Sistemas Distribuídos
Andressa Almeida dos Santos

Geral

1. Sobre o processamento paralelo e distribuído, assinale (V) ou (F) para cada afirmação:

A. A computação paralela é caracterizada pelo uso de vários processadores para executar uma computação de forma mais rápida, baseando-se no fato de que o processo de resolução de um problema pode ser dividido em tarefas menores, que podem ser realizadas simultaneamente através de algum tipo de coordenação.
Verdadeiro (V)
A computação paralela utiliza vários processadores para executar partes de uma computação simultaneamente. Isso é possível quando um problema pode ser dividido em tarefas menores que são coordenadas entre si.

B. A execução de tarefas em um ambiente de processadores distribuídos com acoplamento fraco prevê que a memória seja compartilhada entre os processos trabalhadores.
Falso (F)
Em sistemas distribuídos com acoplamento fraco, cada processador possui sua própria memória local, e a comunicação entre os processos é feita via mensagens, não há compartilhamento de memória entre os processos.

C. Em programação paralela não é necessário se conhecer a arquitetura de comunicação entre processadores para elaborar os programas.
Falso (F)
Para programar de forma eficiente em computação paralela, é fundamental conhecer a arquitetura de comunicação entre os processadores, como compartilhamento de memória ou troca de mensagens, pois isso afeta o desempenho e a correta implementação dos algoritmos.

D. Um grid computacional pode ser formado por diversos computadores, heterogêneos, que não podem estar distribuídos via web.
Falso (F)
Um grid computacional pode ser formado por computadores heterogêneos e estar distribuído via web. O objetivo do grid é utilizar recursos distribuídos e muitas vezes geograficamente dispersos.

E. Um sistema distribuído fortemente acoplado é formado por um ambiente de computadores dotados de memória e sistema operacional próprios, que se comunicam via switch.
Falso (F)
Um sistema distribuído fortemente acoplado geralmente se refere a sistemas com memória compartilhada e comunicação de alta velocidade (como em clusters), e não a computadores com sistemas operacionais próprios que se comunicam via switch, o que caracteriza sistemas fracamente acoplados.

2. Nos anos 1970, os sistemas executavam em mainframes com aplicativos escritos em linguagens estruturadas e com todas as funcionalidades em um único módulo, com grande

quantidade de linhas de código. Acessos a bancos de dados não relacionais, regras de negócios e tratamento de telas para terminais “burros” ficavam no mesmo programa. Posteriormente, uma importante mudança ocorreu: a substituição dos terminais “burros” por microcomputadores, permitindo que todo o tratamento da interface, e de algumas regras de negócios, passassem a ser feitas nas estações clientes. Surgiam as aplicações cliente-servidor. A partir dos anos 1990, até os dias atuais, as mudanças foram radicais, os bancos de dados passaram a ser relacionais e distribuídos. As linguagens passaram a ser orientadas a objetos, cuja modelagem encapsula dados e oferece funcionalidades através de métodos. A interface passou a ser web. Vive-se a era das aplicações em três camadas. Considerando a evolução da arquitetura de software, conforme citado no texto acima, avalie as seguintes informações, marcando V (Verdadeiro) ou F (Falso). Para aquelas que julgar falsas, escreva uma justificativa.

A. **Verdadeiro (V)**
   A separação em três camadas (apresentação, lógica de negócios e dados) aumenta a complexidade, pois exige maior planejamento e especialistas em diferentes áreas (front-end, back-end e banco de dados).
B. **Verdadeiro (V)**
   Essa separação permite maior flexibilidade, pois alterações em uma camada (como a interface) não necessariamente impactam as outras (lógica ou banco de dados), desde que as interfaces entre elas estejam bem definidas.
C. **Falso (F)**
   A separação em três camadas **reduz** o acoplamento, tornando a manutenção mais fácil. Cada camada é projetada para ser independente, e essa modularidade facilita alterações ou substituições em partes específicas do sistema sem interferir nas demais.

3. Segundo Andrew Tanenbaum (2007) “Sistema Distribuído é uma coleção de computadores independentes que se apresenta ao usuário como um sistema único e consistente”. Assinale a alternativa correta a respeito de um sistema de informação distribuído e, para cada incorreta, justifique.

A. A distribuição de tarefas se dá a partir de requisições do usuário, que indica o endereço do servidor onde deseja executar tal tarefa
   Incorreta
   Nos sistemas distribuídos, a distribuição de tarefas não depende do usuário indicar manualmente o endereço do servidor. O sistema utiliza mecanismos automáticos para balancear e alocar tarefas, garantindo maior transparência e eficiência.

B. Em uma rede de computadores há servidores dedicados a atender pedidos dos clientes e estes, por sua vez, têm função exclusiva de requisitantes
   Incorreta
   Em redes de computadores tradicionais, pode haver servidores dedicados e clientes requisitantes, mas isso não é uma regra fixa em sistemas distribuídos. Nestes, os

computadores podem ter papéis dinâmicos, alternando entre cliente e servidor, dependendo do contexto.

C. Todos os computadores de uma rede executam tarefas de cliente e servidor, quando se deseja integrá-los em uma arquitetura de sistemas distribuídos.
Correta
Em sistemas distribuídos, é comum que os computadores atuem tanto como clientes quanto como servidores, dependendo das tarefas específicas que estão realizando, promovendo maior integração e cooperação entre os nós da rede.

D. A transparência de acesso é uma característica dos sistemas distribuídos que permite que recursos sejam acessados sem que sua localização seja determinada.
Correta
A transparência de acesso é uma característica fundamental dos sistemas distribuídos, permitindo que o usuário acesse recursos sem precisar saber onde estão localizados fisicamente na rede.

E. Em um sistema de objetos distribuídos é possível invocar métodos de um objeto, ainda que este não esteja presente no computador do usuário.
Correta
Em sistemas de objetos distribuídos, como o CORBA ou o RMI, é possível invocar métodos de objetos localizados em máquinas remotas, graças à abstração e ao middleware que gerencia essas chamadas.

4. A criação dos serviços em nuvens (clouds) teve como consequência o fato de as tarefas de armazenamento de dados (como arquivos e documentos) e mensagens (webmail) deixarem de ser executadas em estações cliente locais sem conexão à rede e passarem a ser delegadas a equipamentos remotos conectados através da Internet. Cada vez mais surgem empresas que oferecem nuvens de equipamentos conectados através da Internet, com clusters de equipamentos e redundância em múltiplos sites para prestação terceirizada desse tipo de serviço, de modo a oferecer maior desempenho e disponibilidade. Por outro lado, aumentam os riscos de quebra da privacidade dos lados armazenados. Nesse contexto de mudança de um sistema local para a adoção de serviços em nuvens, responda às questões a seguir.

1. Como mudam os requisitos da plataforma do cliente?
A plataforma do cliente precisa de menos recursos locais, ou seja, menos memória, menos processamento, pois as tarefas mais pesadas serão executadas na nuvem. Torna-se necessário o acesso a internet, além de browsers ou aplicações necessárias para se conectar aos serviços em nuvem.

2. Como esse tipo de serviço pode apresentar melhor disponibilidade e menor risco de perda de dados?
Redundância: Os provedores de nuvem utilizam sistemas de backup e replicação em múltiplos datacenters geograficamente distribuídos, permitindo que os dados sejam acessíveis mesmo em caso de falha em um dos locais.
Clusterização: A infraestrutura utiliza clusters de servidores que distribuem a carga, aumentando o desempenho e diminuindo o risco de indisponibilidade por sobrecarga.

Monitoramento contínuo: Os provedores implementam serviços avançados de monitoramento e tratamento de falhas, garantindo disponibilidade.
Soluções de recuperação de desastres: Oferecem planos para recuperação de dados, reduzindo risco de perda.

3. Que benefícios são esperados com a adoção de serviços em nuvens?

Escalabilidade: É possível ajustar os recursos conforme o que usa, pagando apenas o necessário.
Acesso remoto: Dados e serviços podem ser acessados em qualquer lugar com internet.
Colaboração facilitada: Ferramentas em nuvem podem permitir trabalho em grupo, cada um do seu computador.
Atualizações automáticas: As atualizações de segurança são feitas pelo próprio provedor, não sendo necessário que sejam feitas manualmente pelos usuários.

RPC/RMI

1. Sobre invocação remota de método:

a) Por que o desenvolvedor deve descrever a interface remota usando uma IDL?

A IDL define um contrato entre o cliente e servidor. Ela especifica quais métodos podem ser invocados remotamente, parâmetros, tipos de retorno e exceções.
A IDL descreve as interfaces remotas independentes da linguagens de programação utilizada, sendo assim, ela atua como um intermediário entre as linguagens e plataformas, considerando que em SD há vários componentes desenvolvidos em várias linguagens.
Permite geração automática de código auxiliar, como o stub no lado cliente que representa a interface do servidor de forma local, e o skeleton no lado servidor que recebe as chamadas dos clientes e as delegam para os métodos reais.

b) Qual a relação entre a descrição da interface remota e o stub?

A descrição da interface remota define quais os métodos disponíveis, parâmetros, restrições.
O stub é gerado com base na descrição da interface remota, e quando o cliente chama um método na interface remota, o stub encapsula a chamada em uma mensagem serializada, envia para o servidor através de um middleware, rece resposta do servidor e desserializa retornando o resultado ao cliente.
Sendo assim, o stub implementa a interface, tornando-a acessível ao cliente como se fosse um objeto local. A descrição da interface remota inclui detalhes necessários para o stub se comunicar corretamente com o servidor.

c) Por que o stub tem a sua implementação baseada no padrão de projeto proxy?

Ambos tem a finalidade de atuar como intermediários que controlam o acesso a um objeto;
Abstração do local: O stub permite que o cliente chame o método como se fosse local, sem precisar saber detalhes de onde ele encontra;

Esconde a complexidade da comunicação remota, pois lida com serialização, desserialização, aspectos da rede, conversão de dados entre diferentes sistemas; Gerencia o acesso, redirecionando mensagens para o servidor, lida com exceções em casos de falha.

d) Explique o funcionamento do serviço de nomes e o porquê de seu emprego na invocação remota.

O serviço de nomes gerencia a associação entre nomes simbólicos e os endereços físicos ou lógicos de um objeto. Os clientes podem consultá-lo para localizar e acessar objetos por um nome sem precisar saber os detalhes técnicos como endereço IP ou porta.
Quando um servidor cria um objeto remoto, ele o registra no serviço de nomes com um nome simbólico.
Esse registro associa o nome a informações como endereço, porta, e outros dados para localizar um objeto.
Quando um cliente pretende acessar o objeto, ele consulta o sistema de nomes passando o nome simbólico.
O sistema de nomes então retorna a localização para o stub.
O cliente então pode invocar métodos ao objeto como se fosse local.
O serviço de nomes é importante na invocação remota, pois o cliente não precisa saber os detalhes técnicos do objeto; quando o objeto muda de localização não é necessário fazer alterações no código do cliente, apenas no serviço de nomes;

2. Na construção de uma aplicação com objetos distribuídos:

a) Qual a função do stub e do skeleton? Por que também são conhecidos como proxy?

b) Qual a função do Módulo de Referência Remota?

Gerenciar a interação entre objetos remotos, garantindo que chamadas de métodos em um objeto local sejam direcionadas corretamente para seus correspondentes objetos remotos no servidor. O módulo cria uma referência remota para cada objeto que será acessível remotamente. O módulo coverte chamadas de métodos locais para remotas encapsulando os detalhes necessários como: serialização, identificação do objeto remoto, desserialização.
Controla a criação, uso e destruição de objetos remotos.
Fluxo:
Cliente faz chamada no stub.
Stub passa para o módulo de referência remota
O móduto passa para o servidor correspondente.

c) Qual o papel da referência ao objeto? Explique cada uma das informações que a compõem.

A referência ao objeto é um identificador único que permite localizar e acessar um objeto remoto. Ele abstrai os detalhes da localização do objeto. Informações:
Identificador único do objeto;
Endereço do servidor (IP e porta);
Protocolo usada para comunicar com o objeto (RMI, HTTP/REST, gRPC)

Interface remota que o objeto implementa, informando quais métodos podem ser invocados, seus tipos de argumento e retorno.
Middleware responsável por gerenciar a comunicação (RMI, SOAP, CORBA)
Informações sobre o protocolo para transportar as chamadas, como elas devem ser serializadas e desserializadas.

3. As semânticas de invocação do RPC são principalmente três. Explique quais são e como funciona cada uma delas. Inclua na sua discussão os modelos de falhas que empregam, explicando suas características e a relação destes com as respectivas semânticas.

Ao menos uma vez (At least once)
Garante que a mensagem será processada pelo menos uma vez pelo servidor.
O cliente retransmite caso não tenha resposta em um tempo determinado.
Requer implementação de mecanismo de timeout e retransmissão para lidar com falhas.
Modelo de falhas:
Falha de comunicação: Quando ocorre perda ou atraso o cliente retransmite.
Execução duplicada: Se o servidor processar a mesma requisição mais de uma vez pode haver inconsistência.
Uso típico: Sistemas onde é preferível que a ação ocorra sem se importar com a duplicidade, exemplo: email.

Exatamente uma vez (Exactly once)
A mensagem será processada uma única vez, mesmo se houver falha ou retransmissão.
Implementa identificadores únicos para cada requisição e logs de execução para verificar se a requisição já foi processada.
Modelo de falhas:
Falha de comunicação: Pode ser tratada por retransmissão mas sem executar novamente.
Falha no servidor: O estado de processamento é mantido, e duplicadas são descartadas.
Uso: Sistemas críticos onde duplicidades não podem ocorrer, ex: sistemas bancários.

No máximo uma vez (At most once)
Se houver falha o cliente não retransmite a mensagem.
Não garante que a chamada foi processada, apenas que não será duplicada.
Modelo de falhas:
Falha de comunicação: A chamada pode não ser processada.
Execução duplicada não ocorre porque o cliente não retransmite.
Ex: Sistema de notificação

4. Uma aplicação distribuída desenvolvida com base na distribuição de processos é diferente de uma construída com base na distribuição de objetos? A primeira forma de distribuição tem vantagens sobre a segunda, ou vice-versa? Explique as diferenças e vantagens/desvantagens.

Sim. A distribuição de processos foca na execução de processos independentes em diferentes máquinas. A comunicação é geralmente baseada em troca de mensagens explícita (MPI, sockets). Os processos não compartilham memória diretamente, cada um funciona como uma entidade isolada, cada um com baixa abstração de complexidade.

Na distribuição de objetos a comunicação ocorre por invocação remota de métodos. Os objetos podem encapsular estados e comportamentos oferecendo maior modularidade e abstração. A comunicação é mais transparente para os desenvolvedores que interagem com os objetos remotos como se fossem locais.
Distribuição de processos

- Vantagens:Eficiente em termos de desempenho pois evita overhead de middleware; Escalável para sistemas computacionais intensivos.
- Desvantagens: O desenvolvedor precisa lidar diretamente com detalhes de comunicação; Processos independentes tornam difícil a reutilização de código.

Distribuição de objetos

- Vantagens: Abstração facilita o desenvolvimento.Encapsulamento e modularidade promovem reutilização de código.
- Desvantagens: Pode ter maior latência e overhead devido ao middleware; Escalabilidade limitada por dependência entre objetos e latência de chamadas remotas.

5. Um dos mecanismos de comunicação mais usados em sistemas distribuídos é o de chamada remota de procedimento (RPC).

a) Que vantagens o uso de chamada remota de procedimentos pode oferecer ao programador, se comparado com o uso direto de uma API de mensagens como a de sockets? Há desvantagens? Se sim, quais?

b) Em relação ao RMI, que vantagens este pode oferecer ao programador, se comparado com o uso direto de uma API de mensagens como a de sockets? E quando comparado ao RPC?

   Vantagens do RMI em relação ao RPC: O RPC opera em um paradigma procedural (invocação de funções remotas), o RMI permite invocação remota de métodos em objetos, promovendo maior encapsulamento e reutilização de código; Em RMI, objetos podem manter estado persistente o que facilita a manipulação de dados, no RPC não por ser procedural;

6. Dois padrões arquiteturais importantes para sistemas distribuídos são: “Layers” e “Broker”. Explique como cada um deles funciona, incluindo diagramas, exemplos de aplicação e pseudocódigos.

O padrão layers organiza o sistema em camadas hierárquicas, cada uma com responsabilidade específica. Cada camada depende apenas da camada de baixo e oferece serviços para a camada acima. Apenas camadas adjacentes podem se comunicar diretamente.
Exemplo: Camada de Apresentação (front-end), camada de aplicação (back end) e camada de dados.
Aplicação: E-commerce. Onde a camada de apresentação renderiza a página html; a camada de aplicação gerencia carrinho, compra, pagamento; a camada de dados gerencia estoque de produtos, dados dos clientes.

O padrão brokers é projetado para sistemas distribuídos onde diferentes máquinas desejam se comunicar. Ele utiliza um intermediário (broker) para mediar as interações. O broker localiza, conecta e gerencia a comunicação entre clientes e servidores. O cliente não precisa saber onde o servidor está, pois o broker abstrai a complexidade.
Aplicação: Sistema de mensagem distribuído. Clientes são aplicações que enviam e recebem mensagens. Broker é um servidor central que rodeia mensagens entre os clientes. Servidores armazenam os dados históricos ou processam mensagens.

7. A www pode ser considerada um Sistema Distribuído. Neste contexto considere o serviço de páginas, no qual há dois processos, A e B. Suponha que A é o cliente e B é o servidor e que uma invocação consiste de uma mensagem de requisição de A para B, seguida do processamento da requisição feito pelo B, seguido ainda por uma mensagem de resposta de B para A. Explique, com base neste cenário, as classes de falhas que podem ser apresentadas por uma invocação.

8. O modelo de objetos distribuídos estende o modelo de objetos básico em, principalmente, quatro pontos: localidade dos objetos, chamadas de métodos, referência, interface. Descreva detalhadamente estes quatro pontos.

Uma referência deve encapsular informações adicionais para localizar um objeto.

Pub/Sub

1. Descreva o projeto de uma arquitetura para notificação de eventos distribuídos baseada em pub/sub, ECA e RMI. Na sua resposta, apresente diagramas arquiteturais e pseudocódigos.

O publisher publica evento para o broker. O broker irá distribuir para os subscribers interessados. O ECA definirá a lógica da ação a ser tomada a partir daquele evento. O subscriber faz uma invocação a um método remoto por meio do RMI.

2. A indireção é um conceito fundamental em ciência da computação, e, em termos de sistemas distribuídos, é muito aplicado nos paradigmas de comunicação. Explique as possibilidades de implementação de indireção em sistemas distribuídos e as duas propriedades decorrentes do seu uso.

Intermediário broker: Um publisher publica ao broker que fica responsável por entregar aos subscribers sem que seja necessário o publisher conhecer os subscribers.
Serviço de nomes: Quando uma entidade deseja comunicar com outra ela consulta o serviço de nomes, onde estará todas as informações de localização.
Middleware: O middleware oferece abstração para comunicação, ocultando detalhes de transporte. A entidade pode interagir como outra como se fosse local.
Filas de mensagem: Um produtor publica em uma fila, e o consumidor as consome quando disponível.
Pub/Sub: O publisher publica em um intermediário, e os subscribers consomem quando disponível.
Duas propriedades decorrentes:
Transparência: A indireção pode ocultar detalhes sobre a localização ou implementação.

Desacoplamento: As entidades podem interagir sem depender diretamente umas das outras.

3. Se um modelo de comunicação é assíncrono, ele é também desacoplado no tempo (timeuncoupled)? Explique com exemplos.

Não. Há duas possibilidades:
Assíncrono e desacoplado no tempo: O emissor irá enviar mensagens independentemente se o receptor está ativo. E o receptor pode processar a qualquer momento, até um bom tempo depois que o emissor envia. Exemplo: e-mail
Assíncrono e acoplado no tempo: O emissor envia independentemente, mas o receptor deve responder em um limite de tempo. Exemplo: Um sensor que envia mensagens a um servidor, ele enviará um fluxo de dados independentemente da resposta imediata, mas por questões de segurança o servidor não pode demorar a responder.
Também temos:
Síncrona e acoplada no tempo: O emissor envia mensagem e bloqueia para receber a resposta, o remetente deve responder imediatamente. Ex: chamada de um método local, a execução só continua depois que o método for respondido.
Síncrona e desacoplada no tempo: O emissor bloqueia enquanto não recebe a resposta, mas o receptor pode responder a qualquer momento. Ex: Em sistemas bancários quando o cliente solicita uma transferência, e fica bloqueado até o momento que o banco autorizar.

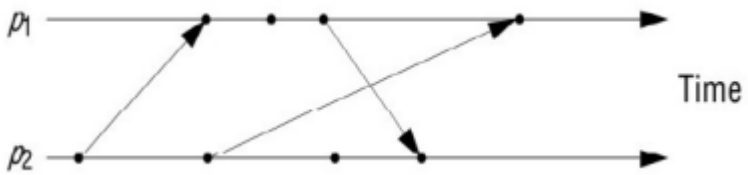
Algoritmos

1. O algoritmo de exclusão mútua distribuída Multicast emprega relógios lógicos na sua solução. Explique a inter-relação entre a ordenação total de mensagens enviadas por multicast e os fundamentos de relógios lógicos.

2. O algoritmo Subscription Flooding para sistemas pub/sub confia na estruturação de tabelas persistidas nos brokers da rede. Descreva o funcionamento do algoritmo e discuta como as tabelas e brokers podem ser estruturados em termos de implementação.

5. Em que, essencialmente, o algoritmo de exclusão mútua de Maekawa difere do algoritmo Usando Multicast e Relógios Lógicos? Explique.

7. A figura a seguir mostra eventos que ocorrem em dois processos p1 e p2. As setas entre os processos denotam a transmissão de mensagem. Identifique a malha de estados consistentes (p1, p2), começando com o estado inicial (0,0).



10. No algoritmo do servidor central para exclusão mútua, descreva uma situação na qual dois pedidos não são processados na ordem hapenned-before.