Fundamentos de Sistemas Distribuídos

Trabalho Prático

2020/2021

Informações gerais

- Cada grupo deve ser constituído por até 5 elementos.
- Não deve ser apresentada uma interface para o utilizador, apenas uma interface para o programador de aplicações (API) e exemplos da sua utilização.
- Deve ser entregue o código fonte e ainda um relatório até 6 páginas em formato pdf.
- O trabalho deve ser entregue até 15 de janeiro de 2021 no *e-Learning*.

Objetivo

Desenvolvimento de um sistema distribuído de armazenamento em memória de pares chave–valor, assumindo um sistema assíncrono e tirando partido de relógios lógicos. O trabalho deve ser elaborado utilizando Java (e opcionalmente com Atomix, considerando apenas as classes já usadas nas aulas). O sistema deve admitir que o conjunto de servidores é fixo, previamente conhecido e não há falhas. A distribuição das chaves pelos servidores pode ser também fixa recorrendo a uma funçãode *hashing*.

Requisitos

1. O sistema deve usar vários servidores para armazenar subconjuntos disjuntos de chaves (`Long`) a que são associados valores (`byte[]`) oferecendo as seguintes operações aos clientes:
   - `CompletableFuture<Void> put(Map<Long,byte[]> values)`: Escreve um conjunto de pares chave–valor.
   - `CompletableFuture<Map<Long,byte[]> get(Collection<Long> keys)`: Lê os valores associados a um conjunto de chaves.
2. O sistema proposto deve garantir que:
   - um cliente observa todas as escritas feitas por si próprio anteriormente;
   - se dois clientes tentarem escrever concorrentemente os mesmos itens, os valores que persistem para serem lidos depois das operações terminarem são todos provenientes do mesmo cliente.

Valorização

O trabalho é valorizado se:

1. A resolução proposta for modular, maximizando o código que é independente desta aplicação em concreto e adequado à utilização em grande escala, sem gargalos ou limitações artificiais.
2. Garantir que leituras concorrentes de itens que estão a ser modificados por outros clientes não observam uma escrita parcial.
3. Incluir uma avaliação de desempenho.
4. Não utilizar Atomix mas apenas *sockets* assíncronos.