EJERCICIOS TIPO

Los siguientes ejercicios agrupados por temas clave, pueden servir para practicar los conocimientos impartidos en clase. Incluyen tanto preguntas teóricas como prácticas para abarcar diversos niveles de dificultad y tipos de evaluación.

MULTIPROCESADORES (SMP Y NUMA)

1. **(Teórica)** Define qué es un multiprocesador y explica cómo se diferencia de un multicomputador.
2. **(Teórica)** Explica el concepto de memoria compartida en un sistema SMP y cómo afecta al rendimiento.
3. **(Teórica)** Describe el principal problema de escalabilidad en sistemas SMP y cómo puede mitigarse.
4. **(Teórica)** ¿Por qué los sistemas NUMA son más escalables que los SMP? Justifica tu respuesta.
5. **(Práctica)** Un sistema NUMA tiene 4 procesadores, cada uno con su propia memoria local. Si un procesador necesita acceder a la memoria de otro, el tiempo de acceso es el doble que a su propia memoria. Explica cómo este comportamiento puede afectar a la programación y optimización del sistema.
6. **(Práctica)** Comparar las ventajas y desventajas de los sistemas SMP y NUMA en términos de latencia, escalabilidad y complejidad de programación.
7. **(Teórica)** ¿Qué es una condición de carrera en multiprocesadores y qué técnicas existen para prevenirla?

MULTICOMPUTADORES Y PASO DE MENSAJES

8. **(Teórica)** Define un multicomputador y sugiere un ejemplo de aplicación en el mundo real.
9. **(Teórica)** Explica cómo funciona el paso de mensajes en un sistema multicomputador y qué protocolos se utilizan habitualmente.
10. **(Práctica)** Supón que un sistema multicomputador tiene 10 nodos conectados mediante una red de alta velocidad. Explica cómo se asignarían las tareas a los nodos utilizando un enfoque de balanceo estático y dinámico.
11. **(Teórica)** ¿Qué ventajas tienen los multicomputadores respecto a los multiprocesadores en términos de tolerancia a fallos?
12. **(Práctica)** Diseña un algoritmo sencillo para distribuir una lista de 1000 tareas entre 5 nodos en un multicomputador utilizando el modelo de paso de mensajes.
13. **(Teórica)** ¿Por qué la latencia es más alta en un multicomputador que en un sistema SMP? Explica las implicaciones prácticas.

BALANCEO DE CARGA

14. **(Teórica)** Explica en qué consiste el balanceo de carga y por qué es importante en sistemas paralelos.

15. **(Teórica)** Compara los algoritmos de balanceo de carga estático y dinámico. Menciona un caso donde cada uno sería el más adecuado.

16. **(Práctica)** Un sistema multiprocesador tiene 8 procesadores y 200 tareas. Si se usa un algoritmo de balanceo estático tipo "round-robin", ¿cuántas tareas asigna inicialmente a cada procesador?

17. **(Práctica)** En un sistema multicomputador, un nodo está sobrecargado mientras otros 3 están inactivos. Diseña un mecanismo de redistribución de la carga basado en balanceo dinámico.

18. **(Práctica)** Explica el funcionamiento del algoritmo "Least Connections" y proporciona un ejemplo práctico donde este enfoque sea preferible.

GPUS (UNIDAD DE PROCESAMIENTO GRÁFICO)

19. **(Teórica)** ¿Qué es una GPU y qué la diferencia de una CPU en términos de arquitectura y rendimiento?

20. **(Teórica)** Explica el concepto de paralelismo masivo en GPUs y su impacto en aplicaciones como redes neuronales y simulaciones físicas.

21. **(Práctica)** Una imagen de 4096 x 4096 píxeles se divide en bloques de 64 x 64 para su procesamiento paralelo. Calcula cuántos bloques se necesitan y explica cómo se asignan a los hilos.

22. **(Teórica)** ¿Cuáles son las ventajas de usar GPUs para tareas de reconocimiento de patrones y aprendizaje profundo en comparación con CPUs?

23. **(Práctica)** En el procesamiento de una simulación física, ¿cómo puede una GPU mejorar el tiempo de ejecución en comparación con una CPU? Proporciona un ejemplo.

24. **(Teórica)** ¿Qué diferencias existen entre CUDA y OpenCL como lenguajes de programación para GPUs?

FPGAS (FIELD PROGRAMMABLE GATE ARRAYS)

25. **(Teórica)** ¿Qué es una FPGA y para qué tipo de aplicaciones es ideal?

26. **(Teórica)** Compara las FPGAs con GPUs y CPUs en términos de flexibilidad, rendimiento y eficiencia energética.

27. **(Práctica)** Describe cómo una FPGA podría utilizarse para el procesamiento de datos en tiempo real en un sistema de conducción autónoma.

28. **(Teórica)** Explica la importancia de los lenguajes de descripción de hardware (HDL) en la programación de FPGAs.

29. **(Práctica)** Proporciona un ejemplo práctico donde una FPGA ofrezca una clara ventaja respecto a una GPU o CPU en términos de procesamiento de señales.

Benchmarking y optimización del rendimiento

30. **(Teórica)** Define el concepto de benchmarking y su importancia en la optimización de sistemas paralelos.

31. **(Práctica)** Un equipo está evaluando el rendimiento de dos sistemas paralelos (Sistema A y Sistema B) utilizando una misma tarea de procesamiento. Los tiempos de ejecución obtenidos son los siguientes:

Sistema A: 10 segundos con 4 procesadores.

Sistema B: 8 segundos con 4 procesadores. Si el tiempo secuencial de la tarea es 40 segundos:

a) Calcula la eficiencia de ambos sistemas.

b) ¿Cuál es más eficiente y por qué?

32. **(Práctica)** Supongamos que tienes una aplicación de simulación científica que puede dividirse en dos partes:

El 70% de la tarea es paralelizable.

El 30% es secuencial.

a) Usando la Ley de Amdahl, calcula el speedup teórico máximo al usar 8 procesadores.

b) ¿Qué ocurriría si el porcentaje paralelizable aumentara al 90%?

33. **(Teórica)** Explica qué son los cuellos de botella en sistemas paralelos y cómo el benchmarking puede ayudar a identificarlos. Menciona dos estrategias para mitigarlos.

34. **(Práctica)** Un proceso de computación paralela se mide utilizando un conjunto de benchmarks, obteniendo los siguientes tiempos parciales:

Tarea 1: 2 segundos

Tarea 2: 4 segundos

Tarea 3: 1 segundo

Tarea 4: 3 segundos

a) Calcula el tiempo total de ejecución.

b) Si se reducen los tiempos de la Tarea 2 y la Tarea 4 en un 50%, ¿cuál es el nuevo tiempo total?

c) Explica cómo este tipo de optimización puede influir en el rendimiento global.