MC833 - Programação de Redes de Computadores

Projeto 1 - Implementação de servidor concorrente sobre TCP

André Luís L. C, Tavares - RA116125
Wesley T. S. T. Ide - RA108268

Introdução

O TCP (Transmission Control Protocol) é um protocolo utilizado para a transmissão segura de dados entre dois computadores. O TCP é um protocolo confiavel, pois reúne garantia de transmissão ordenada, retransmissão de pacotes perdidos, controle de fluxo e controle de congestão. Devido à essas características, o TCP é utilizado em uma ampla gama de aplicações da Internet, como email, transferência de arquivos e navegação web, frequentemente acoplado ao protocolo IP.

Neste trabalho implementamos um servidor e um cliente que se comunicam através de sockets TCP. Essa comunicaçao é concorrente, ou seja, nosso servidor pode atender mais de um cliente ao mesmo tempo. Caso ele receba uma requisição de um cliente enquanto processa a requisição de outro cliente, o servidor faz um fork() (cópia de sí mesmo) e passa a tratar as duas requisições paralelamente, sem congelar uma delas.

O trabalho simula um sistema de gestão de filmes , que possua uma base de dados com informaçoes de vários filmes e permita a realização de algumas operaçoes sobre ela, como consultas e escritas.

Sistema, descrição geral e casos de uso

Nosso sistema está implementado em três arquivos: o programa do servidor (server), o programa do cliente (client) e o arquivo do banco de dados (db.txt). O client é responsavel pela interface com o usuário; ele recebe a opção de consulta desejada, realiza uma solicitação para o servidor e devolve para o usuário a resposta. O server é responsável por atender a múltiplos clientes com as respostas para seus pedidos, para isso acessando as informações contídas no banco de dados. Por fim, o db.txt (banco de dados) é um arquivo txt simples que realiza a persistência das informações do sistema.

O arquivo results.txt é um arquivo adicional, escrito pelo cliente com o tempo que cada consulta ao servidor tomou. Esse tempo é medido do momento em que o socket é

aberto até que a string de resposta seja recebida, e utilizado para efeitos de comparação neste trabalho.

Nosso sistema também contem dois scripts, execute.sh e run30.sh. O primeiro compila o servidor e o cliente e roda uma vez cada tipo de solicitação; o segundo roda trinta vezes um mesmo tipo de solicitação. Ambos foram usados nas medições dos tempos de resposta que nos realizamos.

Ao utilizar nosso sistema, o usuário pode escolher dentre os seguintes casos de uso já implementados:

- Opção 0 - ‘Listar todos os títulos dos filmes e o ano de lançamento’. O cliente imprime uma lista com os títulos e ano de lançamento de cada filme em db.txt
- Opção 1 - 'Listar todas as informações de todos os filmes'. O cliente imprime todas as informações que estao contidas no db.txt
- Opção 2 - 'Listar todos os títulos dos filmes e o ano de lançamento de um gênero determinado’: O cliente imprime uma lista com os títulos e ano de lançamento de cada filme em db.txt que tem o gênero passado como um de seus gêneros.
- Opção 3 - 'Dado o identificador de um filme, retornar a sinopse do filme'. Os filmes sao numerados por identificadores no banco de dados. Sabendo um deles, o usuário pode requisitar ao cliente que retorne sua sinopse.
- Opção 4 - 'Dado o identificador de um filme, retornar todas as informações deste filme'. Da mesma forma, o cliente pode selecionar todos os campos de um filme, ou qualquer combinação deles.
- Opção 5 - 'Dado o identificador de um filme, retornar o número de exemplares em estoque'. Utilizando o identificador, o usuario pode solicitar a quantidade de exemplares de um filme em estoque.
- Opção 6 - 'Alterar o número de exemplares em estoque (apenas cliente locadora)'. O usuario pode sobreescrever o campo de exemplares em estoque de um filme.
- Opção 9 - 'Sair': fecha o cliente

Armazenamento e estruturas de dados do servidor

Os dados do servidor são armazenados em um arquivo (bd.txt). Neste arquivo, cada linha equivale a uma entrada no banco de dados, e cada tabulação a um campo diferente

daquela entrada, ou seja, a uma coluna. A primeira linha é ignorada, tratando-se da descrição do conteúdo de cada coluna.

Os campos guardados de cada filme são: identificador, título, ano de lançamento, gênero (pode ser mais de um), duraçao (em minutos), sinopse, diretor e número de exemplares em estoque.

Detalhes da implementação

Durante a implementação focamos na conexão entre os servidores, em detrimento da implementação do sistema de controle de filmes em si. Por esse motivo, procuramos simplificar ao máximo o sistema. Ao invés de um banco de dados relacional, nós criamos uma base de dados simples, um arquivo em formato de texto, que visa apenas persistir os dados a serem lidos e sobreescritos.

Para simplificar o script que rodava os vários pedidos, deixamos “hard-coded” os casos de uso exemplificados na descrição do projeto. Ao rodar um cliente, o usuário é convidado a selecionar um dentre os sete casos de uso disponíveis, com o proprio sistema do cliente definindo os parâmetros da busca que serão enviados ao servidor.

Entretanto, é importante notar que as funções de busca de filmes dado um identificador ou genero, seleção de campos de um filme e sobreescrita de um campo de filme já estão implementadas no servidor (apenas os parâmetros enviados pelo cliente que atualmente são fixos). Em uma futura expansão do sistema é facil criar dialogos adicionais através dos quais o usuário pode realizar consultas livres.

Nossa implementação foi de um servidor TCP concorrente rodando sobre IP versão 4 (IPv4). Utilizamos a biblioteca <netinet/in.h> para importar as funções socket(), connect(), send(), recv(), bind(), listen(), accept() e close(), além das estruturas de dados sockaddr_in (endereço para IPv4) e socklen_t (tamanho de endereço).

O projeto foi produzido utilizando as seguintes ferramentas: gedit, vim, sublime, google drive, gcc, netstat, linux shell.

Gráficos e análise de resultados

Foram realizados testes para cada caso de uso. Os tempos de comunicação de cada caso podem ser vistos nos gráficos a seguir.

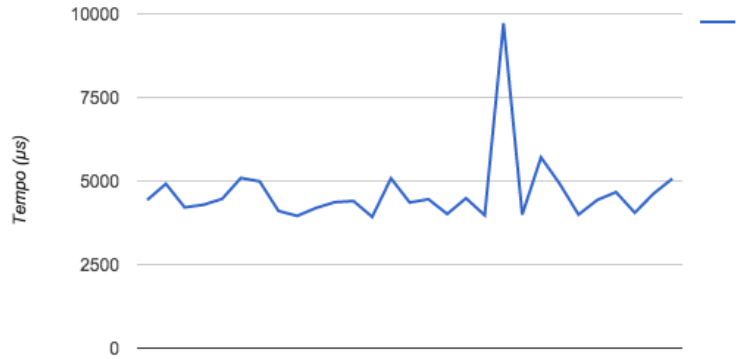

Listar todos os títulos dos filmes e o ano de lançamento
20000
15000
10000
5000
0
Tempo (µs)


Listar todas as informações de todos os filmes
16000
12000
8000
4000
0
Tempo (µs)


Listar todos os títulos dos filmes e o ano de lançamento de um gênero determinado
10000
7500
5000
2500
0
Tempo (µs)

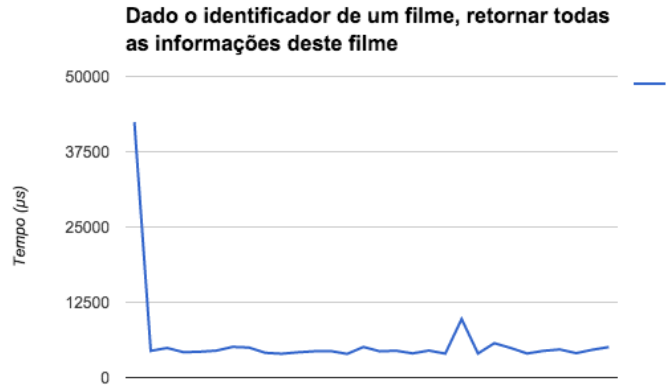
Dado o identificador de um filme, retornar a sinopse do filme

Dado o identificador de um filme, retornar todas as informações deste filme

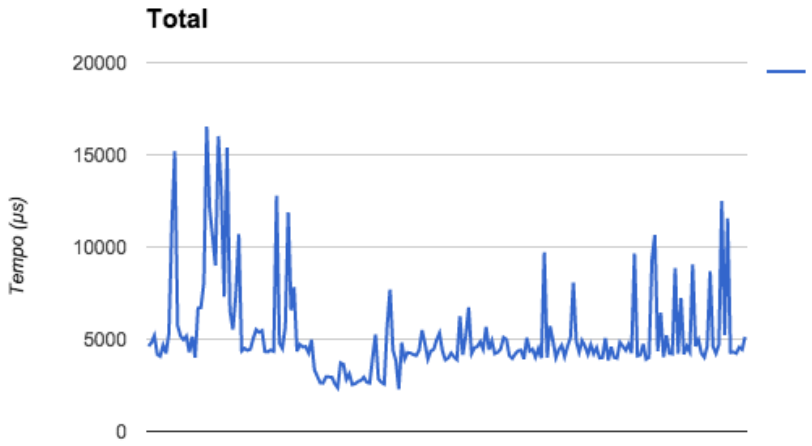


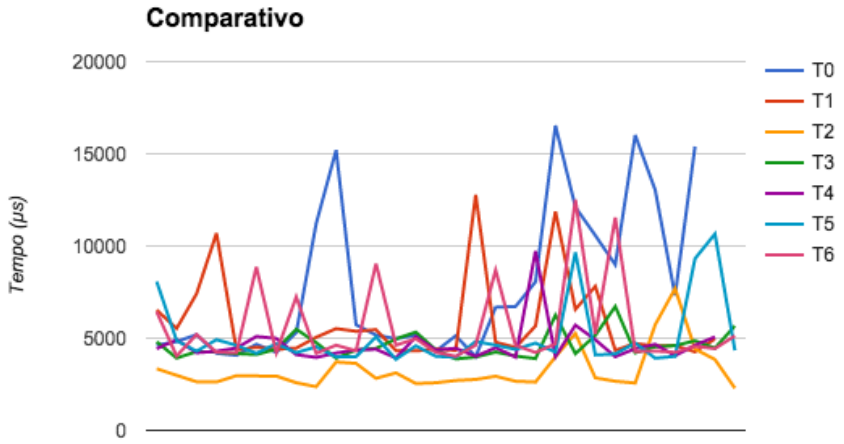
Dado o identificador de um filme, retornar o número de exemplares em estoque

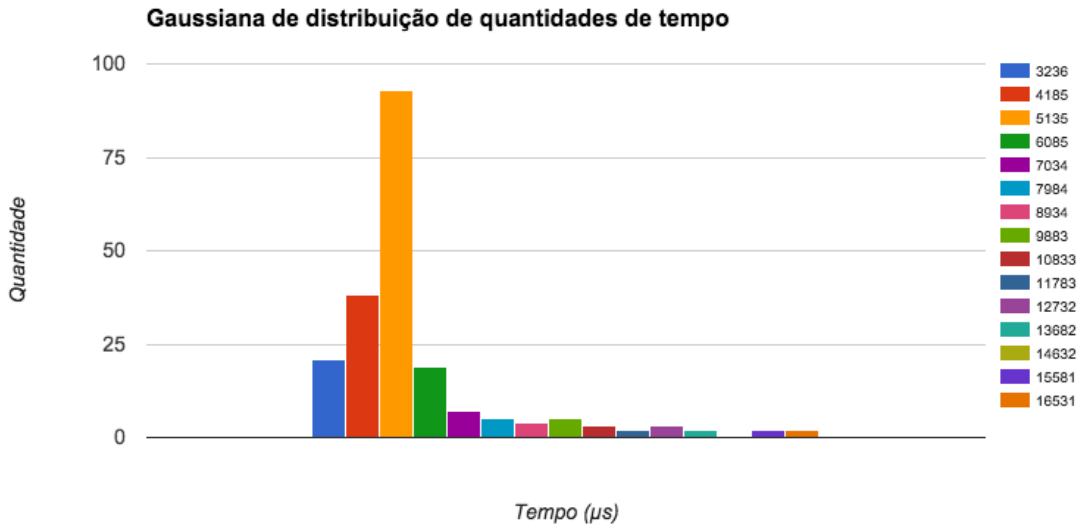
Em cada um dos casos de uso 0, 1 e 4, houve um valor de tempo muito desproporcional aos demais (outlier), o que pode ter sido ocasionado por atrasos e congestionamentos de rede, ou até mesmo timeout. Os gráficos mostrados acima, então, foram mostrados sem esses valores, para que a visualização dos demais tempos ficasse melhor. Se mantivermos os outliers, os gráficos ficam da seguinte forma:



Agora seguem 3 gráficos: um total com todos os tempos, outro comparativo entre cada teste de caso de uso, e uma gaussiana de distribuição de quantidades de tempo.


Total
20000
15000
10000
5000
0
Tempo (µs)


Comparativo
20000
15000
10000
5000
0
Tempo (µs)
T0
T1
T2
T3
T4
T5
T6



Todos os testes foram realizados em um cenário LAN.

Os gráficos mostram que de uma forma geral, os tempos estão bons, e mantém uma certa constância. E há uma pequena porção onde o tempo é maior do que a média. Mas que apesar disso, o resultado final se torna satisfatório.

Conclusão

Analisando os dados obtidos, é possível concluir que os tempos de comunicação se mantém num mesmo intervalo, de forma geral. Tambem é possível observar que, devido à simplicidade do acesso ao banco, as diferenças de tempo de comunicação dependem basicamente da velocidade de conexão entre servidor e cliente. Atrasos na rede acontecem com uma pequena frequência, podendo ser causadas por vários motivos, como congestionamento na rede, baixa taxa de transmissão da rede, perda de conexão (que ocasiona perda de pacotes), entre outros. Apesar disso, a maior parte das transmissões ocorrem com sucesso e em um tempo bom.

Referencias

STEVENS R. W., FENNER B., RUDOFF A. M.: UNIX Network Programming - The Sockets Networking APIs, Vol. 1, Third Edition, Person Education, 2004

Beej's Guide to Network Programming