Sumário

- O que é o RPC.
- Problemas.
- Passagem de parâmetros.
- IDL e rotinas de adaptação.
- Concretização.
- RPC e replicação.

Chamada a procedimentos remotos

- Ideia chave:

  Chamar um procedimento que se executa noutra máquina.

  Birrel and Nelson, "Implementing Remote Procedure Calls", ACM Transactions on Computer Systems, vol 2, pp 39-59, Fevereiro de 1984.

Dificuldades

- Cada processo tem o seu espaço de endereçamento próprio.

  Designação do procedimento.

  Passagem de parâmetros (por valor e por referência).

  Linearização de estruturas complexas.

- Heterogeneidade:

  As máquinas podem possuir arquitecturas diferentes.

- Falhas:

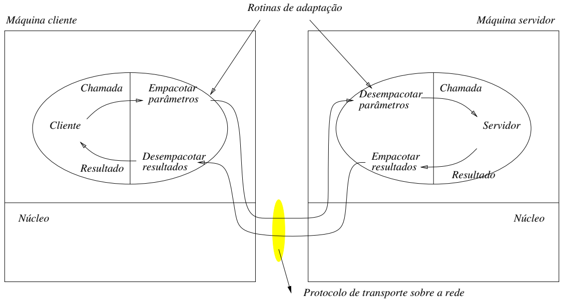
  Na rede (omissões, partições), nos nós e nos processos.

Transparência

- Como tornar um RPC semelhante a uma chamada local?
- Rotinas de adaptação.

  Do lado do cliente e do lado do servidor.

Rotinas de adaptação



Parâmetros: diferentes arquitecturas

- É necessário saber quais as arquitecturas em causa.
- Idêntica representação: *basta copiar dígito a dígito.*
- Representações diferentes:

  Transformar no cliente, para o formato servidor.

  Transformar no servidor.

  Em ambos, para uma forma canónica.

Parâmetros: entrada e saída

- Saber quais os valores que são de entrada e quais os valores que são de saída (ou ambos):

  Por vezes necessário anotar cada parâmetro.

- Passagem por referência: proíbida ou com restrições.

  Usando anotações.

  Só de alguns tipos.

Parâmetros: linearização

- É difícil passar estruturas complexas:

  Listas ligadas.

  Grafos.

- Muitas vezes tem de ser feito manualmente.

Parâmetros: objectos e dados

- Distinguir dados de objectos (referências para servidores).

  Os dados são copiados e acedidos localmente.

  Os servidores são acedidos por RPC.

- Modelos uniformes:

  Tudo são objectos.

Linguagens de definição de interfaces

- "Interface Description Languages":

  Permitem definir a interface de modo independente da linguagem de programação.

  Possível concretizar um dado serviço usando diferentes linguagens e arquitecturas.

  Cliente e servidor podem ser desenvolvidos em diferentes linguagens.

- Mais restritas que linguagens que não consideram a distribuição.

Geração de rotinas de adaptação

- Codificar a linearização de modo automático:

  A partir da descrição da interface geram o código que empacota e desempacota os parâmetros de modo automático.

- Se a partir de um IDL, dois passos:

  Criar os tipos correspondentes aos parâmetros na linguagem alvo;

  Criar as rotinas que des/encapsulam esses tipos.

Associação dinâmica

- Permitir ao cliente saber, em tempo de execução, onde se encontra o servidor.

- Serviço de associação (binding).

- O servidor regista-se no serviço:

  Indicando o tipo de serviço, nome e modo de acesso (ex: protocolo e endereço).

- O cliente obtém o modo de acesso por invocação do serviço de associação.

  Método de acesso a este serviço está pré-definido.

Falhas: o servidor não é localizável

- Possíveis causas:

  O serviço já não é prestado; o servidor mudou de versão; a máquina onde o servidor se executa está em baixo; existe uma partição na rede.

- Solução:

  Pouco há a fazer excepto em casos particulares (voltar a tentar).

- Problema:

  Como indicar o erro ao cliente (está previsto o código de erro?).

  Como manter a transparência?

Falhas: perde-se um pedido ou uma resposta

- Depende da semântica dos protocolos usados na comunicação:

  Se a comunicação não fôr fiável, é necessário retransmitir o pedido.

  Pode-se re-executar o pedido ou o servidor deve descartar os duplicados?

- Se o problema persistir, degenera no caso anterior:

  "Pequeno" problema: será que o serviço foi executado?

Falhas do servidor

- Dois problemas.
- Foi mesmo o servidor que falhou ou foi a rede?
- Falhou antes, durante, ou depois da execução do pedido?

Três tipos de semântica

- Pelo menos uma vez:

  Ir tentando até que o servidor (ou a rede) recupere (assume que irão recuperar em tempo útil).

- No máximo uma vez:

  Abortar mal é detectado um problema.

- Exactamente uma vez:

  É necessário que o servidor recupere e mantenha algum estado. Pode ser impossível se envolver o acções com o mundo exterior.

Falhas do cliente

- O servidor continua a executar um pedido (orfão): *Gasta CPU. Se o clinte re-incarna, podem-se confundir as duas encarnações do mesmo pedido.*
- Soluções:

  O cliente tem memória estável e elimina todos os orfãos explicitamente.

  O cliente possui um número de incarnação, que destingue as várias "épocas" de actividade.

  Definir um tempo máximo para a execução do RPC (qual?).

  Problema: como se mata um orfão sem colocar o sistema num estado incoerente?

Concretização de protocolo de RPC

- Diversos aspectos que necessitam de ser considerados para obter bom desempenho:

  Gestão da rede (e da interface com a mesma): máximizar o débito e minimizar a latência (e o número de omissões).

  Minimizar o número de cópias entre espaços de endereçamento.

  Minimizar o número de trocas de contexto.

  Caminho crítico.

Gestão de temporizadores

- Problema de concretização:

  Listas de espera diferenciais, "rodas do tempo", etc.

- Problema de configuração:

  Que valores dar aos temporizadores?

Servidores replicados

- Que tipo de protocolo utilizar?

  Depende do tipo de estratégia utilizada para manter o servidor replicado.

- Primário/secundário:

  Enviar os pedidos para o primário; re-encaminhar para o secundário em caso de falha do primário, receber a resposta de qualquer réplica.

Servidores replicados (continuação)

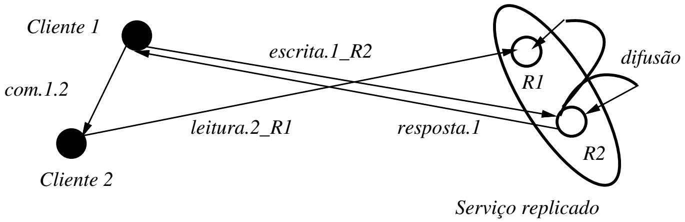
- Máquina de estados:

  Fazer o pedido chegar a todas as réplicas; receber a resposta de qualquer réplica.

  Fazer o pedido chegar a uma réplica que por sua vez o difunde por todas as outras réplicas.

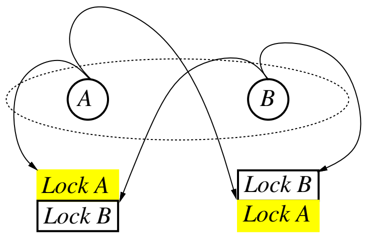
Ordenação causal

Necessária para manter o cliente como uma vista consistente do sistema.



Ordenação total

Necessária para manter os servidores consistentes.



Associação com RPC replicado

- Método de accesso mais complexo.

  Pode ser um endereço lógico de grupo: simplifica este nível mas como se utiliza?

  Pode ser uma lista de endereços: como se mantém coerente com a filiação do servidor?

Clientes replicados

- Detectar "réplicas" do mesmo pedido.
- Problema:

  Como identificar os pedidos para reconhecer "réplicas"?
- Possivelmente, comparar os pedidos.

  "Collation" (escolher um pedido qualquer, ver se são todos equivalentes, votar os pedidos, etc).

Resumo

- RPC ponto a ponto.
- RPC replicado.