Laboratório de Programação Concorrente

Lab8

Objetivo

Nesse laboratório, iremos construir uma aplicação distribuída. Essa aplicação funciona como um buscador de arquivos para filesharing (pense em Bittorrent). Então, um cliente deve procurar que máquinas na rede armazenem o arquivo buscado. Como chave de busca, ele deve passar o hash do arquivo (calcular tal como o **sum** dos labs passados). Considere que haverá, pelo menos, um grupo de quatro máquinas.

O design da solução já foi parcialmente implementado, e você é o responsável por garantir que o sistema esteja protegido para o uso de múltiplas threads. Alguns direcionamentos que podem ser úteis:

1. Proteja a comunicação feita pelo sistema, tanto client-server quanto client-client.
2. Desempenho continua sendo importante. Pense em minimizar o tempo total, do ponto de vista de um cliente, para obter a lista de máquinas (seus IPs). Embora a lista de otimizações possíveis seja enorme, primeiro **FAZ FUNCIONAR!**
3. Considere que os arquivos buscados estão em um diretório no /tmp. Por exemplo, /tmp/dataset

Contextualização do problema

O código disponibilizado para este Lab 8 **já implementa a lógica distribuída**, porém foi fornecido **sem controle de concorrência** (isto é: sem mutex, sem channels, sem sincronização). A proposta do laboratório é que você implemente essas proteções.

O sistema possui dois papéis principais:

1) Tracker/Servidor (porta 8080)

Existe um **servidor central** que atua como um “tracker”. Ele mantém, em memória, um estado equivalente a:

- hashMap: um mapeamento de **hash do arquivo → lista de IPs** que possuem aquele arquivo
- clientData: um mapeamento de **IP do cliente → lista de hashes** que aquele cliente registrou

O protocolo de comunicação (simples) usa TCP + gob:

- O cliente se conecta ao tracker e envia uma operação:
  - "store": envia uma lista inicial de hashes do diretório do cliente
  - "create": informa que um arquivo novo apareceu (hash)
  - "delete": informa que um arquivo foi removido (hash)
  - "query": pergunta “quem tem o hash X?” e recebe a lista de IPs

Ponto importante para o lab: o servidor aceita várias conexões e cria uma goroutine por conexão. Isso significa que o estado (hashMap e clientData) pode ser acessado por múltiplas goroutines ao mesmo tempo, e **hoje está propositalmente sem proteção**.

2) Cliente (porta 9090)

Cada cliente executa duas funções ao mesmo tempo:

1. **Cliente do tracker**: conecta no servidor (porta 8080), calcula hashes do seu diretório de dataset, envia "store", e depois mantém o tracker atualizado com "create"/"delete" ao monitorar o diretório.
2. **Servidor de arquivos (peer)**: abre uma porta local (9090) para que outros peers possam conectar e pedir "download" de um hash.

Assim, o fluxo típico é:

1. Cliente A registra seus hashes no tracker
2. Cliente B consulta o tracker ("query") por um hash
3. Cliente B pega o IP retornado e conecta direto no Cliente A (porta 9090) para baixar o arquivo

Ponto importante para o lab: no cliente, existem goroutines concorrentes (monitoramento do diretório, servidor peer, loop de

interação no terminal) que podem acessar estruturas compartilhadas. Como o código foi fornecido sem sincronização, esse é outro foco de correção.

Como executar o código

A seguir vai um guia **para rodar tudo em uma única máquina**, em terminais separados, simulando múltiplos nós.

Pré-requisitos

- Ter o Go instalado (ex.: go version)
- Estar na pasta do projeto (onde existem server/ e client/)

Estrutura do dataset

O cliente espera um diretório./dataset/ **relativo ao processo**. Para simular dois clientes diferentes no mesmo PC, você deve rodar cada um em um diretório diferente (para não compartilhar o mesmo dataset) e **tomar cuidado com a porta 9090** (ver observação abaixo).

Um jeito simples:

bash

```
mkdir -p /tmp/lab8/client1/dataset
mkdir -p /tmp/lab8/client2/dataset

echo "arquivo do cliente 1" > /tmp/lab8/client1/dataset/a.txt
echo "arquivo do cliente 2" > /tmp/lab8/client2/dataset/b.txt
```

Terminal 1 – Tracker/Servidor (porta 8080)

bash

```
cd BitTorrent-pirata-main/server
go run main.go
```

Você deve ver algo como “Server is listening on port 8080...”.

Terminal 2 – Cliente 1

bash

```
cd /tmp/lab8/client1
# copie/coloque o executável/código do client aqui ou rode apontando
para o path do projeto
# exemplo rodando do repositório:
cd /caminho/para/BitTorrent-pirata-main/client
go run main.go
```

Quando pedir Enter server IP:, digite:

- 127.0.0.1

E garanta que o dataset que ele usa exista (o código usa ./dataset/).

Terminal 3 — Cliente 2

Mesmo procedimento, em outro diretório:

```
bash
cd /tmp/lab8/client2
cd /caminho/para/BitTorrent-pirata-main/client
go run main.go
```

Digite 127.0.0.1 quando solicitado.

Consultar e baixar

No Cliente 2:

1. Escolha opção 1 e informe o hash
2. Depois opção 2 para baixar e informe o caminho de saída

Observação importante (porta 9090)

O código do cliente usa a porta fixa :9090 para o "servidor peer". Se você tentar rodar **dois clientes no mesmo PC**, o segundo provavelmente falhará com "address already in use".

Para fins de laboratório (e mantendo o espírito do exercício), você tem três caminhos:

1. **Rodar clientes em máquinas diferentes** (ideal do lab)
2. **Adaptar o código para permitir escolher a porta do peer** via argumento/flag (recomendado)

3. Rodar apenas 1 cliente por vez no mesmo PC (menos útil)

Detalhes da entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script submit-answer.sh, disponibilizado na estrutura de arquivos do próprio laboratório. Uma vez que você tenha concluído sua resposta, seguem as instruções:

1. **Crie um documento de design chamado de lab8_matr1_matr2.design. Explique as suas modificações**
2. **Crie um arquivo lab8_matr1_matr2.tar.gz** somente com o conteúdo do repositório que vocês trabalharam. Para isso, supondo que o diretório raiz de seu repositório privado chama-se lab7_pc, você deve executar:

bash

```
tar -cvzf lab8_matr1_matr2.tar.gz lab8/
```

2. **Submeta o arquivo lab8_matr1_matr2.tar.gz** usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

bash

```
bash submit-answer.sh lab8 path/lab8_matr1_matr2.tar.gz
```

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Adicione os professores como colaboradores (thiagomanel e giovannifs).

Alterações no código realizadas após o prazo de entrega não serão analisadas.

Prazo

10/fev/26 23:59

Appendix

https://gobyexample.com
https://go.dev
https://go.dev/tour/list