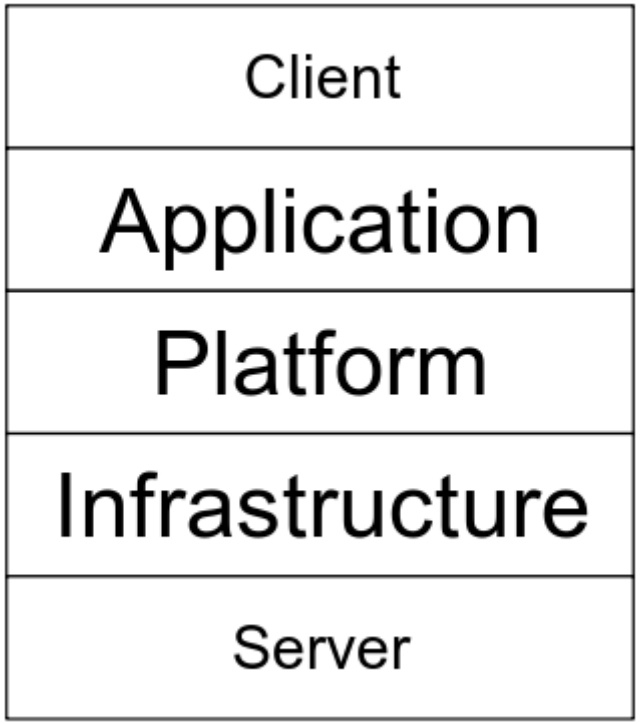
Regroupe les systèmes:

- Application
- Accès aux données
- Stockage

  Sans localisation physique particulière

  des informations/machines

Modèle: Cloud



Accède au services

Fournit des applications sous forme de services (SaaS)

Masque la complexité de la gestion matérielle et logicielle (PaaS)

Permet une virtualisation des infrastructures (IaaS)

Machine matérielle

Modèle: Cloud

Caractéristiques:

- Plus proche des utilisateurs
- Adaptabilité des resources
- Accès par API (Application programming interface)
- Coût plus maîtrisés
- Indépendance matérielle et géographique
- Centralisation des infrastructures
- Gestion des pics de charges
- Déployement des évolutions
- Tolérance aux problèmes
- Passage à l'échelle à la demande
- Surveillance des performances
- Centralisation des données mais perte du conrtôle
- Maintenance

Modèle: Cloud

Déployemenent:

- Public:
  location de services, infrastructure externe Google, Amazon

- Privé:
  infrastructure gérée par l'organisation

- Communautaire:
  regroupement d'organisations autour d'un besoin métier commun

- Hybride:
  Partage des services et infrastructure entre les plusieurs méthodes de déployements

Modèle: Cloud

Problèmes:

- Données privées
- Réglementations différentes selon les pays
- Application des licences
- Encore plus sur des cloud open source
- Interopérabilité entre les cloud inexistante
- Sécurité
- Abus
- Centralisation

Exemple: Amazon Web Services

- Large choix: OpenStack, AppScale, Cloud Foundry, ...
- Elastic Compute Unit (ECU)
- Coûts en bande passante/unité de traitement clairs
- Accessible depuis le web
- Montée en charge automatique

- 2008: spam/malware
- 2010: problèmes matériels
- 2011: coupures de courant
- 2013: faille de sécurité → publication de données privées
- 2014: contournement des règles de sécurité pour récupérer l'ensemble des profils LinkedIn

Résumé

Différents modèles d'architectures présentés:

- Mainframe
- Client-Serveur
- 3-tiers
- N-tiers
- Cluster/Grid
- Peer-to-Peer
- Cloud

Conclusion

- Chaque architecture répond à ses avantages et ses inconvénients.
- Ces différentes architectures peuvent être couplées

  Exemple: Modèle 3-tiers dont la persistence des données est gérée en Peer-to-Peer.
- Evolution des activités et des besoins =>

  Apparitions de nouvelles architectures

  (Recherches sur le flocking de données et la sémantique dans les graphs...)

Références

- Software Architecture: IEEE Standard 1471-2000
- P. Kruchten, Architectural Blueprints—The “4+1” View Model of Software Architecture, IEEE Software 12 (6), Nov. 1995, pp42-50
- Tanenbaum & van Steen, Distributed Systems, Principles and Paradigms, seconde édition
- Architecture of Distributed Systems, cours de Johan Lukkien, 2011
- Architectural Patterns Revisited – A Pattern Language, Paris Avgeriou & Uwe Zdun, 2005
- Software Architecture, Foundations, Theory, and Practice, R.N. Taylor, N. Medvidovic, E.M. Dashofy, Wiley & Sons, 2009
- Software Architecture in Practice, Second Edition, L. Bass, P. Clements, R. Kazman, SEI Series in Software Engineering, Addison-Wesley, 2003