**Introdução**

Em sistemas computacionais, se faz necessário muitas vezes distribuir o processamento de dados em mais de uma máquina. São diversos os fatores que levam a distribuição desses sistemas entre um grupo de computadores.

Alguns desses problemas, está relacionado a cujo funcionamento não pode ser afetado pelo problema individual de uma máquina, assim seu processamento é replicado para que caso uma máquina falhe, as outras possam continuar o processamento dos dados e manter o sistema em funcionamento.

Outro fator são sistemas cujo recursos podem ser requisitados por inúmeras máquinas. Sendo assim o processamento é distribuído para que uma máquina não seja sobrecarregada pelas requisições realizadas por terceiros mantendo a estabilidade do sistema.

Outro fator comum a sistemas é que eles podem ter várias divisões lógicas, capazes de realizar diferentes funções no sistema que podem ser distribuídos para outras máquinas, permitindo que cada máquina se especialize em tais funções a nível de hardware e software aumentando o desempenho do sistema.

Existem vários meios de garantir a distribuição de sistemas. Alguns deles se tratam do uso de ferramentas de distribuição desenvolvidas especificamente para esse fim. Uma dessas ferramentas é conhecida como jGroups. Trata-se de uma ferramenta de código aberto que oferece interface para distribuição de sistemas em escritos em Java.

Um trabalho prático foi realizado utilizando essa ferramenta, um sistema bancário distribuído que deverá ser executado no sistema operacional Linux, e será apresentado no Laboratório de Redes.

**Materiais e métodos**

Durante o desenvolvimento do software, foram utilizadas diferentes ferramentas tanto para programar quanto para realizar testes no software. As ferramentas usadas foram:

IDEs:

Eclipse Mars, Eclipse Neon, NetBeans 8.1, Netbeans 8.0

Editores de texto: gedit;

Sistemas operacionais: Windows 10 64 bits, Linux Ubuntu 14.10, Linux Ubuntu 16.04

**Arquitetura do sistema**

Para o projeto, decidimos pela distribuição física do sistema em 3 partes. A primeira parte consiste em um cluster responsável pela persistência dos dados. Periodicamente ele receberá uma visão dos dados processados e realizará a gravação do mesmo em um arquivo. A segunda parte, consiste num cluster responsável por processar todas as movimentações bancárias. Sua responsabilidade será atender as requisições dos clientes, analisar se tal requisição é possível de ser processada, processar caso seja possível e atualizar o cluster de persistência sobre os dados processados. A terceira parte do sistema consiste em uma interface onde o cliente pode requisitar serviços do sistema bancário através de uma interface gráfica.

**Arquitetura do software**

Durante o desenvolvimento do sistema, foi criado 4 classes. Uma classe Conta, que contém a estrutura de uma conta, além de suas funções e métodos. A classe Conta possui 4 campos, conta, saldo e senha, além de uma lista de movimentações. Ela implementa 4 funções: getMovimento(), setMovimento(), getConta(), setConta(), getSenha, setSenha(), getSaldo e setSaldo.

Outra classe criada foi BancoGui, que implementa as funções relacionada a interface do cliente. Essa classe é responsável por ser a interface de comunicação entre o cliente e o sistema bancário.

Foi criada a classe BancoServidor, responsável por processar os dados relacionados a movimentação bancária, recebendo as requisições dos clientes e verificando quais ações eram possíveis de serem processadas e também por garantir que o sistema não funcionasse de forma ambígua e inconsistente.

Uma classe de persistência também foi criada, responsável por receber as atualizações dos dados processados pelo sistema e armazená-los em arquivos persistentes.

**Configuração do XML**

Para a configuração do arquivo XML, selecionamos uma pilha de protocolos que será descrito abaixo:

UDP

ucast\_recv\_buf\_size="163K"

ucast\_send\_buf\_size="163K"

mcast\_recv\_buf\_size="163K"

mcast\_send\_buf\_size="163K"

PING

FD\_ALL

VERIFY\_SUSPECT

BARRIER

pbcast.NAKACK use\_mcast\_xmit="true"

discard\_delivered\_msgs="true"

UNICAST3

RSVP

pbcast.STABLE stability\_delay="1000"

desired\_avg\_gossip="50000"

max\_bytes="4M"

pbcast.GMS print\_local\_addr="true"

join\_timeout="3000"

view\_bundling="true

MERGE2

UFC

MFC

FRAG2

pbcast.STATE\_TRANSFER