Sistema Distribuído de Log com Ordenação Causal

Sistema distribuído que implementa os algoritmos de **Relógio Lógico de Lamport** e **Algoritmo Bully** para eleição de líder, com replicação de mensagens entre 3 nodos geograficamente distribuídos.

📋 Descrição

Este projeto implementa um sistema de log distribuído que garante:

- **Ordenação causal de mensagens** mediante Relógio Lógico de Lamport
- **Eleição automática de líder** mediante Algoritmo Bully
- **Replicação de mensagens** entre todos os nodos do cluster
- **Tolerância a falhas** com re-eleição automática de líder
- **Dashboard web interativo** para visualização em tempo real

🚀 Início Rápido

Testing Local (Docker)

```
# 1. Iniciar cluster de 3 nodos
./scripts/local/test-local.sh

# 2. Abrir dashboard no navegador
open http://localhost:8001/dashboard

# 3. Parar cluster
./scripts/local/stop-local.sh
```

Deployment em GCP

```
# 1. Configurar projeto
export GCP_PROJECT_ID="seu-projeto-id"

# 2. Deploy completo
./scripts/gcp/deploy-gcp.sh

# 3. Re-deploy de containers com IPs corretas
./scripts/gcp/redeploy-containers.sh

# 4. Verificar estado
./scripts/gcp/check-gcp-status.sh
```

🎨 Dashboard Web

Acesse o dashboard interativo:

- **Local:** `http://localhost:8001/dashboard`
- **GCP:** `http://<IP-DO-NODO>/dashboard`

O dashboard mostra:

- ✅ Estado dos nodos em tempo real
- ✅ Lamport timestamps atuais
- ✅ Identificação do líder (👑)
- ✅ Mensagens ordenadas causalmente
- ✅ Formulário para enviar mensagens
- ✅ Auto-refresh a cada 3 segundos

📁 Estrutura do Projeto

```
distribuidos-trabalho-2/
├── README.md                 # Este arquivo
├── requirements.txt          # Dependências Python
├── dockerfile                # Imagem Docker
├── docs/                     # Documentação completa
│   ├── ARCHITECTURE.md       # Arquitetura do sistema
│   ├── QUICKSTART.md         # Guia rápido
│   ├── gcp-setup.md          # Setup de GCP
│   ├── Trabalho.md           # Especificação do projeto
│   └── PLAN_DETALLADO.md     # Plano de implementação
├── scripts/                  # Scripts de deployment e testing
│   ├── local/               # Scripts para Docker local
│   │   ├── test-local.sh
│   │   ├── stop-local.sh
│   │   └── test-send-messages.sh
│   ├── gcp/                 # Scripts para Google Cloud Platform
│   │   ├── deploy-gcp.sh
│   │   ├── redeploy-containers.sh
│   │   ├── destroy-gcp.sh
│   │   ├── check-gcp-status.sh
│   │   ├── debug-node.sh
│   │   └── test-gcp-system.sh
│   └── monitoring/          # Scripts de monitoramento
│       └── watch-messages.sh
├── src/                      # Código fonte
│   ├── main.py              # Aplicação FastAPI principal
│   ├── server.py            # Modelo de servidor
│   ├── lamport_clock.py     # Implementação Relógio de Lamport
│   └── static/
│       └── dashboard.html # Dashboard web interativo
└── legacy/                  # Arquivos legacy (não usados)
```

📚 Documentação Completa

Para mais detalhes, consulte:

- **docs/ARCHITECTURE.md** - Arquitetura detalhada do sistema
- **docs/QUICKSTART.md** - Guia rápido de deployment
- **docs/gcp-setup.md** - Configuração do Google Cloud Platform

📊 Scripts Disponíveis

Scripts Locais (`scripts/local/`)

- `test-local.sh` - Iniciar cluster local de 3 nodos
- `stop-local.sh` - Parar cluster local
- `test-send-messages.sh` - Enviar mensagens de teste

Scripts GCP (`scripts/gcp/`)

- `deploy-gcp.sh` - Deployment completo em GCP
- `redeploy-containers.sh` - Re-deployar containers com nova configuração
- `destroy-gcp.sh` - Eliminar toda a infraestrutura de GCP
- `check-gcp-status.sh` - Verificar estado de todos os nodos
- `debug-node.sh <num>` - Ver logs detalhados de um nodo
- `test-gcp-system.sh` - Suite de testes para GCP

Scripts de Monitoramento (`scripts/monitoring/`)

- `watch-messages.sh <num>` - Monitorar mensagens em tempo real

🎬 Roteiro para o Vídeo de Demonstração (5 minutos)

Preparação

1. **Verificar deployment em GCP:**

```
export GCP_PROJECT_ID="trabalho2-477920"
./scripts/gcp/check-gcp-status.sh
```

2. **Abrir dashboards nos navegadores:**

```
# URLs dos dashboards (substitua com suas IPs)
http://34.55.87.209/dashboard    # Iowa (us-central1-a)
http://34.95.212.100/dashboard   # São Paulo (southamerica-east1-a)
http://35.201.29.184/dashboard   # Sydney (australia-southeast1-a)
```

Roteiro de Demonstração

Minuto 0-1: Introdução e Arquitetura

- Apresentar o projeto: Sistema de log distribuído com Lamport Clock + Bully
- Mostrar os 3 nodos deployados em regiões geográficas distintas
- Explicar: 3 VMs em Iowa (EUA), São Paulo (Brasil), Sydney (Austrália)
- Total: ~36.000 km de separação

Minuto 1-2: Algoritmo Bully - Eleição de Líder

- Mostrar nos dashboards qual nodo é o líder atual (Node 3 - Sydney, ID 8003)
- Explicar: "O nodo com maior ID é eleito líder automaticamente"
- Mostrar comando para verificar líder:

```
curl http://34.55.87.209/leader  # Retorna 8003
```

Minuto 2-4: Relógio de Lamport - Ordenação Causal

- Enviar mensagens concorrentes de diferentes regiões:

```
# Terminal 1 - Iowa
curl -X POST "http://34.55.87.209/?message=Mensagem_Iowa_1"

# Terminal 2 - São Paulo
curl -X POST "http://34.95.212.100/?message=Mensagem_Brasil_1"

# Terminal 3 - Sydney (Líder)
curl -X POST "http://35.201.29.184/?message=Mensagem_Sydney_1"
```

- Mostrar nos dashboards:
    - Lamport timestamps incrementando (time: 1, 2, 3, ...)
    - Mensagens aparecendo em ordem causal
    - Replicação entre todos os nodos
- Enviar carga de teste:

```
./scripts/gcp/test-gcp-system.sh
```

- Mostrar métricas:
    - Throughput: ~27 msg/s
    - Latências: 19ms (SP), 294ms (Iowa), 652ms (Sydney)

Minuto 4-5: Resultados e Conclusão

- Mostrar dashboard com mensagens replicadas consistentemente
- Destacar Lamport timestamps preservando ordem causal
- Explicar limitações: saturação em 100 mensagens, single-leader
- Mencionar trabalho futuro: multi-leader, tolerância a falhas

Comandos Úteis para Demonstração

```
# Ver mensagens em um nodo
curl http://34.55.87.209/messages | jq

# Ver Lamport time atual
curl http://34.55.87.209/lamport_time

# Enviar 10 mensagens concorrentes
for i in {1..10}; do
  curl -X POST "http://35.201.29.184/?message=Teste_$i" &
done

# Executar suite completa de testes
./scripts/gcp/test-gcp-system.sh

# Coletar métricas detalhadas
./scripts/gcp/collect-metrics.sh
```

Dicas para Gravação

- ✅ Ambos os integrantes devem participar do vídeo
- ✅ Mostrar código-fonte brevemente (main.py, lamport_clock.py)
- ✅ Demonstrar funcionamento prático no GCP
- ✅ Explicar por que as latências são diferentes (distância geográfica)
- ✅ Mostrar consistência: mesmos dados em todos os nodos

👥 Autores

- Sergio Sebastian Pezo Jimenez - RA: 298813
- José Victor Santana Barbosa - RA: 245511

Projeto desenvolvido para a disciplina **MC714 - Sistemas Distribuídos**, Unicamp, 2º Semestre de 2025.

📄 Licença

Este projeto é para uso acadêmico.