

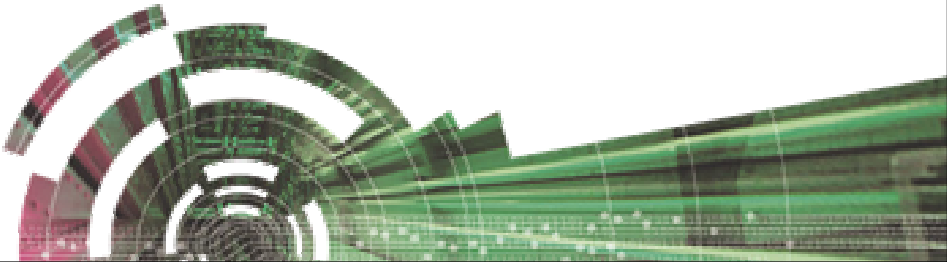


cetic
Your connection to
ICT research



Groupe de discussion

Systemes distribués

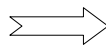


cetic
Your connection to
ICT research

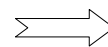
Contexte

- Explosion d'Internet a redéfini les systèmes distribués

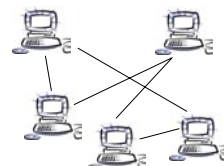
Client-server



Web



Pair à pair



Modèle Pair à Pair

■ Gains

- Absence de serveur
 - diminution des coûts de mise en œuvre
 - suppression d'une dépendance forte, augmentation de la disponibilité
- Équilibrage des participants
 - en ressource de calcul
 - en consommation des ressources réseaux (bande passante)
 - diminution des coûts de mise en œuvre et de fonctionnement

■ Pertes

- Techniques de plus en plus complexes à maîtriser
 - Connectivité, sécurité, disponibilité, tolérance aux pannes, ...



Maîtrise de la complexité

■ Disponibilité de bibliothèques pour les programmeurs

- Sockets tcp/udp (passage de message)
- Corba (appel à distance universel d'objet)
- JXTA (modèle pair à pair)
- Globus (modèle grid computing)
- ...

■ Support de distribution dans les langages de programmation eux-mêmes

- Echange de messages : Erlang
- Appel à distance (RPC/RMI) : Java
- Répartition transparente des entités du langage : Oz
- ...



Thème de la présentation

- Analyse de 3 approches correspondantes à des langages de programmation différents
 - Description des mécanismes utilisés
 - Description de la tolérance aux pannes
 - Conséquences sur les applications
- Évolution des systèmes distribués
 - Mémoire partagée
 - Systèmes pair à pair
 - Grid Computing



Envoi de message : Erlang

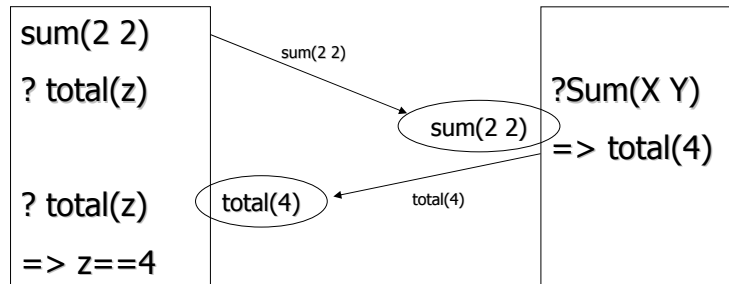
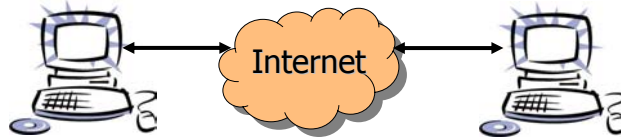
- Système fortement utilisé en téléphonie (Ericsson)
- Notion de process avec identificateur unique
 - Un canal de communication par process
 - Connaître un identificateur permet d'envoyer des messages au process correspondant
 - Une pool de réception par process
 - Extraction de message basée sur le pattern matching (opération bloquante)
 - Gestion des pannes
 - Extraction de message avec timeout





cetic
Your connection to
ICT research

Erlang : exemple



2 octobre 2003

Systèmes distribués

7



cetic
Your connection to
ICT research

Erlang : conclusions

- Tolérance aux pannes
 - Quelle valeur prendre pour le timeout ?
 - Détection pas symétrique
 - Détection pas constante
 - Que se passe-t'il si un message arrive après un timeout ?
- Structure de communication libre
 - Par simplicité : architecture Client-Serveur



2 octobre 2003

Systèmes distribués

8



cetic
Your connection to
ICT research

Appel à distance : Java RMI

- Le plus connu/utilisé !
- Mécanisme d'invocation d'une méthode distante
 - Invocation **synchrone**, comme pour une invocation locale
 - Certain degré de transparence : sous certaines conditions il n'y a pas de différence entre un appel local et un appel distant
 - Pannes gérées par des exceptions



2 octobre 2003

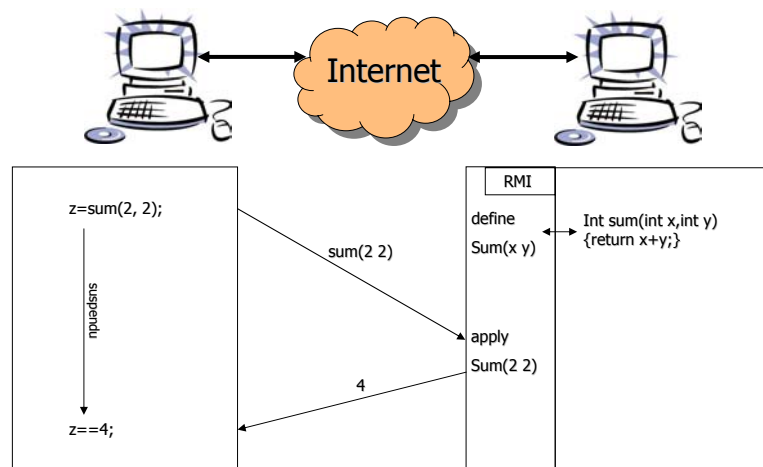
Systèmes distribués

9



cetic
Your connection to
ICT research

Java RMI : exemple



2 octobre 2003

Systèmes distribués

10

Java RMI : conclusions

- Synchronicité
 - Proche du cas local : plus simple
 - Dépendance implicite à des éléments extérieurs
 - Perte d'efficacité
- Tolérance aux pannes
 - Mécanisme d'exception
 - Synchrone
 - Intégration naturelle avec Java
- Structure de communication libre
 - Par simplicité
 - Architecture Client-Serveur
 - Clients bloquants lors des requêtes au Serveur.



Distribution transparente des entités du langage : Mozart

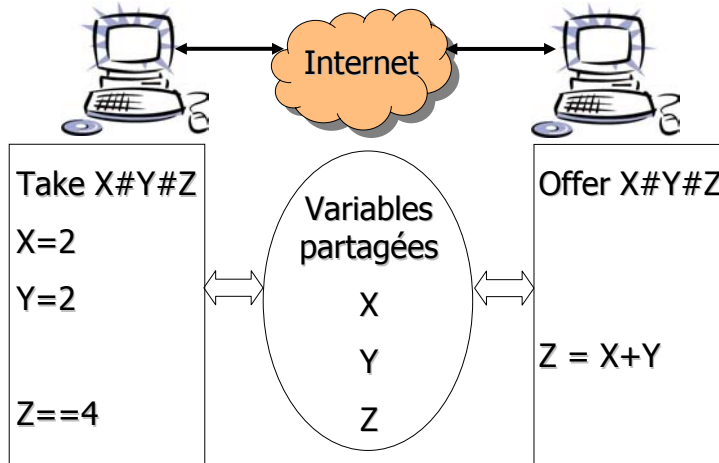
- Etat de l'art en systèmes distribués
- Distribution transparente des entités du langage
 - Chaque type possède son propre algorithme de distribution suivant son type
 - Valeurs sont copiées au vol
 - Variables sont synchronisées dynamiquement
 - A l'utilisation, pas de différence entre une variable locale et une variable distribuée
 - Sauf en cas de panne !





cetic
Your connection to
ICT research

Mozart : exemple



2 octobre 2003

Systèmes distribués

13



cetic
Your connection to
ICT research

Mozart

- Tolérance aux pannes
 - Détection du bon fonctionnement du réseau au niveau des entités elles-mêmes
 - Mécanisme d'exception
 - Mécanisme de remplacement
- Structure de communication par partage d'entités
 - Par simplicité : Architecture Client-Serveur



2 octobre 2003

Systèmes distribués

14



cetic

Your connection to
ICT research

Démos



2 octobre 2003

Systèmes distribués

15



cetic

Your connection to
ICT research

Le futur des systèmes distribués

- Eviter les approches Client-Serveur quand c'est possible :
 - Mémoire virtuelle partagée
 - Modèle Pair à Pair
 - Grid Computing



2 octobre 2003

Systèmes distribués

16

Mémoires virtuelles partagées

- Bande passante
 - mémoire: \sim gigaoctets/s
 - réseau local: \sim megaoctets/s
 - Internet: \sim kilooctets/s
- Mémoire virtuelle partagée pas efficace !
 - Mécanisme supplémentaire pour augmenter l'efficacité :
Global Store
 - Interface transactionnelle
 - Permet le calcul asynchrone anticipatif



Demo Global Store



Pair à Pair

- Décentralisation maximale
 - Pur modèle Pair à Pair : tous les pairs tiennent un rôle identique dans le réseau
 - En pratique, souvent pas le cas
 - Serveur central de Napster
 - Supernode de Kazaa
 - Serveur automatique de Global Store
 - Partage des coûts d'utilisation et de gestion du réseau entre les pairs
- Supporte des millions de participants
 - Reconfiguration automatique du réseau



Exemples Pair à Pair



Grid Computing

- Cluster hétérogène
 - Mécanisme d'interopérabilité entre les différentes plate-formes
- Apparaît comme une entité unique
 - Mécanisme d'identification unique pour tout le réseau
- Basé sur des services
 - Implantés dans le Grid
 - Jobs soumis pour les utilisateurs

