Sistemas Distribuídos — Trabalho Prático
Armazenamento de dados em memória com acesso remoto

Engenharia Informática
Universidade do Minho

2024/2025

Informações gerais

- Cada grupo deve ser constituído por 4 elementos, obrigatoriamente inscritos no *eLearning* até 8 de novembro de 2024.
- Deve ser entregue o código fonte e um relatório de até 6 páginas (A4, 11pt) no formato PDF (a capa não é contabilizada neste limite).
- O trabalho deve ser entregue até às 23:59 do dia 28 de dezembro de 2024 no *eLearning*.
- A apresentação do trabalho será em dia e hora a agendar posteriormente por cada grupo.
- Cada grupo deve organizar a sua apresentação de forma a que todos os elementos participem espontaneamente.

Resumo

Neste projeto pretende-se a implementação de um serviço de armazenamento de dados partilhado, em que a informação é mantida num servidor e acedida remotamente. Clientes interagem com o servidor através de sockets TCP, de forma a inserir e consultar informação. O servidor atende clientes concorrentemente e armazena a informação em memória. A interface disponibilizada, bem como o armazenamento da informação, é no formato chave-valor. *Valorizam-se estratégias que diminuam a contenção* e minimizem o número de threads acordadas. O relatório deverá descrever a arquitetura e protocolos do sistema, bem como justificar as principais decisões de desenho assumidas na elaboração do projeto.

Funcionalidade básica (14 valores)

O serviço deverá suportar as seguintes funcionalidades básicas:

1. Autenticação e registo do utilizador
    - O serviço deverá possibilitar o registo e autenticação de utilizadores, dado o seu nome e palavra passe.
    - Sempre que um utilizador desejar interagir com o serviço, deverá estabelecer uma conexão e ser autenticado pelo servidor.
2. Operações de escrita e leitura simples

- Operação de escrita, enviando par chave-valor: `void put(String key, byte[] value)`. Se a chave não existir, é criada uma nova entrada no servidor, com o par chave-valor enviado. Caso contrário, a entrada deverá ser atualizada com o novo valor.
- Operação de leitura: `byte[] get(String key)`. Para uma chave *key*, deverá devolver ao cliente o respetivo valor, ou *null* caso a chave não exista.

3. Operações de escrita e leitura compostas
   - Operação de escrita composta; `void multiPut(Map<String, byte[]> pairs)`. Todos os pares chave-valor deverão ser atualizados / inseridos atomicamente.
   - Operação de leitura composta: `Map<String, byte[]> multiGet(Set<String> keys)`. Dado um conjunto de chaves, devolve o conjunto de pares chave-valor respetivo.
4. Limite de utilizadores concorrentes
   - Sendo S um parâmetro de configuração, só podem existir no máximo S sessões concorrentes (diferentes clientes a usar o servidor). Quando atingido, a autenticação de um cliente ficará em espera até sair outro cliente.

Funcionalidade avançada (6 valores)

5. Suporte a clientes *multi-threaded*
   - O cliente deverá poder ter várias *threads* concorrentemente a submeter pedidos ao servidor. Um pedido que fique bloqueado no servidor não deverá impedir outros pedidos que o cliente submeta concorrentemente de serem servidos.
6. Operação de leitura condicional
   - Leitura condicional: `byte[] getWhen(String key, String keyCond, byte[] valueCond)`. Deverá ser devolvido o valor da chave *key* quando o valor relativo à chave *keyCond* seja igual a *valueCond*, devendo a operação ficar bloqueada até tal acontecer.

Programa Servidor

Programa servidor deve ser implementado em Java, usando *threads* e *sockets* TCP, mantendo em memória a informação relevante para suportar as funcionalidades acima descritas, receber conexões e input dos clientes, bem como fazer chegar a estes a informação pretendida.

Biblioteca do cliente

Deverá ser disponibilizada uma biblioteca (conjunto de classes e interfaces) que proporcione o acesso à funcionalidade do serviço descrita acima. Esta biblioteca deve ser independente da interface com o utilizador e deverá ser escrita em Java usando *threads* e *sockets* TCP.

Interface do utilizador

Finalmente, deverá ser disponibilizada uma interface com o utilizador que permita interagir com o serviço através da biblioteca cliente. Esta interface deverá também ser escrita em Java e tem como único objetivo a interação com o serviço para testes e durante a apresentação do trabalho.

Avaliação de desempenho

De modo a perceber o desempenho do serviço, conceba diferentes cenários de testes, incluindo cargas de trabalho com diferentes tipos de operações, e testes de escalabilidade (*i.e.,* verificar o desempenho do sistema com o aumento do número de clientes a submeter pedidos). Pode assumir o benchmark YCSB (`https://github.com/brianfrankcooper/YCSB`) como inspiração para o cliente de testes. O relatório deve incluir uma reflexão sobre os resultados dos testes efetuados relacionando-os com as decisões de desenho que tomou.

Requisitos

Na implementação devem ser satisfeitos os seguintes requisitos:

- Para cada cliente, deve haver apenas uma única conexão entre o cliente e servidor.
- O protocolo de comunicação deverá ser num formato binário, através de código desenvolvido no trabalho, podendo recorrer apenas a `Data[Input|Output]Stream`.
- Para o serviço não ficar vulnerável a clientes lentos, não deverá ter *threads* do servidor a escrever em mais do que um socket, devendo as escritas ser feitas por *threads* associadas a esse socket.
- Todas as operações devem ser atómicas.