1. \*\*Podría ser considerat sistema distribuit a un conjunt d'ordinadors sense cap tipus de conexió, directa o indirecta? Justifica aquesta resposta.\*\*

- No, un sistema distribuido implica que hay varios ordenadores (o nodos) que trabajan juntos para lograr un objetivo común, y esto requiere algún tipo de conexión, ya sea directa o indirecta, entre ellos para comunicarse y coordinar acciones.

2. \*\*Explica la diferencia entre els models de consistència data centric i client centric.\*\*

- Modelos \*\*Data Centric\*\*: Están centrados en cómo y cuándo se ve reflejado un cambio o actualización en un dato en todo el sistema. Estos modelos garantizan que todos los nodos vean las actualizaciones en el mismo orden. Ejemplos incluyen consistencia secuencial y consistencia causal.

- Modelos \*\*Client Centric\*\*: Se centran en la vista de un cliente individual respecto a los datos y cómo y cuándo se ven reflejados los cambios para ese cliente. Un ejemplo es la consistencia eventual, donde se garantiza que si no hay más actualizaciones, eventualmente todos los nodos tendrán la misma versión de los datos.

3. \*\*Descriu com es fa l'ajust d'un rellotge (avanc / retard) durant la sincronització de rellotges físics.\*\*

- Durante la sincronización de relojes en sistemas distribuidos, es esencial que todos los relojes muestren un tiempo similar para mantener la consistencia. El algoritmo de Berkeley es un método común utilizado. En él, un nodo maestro recopila la hora de todos los otros nodos, calcula el tiempo promedio y luego informa a cada nodo cuánto debe avanzar o retrasar su reloj para sincronizarse con el tiempo promedio.

4. \*\*En una RPC, com definiríes una crida orfena? Quins problemes representa que hi hagi crides orfenes?\*\*

- Una llamada RPC orfana ocurre cuando el cliente hace una solicitud al servidor y luego falla o se desconecta antes de que el servidor pueda responder. Esto puede resultar en operaciones que el servidor sigue procesando sin que el cliente sepa el resultado, lo que puede causar inconsistencias y desperdicio de recursos.

5. \*\*En el protocolo de consistència primary-based remote-write,...\*\*

- En este protocolo, si un nodo (que actúa como réplica) modifica un dato, primero debe informar y obtener el permiso del nodo principal (primary). Si posteriormente otro nodo lee ese dato, el valor leído será el que el nodo principal tenga, ya que es el encargado de mantener la consistencia.

6. \*\*Quina millora aporta l'algorisme "Enhanced ring algorithm"...\*\*

- El algoritmo "Enhanced ring algorithm" mejora el proceso de elección de líder en sistemas distribuidos en comparación con el algoritmo "Chang & Roberts ring algorithm" al reducir el número de mensajes necesarios y optimizar el proceso de elección en caso de fallos.

7. \*\*Justifica per què la comunicació “Scalable Reliable Multicast” permet més escalabilitat respecte a multicast bàsic.\*\*

- El "Scalable Reliable Multicast" utiliza técnicas avanzadas como la fragmentación de paquetes y la retransmisión selectiva para garantizar la entrega de mensajes. Esto le permite manejar una mayor cantidad de nodos y tráfico que el multicast básico, ofreciendo una escalabilidad y fiabilidad superiores.

8. \*\*Quins avantatges i inconvenients hi hauria en la gestió de la comunicació orientada a stream, si es disposa d'un buffer molt petit?\*\*

- \*\*Ventajas\*\*: El uso de un buffer pequeño puede reducir la latencia ya que los datos se transmiten más rápidamente al no esperar que el buffer se llene.

- \*\*Inconvenientes\*\*: Puede haber más overhead debido a la necesidad de gestionar más paquetes pequeños. También existe un mayor riesgo de pérdida de datos si el buffer se llena y hay problemas de rendimiento o congestión en la red.

9. \*\*¿Por qué se busca el proceso con el ID más grande en un algoritmo de elección?\*\*

En muchos algoritmos distribuidos, el proceso con el ID más grande puede ser elegido como líder porque garantiza una elección determinista y sin ambigüedades. El uso de un identificador único, como el ID, evita conflictos y facilita la elección de un líder en caso de fallos.

10. \*\*Define 'sequential consistency' e identifica las principales diferencias respecto a 'strong consistency'.\*\*

La consistencia secuencial es un modelo de memoria que garantiza que las operaciones de todos los procesos se vean en un orden que respete el orden en el que esas operaciones fueron emitidas por cada proceso. Por otro lado, la consistencia fuerte (strong consistency) garantiza que una operación en un dato será vista por todas las operaciones subsiguientes en ese dato, independientemente del proceso que las emita. La principal diferencia es que la consistencia fuerte es más restrictiva y garantiza una visión unificada del sistema en todo momento, mientras que la consistencia secuencial solo garantiza un orden global coherente.

11. \*\*En una RPC, ¿cómo funciona el paso de parámetros de una matriz? Explica qué opciones de mejora podemos tener.\*\*

En RPC, el paso de parámetros de una matriz generalmente se realiza por copia (marshalling). Esto significa que la matriz se serializa en el lado del cliente y luego se deserializa en el lado del servidor. Las optimizaciones pueden incluir la compresión de datos, la transmisión incremental o el uso de punteros en lugar de copias para matrices grandes si el cliente y el servidor comparten la misma memoria.

12. \*\*Define 'strict consistency' y argumenta por qué no es posible asumir este modelo.\*\*

La consistencia estricta requiere que cualquier lectura en un dato devuelva el valor escrito por la escritura más reciente. No es práctico en sistemas distribuidos debido a la latencia de la red y a los relojes no sincronizados, lo que hace imposible determinar cuál es la "escritura más reciente" en todo momento.

13. \*\*¿Cuál es la diferencia entre un reloj físico y un reloj lógico?\*\*

Un reloj físico mide el tiempo real y se basa en un oscilador o en una fuente de tiempo atómico. Un reloj lógico, en sistemas distribuidos, es una abstracción que representa la causalidad entre eventos, a menudo implementada usando contadores.

14. \*\*En una RMI, ¿dónde se encuentra el 'objeto' sobre el que se ejecuta la RMI, en el cliente o en el servidor?\*\*

En la Invocación de Método Remoto (RMI), el objeto sobre el cual se invoca el método reside en el servidor. El cliente realiza una llamada al objeto remoto como si fuera local, pero la ejecución realmente ocurre en el servidor.

15. \*\*En el algoritmo de Ricart & Agrawala para gestionar una exclusión mutua, teniendo en cuenta que con la conformidad de todos los procesos, ¿cómo se puede garantizar que no habrá deadlock?\*\*

El algoritmo de Ricart & Agrawala evita el deadlock al asegurar que sólo el proceso con el menor timestamp (o en caso de empate, el proceso con el menor ID) puede acceder al recurso. Todos los demás procesos que deseen acceder al recurso deben esperar.

16. \*\*El paso de mensajes es el mecanismo básico (core) que permite a un sistema distribuido operar. ¿Por qué no se utilizan entonces mensajes habitualmente como mecanismo de comunicación entre los procesos del sistema?\*\*

Aunque el paso de mensajes es fundamental para la comunicación en sistemas distribuidos, no se utiliza siempre directamente debido a su complejidad y bajo nivel. En su lugar, se prefieren abstracciones más altas, como RPC o RMI, que proporcionan una interfaz más amigable y manejan automáticamente muchos de los detalles subyacentes del paso de mensajes.

---------------------------------------------------------------------------------------------------------------------------

1. \*\*¿Cuáles son las 3 dificultades más críticas, desde tu punto de vista, para tener un SD transparente?\*\*

- \*\*Localización transparente\*\*: La capacidad de acceder a los recursos sin saber su ubicación física.

- \*\*Migración transparente\*\*: La capacidad de que los recursos se muevan sin afectar la forma en que se accede o se usa.

- \*\*Concurrencia transparente\*\*: Permitir que múltiples usuarios accedan a recursos simultáneamente sin conflictos.

2. \*\*Define "Obertura (Openness)" en un sistema distribuido.\*\*

- La obertura en un sistema distribuido se refiere a la capacidad del sistema para ser flexible, escalable y adaptable. Esto implica que el sistema puede extenderse fácilmente (en términos de agregar más nodos o servicios), interoperar con otros sistemas y adaptarse a los cambios en el entorno.

3. \*\*¿Qué es "Middleware" en el contexto de los sistemas distribuidos?\*\*

- El middleware es un software que actúa como intermediario entre las aplicaciones y el sistema operativo en sistemas distribuidos. Facilita la comunicación y gestión de datos en sistemas distribuidos, ofreciendo servicios como autenticación, autorización, descubrimiento de servicios, y más.

4. \*\*El paso de mensajes es el mecanismo básico que permite a un sistema ser distribuido. ¿Por qué no se usan mensajes habitualmente como mecanismo de comunicación entre los nodos del sistema?\*\*

- El paso de mensajes puede introducir latencia debido al tiempo que toma enviar, recibir y procesar mensajes.

- Puede complicar la programación, ya que los desarrolladores necesitan manejar la asincronía, errores en la entrega de mensajes, entre otros.

- En algunos casos, otros métodos de comunicación, como la memoria compartida, pueden ser más eficientes.

5. \*\*¿Podría ser considerado un sistema distribuido un conjunto de ordenadores sin ningún tipo de comunicación, directa o indirecta?\*\*

- No. La esencia de un sistema distribuido es la capacidad de los nodos para comunicarse y trabajar juntos hacia un objetivo común. Sin comunicación, serían simplemente sistemas independientes.

6. \*\*En una RPC... ¿Cómo funciona el mecanismo de paso de parámetros por referencia (siempre que el sistema lo permita)?\*\*

- En una RPC, el paso de parámetros por referencia generalmente se realiza enviando una dirección de memoria o un identificador único del dato en lugar del dato en sí. Sin embargo, esto puede ser complicado ya que las direcciones de memoria no tienen sentido en diferentes máquinas. En su lugar, se pueden enviar copias de los datos y luego sincronizar cualquier cambio con el original.

7. \*\*En una RPC... ¿Qué problemas podemos encontrar en el mecanismo de paso de parámetros?\*\*

- Problemas de serialización y deserialización de datos.

- Dificultades con el paso por referencia en máquinas diferentes.

- Latencia en la comunicación.

- Problemas de compatibilidad entre diferentes versiones de servicios.

8. \*\*En una RPC... ¿Cómo definirías una llamada huérfana? ¿Qué problemas representa el hecho de que haya llamadas huérfanas?\*\*

- Una llamada huérfana en RPC es cuando el servidor procesa una solicitud, pero el cliente ya no está esperando la respuesta (por ejemplo, debido a un fallo del cliente). Estas llamadas pueden causar problemas como el uso innecesario de recursos y la falta de sincronización entre el cliente y el servidor.

9. \*\*En una RMI... ¿Dónde se encuentra el objeto sobre el cual se ejecuta el RMI, en el cliente o en el servidor?\*\*

- En RMI (Remote Method Invocation), el objeto sobre el cual se invoca el método se encuentra generalmente en el servidor. El cliente invoca un método en ese objeto remoto como si fuera local, pero en realidad, la ejecución ocurre en el servidor.

10. \*\*¿Un sistema de gestión de colas de mensajes es un sistema síncrono o asíncrono?\*\*

- Un sistema de gestión de colas de mensajes generalmente es asíncrono. Permite que los emisores envíen mensajes a una cola sin esperar una respuesta inmediata, y los receptores pueden procesar estos mensajes cuando estén listos.