Aplicando Banco de Dados Distribuídos com Sistemas Heterogêneos em Problemas Reais

Oswaldo Bassani Neto
Escola Politécnica da Universidade de São Paulo (EPUSP)
Laboratório de Arquitetura e Software Básico (LASB)
oswaldo.bassani@poli.usp.br

Prof. Dr. Líria Matsumoto Sato
Escola Politécnica da Universidade de São Paulo (EPUSP)
Laboratório de Arquitetura e Software Básico (LASB)
liria.sato@poli.usp.br

Resumo

Neste trabalho apresenta-se uma proposta de solução para a distribuição de dados entre vários servidores objetivando o uso de servidores menos onerosos e um sistema de maior disponibilidade. Para o teste da nossa proposta criamos uma ferramenta de apoio a testes e outra para execução de 'queries'. Estas ferramentas têm por objetivo tornar possível a familiarização de usuários leigos na linguagem 'SQL' ('Strutured Query Language') e permitir testes genéricos sobre banco de dados distribuídos em uma rede.

A proposta de solução visa atingir problemas reais de banco de dados de pequeno e médio porte, propondo uma otimização para o sistema de dados já existentes.

Nossa solução foi desenvolvida utilizando o 'Remote Method Invocation' (RMI do java) e visa assegurar a consistência e proteção dos dados de problemas de rede e queda de sistemas vizinhos.

1. Introdução

As pesquisas desenvolvidas nos laboratórios possuem em sua maioria um objetivo científico muito maior do que qualquer perspectiva comercial de sua aplicação. Buscando atingir um equilíbrio de objetivos aplicamos os recursos e técnicas disponíveis em ambientes distribuídos sobre uma aplicação real e prática, gerando como resultado um protótipo.

Nosso objetivo é tornar possível que o banco de dados deixe de ser centralizado e passe a ser distribuído. Para isso precisamos de softwares que tornem essa distribuição transparente para o usuário.

Nesse trabalho foi utilizada a linguagem Java, mais especificamente o pacote 'java.rmi' ('Remote Method Invocation' ou simplesmente RMI, usado para invocação de métodos remotos). O RMI é muito usado em arquiteturas cliente-servidor em aplicações de pesquisa, também em projetos de processamento paralelo. Os sistemas implementados em Java vêm se destacando não pelo seu alto desempenho, mas por sua portabilidade, ou seja, são bons candidatos para um projeto de um pequeno 'grid' heterogêneo.

A Java vem se destacando também pelo seu uso na internet o que nos mostra que mais pessoas possuem conhecimentos básicos desta linguagem de programação facilitando assim seu desenvolvimento em aplicações dentro de empresas ou projetos.

Dentro da área de banco de dados em que iremos estar focados destaca-se o uso de pequenos e baratos servidores. Nas aplicações para pequenas e médias bases de dados encontramos muitas vezes um banco de dados em Access, tratado pelos usuários como grandes tabelas. Na maioria dos casos poderíamos usar o mySQL que é um pequeno e gratuito servidor de banco de dados.

A falta de soluções para pequenas e médias bases de dados torna-se um limitante de lucros em muitos casos, dado que os usuários do sistema têm muito trabalho ao serem praticamente gerenciadores destas tabelas.

Nosso software foi formulado sobre o uso do mySQL como banco de dados. Criamos uma coleção de classes para o tratamento da comunicação do Java com o mySQL, usando como base o sistema conhecido como JDBC, neste caso o mysql-connector-java-2.0.14. Visamos com estas classes proteger o sistema de problemas de comunicação entre os nós.

Buscamos um problema simples que poderia ser solucionado usando a solução que propomos. Escolhemos um problema de distribuição de dados que leva em conta a confiabilidade e disponibilidade da rede e equipamentos envolvidos. Nossa escolha foi propor este problema no caso de uma locadora de vídeos.

2. Definição da locadora de vídeos

A locadora de vídeos é composta por duas lojas. Elas se situam numa mesma cidade, a distância entre elas não é

muito grande tornando possível a troca de filmes entre elas através de um serviço de moto que leva cerca de vinte minutos para chegar de uma loja para a outra.

Um cliente cadastrado é reconhecido nas duas lojas podendo efetuar locações, pagamentos e devoluções em ambas.

3. Soluções convencionais

Embora seja um problema simples, muitas são as formas de prejuízo encontradas pelos donos de locadoras. A principal delas é o tão conhecido "Sistema fora do ar", esta falha pode ocorrer em varias soluções que dependam da rede para manter a comunicação entre as lojas. Outro problema sério ocorre em operações maldosas que dependendo da configuração do sistema podem ser mais fáceis de serem realizadas.

A forma mais comum de implementar uma solução para este problema é o uso de um servidor central de dados que pode ficar em uma das lojas enquanto que a outra deve manter uma conexão permanente para acessar e manipular as informações. Esta solução tem como vantagem a inexistência de preocupações com múltiplas bases de dados, no entanto traz como desvantagem a necessidade de um servidor grande e uma conexão permanente precisando que ambos sejam confiáveis.

Uma forma mais moderna é o uso de um pequeno servidor para cada loja, nesse caso a consistência de informações obriga a existência de uma conexão permanente para as eventuais mudanças nos dados.

4. Nossa solução

A solução que propomos torna possível que a conexão entre as lojas possa ser interrompida sem que as lojas parem de funcionar corretamente e caso um dos servidores fique indisponível isso não torna a outra loja indisponível.

Cada loja possui seu pequeno servidor que utiliza para guardar suas informações, a informação dos clientes e parte das informações da fitas de vídeo da outra loja.

Através de um programa de acesso que direciona os acessos ao banco de dados é possível separar os comandos de pesquisa dos comandos de alterações. Comandos de pesquisa são passados diretamente para o servidor local de dados. Já os comandos que alteram os dados são direcionados para um servidor que monitora as alterações visando armazenar comandos importantes para transmiti-los à outra loja.

Este servidor se encarrega de alterar os dados de maneira segura e mantém a coerência de informações entre os servidores de dados.

A figura 1 mostra como a nossa solução se baseia na arquitetura cliente-servidor e como o RMI está sendo usado. Além da comunicação entre clientes e servidor, a comunicação entre Servidores também é feita pelo RMI. Os acessos diretos ao banco de dados só ocorrem no caso de comandos do tipo "SELECT".

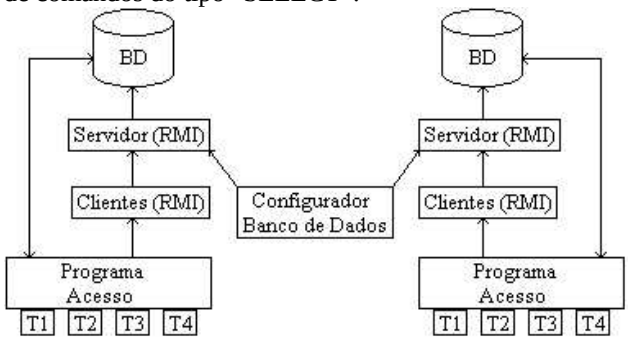

Figura 1. Configuração do Cliente-Servidor-BD da solução

5. Ferramentas desenvolvidas

Visando oferecer uma interface para o usuário e realizar testes sobre o sistema implementado foram desenvolvidas duas ferramentas. A primeira foi desenvolvida para testar a comunicação do Java com o mySQL, o objetivo dela é permitir a execução de 'queries' em bancos de dados e fornecer uma interface simples com a qual um usuário leigo pode aprender os comando SQL. É possível criar uma configuração com o endereço do seu banco de dados seu usuário e senha e se conectar a ele para edição ou leitura.

A segunda foi projetada para testar a configuração dos bancos de dados, objetivando a validação da nossa solução para o problema da locadora de vídeos. É possível criar uma seqüência de comandos que serão executados e através da visualização dos resultados é possível verificar o funcionamento de nossa solução.

5.1. Executor de comandos SQL

Destaca-se pela interface simples e pela facilidade fornecida para o usuário se familiarizar com os comando básico do mySQL tornando possível o rápido aprendizado e a formulação de 'queries' mais complexas.

Esta ferramenta foi desenvolvida para a familiarização com a interface Java-MySQL, tornando possível o aprendizado do JDBC. Vale a pena ressaltar que os programas desenvolvidos em projetos científicos são desenvolvidos em sua maioria para usuários mais experientes ou apenas para o seu autor, isso limita muito o entendimento do programa como software de uso geral

tornando difícil de se visualizar aplicações reais.

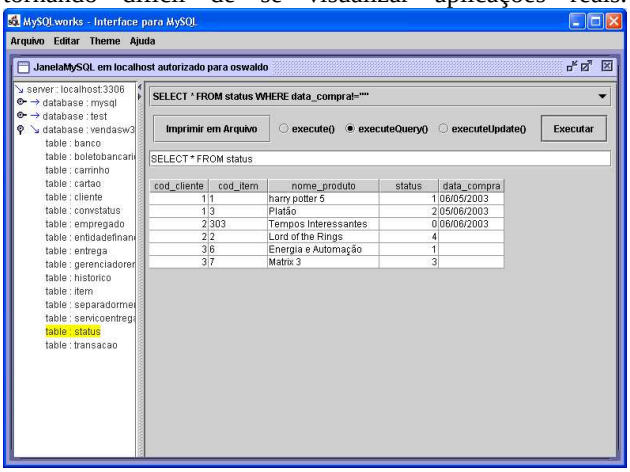

Figura 2: Interface do executor de 'queries'

5.2. Plataforma de teste

Fornece o apoio necessário para a execução dos testes sobre os códigos dos servidores que interligam as lojas, podemos então testar a implementação de nossa solução assim como outras soluções possíveis.

É possível usar um acesso direto pelo Java com o mySQL ou usar o software projetado para ser o servidor da nossa locadora. É claro que para outros problemas deve-se adequar o programa de teste para atingir todas as necessidades da nova configuração.

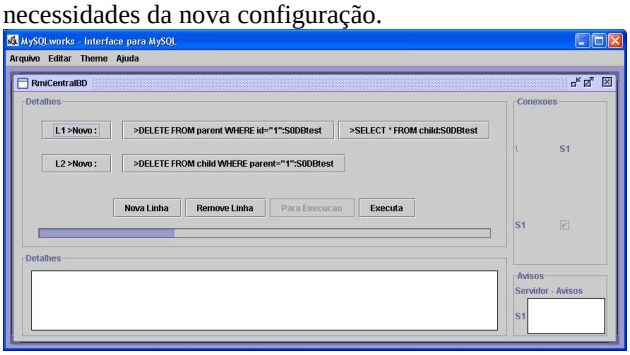

Figura 3: Interface da plataforma de teste

6. Conclusões

Conseguimos atingir com satisfação os pontos alvos do nosso projeto. A implementação por RMI mostrou-se satisfatória embora haja algumas limitações com o uso desse pacote. Um dos problemas com que nos deparamos foi a necessidade de se usar parâmetros serializáveis. Na implementação usando RMI notamos que é preciso definir tudo que se deseja passar de um lado para o outro através do RMI, e após isso, deve-se verificar se tudo é passível de serialização, pois caso haja algum item que não possa ser serializado não será possível usá-lo.

O uso de pequenos bancos de dados distribuídos é uma ótima forma de armazenar dados, no entanto requer inúmeros cuidados para que não apareçam incoerências pelo caminho. Criar soluções padrões nessa linha aponta como algo muito promissor dado a necessidade de se obter maior descentralização para evitar os problemas de comunicação com as redes, além de possibilitar o uso de servidores gratuitos de banco de dados.

Vale lembrar que a grande vantagem do uso da Java como linguagem de programação, que é a possibilidade de usá-la em varias plataformas, não serviu em nosso projeto. A maioria das empresas aplica em suas lojas o mesmo sistema operacional, facilitando o treinamento dos funcionários e possibilitando o uso de outros softwares que não são multi-plataforma. Para as empresas aonde os servidores principais possuem um sistema operacional diferente a Java é uma ótima opção.

O uso do Java acarreta perda de desempenho caso a interface do aplicativo utilize o pacote 'javax.swing', o que normalmente acontece. Esse pacote é uma implementação pesada de interface gráfica, que facilita muito a programação de telas para a interação homem-máquina, mas torna os aplicativos lentos.

Os programas em Java permanecem com seu código desprotegido, não sendo possível criar um arquivo 'exe' a partir das classes, e as soluções para esse problema são caras. Dessa forma, aconselhemos o uso da Java para aplicações não comerciais e que possam permanecer com código aberto. Nas aplicações comerciais, podemos usar as soluções de proteção de código disponíveis ou separar o programa em dois, utilizando uma topologia cliente-servidor e protegendo o acesso às máquinas servidoras.

7. Agradecimentos

Agradeço ao CNPq pelo financiamento do projeto que é base deste resumo e aos colegas que me forneceram dicas e inspirações para chegarmos ao termino do projeto.

8. Referências

[1] Fraizer, Colin; Bond, Jill; API Java Manual de Referencia Pacotes de API java.applet e java.awt, Tradução Álvaro Rodrigues Antunes, Makron Books

[2] Ortali, Robert; Harkey, Dan; Client/Server Programming with Java and Corba, Wiley Computer Publishing

[3] Gosling, James; Yellin, Frank; The Java Application Programming Interface, Vol1 Core Packages, Addison-Wesley Publishing Company

[4] Gosling, James; Yellin, Frank; The Java Application Programming Interface, Vol2 Window Toolkit and Applets, Addison-Wesley Publishing Company

[5] Site oficial da Sun : http://java.sun.com/

[6] Site oficial do mySQL : http://www.mysql.com/