Tabela de Chaves Distribuída

Redes de Computadores e Internet
2º Semestre 2019/2020
Projeto de Laboratório

1. Introdução

Uma tabela de chaves distribuída é uma versão simplificada de uma tabela de *hash* distribuída. Uma *tabela de hash* é uma estrutura de dados constituída por pares (chave, valor), em que chave é um número inteiro e valor referencia um objeto associado à chave. Cada valor é pesquisado pela sua chave. Por exemplo, as chaves podem ser números de cartões de cidadão e os valores podem ser registos com o nome, data de nascimento, sexo e morada dos portadores dos cartões.

Geralmente, o espaço de chaves é demasiado grande para uma pesquisa eficiente. Recorre-se então a uma função de *hash* que mapeia chaves arbitrárias em outras chaves pertencentes a um conjunto pequeno de números inteiros.

Numa *tabela de hash distribuída*, os pares (chave, valor) são repartidos por vários servidores por forma a que a pesquisa de uma chave seja eficiente e a entrada e a saída de um servidor não altere os pares (chave, valor) associados à maioria dos servidores. As tabelas de *hash* distribuídas são usadas, entre outros, em *redes de distribuição de conteúdos*.

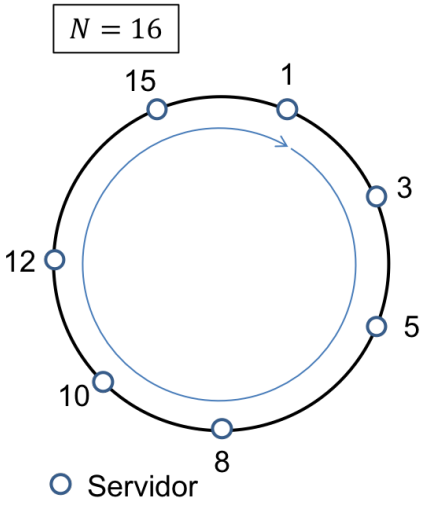
Uma *tabela de chaves distribuída* compreende as funções de pesquisa de uma chave, entrada e saída de servidores, mas ao contrário de uma tabela de *hash* distribuída propriamente dita, dispensa a recuperação, atualização e migração entre servidores dos valores associados às chaves.

1.2 Disposição de chaves em anel

Numa tabela de chaves distribuída, as chaves estão logicamente organizadas num anel orientado, por exemplo no sentido dos ponteiros do relógio. O anel tem capacidade para N chaves, vindo o espaço de chaves dado pelo intervalo $[0, (N-1)]$ dos primeiros N números inteiros. A *distância* da chave k à chave l é $d_N(k,l) = (l-k) \bmod N$, ou seja, o resto da divisão de $(l-k)$ por N. A chave l está *mais próxima* do que a chave m da chave k se a distância de k a l é inferior à distância de k a m, $d_N(k,l) < d_N(k,m)$.

Cada servidor também tem uma chave, pelo que também pertence ao anel. Por simplicidade, identificamos um servidor pela sua chave e referimo-nos à distância de um

servidor a outro como sendo a distância entre as suas respetivas chaves. Cada chave está armazenada no servidor que está mais próximo da dita chave. O *sucessor* de um servidor é o servidor que dele está mais próximo. O *predecessor* de um servidor é o servidor que o tem como sucessor.



A figura mostra um anel de 16 chaves com 7 servidores. Por exemplo, a distância de 5 a 8 é 3; a distância 13 a 2 é 5; e a distância de 4 a 2 é 14. As chaves 6, 7 e 8 estão armazenadas no servidor 8; e as chaves 0 e 1 estão armazenadas no servidor 1. O servidor 8 é o sucessor do servidor 5; e o servidor 1 é o sucessor do servidor 15.

1.3 Pesquisa de uma chave

Cada servidor mantém informação de estado sobre o seu sucessor. Suponhamos que o servidor i dá início à pesquisa da chave k. Essa pesquisa vai circular ao longo do anel até ao predecessor do servidor que armazena a chave k, o qual notificará o servidor i que o seu sucessor armazena a chave k. Tendo este objetivo presente, na pesquisa da chave k cada servidor j procede da seguinte forma:

1. se o sucessor do servidor j estiver mais próximo do que ele próprio da chave k, então o servidor j sabe que a chave k está armazenada no seu sucessor e notifica o servidor i deste facto;
2. caso contrário, se o sucessor do servidor j estiver mais longe do que ele próprio da chave k, então o servidor j delega a pesquisa da chave k no seu sucessor.

1.4 Saída de um servidor

Para precaver contra a saída abrupta de um servidor do anel, reconstruindo o anel sem este servidor, cada servidor mantém informação de estado sobre o sucessor do seu sucessor, também chamado de *segundo sucessor*. Quando o servidor i sai, o seu predecessor h e o predecessor deste têm que atualizar a informação de estado sobre os seus sucessores e segundos sucessores. Concretamente, a saída do servidor i requer as seguintes ações:

1. o servidor h (predecessor de i) deteta a saída de i, atualiza o seu sucessor com o seu segundo sucessor (sucessor de i) e informa o seu predecessor sobre o seu sucessor atualizado, o qual será o segundo sucessor do predecessor de h;
2. o sucessor de i informa o servidor h (novo predecessor do sucessor de i) sobre o seu sucessor que será de ora em diante o segundo sucessor de h.

1.5 Entrada de um servidor

A entrada de um novo servidor pressupõe, em primeiro lugar, que este tenha conhecimento de um qualquer servidor no anel ao qual possa pedir uma pesquisa da chave com a qual pretende entrar. Feita esta pesquisa, o servidor entrante fica a saber o seu futuro sucessor. Seja i o servidor entrante e j o seu futuro sucessor. O servidor i, o predecessor de j e o predecessor deste têm agora que atualizar a informação de estado sobre os seus sucessores e segundos sucessores. Concretamente, a entrada do servidor i tendo o servidor j como futuro sucessor requer as seguintes ações:

1. o servidor i atualiza o seu sucessor para o servidor j;
2. o servidor j informa o servidor i sobre seu sucessor, o qual será o segundo sucessor de i, e informa o seu predecessor sobre a entrada do servidor i;
3. o predecessor de j atualiza o seu sucessor para o servidor i e informa o seu predecessor sobre i, o qual será o segundo sucessor do atual predecessor de i.

2. Especificação

Cada grupo de dois alunos deve concretizar a aplicação **`dkt`** correspondendo a um servidor. O anel é concretizado em sessões TCP que formam uma *rede de sobreposição* à rede e computadores que a suporta. A comunicação entre servidores no anel é por TCP, enquanto que a comunicação entre um servidor que pretende entrar no anel e um servidor pertencente ao anel é por UDP. A aplicação **`dkt`** é composta pelos elementos seguintes:

- Comando de invocação da aplicação
- Interface de utilizador
- Protocolo de pesquisa de uma chave
- Protocolo para saída de um servidor
- Protocolo para entrada de um servidor

2.1 Comando de invocação da aplicação

A aplicação **`dkt`** é invocada com o comando

`dkt` *`IP port`*

em que ***IP*** é o endereço IP da máquina que aloja a aplicação e ***port*** é o porto da aplicação, isto é, o porto tanto de um processo servidor TCP como de um processo servidor UDP a ser instalado na máquina para concretização da aplicação. Em resultado da invocação, a aplicação disponibiliza uma interface de utilizador.

2.2 Interface de utilizador

A interface de utilizador consiste nos seguintes comandos.

- **`new i`**
  Criação de um novo anel composto exclusivamente pelo servidor ***i***.
- **`entry i boot boot.IP boot.TCP`**
  Entrada do servidor ***i*** no anel ao qual pertence o servidor ***boot*** com endereço IP ***boot.IP*** e porto ***boot.TCP***.
- **`sentry i succi succi.IP succi.TCP`**
  Entrada do servidor ***i*** no anel no qual o servidor ***succi*** com endereço IP ***succi.IP*** e porto ***succi.TCP*** será o sucessor do servidor ***i***. Este comando pressupõe que se sabe que a chave ***i*** não é usada por um servidor pertencente ao anel e que ***succi*** será o sucessor do servidor ***i***. Ele serve para construir um anel um servidor à vez sem necessitar da pesquisa de chaves.
- **`leave`**
  Saída do servidor do anel.
- **`show`**
  Mostra do estado do servidor, incluindo a sua chave, endereço IP e porto, bem como os valores correspondentes do seu sucessor e segundo sucessor.
- **`find k`**
  Pesquisa do servidor que armazena a chave ***k***, com a apresentação da sua chave, endereço IP e porto.
- **`exit`**
  Fecho da aplicação.

2.3 Pesquisa de uma chave

Na pesquisa de uma chave são usadas duas mensagens protocolares sobre sessões TCP.

- <**`FND k i i.IP i.port\n`** >
  Um servidor delega no seu sucessor a pesquisa da chave ***k***, a qual foi iniciada pelo servidor ***i*** com endereço IP ***i.IP*** e porto ***i.port***. (O caráter ***\n*** delimita todas as mensagens enviadas sobre TCP.)
- <**`KEY k succ succ.IP succ.port\n`** >
  Um servidor informa o servidor que iniciou a pesquisa da chave ***k*** que esta chave se encontra armazenada no seu sucessor ***succ*** com endereço IP ***succ.IP*** e porto ***succ.port***. Esta mensagem é enviada sobre uma sessão TCP criada para o

efeito, do servidor que pretende enviar a mensagem para o servidor que iniciou a pesquisa.

2.4 Saída de um servidor

Na saída de um servidor são usadas duas mensagens protocolares sobre sessões TCP.

- <**SUCCCONF\n** >
  Um servidor informa outro que este se tornou o seu sucessor.
- <**SUCC *succ succ.IP succ.port*\n** >
  Um servidor informa o seu predecessor que o seu sucessor é ***succ*** com endereço IP ***succ.IP*** e porto ***succ.port***.

2.5 Entrada de um servidor

A entrada de um servidor necessita, antes de mais, que o servidor descubra a chave, endereço IP e porto daquele que será o seu sucessor no anel. Para este efeito, são usadas duas mensagens protocolares sobre UDP.

- <**EFND *i*** >
  Um servidor entrante solicita a um servidor no anel que pesquise a chave ***i*** com a qual pretende entrar no anel.
- <**EKEY *i succ succ.IP succ.port*** >
  Um servidor do anel responde a um servidor entrante informando-o que o sucessor do servidor com chave ***i*** será o servidor ***succ*** com endereço IP ***succ.IP*** e porto ***succ.port***.

Uma vez conhecido o seu futuro sucessor, a entrada de um servidor é baseada em três mensagens protocolares sobre TCP.

- <**SUCCCONF\n** >
  Mesmo significado que anteriormente aquando da saída de um servidor.
- <**SUCC *succ succ.IP succ.port*\n** >
  Mesmo significado que anteriormente aquando da saída de um servidor.
- <**NEW *i i.IP i.port*\n** >
  Esta mensagem é usada em dois contextos diferentes. (1) Um servidor entrante informa o seu futuro sucessor que pretende entrar no anel com chave ***i***, endereço IP ***i.IP*** e porto ***i.port***. (2) Um servidor informa o seu atual predecessor que o servidor de chave ***i***, endereço IP ***i.IP*** e porto ***i.port*** pretende entrar no anel, para que o predecessor estabeleça o servidor entrante como seu sucessor.

3. Desenvolvimento

Cada grupo de alunos deve adquirir a destreza necessária sobre programação em redes para realizar a aplicação proposta. Os passos seguintes podem ajudar a desenvolver a aplicação.

i. Criação de um anel tendo informação do sucessor de cada servidor entrante, comandos ***new*** e ***sentry***.
ii. Mostra do estado, comando ***show***.
iii. Saída de um servidor do anel, comandos ***leave*** e ***exit***.
iv. Pesquisa de uma chave a partir de um servidor, comando ***find***.
v. Entrada de um servidor no anel apenas com informação sobre um qualquer servidor do anel, comando ***entry***.

O código da aplicação deverá ser comentado e testado à medida que é desenvolvido. O projeto será compilado e executado pelo corpo docente **apenas** no ambiente de desenvolvimento disponível no laboratório.

O código da aplicação fará uso das seguintes chamadas de sistema:

- Leitura de informação do utilizador para a aplicação: `fgets()`;
- Decomposição de *strings* em tipos de dados e vice-versa: `sscanf()`, `sprintf()`;
- Gestão de um cliente UDP: `socket()`, `close()`;
- Gestão de um servidor UDP: `socket()`, `bind()`, `close()`;
- Comunicação UDP: `sendto()`, `recvfrom()`;
- Gestão de um cliente TCP: `socket()`, `connect()`, `close()`;
- Gestão de um servidor TCP: `socket()`, `bind()`, `listen()`, `accept()`, `close()`;
- Comunicação TCP: `write()`, `read()`;
- Multiplexagem de informação: `select()`.

Quer os clientes quer os servidores devem terminar graciosamente, pelo menos nas seguintes situações de falha:

- Mensagens de protocolo erradas vindas da entidade par correspondente;
- Sessão TCP fechada abruptamente, quer do lado do cliente quer do lado do servidor;
- Condições de erro nas chamadas de sistema.

4. Bibliografia

- José Sanguino, A Quick Guide to Networking Software, 5ª edição, 2020.
- W. Richard Stevens, Unix Network Programming: Networking APIs: Sockets and XTI (Volume 1), 2ª edição, Prentice-Hall PTR, 1998, ISBN 0-13-490012-X, capítulo 5.

- Michael J. Donahoo, Kenneth L. Calvert, TCP/IP Sockets in C: Practical Guide for Programmers, Morgan Kaufmann, ISBN 1558608265, 2000.
- Manual on-line, comando `man`.

5. Entrega do Projecto

O código fonte da aplicação **`dkt`** deve ser guardado num arquivo `zip` juntamente com a respetiva `makefile`. A entrega do trabalho é feita por e-mail ao seu docente de laboratório. O arquivo deve estar preparado para ser aberto para o directório corrente e compilado com o comando `make`. O arquivo submetido deve ter o seguinte formato: `proj<número_do_grupo>.zip` (ex: `proj07.zip`). A data limite de entrega é 5/04/2020 às 23:59.