ESTUDO DE MECANISMOS DE REPLICAÇÃO DE INFORMAÇÕES DO PBS PARA O LIMA

Bruno Flores Bicca[1]; Nilton Cézar de Paula[2]

[1]Estudante do Curso de Ciência da Computação da UEMS; Unidade Universitária de Dourados;
E-mail: 017348@comp.uems.br
[2]Professor do Curso de Ciência da Computação da UEMS; Unidade Universitária de Dourados;
E-mail: nilton@comp.uems.br
Área Temática: Banco de Dados

Resumo

Este projeto reúne métodos para implementar a replicação de dados para o PBS (*Portable Batch System*) do LIMA. A utilidade do *software* de replicação de dados tem como tarefas a detecção de alteração dos dados no banco de dados e fazer o envio desses dados para outro *software* que gerencie a replicação desses dados para os demais computadores conectados a ele. Neste projeto buscamos uma forma eficaz e rápida de replicação para mantermos os dados de um computador constantemente atualizado. Definiu-se uma parte do conjunto de dados do PBS para ser monitorado, atualizado e replicado, o tipo de base de dados e como seria implementado os *softwares* para a replicação de dados.

Palavras-chave: Banco de Dados. Replicação Total. Troca de Mensagem.

Introdução

Uma grade computacional vem sendo utilizada para melhorar a execução de aplicações, tanto seqüenciais como paralelas, usando vários tipos de recursos espalhados geograficamente e pertencentes a diferentes domínios administrativos (FOSTER; et. al., 2001). Existem vários modelos para obter informações de outros sistemas, uma delas é a replicação de dados. Foi proposto o sistema de monitoramento LIMA (*Light-weIght Monitoring Architecture*) (DE PAULA, 2009), que também tem o objetivo de oferecer acesso fácil às informações. Para o problema de replicação de dados proposto neste trabalho, foram utilizadas informações do PBS (BERMAN; FOX; HEY, 2003) do LIMA para definir a estrutura de dados da replicação.

Materiais e Métodos

Na primeira etapa foram estudados mecanismos para o uso da replicação em bancos de dados, através de pesquisas na internet em busca de artigos e manuais sobre o uso e a implementação de replicação para um banco de dados, além de livros da área. Assim, conseguiu-se coletar informações importantes para gerenciar a atualização dos dados e compreender a programação de comunicação por troca de mensagens. Com o estudo realizado foi escolhida a linguagem Java (NUNES, 2010) para a programação e a conexão entre o cliente e o servidor, por permitir fácil conexão e trocas de mensagens entre computadores. Foi escolhido o MySQL como a plataforma de banco de dados, por ser de fácil programação e por possuir várias ferramentas para a replicação de dados. Os modelos de replicação estudados foram do *software* Replicação de Dados 2.0 da Intesoft (INTESOFT, 2010), que trabalha enviando a linha do banco de dados alterado, e do DBMoto (DBMOTO, 2010), que controla as replicações através de um *software* servidor e trata as alterações de dados no banco de dados usando um *software* cliente.

Resultados e Discussão

Inicialmente, os *softwares* servidor e cliente foram desenvolvidos usando a linguagem Java e de forma conjunta. Desta maneira, pode-se incrementar instruções em cada *software* e conforme os testes iam sendo feitos também iam sendo observadas suas corretas execuções. O primeiro conjunto de instrução desenvolvido foi para resolver a comunicação entre os *softwares* usando *sockets*. Depois, instruções para o processo de conexão do *software* servidor com o banco de dados MySQL usando a biblioteca de conexão JAVA-MySQL.

Em seguida foi construído o mecanismo que trata o evento de alteração das informações do banco de dados (replicação), ilustrado na Figura 1. Cada registro armazenado no banco de dados contém um ID que representa o ID de um cliente e assim, cada *software* cliente verifica apenas a linha do seu banco de dados, apenas a linha que corresponde ao ID do cliente. O cliente verifica o banco de dados a cada 5 segundos, depois o sistema de verificação fica ocioso. Se for detectada a alteração de algum dado do cliente, então o servidor é informado via *socket*, recebe os dados deste cliente e replica para os demais clientes.

Para tratar vários eventos que são necessários na replicação, foram utilizadas *threads*, que são tarefas de um mesmo processo que são executadas concorrentemente. Foi desenvolvido um mecanismo que escuta o servidor através do *socket*. Essa tarefa aguarda receber um sinal do servidor que envia todas as informações da linha que será atualizada no cliente. Através dessa tarefa também pode definir se o servidor se desconectou.

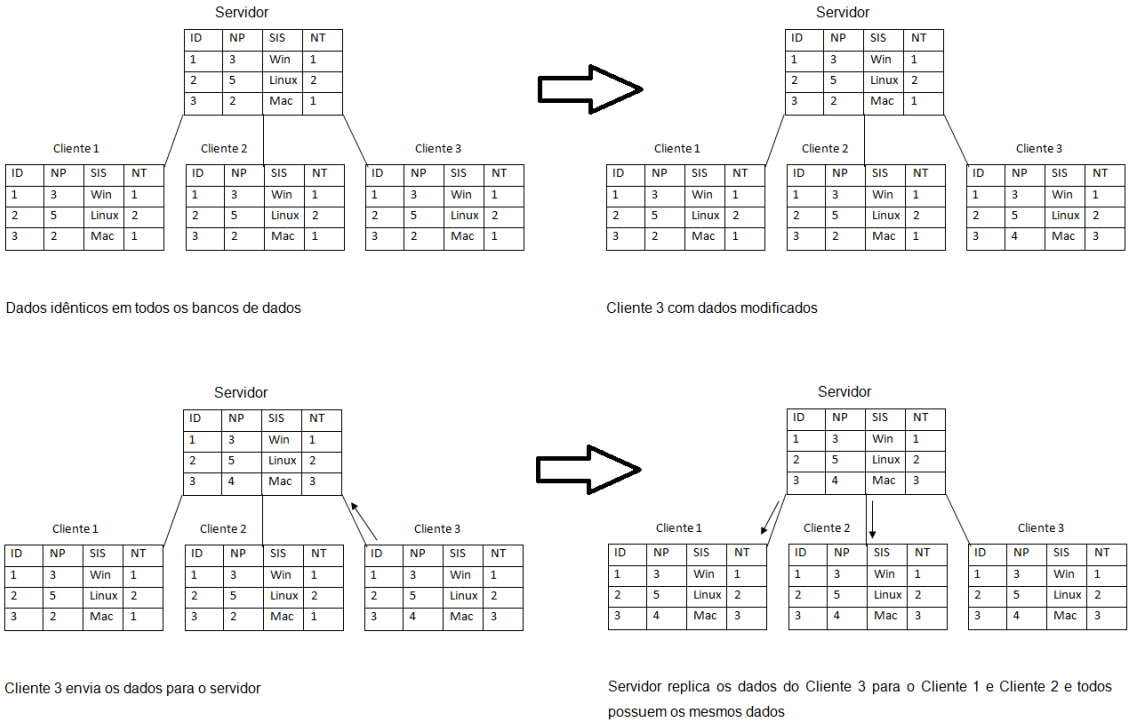

Figura 1: Mecanismo de replicação em um banco de dados desenvolvido.

Os métodos do módulo cliente são detalhados a seguir e apresentados na Figura 2:

- **Main**: Inicia o ambiente e a conexão com o banco de dados.
- **Arq**: Exibe ao usuário a necessidade de informar os dados de entrada para a conexão com o servidor e conexão com o banco de dados.
- **Inicio**: Exibe ao usuário as informações do banco de dados. É necessário que o usuário entre com o ID do cliente, para que possa gerar conexão com o servidor. O status da conexão é exibida em tela, e todas as informações do banco de dados são atualizadas para o usuário.
- **Conexao**: Método que inicia uma conexão com o servidor, e deste modo é possível a troca de informações. Este método fica ocioso, esperando receber alguma informação do servidor, caso o servidor envie um sinal de replicação, o método aguarda receber os dados e faz a atualização dos dados no banco de dados.
- **Verificacao**: Método que verifica modificações dos dados no banco de dados, de forma que quando alterados, é gerado um sinal de replicação para o servidor e logo após, é enviados os dados para o servidor, que replicará os dados para os demais computadores ligados a ele. A verificação é feita a cada 5 segundos.
- **Global**: Método que contém variáveis globais.

- **ConexaoMySQL**: Método que acessa o banco de dados MySQL. Através dele é gerada a conexão com o banco de dados e é feito consultas de dados.

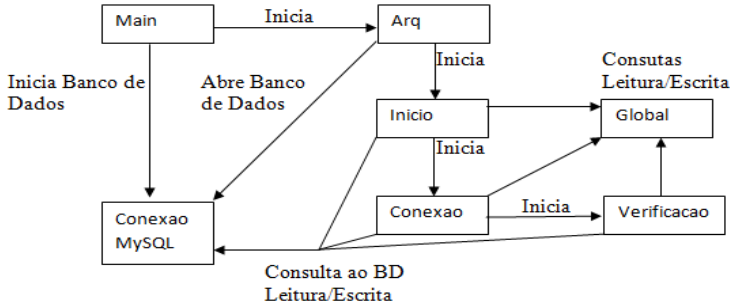

Figura 2: Métodos do módulo cliente desenvolvido.

Como o cliente, no módulo servidor também foi desenvolvido um método de conexão para o servidor com o cliente através de *socket*, mas no caso do servidor, ele irá gerenciar as conexões com vários clientes. Para que se possa ter um controle maior dos dados o servidor também tem uma conexão ao banco de dados MySQL, e guarda todos os valores de todos os clientes que se conectam a ele.

Para cada cliente conectado o servidor gera duas *threads* para gerenciar o cliente. A primeira *thread* tem a tarefa de receber dados do cliente, e com isso ele deve ficar escutando a conexão com o cliente. A segunda *thread* tem a tarefa de enviar dados ao cliente quando necessário. Para não sobrecarregar o sistema com um volume grande de *threads*, o servidor pode atender um limite máximo de 20 clientes.

Quando um cliente é conectado, ele envia a linha que corresponde ao seu ID para o servidor. O servidor atualiza seu campo no banco de dados próprio, e após isso envia todos os outros dados contidos na tabela dele, para que o cliente possa ter os dados atuais de todos os outros clientes. Após isso o servidor replica os dados do cliente conectado a todos os clientes conectados ao servidor.

Posteriormente o servidor só gerencia as conexões, recebendo dados de clientes e os replicando para os demais clientes conectados. E quando o cliente é desconectado, ele não receberá as replicações de dados do servidor.

Os métodos do servidor são descritos a seguir e apresentados na Figura 3:

- **Main**: Gera uma conexão de entrada e inicia a interface gráfica.
- **Inicio**: Apresenta ao usuário o banco de dados do servidor, todos os clientes conectados, as ocorrências e as ações tomadas pela aplicação.

- **Conexao**: Controla a conexão dos clientes. Neste método é utilizada a conexão de entrada para gerenciar até 20 clientes e são geradas *threads* para controlar cada cliente. Cada cliente conectado envia seu ID para controle do servidor.
- **Cliente**: Controla a conexão de entrada de cada cliente. O método Cliente é gerado um para cada cliente que se conecta ao servidor. Neste método inicialmente é enviado ao cliente todos os dados do banco de dados do servidor, menos o campo do ID do cliente. Após isto o método fica ocioso esperando receber dados do cliente para fazer a replicação de dados para os demais clientes conectados. Os dados são recebidos e enviados a uma variável global compartilhada aonde o método Replicacao de cada cliente tem o acesso ao valor da variável, para poder enviar os dados ao cliente.
- **Replicacao**: Controla a conexão de saída de dados para o cliente. O método Replicacao é gerado um para cada cliente. Neste método ele verifica a necessidade de replicação de dados a cada 5 segundos e envia os dados replicados ao cliente.
- **Refresh**: Atualiza a tabela da interface gráfica do servidor. Ele lê o banco de dados e exibe na tela.
- **Global**: Método que contém variáveis globais.
- **ConexaoMySQL**: Acessa o banco de dados MySQL. Através desse método é gerada a conexão com o banco de dados e é feito consultas de dados.

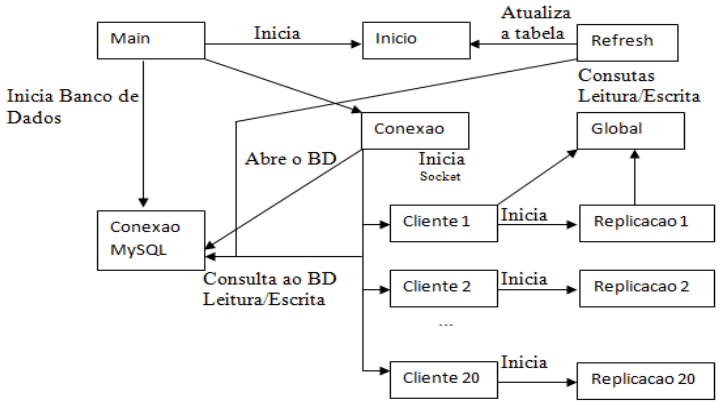

Figura 3: Métodos do módulo servidor desenvolvido.

Conclusão

Com o trabalho foi possível visualizar que a replicação de dados necessita de vários fatores que influenciam na construção de um *software* adequado, como a escolha adequada do banco de dados, o tipo de envio das informações, e como será feita a replicação dos dados. Como o objetivo foi utilizar replicação para as informações do PBS do LIMA, foi necessário

tratar apenas o envio das informações do PBS de um computador ao outro utilizando o MySQL como banco de dados e uma conexão *socket*. Com isso foi concluído que, o uso da replicação de dados depende de cada situação, e todas tem um custo para ser utilizado, como (1) o tempo de verificação dos dados do banco de dados e (2) de quanto em quanto tempo se deve enviar as informações. Para os estudos realizados, foi utilizado um intervalo de tempo pequeno (5 segundos) para verificar alterações dos dados do banco de dados. E, assim que detectada uma alteração nos dados, então a replicação de dados é realizada imediatamente.

Agradecimento

Agradeço pelo apoio financeiro concedido pela Universidade Estadual de Mato Grosso do Sul e também por todos os acadêmicos, coordenadores e professores que, diretamente ou indiretamente, contribuíram para a realização desse trabalho.

Referências

BERMAN, F.; FOX, G. C.; HEY, A. J. G. Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Inc, 2003.

DBMOTO, disponível em http://www.hitsw.com/localized/portuguese/products_services/dbmoto/dbmoto.html – (último acesso em 05/09/2010).

DE PAULA, N.; C. Um Ambiente de Monitoramento de Recursos e Escalonamento Cooperativo de Aplicações Paralelas em Grades Computacionais. Tese (Doutorado), Universidade de São Paulo, 2009.

FOSTER, I.; et al. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001.

INTESOFT. Replicação de Dados 2.0, disponível em http://www.software-replicacao-dados.intesoft.com.br/ - (último acesso em 05/09/2010).

NUNES, L. R, disponível em http://www.sumersoft.com/publicacoes/SocketsEmJAVA.pdf – (último acesso em 25/11/2010).