Universidade Federal do ABC
Prova de MCTA025-13 — Sistemas Distribuídos

Turmas NA1MCTA025-13SA e NA2MCTA025-13SA
Segundo Quadrimestre de 2018

Prof. Dr. Emilio Francesquini

Questão	Pontos	Nota
1	2	
2	2½	
3	2½	
4	3	
5	1	
Total:	11	

Nome:

RA:

- A prova tem duração de **110 minutos**
- Não esqueça de preencher a caneta **todos** os campos na primeira página da folha de prova e seu nome e RA nesta folha de questões.
- Comece a resposta para cada uma das questões em uma nova página.
- Antes da resposta da questão i escreva "Questão i" em letras garrafais.
- As respostas às questões podem ser deixadas a lápis.
- Ao final da prova entregue tanto a folha de questões quanto a folha de prova.
- Em caso de fraude, **TODOS** os envolvidos:
  - **Receberão conceito final F (reprovado) na disciplina**
  - Serão **denunciados** à Comissão de Transgressões Disciplinares Discentes da Graduação e à Comissão de Ética da UFABC cuja punição pode resultar em **advertência, suspensão ou desligamento**, de acordo com os artigos 78-82 do Regimento Geral da UFABC e do artigo 25 do Código de Ética da UFABC.

Boa Prova!

Questão 1. Middlewares orientados a mensagem (também chamados de sistemas de enfileiramento de mensagens) são sistemas que proporcionam comunicação assíncrona e persistente. Neste contexto, responda:

(a) (1 ponto) O que significa exatamente dizer que um sistema distribuído garante (i) *comunicação assíncrona* e (ii) *persistente*?

Solução: Comunicação assíncrona significa que o remetente de uma mensagem não precisa esperar até que o destinatário tenha recebido (ou até mesmo processado) a sua mensagem antes de continuar a sua execução. Comunicação persistente se refere ao fato do destinatário não precisar estar, necessariamente, em execução no momento que a mensagem é enviada assim como o remetente não precisa estar em execução no momento em que a mensagem é entregue. A mensagem é persistida (salva) em uma terceira entidade (o middleware orientado a mensagem) que se encarrega de efetuar a sua transferência.

(b) (1 ponto) Descreva com detalhes o funcionamento de um middleware orientado a mensagens. Não deixe de explicar o papel de cada uma de suas operações primitivas (`put`, `get`, `poll` e `notify`) no seu funcionamento.

Solução: Veja slides das aulas 06 e 07.

Middlewares orientados à mensagens efetuam comunicação assíncrona e persistente com o uso de filas persistentes (buffers) de comunicação. A operação PUT Adiciona uma mensagem à fila especificada; GET Bloqueia até que a fila especificada tenha alguma mensagem e remove a primeira mensagem; POLL Verifica se a fila especificada tem alguma mensagem e remove a primeira, é uma chamada não bloqueante; NOTIFY Instala um tratador para ser chamado sempre que uma mensagem for inserida em uma determinada fila.

Questão 2. Sobre escalabilidade de sistemas distribuídos

(a) (1 ponto) Descreva com exatidão o que quer dizer *"sistema escalável"*

Solução: [ST] Seção 1.2.4
A escalabilidade de um sistema pode ser medida em basicamente 3 dimensões. Escalabilidade em relação ao seu tamanho, que indica que o sistema pode ser facilmente adaptado a uma quantidade maior de usuários pela simples adição de recursos computacionais. Escalabilidade geografica, que indica a capacidade de um sistema atender usuários que não se localizam perto geograficamente. Escalabilidade administrativa, que se refere a um sistema que é de fácil manutenção ainda que ele abranja diferentes organizações e se estenda geograficamente.

(b) (1½ pontos) Descreva sucintamente três técnicas usadas para atingir escalabilidade em sistemas distribuídos.

Solução:

1. Comunicação assíncrona - Ponto principal: Um sistema não mais é limitado pelo desempenho de outro ou pela comunicação.

2. Distribuição - Ponto principal: Divisão de um componente em múltiplas partes e espalhá-las pela rede.

3. Replicação - Ponto principal: Aumento da disponibilidade e melhoria no equilíbrio de carga entre os componentes aumentando o desempenho.

Questão 3. Um cliente faz chamadas RPC a um servidor. O cliente leva 3 milissegundos para computar os argumentos para cada requisição e o servidor leva 7 milissegundos para processar cada requisição. O sistema operacional leva 0.2 milissegundos para processar cada uma das operações send ou receive. A rede leva 3 milissegundos para transmitir cada uma das requisições e 3 milissegundos para cada uma das respostas. Empacotamento (marshalling) e desempacotamento (unmarshalling) levam o mesmo tempo: 0.5 milissegundos por mensagem.

(a) (2 pontos) Estime o tempo gasto pelo cliente para efetuar (e obter os resultados) de 2 requisições ao servidor nos seguintes casos:

- O cliente é single-threaded
- O cliente tem 2 threads que são capazes de fazer requisições concorrentes mesmo em um único processador

Solução:
Caso single-threaded:

- Cliente efetuando a requisição: 3 (argumentos) + 0,5 (marshalling) + 0,2 (send - SO)
- Rede: 3
- Servidor: 0,2 (receive - SO) + 0.5 (unmarshalling) + 7 (processamento da requisição) + 0,5 (marshalling) + 0,2 (send - SO)
- Rede: 3
- Cliente recebendo a resposta: 0,2 (receive - SO) + 0.5 (unmarshalling)

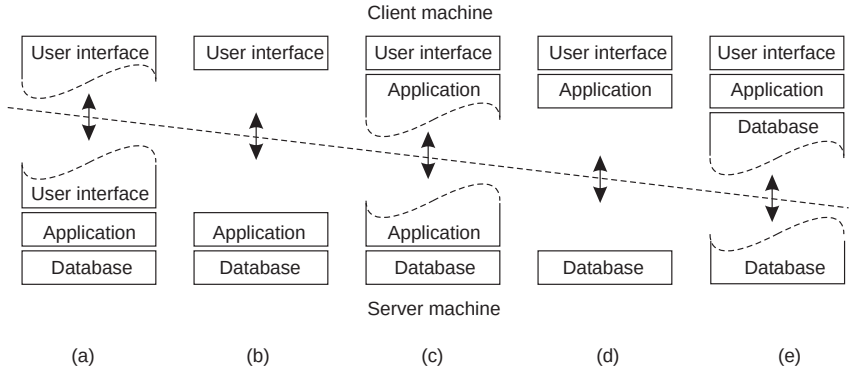
Total: 18,8 ms por chamada
Como são feitas 2 chamadas em sequência: $2 \times 18{,}8 =$ **37,6 ms**

2 threads:
O fato do cliente possuir dois threads significa que ele será capaz de duas chamadas simultâneas. Ou seja, como cada chamada leva 18,8 ms, e como ambas serão feitas em paralelo, o tempo total neste caso (assumindo que o servidor também é capaz de tratar as chamadas em paralelo) será de **18,8ms**.

(b) (½ ponto) É necessária a existência de chamadas RPC assíncronas se o cliente for multi-threaded? Justifique.

Solução: Não é necessária já que, caso o usuário utilize um thread por requisição é possível simular a existência de RPC assíncrono.

Questão 4. (a) (1 ponto) Dê um exemplo de aplicação para cada uma das arquiteturas tradicionais de duas camadas mostradas na figura abaixo:



Solução: a) terminal burro, (b) terminal X, (c) CMS ou aplicações Web, (d) aplicações transacionais onde o controle da transação fica do lado do cliente, (e) Web browser com caches.

(b) (1 ponto) Por muito tempo pudemos observar uma tendência de mudança nas arquiteturas de aplicações que usavam clientes magros (*thin clients*), que passaram a usar clientes gordos (*fat clients*). Mas depois essa tendência se inverteu. Por que os desenvolvedores passaram a evitar o uso de *fat clients*?

Solução: A razão é simples: fat clients implicam um grande esforço para que as versões mais atualizadas e corretas sejam instaladas e mantidas nas máquinas dos clientes. Essa manutenção não é alcançada facilmente já que as máquinas dos clientes são geralmente mantidas pelos próprios usuários. A quantidade de trabalho envolvida pelo fornecedor de software e os problemas invariavelmente causados por instalações incorretas ou defasadas tornava este modo de trabalho pouco atraente.

(c) (1 ponto) Recentemente, as aplicações Web estão fazendo com que as aplicações *fat clients* voltem a ficar mais populares. O que mudou?

Solução: Web browsers modernos (assim como os principais sistemas operacionais móveis) permitem instalação automática, verificação de versão e atualização transparente de plugins/aplicativos. Não raramente estes plugins/apps são aplicações completas. O advento destas novas tecnologias facilitaram em muito o gerenciamento de instalações de software distribuídas.

Questão 5. (1 ponto) Comente, critique, reclame, e/ou elogie a disciplina, o material, o projeto e/ou o professor. Caso deseje, sugira mudanças que você gostaria de ver na segunda metade do quadrimestre.