Universidade Estadual de Mato Grosso do Sul

Curso de Ciência da Computação

Disciplina de Redes de Computadores

Trabalho de Redes de Computadores

IMPLEMENTAÇÃO DE UM SISTEMA ESCALONADOR PROCESSADOR NO MODELO CLIENTE-SERVIDOR.

Dourados (MS), 15 de maio de 2024.

Escalonadora-Processadora

Computação em nuvem é um termo coloquial para a disponibilidade sob demanda de recursos do sistema de computação, especialmente armazenamento de dados e capacidade de computação, sem o gerenciamento ativo direto do utilizador. O termo geralmente é usado para descrever centros de dados disponíveis para muitos utilizadores pela internet.

Nuvens em grande escala, predominantemente hoje em dia, geralmente têm funções distribuídas em vários locais dos servidores centrais. Se a conexão com o utilizador for relativamente próxima, pode ser designado um servidor de borda. O armazenamento de dados é feito em serviços que poderão ser acedidos de qualquer lugar do mundo, a qualquer hora, não havendo necessidade de instalação de programas ou de armazenar dados. O acesso a programas, serviços e arquivos é remoto, através da internet.

Objetivos do Trabalho

O principal objetivo do trabalho é praticar a programação com a biblioteca Socket utilizando o protocolo TCP com o servidor tratando múltiplas conexões. O trabalho consiste em desenvolver um sistema que permita ao usuário enviar enviar um arquivo fonte a uma máquina escalonadora e, posteriormente, esta máquina escalonadora enviar o arquivo a uma máquina processadora. A máquina processadora deverá compilar, executar e devolver ao cliente, o resultado da execução.

Em outras palavras, deverá ser implementado uma nuvem para a compilação e execução de programas em uma determinada linguagem da sua escolha. Você vai construir um cliente que recebe como parâmetro de entrada o nome do arquivo onde está o programa fonte. Este cliente se conecta com um servidor que funciona como um portal da nuvem e recebe do programa fonte do cliente.

Pode-se usar TCP ou UDP. Além do servidor portal você deve também implementar um servidor de processamento (também pode ser TCP ou UDP). Este servidor de processamento deve ser executado em pelo menos 3 máquinas diferentes do Lab1. O servidor portal escolhe um dos servidores de processamento para compilar e executar o programa. Veja que o servidor de processamento retorna o resultado para o servidor portal da nuvem que, por sua vez, retorna o resultado para o cliente. Além da saída correta do programa, devem também ser retornados erros de compilação ou execução, caso ocorram.

Os programas poderão funcionar tanto em IPv4 ou IPv6 utilizando o protocolo TCP.

Programas a serem Desenvolvidos

O trabalho prático consistirá na implementação de três programas, o cliente, o portal e o servidor. Ambos receberão parâmetros de entrada como explicados a seguir:

cliente <ip/nome_maquina> <porta>

O primeiro programa (cliente) irá se conectar ao portal por meio do endereço IP (ou nome da máquina) e porta passados como parâmetro. Após a conexão deverá ser fornecida a opção de envio de um ou mais arquivos fontes de uma única vez. Essa opção de envio deverá ser seguida da possibilidade de escolha dos arquivos.

O porta não possuirá qualquer interação do usuário. Ele receberá a conexão do cliente e se conectará ao servidor de processamento. Para a seleção do servidor que irá compilar e executar o programa fonte, deverá ser implementados dois critérios. Um aleatório e um Round-Robin. No método Round-Robin primeiro é escolhido o servidor 1, depois o servidor 2 e posteriormente o servidor 3. Depois volta-se a escolher o servidor 1, 2 e assim por diante. No caso do envio de mais de um arquivo simultaneamente pelo cliente, o portal deverá fazer a distribuição (1 arquivo para 1 servidor) de acordo com o método escolhido.

Já o servidor (processadora) receberá a conexão do portal. Não há interação do usuário com o servidor. Este deverá receber o arquivo fonte, compilá-lo, executá-lo e devolver ao portal o resultado da execução. Este resultado pode ser um erro de compilação, um resultado errado ou, um resultado correto de execução. Por fim, o portal devolve ao cliente o resultado recebido o servidor (processadora).

Protocolo Rede

O cliente enviará apenas um tipo de mensagem ao portal: **envio (S) [arq1, arq2, …, arqN]** e terá um comando para listar os arquivos presentes no cliente **lista (L).** O cliente obrigatoriamente deverá ficar preso em um loop infinito, recebendo as mensagens, imprimindo seus resultados e, estar pronto para novamente para uma nova mensagem.

O modo como o cliente e o portal se comunicam (se usam string ou struct) não faz diferença, porém a saída do programa cliente deve seguir exatamente as mensagens da tabela a seguir e nada mais.

A mensagem de envio (S) deverá ser lida da seguinte maneira:

S [nomes_dos_arquivos]

Exemplo: S [soma.c, quadrado.c, inverso.c]

O programa cliente deverá imprimir na tela **APENAS** a seguinte mensagem:

Mensagem na Tela	Explicação
[retorno]	Sucesso, no lugar de [retorno], coloque o conteúdo recebido pelo portal referente à execução do arquivo fonte.

A mensagem de lista (L) deverá ser lida da seguinte maneira:

L

Exemplo: L

Ao receber o comando (L), o cliente deverá listar na tela os arquivos fontes disponíveis para envio.

O programa cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de lista:

Mensagem na Tela	Explicação
L [arquivo 1] [arquivo 2] ……..	Sucesso, no lugar de [arquivo 1] coloque o nome dos arquivos. Se não houver nenhum arquivo, retorne apenas “L 0”.

Tanto o portal quanto os servidores não deverão imprimir texto algum na tela. O portal apenas receberá o(s) arquivo(s) e encaminhará ao servidor e retornará o cliente o resultado da computação feita no servidor. E o servidor apenas receberá o arquivo fonte, compilará o arquivo, executará o arquivo e devolverá o portal o resultado.

IMPORTANTE 1: A maneira como as mensagens serão gerenciadas pelo seu programa não fará parte da avaliação. Não é necessário que se guarde os dados em arquivos. Sendo permitida a perda de todos os dados ao fechar o servidor.

IMPORTANTE 2: Não colocar nenhum outro tipo de saída na tela, como mensagens de boas vindas ou "esperando comando". Ao iniciar o cliente, não imprima nada e apenas espere a entrada de dados do usuário e imprima o resultado de acordo com as tabelas apresentadas.

EXECUÇÃO DO TRABALHO

Este trabalho deverá ser feito da seguinte forma:

1. Trabalho individual;
2. Poderá ser feito em C, C++ ou Python. Para qualquer outra linguagem, falar com o professor para discutir os detalhes.
3. Você poderá escolher entre multiplexing, fork ou threads.
4. Fazendo em C, C++ ou Python você deverá enviar sua solução com um Makefile.

ENTREGA DO CÓDIGO

O código e a documentação devem ser entregues em um arquivo Zip. Dentro deste arquivo Zip devem conter um **readme.txt** com o nome do estudante e o comando para a execução do código e, um arquivo PDF da documentação. Inclua todos os arquivos fontes (.c, .h, makefile, **não inclua executáveis ou arquivos objetos**), em um único diretório. Um **Makefile** deve ser fornecido para a compilação do código.

Parte desse trabalho envolve o aprendizado de como construir um makefile e utilizar a ferramenta make. Este makefile, quando executado sem parâmetros, irá gerar os três programas: cliente, portal e servidor, EXATAMENTE com esses nomes. No caso da linguagem python você poderá gerar um binário por meio do comando chmod +x.

Submissões onde os programas não sigam as especificações de parâmetro e nomes, makefile não funcionando ou arquivos necessários faltando **NÃO SERÃO CORRIGIDOS.** Programas que não compilem também não serão corrigidos.

DOCUMENTAÇÃO

O texto da documentação deve ser breve, de forma que o professor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deverá conter os seguintes itens:

1. O sumário do problema a ser tratado;
2. Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, procedimentos e das decisões de implementação;
3. Decisões de implementação que porventura estejam omissos na especificação;
4. Como foi tratada a retransmissão de mensagens;
5. Testes, mostrando que o programa está funcionando de acordo com as especificações, seguidos de sua análise;

6. Print screens mostrando o correto funcionamento do cliente e do servidor e exemplos de testes executados;
7. Conclusão e referências bibliográficas.

AVALIAÇÃO

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Todos os alunos deverão apresentar e explicar em laboratório o trabalho.

Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, estruturado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc);
- Execução correta do código com entrada de testes a serem definidas no momento da avaliação. As entradas de testes irão exercitar a funcionalidade completa do código e, testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto;
- A aderência ao protocolo especificado. Usarei ferramentas de captura de tráfego da rede (wireshark) e analisarei o código para verificar se a comunicação está da forma definida na documentação;
- Conteúdo da documentação que deve conter os itens mencionados anteriormente;
- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua portuguesa, qualidade textual e facilidade de compreensão);
- CASOS DE CÓPIA NÃO SERÃO TOLERADOS;
- Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o protocolo funcione, deve ser descrito na documentação de forma explícita.

REFERÊNCIA

Para uso do programa Make e escrita de Makefiles:
http://www.gnu.org/software/make/manual/make.html
http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

ENTREGA E APRESENTAÇÃO DO TRABALHO

Data: 10/07/2024
O trabalho deve ser feito individualmente.