MC833 - Programação de Redes de Computadores

Projeto 3 - Implementação de servidor concorrente utilizando Java RMI

André Luís L. C, Tavares - RA116125

Wesley T. S. T. Ide - RA108268

Introdução

Java RMI (Remote Method Invocation) é uma API de Java que realiza o equivalente a chamadas de procedimento remotas (RPCs) em orientação a objeto. Essa API pode ser utilizada para implementar a um sistema cliente/servidor para transmissão de dados entre dois computadores. O RMI é implementado sobre TCP, e por isso possui as confiabilidade instrinseca a este protocolo, garantida pela retransmissão de pacotes perdidos, transmissão ordenada, controle de fluxo e controle de congestão. Por usar a linguagem Java, a comunicação em RMI tambem tende a ser mais simples de implementar.

Neste trabalho implementamos um servidor e um cliente que se comunicam através de RMI. Esta comunicação é concorrente, ou seja, nosso servidor atende mais de um cliente ao mesmo tempo. Caso ele receba uma requisição de um cliente enquanto processa a requisição de outro cliente, o servidor trata as duas requisições paralelamente, sem congelar uma delas.

O trabalho simula um sistema de gestão de filmes, que possui uma base de dados com informações de vários filmes e permite a realização de algumas operações sobre ela, como consultas e escritas.

Sistema, descrição geral e casos de uso

Nosso sistema está implementado em quatro arquivos: a classe que implementa o servidor (FilmSystemServer), a classe que implementa o cliente (FilmSystemClient), a interface que define as funcionalidades disponiveis no sistema (FilmSystem), a classe que implementa essas funcionalidades (FilmSystemImpl). O banco de dados é uma base criada usando o postgreSQL, a qual é conectada no servidor.

FilmSystemClient é responsável pela interface com o usuário; este programa recebe a opção de consulta desejada, realiza uma solicitação para o servidor e devolve para o usuário a resposta.

FilmSystemServer é o servidor, o programa que disponibiliza na rede os serviços definidos na interface FilmSystem e implementados em FilmSystemImpl. O servidor é responsável por atender aos clientes com a resposta para seus pedidos; ele interpreta cada chamada e invoca a função correspondente de FilmSystemImpl.

FilmSystemImpl é a classe que efetivamente implementa a lógica do nosso sistema. Ela acessa o banco de dados "Filme", e lê ou modifica suas informações de acordo com o pedido recebido do cliente.

O intervalo medido vai do momento em que a requisição é enviada (System.nanoTime()) até que o momento em que a string de resposta é recebida. Este tempo é medido com precisão de nanosegundos, mas para facilitar a comparação com trabalhos anteriores arredondamos para microssegundos..

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 no banco de dados.
- Opção 1 - 'Listar todas as informações de todos os filmes'. O cliente imprime todas as informações que estão contidas no banco de dados.
- 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 no banco de dados 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 são 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 usuário pode solicitar a quantidade de exemplares de um filme em estoque.
- Opção 6 - 'Alterar o número de exemplares em estoque (apenas cliente locadora)'. O usuário pode sobrescrever o campo de exemplares em estoque de um filme.

Armazenamento e estruturas de dados do servidor

Os dados do servidor são armazenados em um banco de dados relacional no sistema de gerenciamento de banco de dados, o PostgreSQL. O banco é "mc833", o qual possui a tabela "Filme". Nesta estão as colunas referentes aos filmes, com seus respectivos conteúdos.

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

Detalhes da implementação

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 próprio sistema do cliente defininndo os parâmetros da busca que serão enviados ao servidor.

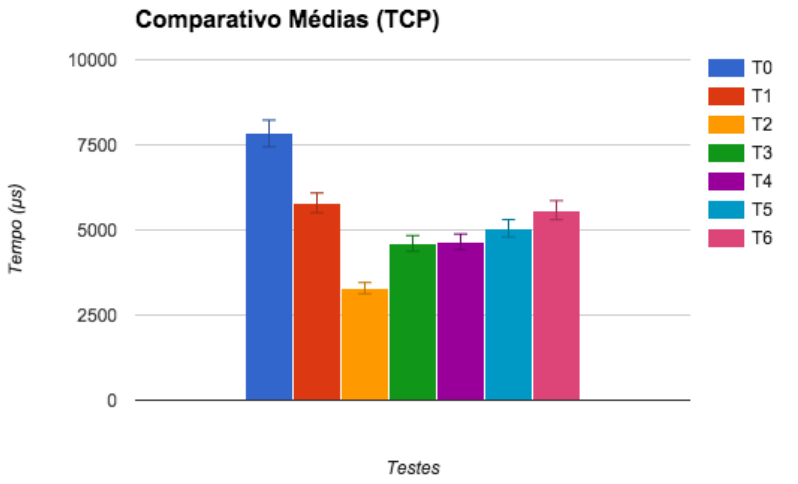
É importante notar que as funções de busca de filmes dado um identificador ou gênero, seleção de campos de um filme e sobrescrita de um campo de filme já estão implementadas no servidor (apenas os parâmetros enviados pelo cliente que atualmente são fixos). Em uma futura expansão do sistema é fácil criar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

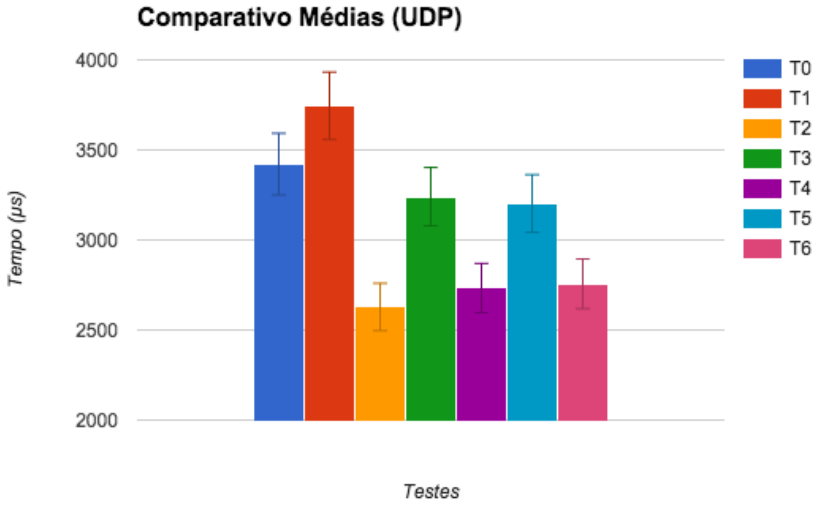
Nossa implementação foi de um conjunto cliente/servidor que utiliza a API de RMIs em Java para se comunicar. Essa API roda sobre TCP/IP concorrente.. Utilizamos a biblioteca java.rmi para importar as funções relacionadas ao RMI, e a biblioteca java.sql para importar as funcionalidades relacionadas ao banco.

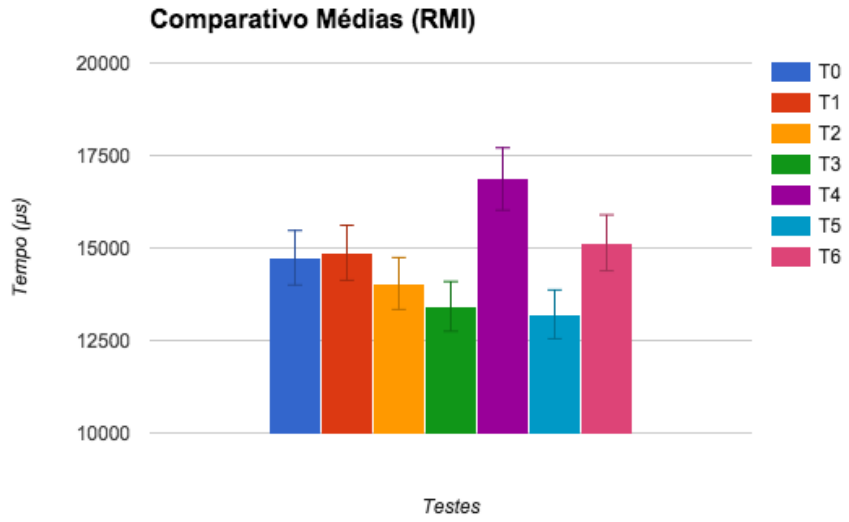
O projeto foi produzido utilizando as seguintes ferramentas: gedit, vim, eclipse, google drive, linux shell.

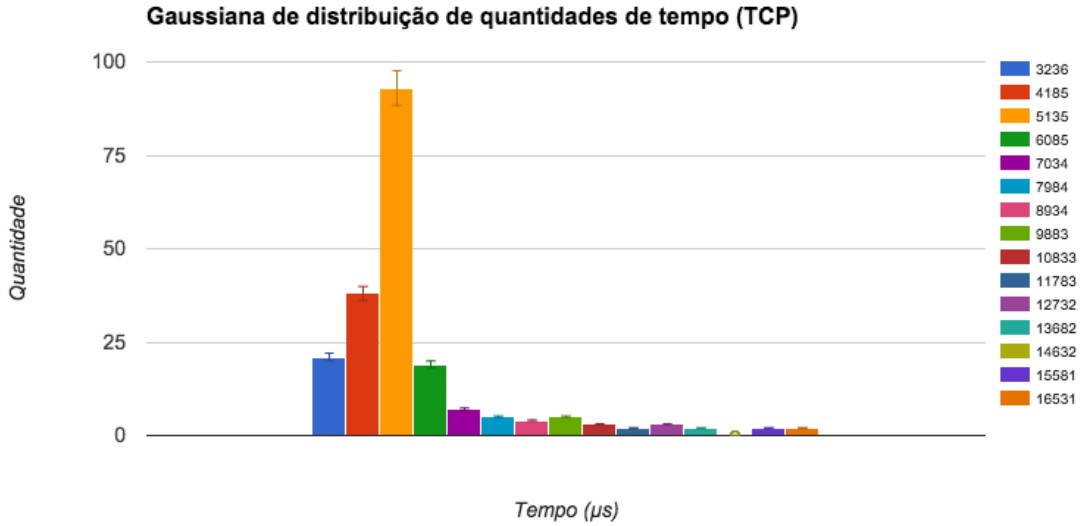
Gráficos e análise de resultados

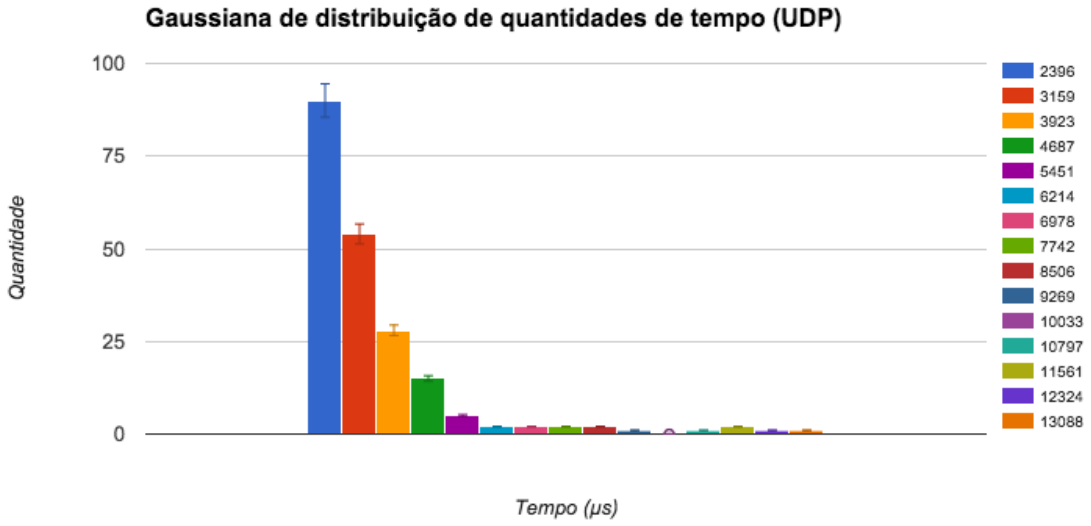
Foram realizados testes para cada caso de uso. Todos os testes foram realizados em um cenário LAN. Seguem 4 gráficos: comparativo de médias do UDP, TCP e RMI, e gaussiana de distribuição de quantidades de tempo do UDP, TCP e RMI.

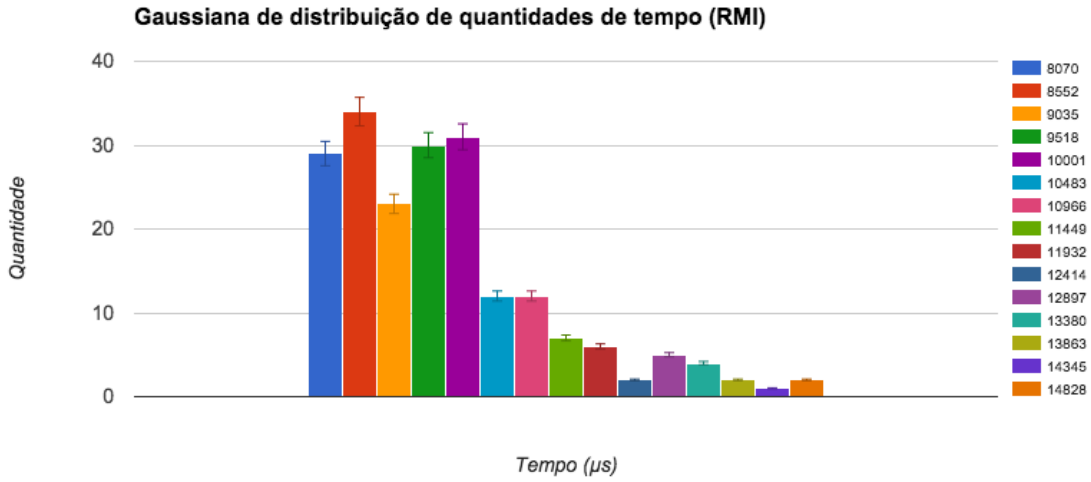





Comparativo Médias (RMI)
20000
17500
15000
12500
10000
Tempo (µs)
Testes
T0
T1
T2
T3
T4
T5
T6

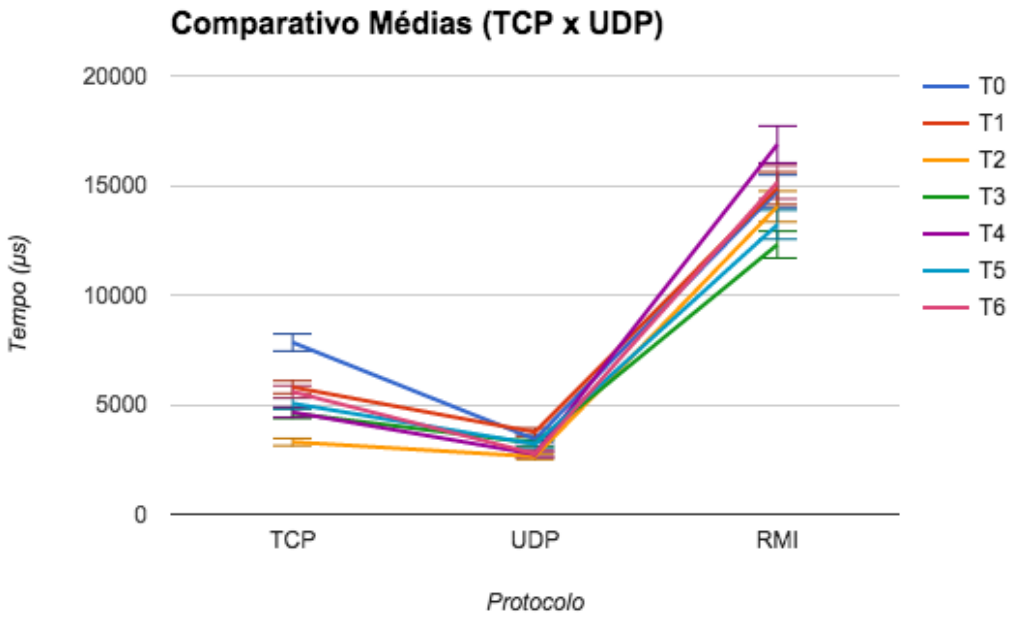

Gaussiana de distribuição de quantidades de tempo (TCP)
100
75
50
25
0
Quantidade
Tempo (µs)
3236
4185
5135
6085
7034
7984
8934
9883
10833
11783
12732
13682
14632
15581
16531





Pelos gráficos das gaussianas é possível verificar que enquanto ⅔ aproximadamente dos tempos do TCP e UDP tem tempos inferiores a 5135µs e 2396µs, respectivamente, no RMI, apenas 15% aproximadamente tem tempo inferior a 8070µs. É bem evidente que utilizando RMI, o tempo de execução é muito maior.

Comparação TCP x UDP x RMI



Através do gráfico acima podemos comparar o desempenho entre o projeto 1, onde foi utilizado o protocolo TCP e a linguagem C, o projeto 2, onde foi utilizado o protocolo UDP e a linguagem C, e o projeto 3, onde foi utilizada a API RMI e a linguagem Java. O gráfico possui uma representação das médias de tempo de cada teste para cada protocolo; todos os testes foram mais rápidos rodando sobre UDP, depois sobre TCP e por último com RMI. Em alguns testes o tempo de comunicação do RMI chegou a ser três vezes maior que do TCP, e quatro vezes maior que do UDP.

Conclusão

Após os testes e comparações feitas, é possível concluir que uma implementação que utiliza RMI na linguagem Java é mais lenta que uma utilizando TCP ou UDP na linguagem C. Este resultado já era esperado, pois tanto a linguagem Java quanto a API RMI são de mais alto nivel do que seus equivalentes TCP/UDP e C, possuindo um overhead de comunicação maior introduzido pelas facilidades de programação introduzidas.

Por outro lado, foi notável a maior facilidade na implementação desta versão do projeto em relação às anteriores, que utilizavam TCP e UDP e a linguagem C. Também é notavel a

redução no tamanho do código, o que, aliada as IDEs disponíveis para Java, torna muito mais fácil a manutenção da versão que utiliza RMI.

Cabe portanto ao arquiteto do sistema ponderar quais são suas prioridades ao escolher qual tecnologia de comunicação e linguagem utilizar. Se o objetivo for minimizar os recursos gastos com o desenvolvimento, vale a pena utilizar o Java com RMI. Se for necessário maximizar a velocidade, mas mantendo as características de confiabilidade do canal, pode-se optar por utilizar o TCP em uma aplicação C. Se o ponto crucial for a velocidade, é melhor utilizar o UDP rodando sobre C.

Referencias

http://www.darwinsys.com/java/rmi/