LICENCIATURA EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Teste[1]

18 de dezembro de 2025 Duração: 2h00m

Responda a cada grupo numa folha separada, entregando obrigatoriamente três folhas.

I

Responda diretamente a cada pergunta e com grafia bem legível. Omita preâmbulos e considerações genéricas sobre cada um dos assuntos.

1 O controlo de concorrência, por exemplo, com trincos (*locks*), tem impacto no desempenho. Explique quais as estratégias genéricas para minimizar esse impacto. Use exemplos.

2 Considere um sistema em três camadas (*3-tier*) em que os clientes contactam um servidor aplicacional A, sem estado persistente, que usa por sua vez um servidor B de base de dados. Discuta as estratégias de *threading* adequadas a cada um dos servidores.

3 Explique como é que o protocolo 2PC (*two phase commit*) resolve uma falha de um participante durante a segunda fase.

4 Identifique o desafio principal na procura de informação num sistema distribuído e explique como é resolvido por uma DHT como o *Chord*.

II

Considere um sistema de gestão de projetos, que permite a adição e execução de tarefas por parte de utilizadores. Para adicionar uma tarefa a ser executada, um utilizador fornece ao sistema o identificador da mesma, o número de passos necessários para ser concluída, e o conjunto de dependências (i.e., identificadores de outras tarefas). Para executar um passo de uma tarefa, um utilizador indica ao sistema o identificador da tarefa, recebendo posteriormente a confirmação da alocação quando for possível iniciar a execução. Finalmente, o utilizador indicará ao sistema quando terminar a execução do passo da tarefa. Uma tarefa é considerada concluída quando todas os seus passos estiverem completos.

Apresente uma classe Java (para ser usada no servidor), usando primitivas baseadas em monitores, que implementa a interface abaixo, tendo em conta que os seus métodos serão invocados num ambiente *multi-threaded*.

```
interface Manager {
    void add(String id, int nsteps, Set<String> deps);
    bool acquire(String id) throws InterruptedException;
    void complete(String id);
    Set<String> getCompleted();
}
```

O método `add` serve para adicionar uma tarefa, recebendo como argumento o identificador, o número de passos e o conjunto de dependências. O método `acquire` serve para adquirir um passo da tarefa identificada pelo argumento. Deverá retornar `true` em caso de sucesso, ou `false` caso a tarefa não exista, tenha sido já concluída, ou não existam mais passos disponíveis. Em caso de sucesso, deve bloquear até a tarefa poder ser iniciada, i.e., todas as suas dependências tiverem sido concluídas. O método `complete` serve para indicar que um passo da tarefa identificada pelo argumento foi completado. O método `getCompleted` retorna o conjunto de tarefas que já concluíram. Minimize as *threads* que são acordadas desnecessariamente.

III

Considere o sistema descrito acima, ao qual os utilizadores ligam-se por TCP. Implemente só o programa servidor usando *threads*, *sockets* TCP, e a interface apresentada na pergunta anterior. Considere que um utilizador pode adquirir passos de várias tarefas mesmo enquanto espera pela resposta a outro pedido. Descreva o protocolo usado, que deve ser o mais simples possível, por exemplo, baseado em linhas de texto.

---

[1] Cotação — $4 \times 2.5 + 7 + 3$