



Arquitetura do Sistema Distribuído

Distribuição Geográfica Global

Regiões Seleccionadas (Máxima Distância)

DISTRIBUIÇÃO GLOBAL



Node 1: Iowa, USA (us-central1-a)

Latitude: 41.8780° N

Longitude: 93.0977° W



Node 2: São Paulo, Brasil (southamerica-east1-a)

Latitude: 23.5505° S

Longitude: 46.6333° W



Node 3: Sydney, Austrália (australia-southeast1-a)

Latitude: 33.8688° S

Longitude: 151.2093° E

Distâncias Geográficas

De → Para	Distância (km)	Latência Estimada (ms)
Iowa → São Paulo	~9,500 km	150-200 ms
Iowa → Sydney	~13,300 km	200-250 ms
São Paulo → Sydney	~13,600 km	250-300 ms

Distância total percorrida: >36.000 km (quase a circunferência da Terra!)

Por que estas regiões?

1. **Máxima separação geográfica:**

- Cobrimos 3 continentes diferentes
- Hemisférios norte e sul representados
- Múltiplos fusos horários (diferença de ~15 horas entre Iowa e Sydney)

2. **Simula um sistema distribuído REAL:**

- Latências altas (150-300ms) similares a aplicações globais reais
- Diferentes condições de rede
- Teste real do algoritmo Bully e Lamport sob condições adversas

3. **Demonstra propriedades do sistema:**

- **Relógio Lógico de Lamport:** NÃO depende de sincronização de relógios físicos
- **Algoritmo Bully:** Funciona mesmo com latências altas
- **Tolerância a falhas:** Se uma região falha, as outras 2 continuam



Arquitetura de Deployment

Google Cloud Platform

us-central1-a

(Iowa, USA)

IP: 34.55.87.209

southamerica-east1-a

(São Paulo, Brasil)

IP: 34.95.212.100

australia-southeast1-a

(Sydney, Austrália)

IP: 35.201.29.184

VM: e2-micro

2 vCPU

1GB RAM

20GB Disk

VM: e2-micro

2 vCPU

1GB RAM

20GB Disk

VM: e2-micro

2 vCPU

1GB RAM

20GB Disk

Docker:

- FastAPI

- Lamport

- Bully

Docker:

- FastAPI

- Lamport

- Bully

Docker:

- FastAPI

- Lamport

- Bully

NODE_ID:8001

Port: 80

NODE_ID:8002

Port: 80

NODE_ID:8003

Port: 80

OTHER_SERVERS

= "IPs:..."

OTHER_SERVERS

= "IPs:..."

OTHER_SERVERS

= "IPs:..."

Comunicação HTTP/REST usando IPs públicos

(Internet - Latências reais de 150-300ms)

Nota Importante sobre Networking:

- No **Docker local**: Os nodos usam nomes de container (node1, node2, node3)
- No **GCP**: Os nodos usam IPs públicos passados via variável `OTHER_SERVERS`

- O código detecta automaticamente o ambiente e se configura apropriadamente



Fluxo de Comunicação

1. Eleição de Líder (Algoritmo Bully)

Início: Todos os nodos iniciam simultaneamente

Node 8001 (Iowa): "Há alguém com ID maior?"
→ Consulta a 8002 (Brasil) [~180ms RTT]
→ Consulta a 8003 (Sydney) [~230ms RTT]

Node 8002 (Brasil): "Há alguém com ID maior?"
→ Consulta a 8003 (Sydney) [~270ms RTT]

Node 8003 (Sydney): "Não há ninguém maior, sou o líder"
→ Notifica a todos [~250ms promedio]

Resultado: Node 8003 é o LÍDER

Tempo total de eleição: ~1-2 segundos

2. Replicação de Mensagens (Relógio Lógico de Lamport)

Cliente → Node 8001 (Iowa):

POST /?message=Hello

Node 8001:

1. Detecta que NÃO é líder
2. Forward a Node 8003 (Sydney) [~230ms]

Node 8003 (Líder):

1. Incrementa Lamport Clock: $t=1$
2. Cria mensagem: {id: 2, lamport: 1, node: 8003}
3. Replica em PARALELO:
 - Node 8001 (Iowa) [~230ms]
 - Node 8002 (Brasil) [~270ms]

Node 8001 e 8002:

1. Recebem mensagem com lamport=1
2. Atualizam relógio: $\max(\text{local}, 1) + 1$
3. Guardam mensagem ordenada por Lamport
4. Respondem ao líder

Tempo total: ~500-600ms (incluindo latências globais)



Métricas Observáveis

Latências Esperadas

Operação	Latência Estimada
Leitura local (GET /messages)	1-5 ms
Escrita no líder (POST /)	10-20 ms
Replicação global completa	300-600 ms
Eleição de líder (re-election)	1-2 segundos

Operação	Latência Estimada
Health check entre nodos	150-300 ms

Propriedades Garantidas

✓ Consistência Causal (Lamport):

- Se mensagem $A \rightarrow B$ (causalmente), então $\text{Lamport}(A) < \text{Lamport}(B)$
- SEMPRE, independentemente de latências de rede

✓ Disponibilidade (Bully):

- Se 2 de 3 nodos estão vivos, o sistema funciona
- Re-eleição automática em ~1-2 segundos

✓ Tolerância a Partições:

- Cada nodo pode seguir operando localmente
- Consistência eventual quando a rede se recupera



APIs Requeridas no GCP

Para que o deployment funcione corretamente, você precisa habilitar estas APIs:

```
# Compute Engine API - Para criar e gerenciar VMs
gcloud services enable compute.googleapis.com

# Artifact Registry API - Para armazenar imagens Docker (novo sistema)
gcloud services enable artifactregistry.googleapis.com

# Container Registry API - Para backward compatibility com gcr.io
gcloud services enable containerregistry.googleapis.com
```

Nota: Embora usemos `gcr.io` no código, Google Cloud internamente redireciona para Artifact Registry, portanto ambas as APIs são necessárias.



Custos Estimados (GCP)

VM e2-micro (3 instâncias):

- Preço: ~\$6.11/mês por instância
- Total VMs: ~\$18.33/mês

Egress Traffic (dados saindo do GCP):

- Primeiros 1GB/mês: Grátis
- Próximos 10TB: \$0.12/GB
- Estimado para testing: ~\$5/mês

TOTAL ESTIMADO: ~\$25/mês

Para este projeto (algumas horas): < \$1



Segurança

Firewall Rules

allow-distributed-log:

- Protocolo: TCP
- Portas: 80, 443, 8000-8100
- Fonte: 0.0.0.0/0 (qualquer IP)
- Target: VMs com tag "distributed-log"

allow-ssh-distributed-log:

- Protocolo: TCP
- Porta: 22
- Fonte: 0.0.0.0/0
- Target: VMs com tag "distributed-log"

Melhorias de Segurança (Produção)

⚠ Para um sistema de produção, implementar:

- HTTPS com certificados TLS

- Autenticação entre nodos (tokens JWT)
- IP whitelisting (somente IPs de nodos conhecidos)
- VPN ou VPC peering privado
- Rate limiting
- DDoS protection (Cloud Armor)