UNIVERSIDADE FEDERAL DE VIÇOSA

Departamento de Informática

INF310 – PROGRAMAÇÃO CONCORRENTE E DISTRIBUÍDA

LISTA DE QUESTÕES TEÓRICAS

Fundamentos de Programação Concorrente

---

1. (3pts) Desempenho e Tipos de Processos

Em um sistema operacional, processos podem ser classificados como CPU-bound ou I/O-bound. Um administrador de sistema observou que dar prioridade aos processos I/O-bound resulta em melhor eficiência geral do sistema.

a) Explique por que essa estratégia de priorização aumenta a eficiência da utilização dos recursos do sistema.

b) Descreva o que aconteceria com o desempenho geral se fosse dada prioridade aos processos CPU-bound. Justifique sua resposta considerando o comportamento de cada tipo de processo.

---

2. (4pts) Análise Comparativa de Paradigmas

Compare e contraste os três paradigmas computacionais: programação concorrente, programação paralela e sistemas distribuídos.

a) Para cada paradigma, explique suas características principais e objetivos.

b) Identifique uma situação prática específica onde cada paradigma seria mais apropriado e justifique sua escolha.

c) Explique quais são os principais desafios únicos de cada paradigma que não existem nos outros.

---

3. (3pts) Condições de Corrida em Diferentes Arquiteturas

As condições de corrida (race conditions) são um problema fundamental em programação concorrente.

a) Defina precisamente o que é uma condição de corrida e explique por que ela ocorre.

b) Uma condição de corrida pode ocorrer em sistemas monoprocessados? Justifique sua resposta explicando o mecanismo pelo qual isso acontece ou por que não é possível.

c) Compare como as condições de corrida se manifestam diferentemente em arquiteturas monoprocessadas versus multiprocessadas.

---

4. (4pts) Análise de Desempenho Temporal

Considere dois processos P1 e P2 que, quando executados individualmente em um sistema monoprocessado, levam tempos T1 = 100ms e T2 = 150ms respectivamente. Assuma que P1 é CPU-bound e P2 gasta 60% do seu tempo em operações de I/O.

a) Qual seria o tempo total de execução se os processos fossem executados sequencialmente?

b) É possível que a execução concorrente desses dois processos no mesmo sistema monoprocessado resulte em um tempo total menor que T1 + T2? Explique detalhadamente o mecanismo que tornaria isso possível.

c) Calcule o tempo mínimo teórico para execução concorrente, assumindo sobreposição perfeita de CPU e I/O.

d) Como a resposta mudaria se ambos os processos fossem CPU-bound? Justifique.

---

5. (3pts) Threads versus Processos

Explique as principais diferenças entre threads e processos, considerando os seguintes aspectos:

a) Compartilhamento de recursos e espaço de memória.

b) Overhead de criação e troca de contexto.

c) Em que situações você escolheria usar threads ao invés de processos e vice-versa? Forneça exemplos práticos específicos.

---

6. (4pts) Tipos de Sincronização

Identifique e explique os três tipos principais de sincronização entre threads/processos:

a) Para cada tipo, descreva sua característica fundamental e quando é necessário.

b) Forneça um exemplo prático específico (não código, mas situação real) onde cada tipo de sincronização seria aplicado.

c) Explique qual tipo é mais complexo de implementar e por quê.

---

7. (3pts) Espera Ocupada versus Bloqueio

A espera ocupada (busy waiting) é uma técnica de sincronização onde uma thread/processo continuamente verifica uma condição.

a) Explique o que caracteriza um mecanismo de exclusão mútua com espera ocupada.

b) Quais são os principais problemas desta abordagem, considerando tanto eficiência quanto comportamento do sistema?

c) Em que cenários específicos a espera ocupada poderia ser preferível ao bloqueio? Justifique sua resposta.

---

8. (4pts) Análise de Eficiência em Multiprocessadores

Um programador desenvolveu um algoritmo que, quando executado sequencialmente, leva 200 segundos. Ele conseguiu paralelizar o algoritmo e observou os seguintes tempos de execução:

- 2 processadores: 120 segundos
- 4 processadores: 80 segundos
- 8 processadores: 70 segundos

a) Calcule o speedup e a eficiência para cada configuração.

b) Explique por que a eficiência diminui conforme aumenta o número de processadores.

c) Baseado nos dados, estime qual seria o tempo com 16 processadores e explique seu raciocínio.

d) O que limitaria o speedup máximo teórico deste algoritmo?

---

9. (3pts) Arquitetura de Memória e Concorrência

a) Explique como a arquitetura de memória (compartilhada vs distribuída) influencia o design de programas concorrentes.

b) Descreva os principais desafios de sincronização que surgem especificamente em sistemas de memória compartilhada.

c) Como esses desafios diferem daqueles encontrados em sistemas de memória distribuída?

---

10. (4pts) Overhead de Concorrência

A introdução de concorrência em um programa sempre traz overhead adicional.

a) Identifique e explique pelo menos quatro fontes diferentes de overhead em programas concorrentes.

b) Para cada fonte de overhead identificada, descreva uma estratégia específica para minimizá-la.

c) Explique por que, mesmo com esses overheads, a programação concorrente ainda é vantajosa em muitos cenários.

d) Em que situações o overhead da concorrência poderia ser maior que os benefícios? Dê um exemplo específico.

---

11. (3pts) Granularidade de Paralelização

a) Explique o conceito de granularidade em paralelização (grossa vs fina).

b) Como a escolha da granularidade afeta o desempenho e a complexidade do programa?

c) Descreva um cenário onde granularidade grossa seria preferível e outro onde granularidade fina seria melhor. Justifique suas escolhas.

---

12. (4pts) Modelo de Programação Concorrente

Considere um sistema que processa transações bancárias onde múltiplas threads acessam contas para realizar transferências.

a) Identifique os principais problemas de concorrência que podem ocorrer neste cenário.

b) Explique por que simplesmente usar um mutex global para todas as operações não seria uma solução eficiente.

c) Proponha uma estratégia de sincronização mais eficiente (em alto nível, sem código) que minimize contenção enquanto garante consistência.

d) Como a estratégia proposta lidaria com o problema de deadlock potencial?

---

13. (3pts) Escalabilidade e Lei de Amdahl

a) Enuncie a Lei de Amdahl e explique seu significado no contexto de programação paralela.

b) Um programa tem 30% de código que não pode ser paralelizado. Qual é o speedup máximo teórico possível, independentemente do número de processadores?

c) Explique por que a Lei de Amdahl é considerada uma limitação fundamental na paralelização e como os programadores podem trabalhar para minimizar seu impacto.

---

14. (4pts) Modelos de Comunicação

a) Compare os dois modelos principais de comunicação em sistemas concorrentes: memória compartilhada e troca de mensagens.

b) Para cada modelo, explique suas vantagens e desvantagens principais.

c) Descreva um problema específico que seria naturalmente mais adequado para cada modelo e justifique sua escolha.

d) Como a escolha do modelo de comunicação afeta a complexidade de debugging e manutenção do código?

---

15. (3pts) Determinismo em Programas Concorrentes

a) O que significa dizer que um programa concorrente é determinístico versus não-determinístico?

b) Quais fatores contribuem para o não-determinismo em programas concorrentes?

c) Em que situações o não-determinismo é aceitável e em quais é problemático? Forneça exemplos específicos de cada caso.

INSTRUÇÕES:

- Todas as respostas devem ser fundamentadas tecnicamente
- Use terminologia precisa e adequada
- Quando solicitados exemplos, seja específico e realista
- Cálculos devem incluir o desenvolvimento
- Justifique sempre suas afirmações

Tempo sugerido: 3-4 horas