Lab01 – Construindo aplicações distribuídas usando RPC

Disciplina: Programação de sistemas paralelos e distribuídos
Professor: Fernando W Cruz Turma A

Aluno: Felipe Boccardi Silva Agustini Matrícula: 180119818
Aluno: Giovanna Borges Bottino Matrícula: 170011267

Introdução:

Chamada remota de procedimento (RPC) é uma tecnologia de comunicação entre processos que permite a um software chamar um procedimento em outro espaço de endereçamento. O programador não se preocupa com detalhes de implementação dessa interação remota: do ponto de vista do código, a chamada se assemelha a chamadas de procedimentos locais.

Objetivo do trabalho:

Neste trabalho é proposto então o desafio de construir uma pequena aplicação distribuída (linguagem C), que descobre o maior e o menor valor em um vetor de 10 posições, incluindo passagem de parâmetros, envolvendo módulos cliente e servidor usando a arquitetura RPC.

Descrição da solução:

O git pode ser encontrado no seguinte link:
<https://github.com/giovannabbottino/pspd_unb>

Nossas contribuições e reuniões podem ser observadas pela contribuição no git em <https://github.com/giovannabbottino/pspd_unb/commits/main>. Para avaliar nosso projeto recomendo uma leitura do nosso server.c e do client.c que tentamos deixar comentado para o entendimento. Além disso, também recomendo seguir o README enviado.

Em nosso trabalho, não utilizamos do **rpcgen** da SUNMicrosystens, por isso não fizemos um arquivo de definição de interface. Fizemos isso, porque queríamos gerar a aplicação de forma modular usando programação convencional.

Fizemos um cliente servidor utilizando sockets, usamos o protocolo TCP, orientado a conexão. Como não estava definido na atividade, usamos de um arquivo <*properties.h*> para definir o host, a porta, o tamanho do vetor e o pdu. Dessa forma, não precisamos enviar como argumento sempre. Isso também permite o client e o server terem acesso ao tamanho do vetor e permite alterar para 1.000.000.000 com facilidade.

Além disso, enviamos cada posição do vetor em mensagens individuais para o server. Quando se considera o TCP, uma conexão de fluxo de bytes que não envia pacotes, não é uma boa escolha enviar uma *scruct* ou vetor de *float*, já que você enfileira um buffer de dados para transmissão e a pilha TCP envia os bytes no buffer conforme escolher.

Isso significa que você pode enviar 2 estruturas de 11 bytes cada, mas dependendo do tempo e do meio de transporte, a extremidade receptora pode receber 1 pedaço de 22 bytes, 2 pedaços de 11 bytes, 22 pedaços de 1 byte ou qualquer outra combinação somando 44 bytes no total. O software receptor tem que conseguir lidar com isso e reconstituir os bytes de volta em sua estrutura. Por essa dificuldade, enviamos as posições individualmente.

A diferença entre a solução com um worker da solução com dois workers. Para isso, adicionamos um novo socket no client que se conecta ao segundo server. Não é uma solução viável para muitos workers, por isso não conseguimos fazer a opção bônus.

Felipe - Achei o laboratório muito bom, pudemos colocar em prática o que foi visto em sala além de conhecimentos obtidos em fundamentos de redes. tivemos um desentendimento na primeira parte onde não usamos o rpcGen. Acredito que ambos tivemos uma participação equilibrada. Nota 10;

Giovanna - Foi visto desde o começo que o projeto é relativamente mais complicado que os feitos em fundamentos de redes, mas que os conceitos são reaproveitáveis. Tivemos algumas dificuldades no começo do projeto por fazermos o código. A participação mútua foi bem dividida e equilibrada, não havendo muito peso para nenhum dos lados, tanto na parte de pesquisa quanto na de codificação. Nota 10.