Plan

Principe de conception

Une idée clé : la transparence

Principe de conception 1

Un bon système réparti
est un système
qui semble centralisé
(qui s'utilise comme)

Principe de conception 2

Un bon système réparti
n'est pas un système centralisé

[Précis 1.5 pp.13–18]

Idée : masquer la répartition

Niveaux de transparence

- Accès
- Localisation
- Partage
- Réplication
- Fautes
- Migration
- Charge
- Échelle

Mécanismes

- Interface
- Nommage
- Synchronisation
- Groupe
- Atomicité
- Mobilité
- Réflexivité
- Reconfiguration

Transparence d'accès

Propriété

Accès à une ressource distante $\equiv$ accès à une ressource locale

Exemple

- Niveau langage de commande : sh $\neq$ ssh (non transparence)
- Niveau service système : read,write identiques que le fichier opérande soit local ou distant (transparence)
- Niveau langage à objet : appel de méthode local ou à distance identique pour l'appelant (transparence)

Solution : Notion d'interface

Cas des intergiciels à objets : langage IDL et bus logiciel

Transparence de localisation

Propriété

La localisation d'une ressource reste cachée.

Exemple

- Non transparence : commande scp bach.enseeiht.fr:/foo .
- Transparence :
  - Niveau service système : open("nom-fichier",...) : nom du fichier indépendant de la localisation du fichier
  - Niveau langage à objet : références aux objets distants sans nécessité de connaître leur localisation

Solution : Services de nommage gérant des noms globaux

Cas des intergiciels à objets : serveurs de noms

Transparence du partage

Propriété

L'usage partagé (et en parallèle) d'une ressource doit rester cohérent ($\equiv$ sémantique équivalente au cas centralisé).

Exemple

- Niveau service système : cohérence d'accès à un fichier partagé : assurer les contraintes d'exclusion mutuelle des lecteurs/rédacteurs, mais coûteux
- Niveau langage à objets : limiter l'exécution en parallèle des méthodes sur un objet

Solution : Mécanismes de synchronisation

Problème : mécanismes connus mais souvent coûteux en réparti

Transparence de la réplication

Propriété

La répartition permet la redondance pour plus de fiabilité

Exemple

- Niveau service système : assurer le maintien de plusieurs copies cohérentes d'un même fichier
- Niveau langage à objets : assurer la réplication transparente d'un objet
- Niveau intergiciel : assurer que plusieurs serveurs répliqués évoluent en cohérence

Solution : Synchronisme virtuel

Notion de groupe et de protocoles de diffusion atomique

Transparence des fautes

Propriété

La répartition induit un contexte moins fiable que celui du centralisé : panne partielle

Exemple

- Niveau service système : un service n'est plus accessible (serveur de noms !)
- Niveau langage à objets : un appel à distance de méthode peut échouer

Solution : Traitement d'exception et atomicité

Atomicité : un traitement s'exécute en entier ou pas du tout

Transparence de la migration

Propriété

Permettre la migration de code, de processus, d'agents, d'objets.

Exemple

- Niveau service système : déplacer un serveur d'une machine chargée à une machine sous-utilisée
- Niveau langage à objets :
  - code mobile : exemple des applets Java, exemple des fichiers postscript
  - objets mobiles (ou agents mobiles)

Solution : la mobilité des traitements et/ou des données

Agents mobiles (contexte d'exécution mobile), code mobile

Transparence de charge

Propriété

Masquer (et empêcher) les phénomènes de surcharge, écroulement

Exemple

La répartition permet naturellement la mise en œuvre de techniques d'équilibrage de charge

- Niveau système : reconfigurer dynamiquement les services sur les machines disponibles selon la charge des serveurs
- Niveau grappe (cluster) : répartir les traitements parallèles de façon équilibrée sur les différents processeurs

Solutions : réflexivité, machine virtuelle

Réflexivité : possibilité d'auto observation des composants
Machine virtuelle : dissocier environnement d'exécution et support matériel

Transparence d'échelle

Propriété

Permettre l'extension d'un système sans remettre en cause son fonctionnement global

Exemple

- Niveau système : introduire de nouveaux serveurs sur de nouvelles machines pour s'adapter à une augmentation de l'activité applicative

Solution : Adaptabilité et autonomie

Adaptabilité et autonomie : mise en œuvre de mécanismes automatique d'adaptation dynamique

En résumé

Répartition

$\equiv$

Accès et partage de ressources
via un réseau de communication
à tout usager qui en a le droit
et où qu'il soit