Exercício de Programação 2
Armazenamento chave-valor distribuído

Arthur Souto Lima

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

arthursl@ufmg.br

1. Introdução

Em sistemas distribuídos, há vários modelos de comunicação, um dos mais usados em serviços de nuvem, por exemplo, é o Remote Procedure Call (RPC). Esse paradigma prevê mecanismos para permitir que procedimentos sejam chamados em máquinas diferentes, de uma forma tão próxima de uma chamada local quanto possível. Como visto em aula, há vários detalhes e cuidados a serem tomados nesse tipo de abordagem. Felizmente, com o uso de bibliotecas como o gRPC, muitos desses detalhes estão encapsulados e permitem um desenvolvimento mais direto do programador. Neste projeto, tem-se como objetivo usar os conceitos de RPC por meio da biblioteca gRPC a fim de implementar um sistema que armazena pares chaves-valor de forma distribuída. Nele, há servidores, que armazenarão esses dados, e clientesc que interagem com esses outros agentes inserindo ou consultado chaves e valores, por exemplo.

2. Compilação e Execução

O código entregue é acompanhado de um makefile para auxílio de compilação e execução do programa. O comando make compila os procedimentos do RPC e o restante dos arquivos python não precisam ser compilados.

Seguindo a especificação, pode-se usar os comandos make seguintes para executar cada um dos agentes das duas partes do exercício: *run_cli_pares, run_serv_pares_1, run_serv_pares_2, run_serv_central, run_cli_central*. Cada um possui seus argumentos específicos, como previstos nas orientações do trabalho, os quais podem ser passados para o comando via construção "arg= <parâmetros>". Alguns exemplos de comandos se encontram listados abaixo.

```
$ make run_cli_pares arg=nome_do_host_do_serv_pares:5555
$ make run_serv_pares_1 arg=5555
$ make run_serv_pares_2 arg=5555
```

2.1. Questão do chmod

Uma observação relevante quanto à execução deve ser feita acerca das permissões para tanto. Os scripts python entregues contam com shebangs nas suas primeiras linhas a fim de indicar ao sistema qual interpretador (no caso, o de python tradicional mesmo) usar para executar aquele script. Apesar disso, em algumas situações, acusava-se permissão de execução negada. Para remediar isso, pode-se usar o comando

```
$ chmod +x <nome do arquivo>
```

para conceder permissão de execução ao programa.

3. Decisões de Projeto

O objetivo do trabalho foi exercitar os conceitos de RPC na prática com a biblioteca gRPC no ambiente Python. Apesar da especificação estar repleta de detalhes, sua correspondência para a lógica foi relativamente direta. A parte mais trabalhosa foi compreender e se adaptar à interface da gRPC. Felizmente, a aula e o exemplo apresentado nela serviram de excelente ponto de partida para ulteriores pesquisas na documentação. Além disso, eventuais desafios naturais que viriam com o desenvolver do programa foram previamente endereçados com recomendações de links para guiar as soluções, o que também auxiliou no desenrolar do projeto.

Dada essa situação, não foram necessárias estruturas de dados complexas ou mesmo bibliotecas auxiliares diversas. O grande foco é realmente no gRPC com seus arquivos de definição dos procedimentos remotos. Desse modo, também como reitera a especificação, não é necessário uma grandiosa documentação, assim, este documento não entrará em muitos detalhes quanto decisões de projeto, as quais não foram muitas neste trabalho. Apesar disso, tentou-se, sempre que possível, comentar o código e usar nomes descritivos para variáveis, funções e métodos. Com isso, espera-se que o leitor não tenha dificuldades em eventuais inspeções no código entregue.

O programa foi implementado e testado em Python 3.8.10 e gRPC 1.43.0 (pacotes *grpcio* e *grpcio-tools*). O ambiente de desenvolvimento foi o Windows no WSL (*Windows Subsystem for Linux*).