Perguntas testes

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

Vantagens:

Mais fácil de entender, mais legível e, portanto, permite um debug mais fácil. É mais genérico. É robusta na medida em que uma pequena alteração no conteúdo de dados não invalidade totalmente a sua descodificação.

Desvantagens:

É necessário fazer parsing, têm um tamanho maior e a falta de encriptação de dados leva a uma menor segurança de dados. Possui alguma redundância, sendo mais lenta a descodificar.

Extra:

Formatos Binários:

Vantagens:

São mais compactos e, por isso, mais eficientes;

Desvantagens:

São de mais difícil utilização durante o debug porque não é fácil entender diretamente o formato binário. Podem ser mais frágeis no sentido em que a pequena alteração de um byte do seu conteúdo poderá invalidar toda a mensagem (ex. Java Data*Stream streams). Preocupação em alinhar o endereçamento em memória, uma vez que acessos à memoria que não estejam alinhados são substancialmente mais lentos.

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

Transparência de acesso faz com que se oculte ao usuário/programador quais os recursos que o sistema pode ter acesso. Para além disso, são também ocultadas as diferenças de representação de dados. O usuário não deve saber se o 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 desta maneira que o RPC ajuda a cumprir a transparência de acesso, pois tanto 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.

Dentro de um sistema onde os recursos e os utilizadores se encontrem afastados, a escala geográfica faz com que a distância não cause delays significativos na comunicação entre ambos. Ou seja, independentemente da localização física, a comunicação não é gravemente afetada.

Uma das técnicas para a resolução é a replicação de dados, ou a criação de caches para os locais mais próximos dos recursos/utilizadores. Como os dados passam a encontrar-se mais próximos, os problemas relativos à latência são diminuídos.

Extra:

Escala numérica: Dentro de um sistema, mesmo que a quantidade de clientes/pedidos aumente, a performance não diminui. Isto é, se um sistema se diz escalado numericamente, é possível inserir utilizadores e recursos sem a diminuição significativa de performance. Uma técnica para atingir este tipo de escalabilidade é aumentar a quantidade de hardware/servidores para poderem corresponder a uma maior quantidade de utilizadores.

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

Síncrona: Para haver comunicação, o cliente e o servidor têm de estar sincronizados. O envio e a receção de mensagens são ações bloqueantes, isto é, quando um cliente emite uma mensagem, este fica bloqueado até obter uma resposta e enquanto o servidor não receber a mensagem, também fica bloqueado.

Assíncrona: Neste caso, o envio não é uma ação bloqueante, ou seja, o cliente pode enviar uma mensagem e continuar a sua execução logo após o envio da mesma. Isto acontece porque a mensagem é copiada para um buffer/quele de mensagens. A transmissão destas mensagens ocorre em paralelo com a execução do emissor. No caso da receção de mensagens, a receção tanto pode ser uma ação bloqueante ou não.

Exemplos middleware para síncrona: Message passing em MOM (middleware orientado a mensagens);
Exemplos middleware para assíncrona: Message Queuing em MOM.

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

O modelo em 3 camadas, derivado do modelo 'n' camadas, recebe esta denominação quando um sistema cliente/servidor é desenvolvido retirando-se a camada de negócio do lado do cliente. O desenvolvimento é mais demorado no início quando comparado ao modelo de 2 camadas porque é necessário dar suporte a uma maior quantidade de plataformas e ambientes diferentes. Em contrapartida, o retorno vem em forma de respostas mais rápidas nas requisições, tanto em sistemas que rodam na internet ou em intranet, e há uma maior controlo no crescimento do sistema.
As três partes de um modelo em 3 camadas são: apresentação (interface), negócio e dados.
Característica:

O software executado em cada camada pode ser substituído sem prejuízo para o sistema;

Atualizações e correções de defeitos podem ser feitas em prejudicar as demais camadas.
Por exemplo, alterações da interface podem ser realizadas sem comprometer as informações contidas na camada dos dados. A separação em camadas lógicas torna os sistemas mais flexíveis, permitindo que as partes possam ser alteradas de forma independente. As funcionalidades da camada de negócio podem ser divididas em classes e essas classes podem ser agrupadas em packages, reduzindo as dependências entre as classes e packages;

O modelo de 3 camadas tornou-se a arquitetura padrão para sistemas corporativos com base na web.

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

O protocolo de exclusão mútua tem base na assunção de que o sistema consiste de n 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 se encontra em execução dentro da sua zona crítica. Qualquer processo que queira executar a sua zona critica tem de enviar um pedido ao processo coordenador, sendo este quem decide se o processo pode ou não entrar na zona crítica e enviar uma resposta. Quando o processo recebe a resposta do coordenador, inicia a execução da zona crítica. Quando este acaba a execução, o processo envia uma mensagem ao coordenador para libertar a zona.

Desvantagens:

Se o servidor falhar, o sistema cai;

Num sistema de maiores dimensões, um servidor pode-se tornar uma performance bottleneck;

Vantagens:

Algoritmo justo. Os processos executam a zona critica por forma de chegada;

Garante que apenas 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 lógicos, Lamport definiu uma relação Happened-Before (->).

- 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 o evento de receção dessa mensagem por outro processo, então A -> B.
- Se A -> B e B -> C, então A -> C.

Para a realização destas relações, é associado uma etiqueta temporal a cada evento do sistema, de forma que se A -> B então a etiqueta de A é menor que a etiqueta de B, isto é:

- Cada processo tem um relógio lógico associado. O 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 este último for mais recente.

Um exemplo de aplicação dos relógios de Lamport é uma 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**.

Um método é através do protocolo dos relógios de Lamport.

Para implementar o algoritmo dos relógios de Lamport, cada processo "A" mantém um contador C(i). Estes contadores são atualizados conforme as seguintes etapas:

- Antes de executar um evento (enviar uma mensagem para uma rede, entregar uma mensagem a uma aplicação ou qualquer outro evento interno), "A" incrementa C(i) <- C(i) +=1;
- Quando o processo "A" envia uma mensagem "M" para uma 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 aplicação.

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 na resolução do problema de exclusão mútua tem por base uma eficiente gestão dos recursos. Este é responsável por bloquear processos, prevenindo-os de consumir tempo de CPU, enquanto não tiverem permissão para avanção para a região crítica.

A ação do sistema operativo neste tipo de problemas no modelo de memória partilhada e no modelo de passagem de mensagens apenas interfere no momento em que os processos são bloqueados e libertados.

Em memória partilhada, tal acontece quando se tenta obter o lock. No caso de passagem de mensagens, os processos são bloqueados desde que enviam os pedidos até à receção de resposta.

Em ambos os casos evitam-se esperas ativas.