Sistema de Echo com Replicação Passiva e Tolerância a Falhas

Descrição do Projeto

Este projeto implementa um serviço de Echo distribuído com replicação passiva e tolerância a falhas, utilizando:

- **Java RMI** para comunicação cliente-servidor
- **MQTT (Mosquitto)** para replicação de mensagens entre servidores
- **Arquitetura Master-Clone** com eleição automática de novo master em caso de falha

Arquitetura do Sistema

Componentes Principais

1. **ServerListManager**: Gerenciador centralizado que mantém a lista de servidores ativos e coordena eleições
2. **EchoServer**: Servidor de echo que pode atuar como Master ou Clone
3. **EchoClient**: Cliente que se conecta ao servidor Master para enviar mensagens
4. **MQTT Broker (Mosquitto)**: Middleware de mensagens para replicação assíncrona

Fluxo de Funcionamento

1. **Inicialização**:
   - ServerListManager é iniciado primeiro
   - Servidores se registram e o primeiro se torna Master
   - Clones se inscrevem no tópico MQTT para receber replicações
2. **Operação Normal**:
   - Cliente envia mensagens apenas para o Master
   - Master processa e publica no MQTT
   - Clones recebem e armazenam as mensagens replicadas
3. **Tolerância a Falhas**:
   - Clones monitoram o Master via heartbeat
   - Ao detectar falha, iniciam processo de eleição
   - Novo Master é eleito e se desinscreve do MQTT
   - Cliente reconecta automaticamente ao novo Master

Pré-requisitos

Software Necessário

- **Java JDK 11** ou superior
- **Maven 3.6** ou superior
- **Mosquitto MQTT Broker**
- **Ubuntu 20.04** (recomendado) ou outro Linux

Instalação das Dependências

```bash
# Instalar Java 11
sudo apt update
sudo apt install openjdk-11-jdk

# Instalar Maven
sudo apt install maven

# Instalar Mosquitto
sudo apt install mosquitto mosquitto-clients

# Verificar instalações
java --version
mvn --version
mosquitto -h
```

Configuração do Ambiente

1. Configurar Mosquitto

```bash
# Iniciar o serviço Mosquitto
sudo systemctl start mosquitto

# Habilitar inicialização automática
sudo systemctl enable mosquitto

# Verificar status
sudo systemctl status mosquitto
```

2. Configurar o Projeto no IntelliJ IDEA

1. **Clonar/Extrair o projeto**

```bash
```

```
cd ~/workspace
unzip echo-service-replication.zip
cd echo-service-replication
```

2. **Abrir no IntelliJ IDEA**:
   - File → Open → Selecione a pasta do projeto
   - Aguarde o IntelliJ indexar o projeto
3. **Configurar SDK**:
   - File → Project Structure → Project
   - Project SDK: Selecione Java 11
   - Project Language Level: 11
4. **Importar como projeto Maven**:
   - O IntelliJ deve detectar automaticamente o pom.xml
   - Se não, clique com botão direito no pom.xml → Add as Maven Project

Compilação

Via Maven (Terminal)

```bash
# Na pasta raiz do projeto
mvn clean package

# Os JARs executáveis serão gerados em:
# target/server-manager-jar-with-dependencies.jar
# target/echo-server-jar-with-dependencies.jar
# target/echo-client-jar-with-dependencies.jar
```

Via IntelliJ IDEA

1. Abrir a janela Maven (View → Tool Windows → Maven)
2. Executar: Lifecycle → clean
3. Executar: Lifecycle → package

Execução

Método 1: Script Automatizado

```bash
```

```
# Dar permissão de execução ao script
chmod +x run_scripts.sh

# Executar o script
./run_scripts.sh

# Escolher opção 2 para iniciar sistema completo
```

Método 2: Execução Manual

1. Iniciar ServerListManager

```
bash

java -jar target/server-manager-jar-with-dependencies.jar
```

2. Iniciar Servidores (em terminais separados)

```
bash

# Servidor 1 (será o Master inicial)
java -jar target/echo-server-jar-with-dependencies.jar Server1

# Servidor 2 (Clone)
java -jar target/echo-server-jar-with-dependencies.jar Server2

# Servidor 3 (Clone)
java -jar target/echo-server-jar-with-dependencies.jar Server3
```

3. Iniciar Cliente

```
bash

java -jar target/echo-client-jar-with-dependencies.jar
```

Testes do Sistema

Teste 1: Operação Normal

1. Inicie o sistema completo
2. No cliente, envie algumas mensagens (opção 1)
3. Liste as mensagens (opção 2)
4. Verifique que todas foram armazenadas

Teste 2: Tolerância a Falhas

1. Com o sistema rodando, identifique o servidor Master
2. Feche o terminal do servidor Master (Ctrl+C)
3. Aguarde alguns segundos
4. No cliente, tente enviar nova mensagem
5. Verifique que o sistema continua funcionando com novo Master

Teste 3: Adição de Novo Servidor

1. Com o sistema rodando, adicione novo servidor:

```bash
java -jar target/echo-server-jar-with-dependencies.jar Server4
```

2. Verifique no cliente (opção 3) que o novo servidor aparece na lista
3. O novo servidor deve sincronizar automaticamente as mensagens existentes

Teste 4: Replicação MQTT

1. Para monitorar as mensagens MQTT:

```bash
mosquitto_sub -t "echo/replication" -v
```

2. Envie mensagens pelo cliente
3. Observe as mensagens sendo publicadas no tópico

Estrutura do Projeto

```
echo-service-replication/
├── src/
│   └── main/
│       └── java/
│           ├── EchoService.java          # Interface RMI principal
│           ├── EchoServer.java           # Implementação do servidor
│           ├── EchoClient.java           # Implementação do cliente
│           ├── ServerListManager.java    # Interface do gerenciador
│           └── ServerListManagerImpl.java # Implementação do gerenciador
├── pom.xml                               # Configuração Maven
├── run_scripts.sh                        # Script de execução
└── README.md                             # Esta documentação
```

Funcionalidades Implementadas

✅ **Operações do Cliente**:

- Echo de mensagens
- Obter lista de mensagens
- Visualizar status do sistema

✅ **Replicação Passiva**:

- Master processa requisições
- Clones recebem replicação via MQTT
- Sincronização de mensagens

✅ **Tolerância a Falhas**:

- Detecção de falha do Master via heartbeat
- Eleição automática de novo Master
- Reconexão transparente do cliente

✅ **Escalabilidade**:

- Adição dinâmica de novos servidores
- Sincronização automática de estado

Troubleshooting

Problema: "Connection refused" ao iniciar servidor

Solução: Certifique-se que o ServerListManager está rodando primeiro

Problema: "MQTT connection failed"

Solução: Verifique se o Mosquitto está rodando:

```bash
sudo systemctl status mosquitto
sudo systemctl start mosquitto
```

Problema: "RMI Registry já existe"

Solução: Normal se múltiplos componentes tentam criar. Pode ignorar.

Problema: Cliente não reconecta após falha do Master

Solução: Aguarde 5-10 segundos para eleição completar

Observações Importantes

1. **Ordem de Inicialização**: Sempre inicie nesta ordem:
   - Mosquitto
   - ServerListManager
   - Servidores
   - Cliente
2. **Portas Utilizadas**:
   - RMI Registry: 1099
   - MQTT Broker: 1883
3. **Limitações**:
   - A eleição é baseada em ordem de registro (FIFO)
   - Requer conectividade de rede estável
   - Mensagens podem ser perdidas durante eleição

Autor

Professor: Adriano Fiorese
Disciplina: Sistemas Distribuídos
Instituição: UDESC

Licença