Universidade Federal de Santa Catarina (UFSC)
Campus Reitor João David Ferreira Lima
Departamento de Informática e Estatística
Bacharelado em Ciência da Computação

Disciplina INE5418 - Computação Distribuída - 2024.1

Prof. Odorico Machado Mendizabal
Cursandos: Enzo da Rosa Brum,
Gustavo Fukushima de Salles,
Gustavo Konescki Führ

15/06/2024

Serviço de espaço de tuplas tolerante a falhas

Florianópolis

1. INTRODUÇÃO

O serviço de espaço de tuplas foi desenvolvido em linguagem GO utilizando a biblioteca Dragonboat, que fornece uma implementação do protocolo Raft e suas funções de eleição de líder, controle de processos e tolerância a falhas.

A aplicação também conta com um cliente desenvolvido em Python, o qual permite o cadastro de alunos com seu nome, matrícula, disciplina, média, frequência. Também é possível a remoção de alunos e a leitura de um ou todos os alunos inseridos.

2. IMPLEMENTAÇÃO DO SERVIÇO

2.1. GARANTIA E TOLERÂNCIA A FALHAS

2.1.1. REPLICAÇÃO

São iniciados 5 nós, os quais usam o algoritmo Raft para eleição de um líder. Alguns nós se propõem como candidatos e enviam mensagens para outros nós pedindo votos. Os nós enviam uma confirmação de voto para o nó candidato e se declaram seguidores dele, negando todas outras propostas de votos. Se um nó candidato receber confirmação da maioria, ele se denomina líder e envia uma mensagem para os outros nós se proclamando o líder.

O líder frequentemente envia mensagens de heartbeat aos seguidores como indicador de autoridade. Caso um seguidor não receba o heartbeat no intervalo de timeout determinado (geralmente de 100 a 300 ms), ele se declara candidato e convoca uma nova eleição de líder.

Sendo necessário a confirmação da maioria, o servidor assegura confiabilidade capaz de tolerar até 2 falhas, pressupondo falhas do tipo crash, e pode ser escalado adicionando novos nós conforme o necessário.

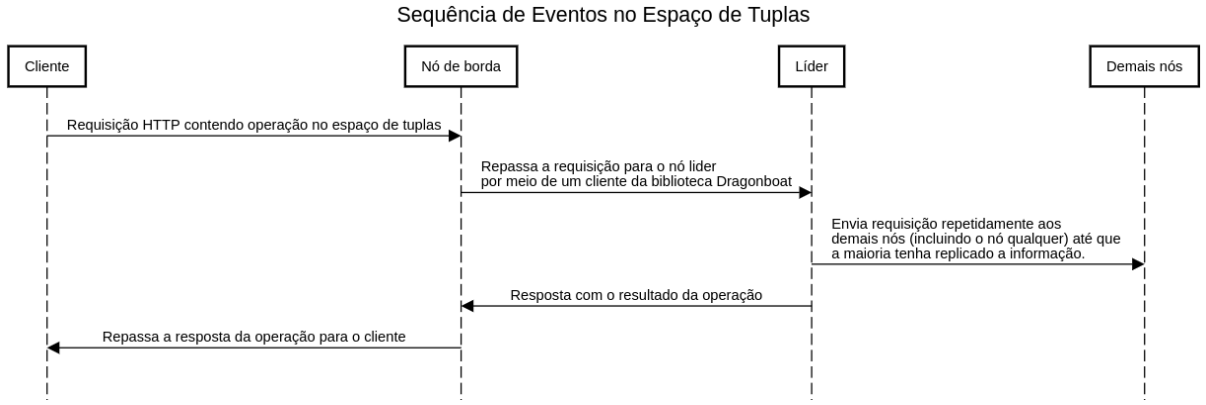
2.1.2. ATOMICIDADE

Raft utiliza “Log Replication” para garantir a atomicidade do sistema. Ao receber uma requisição de escrita ou remoção, o líder adiciona um registro não “commitado” no seu log e envia o registro para os demais nós. Posteriormente, se o nó líder receber a confirmação da maioria, ele realiza o “commit” do registro, aplica a alteração na sua máquina de estados e envia uma mensagem para seus seguidores, solicitando que também façam o “commit” do registro.

Se o líder não receber a confirmação da maioria, ele utiliza o mecanismo de rollback, no qual reverte o seu log para um estado consistente anterior e reenvia as alterações do estado.

2.1.3. ORDEM DE ENTREGA

Inicialmente, o cliente C1 faz uma requisição HTTP a um servidor qualquer pertencente ao cluster Raft. Ao receber tal requisição, o servidor age como um cliente C2 ao enviar uma mensagem ao líder propondo a operação requisitada por C1. O líder, por sua vez, é responsável por receber a requisição e repassá-la a todos os nós do grupo. Portanto, o líder funciona como um sequenciador, e assim, o sistema garante as prioridades de ordem Total. Isso significa que a ordem dos envios do nó líder é a mesma ordem que os demais nós entregam.
O diagrama abaixo ilustra a comunicação entre as diferentes partes do sistema:



2.2. CASO DE USO DA SOLUÇÃO

Para obter instruções de como executar a aplicação, veja o README.md enviado junto ao código.

Um possível caso de uso para a solução desenvolvida é um sistema para cadastro e busca de alunos. Em tal sistema, as tuplas poderiam ser organizadas no seguinte formato: (Código da disciplina, Matrícula do(a) Estudante, Nome do(a) Estudante, Frequência, Média do(a) estudante). Este foi o caso de uso escolhido para este trabalho.

O restante desta seção indica como o usuário pode interagir com a aplicação desenvolvida:

Para a criação dos nós é utilizado o seguinte comando, o qual retorna o endereço e a porta que estará ouvindo.

```
./server_executable -nodeid 5
2024-06-29 17:12:53.376757 I | config: using default EngineConfig
2024-06-29 17:12:53.376822 I | config: using default LogDBConfig
2024-06-29 17:12:53.563130 I | logdb: using plain logdb
Server listening at -> localhost:60009
```

Inicialmente o usuário escolhe qual será o nó que receberá as suas requisições. A aplicação irá perguntar o endereço até o usuário digitar um valor válido.

```
ENDEREÇO DO SERVIDOR: localhost:60001
```

Ao selecionar o servidor, o usuário poderá optar por realizar uma das seis operações possíveis.

```
1 - ADICIONAR
2 - LER
3 - REMOVER
4 - LER TODAS
5 - ALTERAR ENDEREÇO DO SERVIDOR
6 - SAIR
```

Na operação de adicionar, o usuário deverá entrar obrigatoriamente com os cinco valores de descrição. A aplicação irá criar uma tupla e a enviará para o servidor.

```
1 - ADICIONAR
2 - LER
3 - REMOVER
4 - LER TODAS
5 - ALTERAR ENDEREÇO DO SERVIDOR
6 - SAIR
1
Nome da disciplina: INE5418
Matrícula do(a) estudante: 22100613
Nome do(a) estudante: Enzo
Frequência: 100
Média do estudante: 9.0
==========
```

Na operação de leitura, o usuário poderá preencher os valores descritivos com ‘*’, o qual significa que o valor não importa para a análise. A aplicação irá criar uma tupla, enviará para o servidor e o servidor buscará uma correspondência com os valores propostos. Se a

tupla existir o servidor irá enviá-la ao cliente. Se não existir, enviará uma mensagem que a tupla não foi encontrada.

```
1 - ADICIONAR
2 - LER
3 - REMOVER
4 - LER TODAS
5 - ALTERAR ENDEREÇO DO SERVIDOR
6 - SAIR
2
Nome da disciplina: INE5418
Matrícula do(a) estudante: *
Nome do(a) estudante: *
Frequência: *
Média do estudante: *
==========
Aluno recebido: ['INE5418', '22100613', 'Enzo', '100', '9.0']
==========
```

A operação de remoção ocorre de maneira semelhante à leitura, no entanto, ao invés de apenas ler a tupla se ela existir, o servidor também irá removê-la.

```
1 - ADICIONAR
2 - LER
3 - REMOVER
4 - LER TODAS
5 - ALTERAR ENDEREÇO DO SERVIDOR
6 - SAIR
3
Nome da disciplina: INE5418
Matrícula do(a) estudante: *
Nome do(a) estudante: *
Frequência: *
Média do estudante: *
==========
Aluno removido: ['INE5418', '22100613', 'Enzo', '100', '9.0']
==========
1 - ADICIONAR
2 - LER
3 - REMOVER
4 - LER TODAS
5 - ALTERAR ENDEREÇO DO SERVIDOR
6 - SAIR
4
==========
Não há tuplas no servidor
==========
```

O usuário poderá trocar o nó que enviará as requisições, apertando a tecla 5.

```
1 - ADICIONAR
2 - LER
3 - REMOVER
4 - LER TODAS
5 - ALTERAR ENDEREÇO DO SERVIDOR
6 - SAIR
5
ENDEREÇO DO SERVIDOR: localhost:60003
==========
```