Estudo de caso sobre o RPC.

O RPC *(Remote Procedure Call)* ou chamada de procedimento remoto é uma tecnologia usada para que processos se comuniquem mesmo que um processo esteja em uma máquina e chame remotamente um procedimento em outra máquina. Usado na computação distribuída, o RPC é bastante usado na implementação do modelo cliente-servidor onde o cliente chama remotamente um procedimento do servidor que executa o mesmo e retorna algo para o cliente. Um detalhe importante é que se o servidor chamado remotamente não estiver disponível a chamada irá falhar pois não há garantia de que o procedimento foi invocado, o modelo de chamada remota se assimila com o de chamada local onde os argumentos de uma rotina são colocados em uma área conhecida da memória que depois é transferida para o processo em execução que irá ler os mesmos. Esse processo de invocação ocorre de forma similar. É usado um *Thread* para controlar os processos cliente e servidor, o cliente manda uma mensagem para o servidor e aguarda a resposta pois é ***AUTO BLOQUEANTE***, a mensagem do cliente contem parâmetros de procedimento e a mensagem que contem a resposta do procedimento.

O processo servidor aguarda uma uma mensagem de invocação, quando recebida o servidor extrai os parâmetros e processa os resultados e envia a resposta ao cliente, o servidor novamente voltará ao estado de espera pois apenas um dos processos permanece ativo o cliente por padrão finaliza sua execução quando recebe uma mensagem de resposta.

As diferenças entre chamadas locais e remotas estão nos pontos de: Tratamento de erros; onde falhas na rede ou no servidor devem ser tratadas. Variáveis globais e efeitos colaterais; onde não permite a passagem de parâmetros do cliente para o servidor caso o servidor não possua acesso aos espaços de memória do cliente. Desempenho; Chamadas locais operam em velocidades maiores que as remotas.

Autenticação; Essas chamadas são transportadas pela rede de forma INSEGURA muitas das vezes, necessitando de autenticação, por isso a escolha do protocolo de transporte deve ser feita de forma cuidadosa pois os códigos gerados por ferramentas do RPC não garantem a autenticação.

Formas de Transporte.

O RCP pode ser implementado sobre diferentes tipos de protocolo pois há diferença entre as mensagens transmitidas pelos processos, já sabemos que o RPC não implementa nenhum tipo de confiabilidade por isso cabe a aplicação tomar cuidado com o tipo de protocolo a ser utilizado. Se for implementado pelo ***TCP*** as preocupações são resolvidas pois s trata de um protocolo confiável, caso o protocolo de transporte não seja confiável como o ***UDP***, alguns mecanismos: timeout, retransmissão e detecção de duplicatas devem ser implementadas.

O RCP não muda sua forma de chamadas remotas nem seus requisitos de execução pelo fato de que o protocolo da camada de transporte é escolhido de forma independente.

Implementação do RCP.

Afim de criar uma forma de acesso transparente pelo cliente e servidor, foram criados vários sistemas padronizados de RCP, a maioria usa IDL (*Linguagem de Descrição de Interface*) para que as diferentes plataformas possam chamar os procedimentos.

O *RCPGEN* é uma ferramenta que permite a criação de uma interface entre o cliente e servidor a partir da IDL, essa interface chamada de stub é embarcada nas aplicações cliente e servidor, portanto o RPC não é uma camada de middleware.

O procedimento cliente chama o stub(cliente) como um simples procedimento local, o stub de cliente inicia o processo de transmissão para o servidor empacotando a mensagem, quando a mensagem chega ao servidor a mesma é desempacotada e o processo é invocado localmente o processo pelo servidor, aguarda a resposta da chamada local onde o stup do servidor empacota a resposta e envia para o cliente, onde o stub do cliente irá desempacotar a resposta e retornar ao procedimento cliente local que fez a chamada remota.

No momento em que uma chamada remota é realizada devemos nos atentar que os processos tanto cliente quanto servidor podem estar rodando em plataformas diferentes onde a representação de dados também é feita de forma diferente, neste caso é necessário um protocolo comum de representação de dados, exemplo; *XDR* ou então garantir que ambos os processos saibam converter os dados para os tipos suportados.

Autor: Lucas Alves da Costa.
Disciplina: Sistemas Distribuídos – 2018/02 – T.A.D.S