Resolução de Perguntas Teóricas

1. Vantagens e desvantagens relativas à utilização de formatos de texto (por oposição a binários) para a serialização de dados em sistemas distribuídos.

Em relação às vantagens:

- Human-readable: formatos de texto são mais fáceis de entender, mais legíveis para o ser humano, logo, permitem um debug mais fácil.
- É mais genérico: pode ser facilmente lido por software em diversos sistemas operativos.
- É robusto: uma pequena alteração no conteúdo de dados não invalida totalmente a sua descodificação, porque o erro é facilmente detetado e corrigido. Em relação às desvantagens:
- Socorre-se de parsing.
- Menos compacto: gera um tamanho maior de dados, porque é mais complexo.
- Menos seguro: existe falta de encriptação de dados, pois todos o entendem.
- Redundante: levando a uma conversão mais lenta, é mais difícil de descodificar.

Nos formatos binários

Em relação às vantagens:

- Compactos: o que faz com que sejam mais eficientes, mesmo em comunicação, não se perde espaço na representação dos dados.
- Opacos: têm um formato que não permite que qualquer utilizador entenda.
- Difícil Debug: devido a ser opaco.
- Mais Seguro: devido a ser opaco e de mais difícil compreensão.
- Padding: leva a uma maior eficiência, havendo alinhamento de endereços na memória. Em relação às desvantagens:
- Menos genérico: e, por isso, de mais difícil compreensão, durante o debug, pois não é facil entender o formato binário.
- Frágeis: uma pequena alteração no seu conteúdo pode invalidar toda a descodificação da mensagem.

2. Defina transparência de acesso e explique em que medida é que a invocação remota (RPC) contribui para a obter.

Transparência de acesso é aquilo que permite que se oculte do usuário/programador o conjunto de recursos a que o sistema pode ter acesso, sendo também ocultadas as diferênças na representação de dados. O usuário não deve saber se um recurso acedido é local ou remoto. A transparência de acesso faz com que o sistema não tenha que fornecer a localização dos recursos, ou seja, os programas devem executar os processos de leitura e escrita de arquivos remotos da mesma maneira que operam sobre os arquivos locais, sem qualquer modificação no programa. É assim que o RPC ajuda a cumprir a transparência, porque não só encapsula as rotinas de acesso e consulta como efetua o controlo de concorrência do sistema distribuído.

3. Identifique a principal dificuldade criada pela escala geográfica a aplicações cliente/servidor interativas e explique uma forma de a resolver.

Num sistema onde os recursos e os utilizadores se encontram afastados, a escala geográfica faz com que a distância não cause delays significativos na cimunicação entre ambos, isto é, independentemente da localização física, a comunicação não é gravemente afetada. Uma das formas de resolver este delay e a replicação dos dados, ou a criação de caches nos locais mais próximos dos recursos/utilizadores. Assim, os dados passam a encontrar-se mais próximos, os problemas relativos à latência são diminuidos. Isto segue claramente tem por base que quanto mais próximo está programa do nível de hardware, obtido por exemplo por migração, mais eficiente é a comunicação cliente-servidor.

Em relação à escala numérica

Num sistema, mesmo que a quantidade de clientes/pedidos aumente, a performance não diminui, ou seja, se um sistema se diz escalado numericamente, é possível inserir utilizadores e recursos sem diminuição significativa de performance. Uma técnica para o obter é aumentar a quantidade de hardware/servidores que possam corresponder a uma maior quantidade de utilizadores. Esta técnica, também conhecida como Fog computing (conjugando cloud computing com edge computing), torna os sistemas distribuídos mais eficazes.

4. Distinga comunicação síncrona de assíncrona em sistemas distribuídos. Dê exemplos de middleware para cada uma delas.

A diferença entre os tipos de comunicação apresentados reside no tipo de ações bloqueantes na interação cliente-servidor. No caso da comunicação síncrona, para haver comunicação, o cliente e o servidor têm de estar sincronizados, ou seja, o envio e a receção de mensagens são ações bloqueantes: quando um cliente emite uma mensagem, este fica bloqueado até que o servidor lhe responda. Do mesmo modo, enquanto o servidor não receber a mensagem também fica bloqueado.

- Exemplos de middleware: Message passing em MOM (middleware orientado a mensagens); No caso da comunicação assíncrona, para haver comunicação, já não é necessária esta comunicação restrita, ou seja, o envio não é uma ação bloqueante, pelo que o cliente pode enviar uma mensagem, sem ficar bloqueado até que o servidor lhe responda, continuando a sua execução logo após o envio da mesma. Não é preciso que o envio seja bloqueante, porque a mensagem é guardada num buffer/queue de mensagens. A transmissão destas mensagens ocorre em paralelo com a execução do emissor. Quanto à receção de mensagens, esta ação pode ou não ser bloqueante.
- Exemplos de middleware: Message Queueing em MOM.

5. Qual a razão para estruturar uma aplicação distribuídas em camadas? Use um exemplo.

O modelo em 3 camadas é o quando o cliente/servidor são desenvolvidos retirando-se a camada de negócio do lado do cliente. O desenvolvimento é mais demorado no início em relação ao modelo bicamadas pois é necessário dar suporte a mais plataformas e ambientes diferentes. Por outro lado, as respostas são mais rápidas nas requesições, havendo maior controlo sobre o crescimento do sistema. As suas 3 camadas são: apresentação (interface), negócio e dados. Alterações numa camada não afetam outra camada, sendo que a separação em camadas lógicas torna estes sistemas mais flexíveis, reduzindo a dependência entre elementos dentro de uma mesma camada (classes e packages). Este modelo tornou-se a arquitetura padrão em sistemas corporativos web.

6. Explique como funciona um protocolo de exclusão mútua distribuída centralizado. Identifique as principais vantagens e desvantagens.

Este protocolo assume que o sistema é um conjunto de processos em que cada processo está no seu processador. Cada processo tem uma zona crítica que requer exclusão mútua. O principal requisito é o facto de que se um processo se encontra a executar na zona crítica, então, mais nenhum processo está a executar dentro da sua zona crítica. Qualquer processo que queira executar a sua zona critica tem de enviar o seu pedido ao processo coordenador e este que decide se o devido processo pode ou não entrar e enviar uma resposta. Assim que o processo recebe resposta, inicia a execução critica. Quando termina, este processo envia uma mensagem ao coordenador para libertar a zona. Em relação às desvantagens:

- Dependência: se o servidor falhar, o sistema cai;
- Bottleneck: num grande sistema, um servidor pode tornar uma performance bottleneck;
- Tempo para entrar na secção critica é maior ou igual que o RTT;
- há um processo que lida com pelo menos 4 mensagens (lock, reply, unlock, reply); Em relação às vantagens:
- Algoritmo justo: Processos executam a secção critica por ordem de chegada;
- Exclusão mútua simultanea: só um processo entra na zona critica simultaneamente;
- Simples e fácil de implementar;
- Requer apenas 3 mensagens por uso de recurso.

7. Identifique uma aplicação e descreva sucintamente o funcionamento de um relógio de Lamport num sistema distribuído.

Para sincronizar relógios, Lamport definiu uma relação Happened-Before (->), onde:

- Se A e B são eventos do mesmo processo e A foi executado antes de B, então, A->B.
- Se A é o evento de envio de uma mensagem por um processo e B é o evento de receção dessa mensagem por outro processo, então, A->B.
- Se A->B e B->C, então A->C. Lamport associa também uma tag temporal a cada evento de forma a que se A->B, então a tag de A é menor que a tag de B, ou seja:
- Cada processo tem um relógio lógico associado, onde este relógio é um contador que é incrementado entre cada dois eventos sucessivos do processo;
- Cada mensagem enviada transporta o instante lógico em que foi enviada;
- Ao receber uma mensagem, o processo acerta o seu relógio com o instante da mensagem se o último for mais recente; Um exemplo da aplicação é numa Base de Dados replicada em várias cidades.

8. Explique uma forma de mitigar a incerteza quanto ao tempo de transmissão de mensagens para conseguir sincronizar relógios em sistemas distribuídos.

Uma forma de mitigar esta incerteza é através do protocolo de relógios de lamport. Como referido anteriormente, cada processo tem o seu contador C(i), onde este contador é atualizado de acordo com as etapas seguintes:

- Antes de executar um evento, A incrementa C(i) <- C(i) + 1;
- Quando o processo A envia uma mensagem M para um processo P(j), define um timestamp ts(m) em M igual a C(i) após ter executado a última ação;
- Na receção de uma mensagem M, o processo P(j) ajusta o seu contador local para C(j) - max{C(j).ts(m)}, e após executar a primeira etapa, envia a mensagem para a aplicação. Isto garante que sabemos tudo o que aconteceu antes de uma certa mensagem ter sido enviada, pelo que temos a certeza que é a próxima a adicionar à queue.

9. Qual a relevância do sistema operativo na resolução do problema de exclusão mútua no modelo de memória partilhada e no modelo de passagem de mensagens?

A função do sistema operativo tem por base uma eficiente gestão dos recursos. O SO é responsável por bloquear processos, prevenindo-os de consumir tempo de CPU, enquanto não tiverem permissão para entrar na secção crítica.

- No modelo de memória partilhada: os processos são bloqueados ao tentar obter o lock.
- No modelo de passagem de mensagens: os processos são bloqueados desde do momento em que enviam os pedidos até receberem a resposta. Em ambos os casos, evitam-se esperas ativas. O que distingue a ação do SO nos dois modelos é o momento em que os processos são bloqueados e libertados.

10. Distinga em termos de objetivos e forma de utilização as primitivas lock/unlock e wait/notify.

O objetivo das primitivas lock/unlock é garantir que apenas um processo se encontra na zona limitada por estas primitivas (secção critica), já o das primitivas wait/notify é que os processos voluntariamente adormeçam ou suspendam enquanto o predicado de uma variável de condição não se verifique, sendo acordados por outros processos quando estes alteram algo ou existe possibilidade de o predicado ser válido. A forma de utilizar lock/unlock é envolvida com um try/finally que garante que, primeiro se adquira o lock, e só no final da execução do código da região crítica se liberte o lock com unlock(), garantindo que o lock é efetivamente libertado. A forma de utilizar o wait/notify está normalmente relacionada com ter um lock para verificar um predicado e caso aconteça, liberte o lock, suspenda e, quando acorde adquira o lock. Quando "passa" o predicado liberta o lock.

11. Das arquiteturas estudadas de sistemas distribuidos, quais se adequariam melhor a um sistema de suporte a redes sociais?

As redes sociais são baseadas em eventos que podem gerar reações em determinados componentes. É neste contexto que foram criadas as arquteturas baseadas em eventos e que se adequam melhor a este tipo de sistema. Esta arquitetura é constituida por um grande repositório de eventos/dados, que são homogéneos, orientados a um aplicação em tempo real. Há um barramento de eventos e componentes que de forma assíncrona agem perante estes. Este comportamente é facilmente identificado numa rede social atual. Os processos publicam então os eventos e o middleware garante que os processos recebem os eventos em que se encontram subscritos.

12. Diga o que entende por middleware orientado a mensagens, identificando os principais componentes.

Esse é um método de comunicação assíncrona e persistente baseado em trocas de mensagens entre a componente de software num sistema distribuido. Cada componente tem duas filas de mensagens que armazenam as mensagens a enviar e recebidas. A mensagens a enviar são enviadas para um gestor de mensagens ("Message broker"), que gere várias filas de mensagens e realiza as trocas entre as filas. A transmissão das mensagens ocorre por canais MCA's, que empacotam/desempacotam as mensagens a receber/enviar pelo canal de rede.

13. Descreva as funcionalidades de um servidor de objetos distribuidos.

A principal funcionalidade deste servidor é gerir um conjunto de objetos e intermediar a realização de pedidos aos objetos. Este é responsável por:

- Verificar se certo pedido se destina a um dos objetos que possui;
- Assegurar que tal objeto se encontre carregado;
- Realizar o pedido ao objeto;
- Retomar a resposta ao ciente;
- Remover da cache os objetos que já não são utilizados há muito tempo.

15. Descreva o algoritmo distribuido de exlcusão mútua em anel.

O algoritmo em anel é aplicado em sistemas organizados fisica ou logicamente em anel. Este algoritmo assume que os canais sao unidirecionais. Cada processo mantém uma lista de ativos que consiste nas prioridades de todos os processos ativos quando este algoritmo terminar. Quando um processo A suspeita a falha do coordenador, A cria uma lista de ativos vazia, envia uma mensagem de eleição M(i) ao seu vizinho e insere i na lista de ativos. Se A receber uma mensagem de eleição M(j) pode responder de 3 formas:

- Se foi a primeira mensagem de eleição que recebeu, então cria uma lista de ativos com i e j. Envia as mensagens de M(i) e M(j) (por esta ordem) ao vizinho.
- Se i!=j entao junta j à sua lista de ativos e reenvia a mensagem ao seu vizinho.
- Se i==j então a sua lista de ativos já contém todos os processos ativos no sistema e A pode determinar o coordenador.

16. Indique sucintamente as arquiteturas descentralizadas estudadas.

Existem arquiteturas estruturadas e não estruturadas:

- Estruturadas: têm uma organização bem definida, onde todo o domínio dos dados é dividido pelo contradomínio dos constituintes da arquitetura através de uma função de hash (DHT em p2p).
- Não Estruturadas: não tem organização bem definida,são um conjunto de nós dispersos, cada um com uma determinada visão parcial da rede. Têm em conta aleatoriedade, devido aos modos das vistas de qualquer nó constituinte (p2p por métodos aleatórios).

17. Discuta as diferenças ao assegurar exclusão mútua entre processos numa máquina e num sistema distribuido.

A forma de assegurar a exclusão mútua é diferente para cada situação. Numa única máquina é alcançada através das primitivas de sincronização lock/unlock disponibilizados pelo SO, permitindo o bloqueio de processos enquanto não conseguem entrar na secção critica. Em sistemas distribuidos, centralizados ou descentralizados, obtém-se através da troca de mensagens. O cliente envia uma mensagem a pedir permissão para "escutar" a região e só a pode executar se o coordenador (nos centralizados), ou todos os outros constituintes (nos descentralizados), enviarem uma mensagem de resposta com essa permissão.

18. "O monitor é uma primitiva estruturada de controlo de concorrência de alto nível". Comente a afirmação e indique uma vantagem e desvantagem deste elevado nível de abstração.

A afirmação é verdadeira. Um monitor é uma primitiva estruturada de controlo de concorrência, oferencendo um tipo de dados com controlo de concorrência implícito em todas as operações de exclusão

mútua. Com um modelo de concorrência baseado em monitores, o compilador pode inserir mecanismo de exclusão mútua transparente, isto é, sem que o programador tenha de aceder às primitivas de controlo e realizar o bloqueio/libertação de recursos manuais. Em java, esta utilização de monitores baseia-se através de métodos synchronized que fazem uso de ReentrantLocks.

- Vantagens: Controlo de concorrência implícita (transparente);
- Desvantagens: Os locks são reentrantes e podem levar à starvation.

19. Caracterize as funções do cliente e do servidor, num ambiente cliente-servidor.

Cliente:

- Interface com o utilizador;
- Caching;

Servidor:

- Grande capacidade de armazenamento ou processamento;
- Periféricos especiais (e.g. impressão, salvaguarda);
- Periféricos particulares (e.g. pesquisa, autenticação);
- Servidores vulgares: sistema de ficheiros, base de dados, informação de redes, sistema de nomes, email, etc.

20. Caracterize as vantagens e inconvenientes da conexão TCP e UDP.

De modo resumido, deixa-se a seguinte tabela que tem as principais diferenças entre os dois protoclos de conexão:

Protocolo/Funções	**UDP**	**TCP**
Velocidade	Rápido (mais eficiente)	Lento (menos eficiente)
Controlo de fluxo	Não	Sim
Controlo de erros	Não	Sim
Controlo de congestão	Não	Sim
Deteção de erros	Não	Sim
Orientado à conexão	Não (não fiável)	Sim (fiável)

De um modo geral, o protocolo TCP é mais complexo que o UDP.

21. Descreva com detalhe o fluxo de operações e dados na realização de um invocação remota RPC entre cliente e servidor. Use a invocação local da função "int somar(inta a, int b)".

Seguem-se os seguintes passos detalhados em RPC.

1. O processo Cliente chama o Client Stub de modo normal (Call somar(a,b)).
2. O Client Stub constroi a mensagem e chama o Sistema Operativo Local.

3. O Sistema Operativo local envia a mensagem ao Sistema Operativo Remoto.
4. O Sistema Operativo Remoto dá a mensagem ao Server Stub.
5. O Server Stub desempacota os parâmetros, chamando o Server.
6. O Server faz o trabalho, retornando o resultado ao Stub.
7. O Server Stub empacota estes parâmetros numa mensagem, chamando o Sistema Operativo Local.
8. O Sistema Operativo do Servidor envia a mensagem ao Sistema Operativo do Cliente.
9. O Sistema Operativo do Cliente dá a mensagem ao seu Stub.
10. O Stub do Cliente desempacota o resultado e retorna-o ao Cliente (Return (somar)).

(Contém todas as perguntas desde a existência da cadeira. Note-se que algumas constumam sair repetidas nos testes, Good Luck ❤️).