Universidade Federal de Santa Catarina – UFSC
Departamento em Informática e Estatística
INE 5645 – Programação Paralela e Distribuída
Semestre 2024/2

Definição do Trabalho 2: Programação Distribuída

Este trabalho visa exercitar programação distribuída com sockets e explorar o uso de padrões de projeto para programação distribuída.

Implementação de um serviço de transferência de arquivos em rede

Este trabalho consiste em desenvolver um serviço de troca de arquivos em rede. Este serviço tem funcionalidade semelhante a dos comandos scp e FTP, por exemplo, onde um comando executado permite que um cliente envie (receba) um arquivo para (de) um nodo remoto. Para a comunicação distribuída, serão utilizadas APIs de comunicação que implementam Berkeley Sockets.

O exemplo a seguir ilustra o uso do comando `remcp` (acrônimo para *remote copy*), que será implementado. Na primeira linha, o arquivo `meu_arquivo.txt` é enviado do diretório corrente para o nodo remoto `192.168.0.5` e é salvo no diretório `/home/usuario/teste`. Na segunda linha, o comando `remcp` envia o arquivo `arq1.png` localizado no diretório `/home/usuario/teste/` para a pasta `/tmp` do nodo remoto. Para a identificação dos caminhos de origem e destino, assuma a representação de caminhos de diretório e arquivos utilizados em sistemas baseados em UNIX.

```
$ ./remcp meu_arquivo.txt 192.168.0.5:/home/usuario/teste

$ ./remcp 192.168.0.5:/home/usuario/teste/arq1.png /tmp/
```

O comando `remcp` permite a parada e reinicialização da transferência de arquivo, no caso de falhas ou interrupção do serviço durante o uso. Para isso, a cada 128 bytes, o comando cria ou atualiza um arquivo temporário, representado pelo nome do arquivo seguido pela extensão `.part`. Por exemplo, o arquivo `arq1.png` manteria a cópia parcial do conteúdo recebido no arquivo `arq1.png.part`,o qual seria criado após receber 128 bytes e atualizado ao alcançar os tamanhos 256, 384, 512, e assim sucessivamente, até que não tenha mais conteúdo para transferir. Ao completar a transferência, o arquivo nome `arq1.png.part` é renomeado para nome `arq1.png`.

Ao iniciar uma transferência, o serviço verifica se há uma cópia parcial do arquivo em curso. Caso encontre o respectivo arquivo `.part`, o serviço verifica o tamanho do arquivo parcial e transfere apenas o conteúdo restante.

Para que possa atender requisições de transferência de arquivos, cada nodo deve executar um serviço servidor (*daemon*), chamado `remcp-serv` em *background*. O servidor é **orientado a conexão** e atende requisições em uma porta conhecida pelos clientes.

O servidor pode atender **múltiplos clientes simultaneamente**. Para evitar sobrecarga e monopolização da rede, o servidor transfere arquivos respeitando uma **taxa de transferência** limitada e configurável, dada em **bytes/s**. Por exemplo, se a taxa de transferência for de 256 bytes por segundo, o servidor não pode enviar mais do que 256 bytes por segundo. Pode-se utilizar a operação `sleep` para garantir que o servidor interrompa momentaneamente as escritas nos *sockets* dos clientes, caso o limite de transferência seja alcançado, retomando próximas escritas no socket no segundo subsequente. Se o servidor estiver atendendo mais de um cliente simultaneamente, a taxa de transferência deve ser dividida, de modo a permitir transferências simultâneas sem exceder a taxa de transferência. O padrão de projeto ***throttling*** pode servir como referência para auxiliar na implementação deste recurso.

O servidor atende um número máximo de transferências simultâneas. No caso de um cliente tentar transferir um arquivo de/para um servidor que esteja operando no seu limite, o servidor retorna uma mensagem de erro e o cliente irá tentar retomar a transferência novamente. O padrão de projeto ***retry*** pode servir como referência para auxiliar na implementação deste recurso.

Execução

Para compreender os impactos na execução do serviço, os arquivos utilizados, número de clientes simultâneos e parâmetros do servidor podem ser alterados. É importante que o grupo explore diferentes cenários de execução para avaliar a influência dos parâmetros na divisão de carga e retentativas de transferência. O programa pode ser desenvolvido em qualquer linguagem de programação, mas a programação distribuída deve utilizar Berkeley Sockets.

Entrega

O trabalho consiste em:

1. Implementar um programa que simule a execução do serviço seguindo os requisitos descritos acima;
2. Breve relatório que indique as principais decisões e estratégias de implementação utilizadas, instruções sobre como compilar e executar o código produzido, além de exemplos de saídas de execução com diferentes parametrizações. Ao final, discuta quais conclusões você observa ao variar parâmetros do sistema.

O trabalho pode ser realizado em **grupos de até 3 participantes**. O trabalho será apresentado em sala de aula.

O código-fonte e relatório devem ser enviados pelo Moodle para análise e avaliação. Os nomes dos participantes do grupo devem constar no relatório entregue no Moodle. Participantes com nomes não referenciados não serão considerados membros do grupo.

Referência de apoio

https://learn.microsoft.com/en-us/azure/architecture/patterns/