Systèmes d'exploitation, 2ème année

Appel de procédures distantes

Yves Stadler

Université de Lorraine - IUT de Metz

7 février 2012

Agenda

Plan du cours

- Qu'est-ce que les RPC
- Usage
- Implémenation en C

Principe des RPC

Transparence

- L'objectif des appels de procédures distantes :
  - Fournir un appel de fonction (`func(a,b)`)
  - S'exécuter à distance
  - Renvoyer un résultat au programme appelant.

Avantage

- Le programmeur ne se soucie pas de la gestion du réseau ! \o/
- On peut proposer une liste de service réseaux
- Le protocol est assisté

Définir le protocol

Langage RPC

```
 1 enum result_t {SUCESS, FAILURE};
 2
 3 struct etudiant {
 4     string name<>;
 5     int age;
 6 };
 7
 8 program ETUDDB {
 9     version ETUDDB_V1 {
10         result_t ADD_ETUD(etudiant) = 1;
11         result_t PRINT() = 2;
12     } = 1;
13 } = 0x2fffffff;
```

Génération des données externes

External Data Representation (XDR)

- Un fichier `nom_xdr.c` contenant les types définis
- Ne pas modifier, inclure le `.o`
- RFC 1831

Server Stub

- Contient la gestion des appels client et la liaison avec le service
- Appel des fonctions non implémentées
- Ne pas modifier, inclure le `.o`

Client Stub

- Contient les appels aux fonctions définies précédement
- Elle appelles les fonction serveur
- Ne pas modifier, inclure le `.o`

Header

- Définis les fonctions
- inclure le `.h`

Approches

Traditionnelle

- Sun ONC/RPC
- OSF DCE
- Procédures stockée des BdD

Objets répartis

- CORBA
- Java RMI
- DCOM

Systèmes de composants

- Sun J2EE EJB
- Corba Component Model
- WS-SOAP

Problèmes de réalisation

Paramètres

- Passage par référence impossible
- Conversion des données (standardisation)

Traitement des erreurs

- Temps de réponse trop long
  - Appel perdu
  - Réponse perdue
  - Défaillance serveur
- Résolution
  - Le client renvoie la demande
  - Problèmes (double réponses, etc.)

Sans état

- Le serveur ne peut pas gérer d'états
- Il faut enregistrer les données qui doivent être persistantes
- Toute requête doit contenir tout ce qu'il faut pour exécuter correctement les appels
- Prévoir des fonctions qui ont toujours le même effet avec les même paramètres.

Serialisation

Usage

- Stockage d'objet sous une forme exploitable
- Fichier, RPC, réseau, inter-langage.
- Conventions
- Marshalling, Serialization, deflating

Fonctions

- Fournies par beaucoup de framework/langages
- Manuelle (écriture de flux binaires)

Stratégie d'exécution

Migration / Rapatriement

- On amène le code sur la machine locale

Critiques

- Efficaces
- Problème d'homogénéité
- La taille du code fait varier les performances

Stratégie d'exécution

Mémoire virtuelle

- L'appel côté client génère un défaut de page
- On va chercher la page sur le serveur

Critiques

- Efficaces pour de nombreux appels
- On peut utiliser des pointeurs
- Systèmes homogènes requis
- La mémoire répartie doit rester cohérente.

Stratégie d'exécution

Messages asynchrones

- La requête client est interceptée par un wrapper
- Le wrapper gère l'appel distant
- Le serveur reçoit les infos de l'appel et l'exécute
- Le serveur renvoie la réponse au wrapper.

Critiques

- Taille quelconque
- Hétérogénéité possible
- Transactionel
- Pas de pointeurs
- Échanges de types complexes compliqué
- Peu efficace pour beaucoup d'appels

Stratégie d'exécution

RPC léger

- On se passe des wrappers

Critiques

- Uniquement en local
- Si threads, ok
- Si fork, Segment de mémoire partagée

Client bloqué

Mode asynchrone

- Mon client est bloqué lors des appels
- Solution : créer un thread pour l'exécuter pendant que le client continu son boulot.
- Le client récupère l'information quand ça l'arrange.

Serveur bloqué

Parallélisation du serveru

- Le serveur peut-être en mode séquentiel
- Si le serveur est en mode parallèle, il faut gérer la concurrence !

Gestion des données

Stateless

- Les paramètres sont suffisants pour générer la sortie (maths)
- Cas simple, peu de problème

Manipulation de données

- Contrôle de concurrence
- Que se passe-t-il si panne ?
- Il faut gérer des transactions

Information d'historique

Sans état

- Une opération n'a pas besoin de connaître des informations d'une opération précédente

Avec état

- Nécessite des informations historiques
- Usage d'un descriptif pour chaque relation client-serveur