RPC

mardi 15 octobre 2024 15:25

Applications structurées en 2 parties :
- Programme principal
- Ensemble de procédures

Programme principal et procédures sont compilés et liés
Au moment de l'exécution :
- Le programme principal appelle les procédures
- Les procédures s'exécutent et retournent leurs résultats

Le programme principal se comporte comme un client
l'ensembles des procédures est assimilable à un ensemble de services disponibles sur un serveur
- Interface d'un service = signature de la procédure (au sens large : fonctions, méthodes …)
- Interface du serveur = ensemble des signatures des procédures

Si dans la machine le client fait appel des services d'un partenaire distant il y aura des erreurs
Les procédures seront remplacées par des fausses procédures en local. Ces fausses procédures sont appelées stubs client

Le stub client :
- Ils portent le nom de la vrai procédure qu'il remplace
- Donne l'illusion au programme principal que la procédure est locale
- Remplace le code de la vraie procédure par un autre code
    - Gère la connexion avec le bus middleware
    - Transmet les paramètres vers la machine où se trouve la procédure
    - Récupère le(s) résultat(s)

Il est impossible d'avoir un programme exécutable composé uniquement de procédures
Il est donc nécessaire d'avoir un programme principal appelé stub serveur

Le stub serveur :
- Permet de créer un exécutable contenant les procédures du serveur
- Gère la communication avec les stubs client
    - Active la procédure désignée en lui transmettant les paramètres s'appel
    - Retourne les valeurs de résultat au stub client

Middleware par appel de procédures distantes :
Assure la transparence de localisation :
- Le client appelle les procédures comme si elles étaient locales
- Le middleware assure la communication avec le serveur
- l'interface du serveur est décrite en IDL (langage de définition d'interface)
- Le code de préparation d'une requête (stub client) est généré automatiquement à partir de la description IDL

Caractéristiques :
- Codes client et serveur indépendants du système de communication
  Le client ne sait pas si la procédure est locale ou distante
- Le code client n'a pas à préparer le message ni à localiser le serveur
  => à la charge de middleware RPC
- Système de dialogue entièrement externe au client et au serveur décrit dans un langage spécifique (IDL) à partir duquel est généré automatiquement le code nécessaire
- Structure de communication construite au moment de la compilation

- Communication synchrone --> le client attend la réponse à son appel de procédure avant de continuer son traitement
- Technologie RPC entièrement standardisée (inclure IDL + services nécessaires à le communication)

Communication entre les deux stubs :

1. Le client récupère tous les paramètres de l'appel et fait un paquet avec ces paramètres => marshalling des paramètres
2. Le paquet est envoyé par le protocole de transport au serveur RPC (talon serveur / skeleton)
3. Le serveur reçoit le paquet et le déballe (unmarshalling)
4. Appel de la procédure / exécution
5. Le résultat est reçu à la fin de l'appel et emballé
6. Le paquet contenant le résultat est envoyé au client
7. Le client déballe le résultat

Le contrat :

- C'est le point le plus important dans le RPC car c'est lui qui va permettre de générer les différents stubs qui communiquent ensemble
- Le contrat est possédé par le client et le serveur, c'est l'élément comment qui permet de les liés
- Formalise le dialogue entre le client et le serveur
- Répond aux questions : Que transmet-on ?, Où envoi-t-on les données ?, Qui reçoit les donnée ?

Construction du client et du serveur :
Élément à développer

- Le programme principal de l'application (le client)
- Les procédures composant l'application (le serveur)
- Le contrat décrivant les échanges entre le client et le serveur (écrit en langage IDL)

Étapes de déploiement

- Générer les stubs client et serveur (compilateur IDL)
- Construire l'exécutable client
  - Compiler le programme principal et les stubs client
  - Lier les deux
- Construire l'exécutables serveur
  - Compiler le stub serveur et les procédures
  - Lier les deux

Caractéristiques du contrat :
…

La représentation des données est un problème classique dans les réseaux
Conversion nécessaire si le site client et le site serveur

- n'utilisent pas le même codage
- Utilisent des formats internes différents (type caractère, entier, flottant, …)
- Solution placée classiquement dans la couche 6 du modèle OSI présentation

Les difficultés :

- Appel de procédure local
  - Appelant et appelé sont dans le même espace virtuel
  - Même mode de pannes
  - Appel et retour de procédure sont des mécanismes internes considérés comme fiables
  - Dans certains langages mécanismes d'exception
- Appel de procédure à distance
  - Appelant et appelé sont dans 2 espaces virtuels différents
  - Pannes du client et du serveur sont indépendantes

    - Possibles pannes du réseau de communication (perte du message d'appel ou de réponse)
    - Temps de réponse long (charge du réseau ou du site)
- Passage des paramètres
    - Valeur (pas de problème en particulier)
    - Copie / restauration
    - Référence
        - Utilise une adresse mémoire centrale du site de l'appelant
        - Aucun sens pour l'appelé
        - Solutions :
            - Interdiction totale
            - Simulation en utilisant une copie de restauration
    - Solutions généralement prises
        - IN
        - OUT
        - …

Désignation :
Objets à désigner

- Le site d'exécution + le serveur + la procédure
- Désignation globale indépendante de la localisation

Désignation statique ou dynamique

- Statique : localisation du serveur connue à la compilation (dans les stubs lors de la génération)
- Dynamique : non connue à la compilation (dans la programme et non dans les stubs)
  Objectif :
    - Séparer la connaissance du nom du service de la sélection de la procédure qi va l'exécuter
    - Permettre l'implémentation retardée

Liaison et fonctionnement :
Liaison (détermination de l'adresse du serveur)

- Liaison statique (pas d'appel à un serveur de nom ou appel lors de la compilation)
- Liaison au premier appel
- Liaison à chaque appel
- DNS internet (solution classique)

Les limites :

- Mécanisme de bas niveau (la notion de procédure n'existe pas dans les méthodes d'analyse)
- n'assure pas tous les services souhaités d'un bus de communication
- Outils de développement
    - Limités à la génération automatiques des stubs
    - Peu d'outils pour le déploiement

Les problèmes :

- Traitement des défaillances
    - Cogestion de réseau ou serveur
    - Panne du client pendant le traitement de la requête
    - Panne du serveur…
- Problèmes de sécurité
    - Authentification du client
    - Authentification du serveur
    - Confidentialité des échanges
- Désignation et liaison
- Aspects pratiques
    - Adaptation à des conditions multiples (protocoles, langages, matériels)
    - Gestion de l'hétérogénéité