PROGRAMACION CONCURRENTE - EXAMEN FINAL - 5-8-2009

En todos los casos, responda con claridad y sintéticamente. En los casos que corresponda, NO SE CONSIDERARAN RESPUESTAN SIN JUSTIFICAR. Tiempo Máximo 2 hs 15 min.

1. Sea "ocupados" una variable entera inicializada en N que representa la cantidad de slots ocupados de un buffer, y sean P1 y P2 dos programas que se ejecutan de manera concurrente, donde cada una de las instrucciones que los componen son atómicas.

P1::	P2::

if (ocupados < N) then
begin
buffer := elemento_a_agregar;
ocupados := ocupados + 1;
end;
	if (ocupados > 0) then
	begin
	ocupados := ocupados - 1;
	elemento_a_sacar:= buffer;
	end;

El programa funciona correctamente para asegurar el manejo del buffer? Si su respuesta es afirmativa justifique. Sino, encuentre una secuencia de ejecución que lo verifique y escríbala, y además modifique la solución para que funcione correctamente (Suponga buffer, elemento_a_agregar y elemento_a_sacar variables declaradas).

2. Defina el concepto de granularidad. Qué relación existe entre la granularidad de programas y de procesadores?

Granularidad: Relación entre el número de procesadores y el tamaño de memoria total.
Puede verse también como la relación entre cómputo y comunicación
Grano fino y grano grueso.
Tres granularidades:

I. De grano grueso (coarse-grained): pocos procesadores muy poderosos.
II. De grano fino (fine-grained): gran número de procesadores menos potentes.
III. De grano medio (medium-grained).

La relación que existe es la siguiente:

Si tenemos concurrencia limitada entre los procesos de un sistema, se puede optar por usar pocos procesadores muy poderosos (o sea máquinas de grano grueso).

Si tenemos alta concurrencia debemos aplicar grano fino (muchas procesadores).

Lo ideal es machear entre la arquitectura y la aplicación para obtener la mejor performance posible.

3. Sea la siguiente solución propuesta al problema de alocación LJN (Longest Job Next):

```
monitor LJN {
  bool libre = true;
  cond turno;
  procedure request(int tiempo) {
    if (not libre) wait(turno, (MAXVALOR - tiempo));
    libre = false;       }
  procedure release() {
    libre = true;
```

```
    signal(turno);      }
}
```

a) Funciona correctamente con disciplina de señalización Signal and Continue? Justifique.

b) Funciona correctamente con disciplina de señalización Signal and Wait? Justifique
Nota: suponga que MAXVALOR es el máximo valor de tiempo por el cual un proceso puede solicitar el recurso.

4. a) Qué significa que un problema sea de "exclusión mutua selectiva"?

En los problemas de exclusión mutua selectiva, cada proceso compite por sus recursos no con todos los demás procesos sino con un subconjunto de ellos. Dos casos típicos de dicha competencia se producen cuando los procesos compiten por los recursos según su tipo de proceso o por su proximidad.

b) El problema de los lectores-escritores es de exclusión mutua selectiva? Por qué?

El problema de los lectores-escritores es de exclusión mutua selectiva, ya que los lectores compiten solamente con los escritores para acceder al recurso, y en cambio los escritores compiten tanto con sus colegas como con los lectores.

c) De los problemas de los baños planteados en teoría, cuál podría ser de exclusión mutua selectiva? Por qué?

Dos baños utilizables simultáneamente por un número máximo de varones (K1) y de mujeres (K2), con una restricción adicional respecto que el total de usuarios debe ser menor que K3 (K3 < K1 + K2). Es de EMS porque se compite con un subconjunto de procesos (las mujeres solo con las mujeres y los varones solo con los varones).

d) Por qué el problema de los filósofos es de exclusión mutua selectiva? Si en lugar de 5 filósofos fueran 3, el problema seguiría siendo de exclusión mutua selectiva? Por qué?

El problema de los filósofos es de EMS, ya que se basa en que 5 filósofos tratan de comer usando dos tenedores, el de su izquierda y el de su derecha; cuando sólo hay 5 tenedores en total, por lo que se tienen procesos (un filósofo) que compiten con un subconjunto de procesos (los filósofos que lo rodean) por recursos (el tenedor a la izquierda y a la derecha de cada filósofo).

Si en lugar de 5 filósofos fueran 3 el problema dejaría de ser de EMS ya que cada filósofo competiría con todos los demás filósofos por los mismos recursos.

e) El problema de los filósofos resuelto en forma centralizada y sin posiciones fijas es de exclusión mutua selectiva? Por qué?

El problema de los filósofos resuelto en forma centralizada y sin posiciones fijas no es de exclusión mutua selectiva ya que al no haber posiciones fijas cualquiera puede agarrar los cubiertos para comer, por lo tanto todos los procesos compiten entre si.

f) Si en el problema de los lectores-escritores se acepta sólo 1 escritor o 1 lector en la BD, tenemos un problema de exclusión mutua selectiva? Por qué?

Si en el problema de los lectores-escritores se acepta sólo 1 escritor o 1 lector en la BD, no tenemos un problema de exclusión mutua selectiva ya que ambos compiten entre sí (o sea entre todos los procesos).

5. Resuelva con monitores el siguiente problema:
Tres clases de procesos comparten el acceso a una lista enlazada: *searchers*, *inserters* y *deleters*. Los searchers sólo examinan la lista, y por lo tanto pueden ejecutar concurrentemente unos con otros. Los inserters agregan nuevos items al final de la lista; las inserciones deben ser mutuamente exclusivas para evitar insertar dos items casi al mismo tiempo. Sin embargo un insert puede hacerse en paralelo con uno o más searchers. Por último, los deleters remueven items de cualquier lugar de la lista. A lo sumo un deleter puede acceder la lista a la vez, y el borrado también debe ser mutuamente exclusivo con searchers e inserciones.

6. Explique sintéticamente los 7 paradigmas de interacción entre procesos en programación distribuida. Ejemplifique en cada caso.

I. Servidores Replicados: Los servidores manejan (mediante múltiples instancias) recursos compartidos tales como dispositivos o archivos. Ejemplo: File servers con múltiples clientes y una instancia de servidor por cliente (o por File, o por Unidad de almacenamiento).
II. Algoritmos Heartbeat: Los procesos periódicamente deben intercambiar información con mecanismos tipo send/receive. Ejemplo: Modelos biológicos / Problema de los N Cuerpos / Modelos de simulación paramétrica.
III. Algoritmos Pipeline: La información recorre una serie de procesos utilizando alguna forma de receive/send. Se supone una arquitectura de procesos/procesadores donde la salida de uno es entrada del siguiente. Los procesadores pueden distribuirse DATOS o FUNCIONES. Ejemplo: Redes de Filtros, Tratamiento de Imágenes.
IV. Probes (send) y Echoes (receive): La interacción entre los procesos permite recorrer grafos o árboles (o estructuras dinámicas) diseminando y juntando información. Un ejemplo clásico es recuperar la topología activa de una red móvil o hacer un broadcast desde un nodo cuando no se "alcanzan" o "ven" directamente todos los destinatarios.
V. Algoritmos Broadcast: Permiten alcanzar una información global en una arquitectura distribuida. Sirven para toma de decisiones descentralizadas. En general en sistemas distribuidos con múltiples procesadores, las comunicaciones colectivas representan un costo crítico en tiempo. Un ejemplo típico es la sincronización de relojes en un Sistema Distribuido de Tiempo Real.
VI. Token Passing: En muchos casos la arquitectura distribuida recibe una información global a través del viaje de tokens de control o datos. La arquitectura puede ser un anillo, caso en el cual el manejo se asemeja a un pipeline, pero también puede ser cualquier topología (tipo objetos distribuidos). Los tokens normalmente habilitan el control para la toma de decisiones distribuidas.
VII. Manager/Workers: Implementación distribuida del modelo de bag of tasks. Un procesador controla los datos y/o procesos a ejecutarse y múltiples procesadores acceden a él para acceder a datos/procesos y ejecutar las tareas distribuidas.

7. a) Cuál es el objetivo de la programación paralela?

El objetivo de la programación paralela el de resolver un problema en el menor tiempo (o un problema más grande en aproximadamente el mismo tiempo) usando una arquitectura multiprocesador en la que se pueda distribuir la tarea global en tareas que puedan ejecutarse en distintos procesadores.

b) Defina las métricas de speedup y eficiencia. Cuál es el significado de cada una de ellas (que miden)? Ejemplifique.

La métrica de speedup representa la ganancia que tenemos al usar más procesadores para un mismo problema. Se define por la siguiente fórmula:

$S = T_s / T_p$

I. p es el número de procesadores.
II. T_s es el tiempo de ejecución del algoritmo secuencial.
III. T_p es el tiempo de ejecución del algoritmo paralelo con p procesadores.

El rango de valores en general se encuentra entre 0 y p. Aunque hay casos de speedup superlineal, es decir mayor que p.
Por ejemplo si obtenemos un speedup S = p estamos frente a un speedup lineal, esto quiere decir que duplicando el número de procesadores duplicamos la velocidad, lo cual es ideal.

La métrica de eficiencia mide la fracción de tiempo en que los procesadores son útiles para el cómputo. Se define por la siguiente fórmula:

$E = S / p = T_s / pT_p$

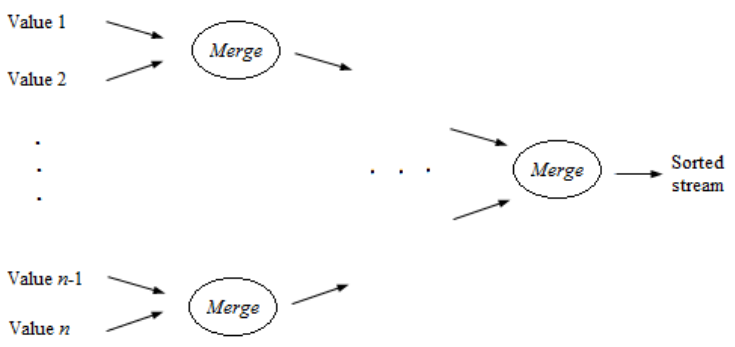
El rango de valores se encuentra entre 0 y 1, dependiendo de la efectividad de uso de los procesadores.
Por ejemplo: Algoritmos con speedup linear y algoritmos que corren en un procesador simple, tienen eficiencia 1. Muchos algoritmos difíciles de paralelizar tienen eficiencia cercana a 1/log p, que se aproxima a cero a medida que el número de procesadores se incrementa.

c) Suponga que el tiempo de ejecución de un algoritmo secuencial es de 1000 unidades de tiempo, de las cuales el 80% corresponden a código paralelizable. Cuál es el límite en la mejora que puede obtenerse paralelizando el algoritmo?

Si el tiempo de ejecución de un algoritmo secuencial es de 10000 unidades y el porcentaje de unidades paralelizables es del 80%, quiere decir que 2000 unidades (el 20%) deben ejecutarse secuencialmente, por lo tanto el tiempo mínimo esperable es de 2000 para un procesador. Por esta razón nos conviene utilizar una cantidad de procesadores tal que los mantenga a todos ocupados esa cantidad de tiempo, es decir en este caso conviene tener 5 procesadores porque si tuviéramos más un procesador estaría trabajando 2000 unidades de tiempo y los demás terminarían de ejecutar las instrucciones antes, y tendrían que esperarlo.

8. Suponga los siguientes métodos de ordenación de menor a mayor para n valores (n par y potencia de 2), utilizando pasaje de mensajes:

i- Un pipeline de filtros. El primero hace input de los valores de a uno por vez, mantiene el mínimo y le pasa los otros al siguiente. Cada filtro hace lo mismo:



recibe un stream de valores desde el predecesor, mantiene el más chico y pasa los otros al sucesor.

ii- Una red de procesos filtro (como la de la figura).

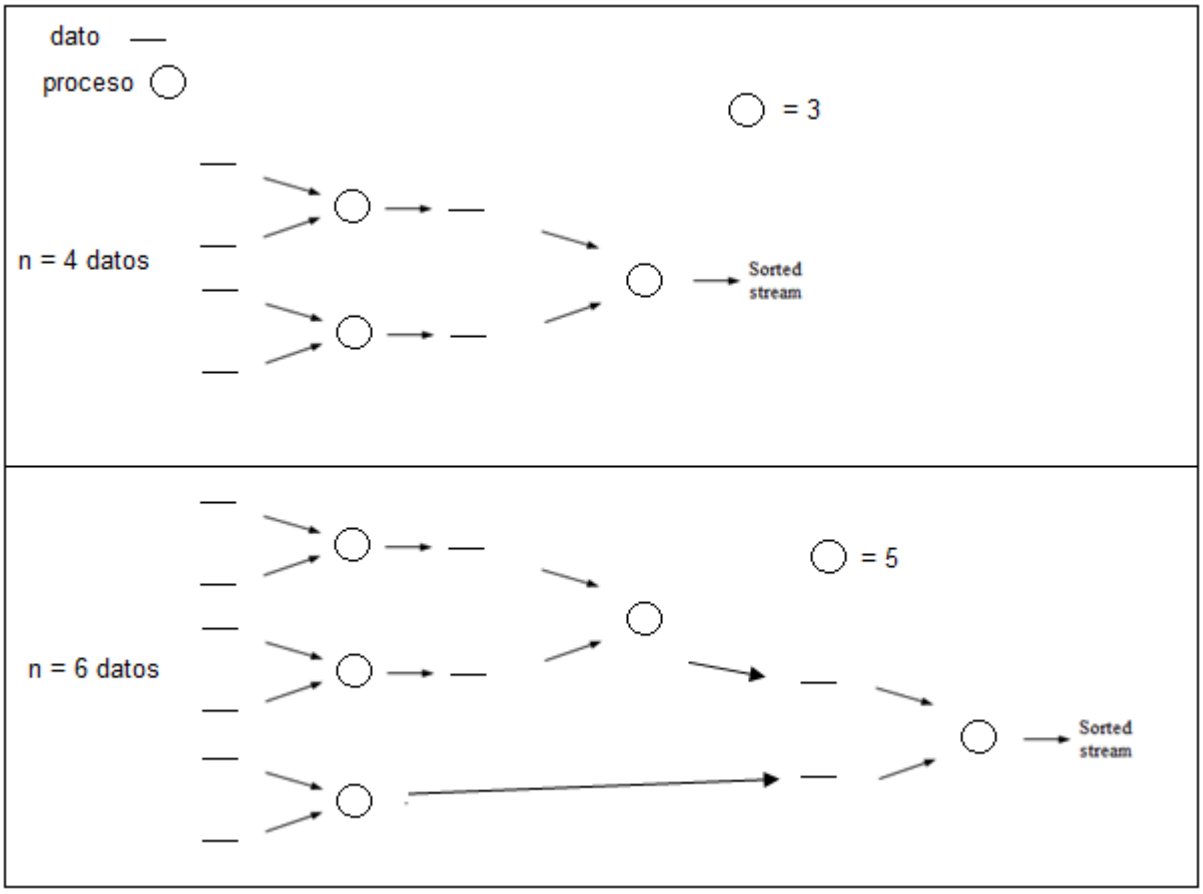
iii- Odd/even exchange sort. Hay n procesos P[1:n], Cada uno ejecuta una serie de rondas. En las rondas "impares", los procesos con número impar P[impar] intercambian valores con P[impar+1]. En las rondas "pares", los procesos con número par P[par] intercambian valores con P[par+1] (P[1] y P[n] no hacen nada en las rondas "pares"). En cada caso, si los números están desordenados actualizan su valor con el recibido.

Asuma que cada proceso tiene almacenamiento local sólo para dos valores (el próximo y el mantenido hasta ese momento).

a) Cuántos procesos son necesarios en i e ii? Justifique.

Algoritmo i: Se necesitan n procesos ya que cada proceso almacena el valor más pequeño de todos los que recibe, es decir que como se envían n valores, se necesitan n procesos.

Algoritmo ii: Como en un principio se necesitan procesar n datos, y cada proceso recibe dos, entonces se necesitan n/2 procesos. Por consiguiente necesitaremos además n/2 - 1 procesos para hacer un sort del resultado de los anteriores. Por lo tanto tenemos n/2 + n/2 -1 = n-1, y entonces se necesitan n-1 procesos en total. Por ejemplo observemos dos árboles uno con n=4 y otro con n=6, la cantidad de nodos que representan procesos es 3 y 5 respectivamente.



b) Cuántos mensajes envía cada algoritmo para ordenar los valores? Justifique.

Algoritmo i: Asignándole a cada proceso un número entero consecutivo, siendo el primer proceso el 1, el segundo el 2, así sucesivamente hasta el último proceso n, tenemos que:
Al proceso 1 le llegan n mensajes, y envía n-1 mensajes, porque se queda con el valor más chico de todos los que le llegan y va enviando los demás.
Al proceso 2 le llegan n-1 mensajes y envía n-2 mensajes, por la misma razón del proceso anterior.
Y así sucesivamente, hasta el proceso n, el cual recibe 1 (n-(n-1)) mensaje y no manda ningún (n-n) mensajes.
Observando dicho patrón en la sucesión de envíos, podemos afirmar que el número total de mensajes que se envían es:

$$\sum_{i=1}^{n} i = \frac{n\ (n+1)}{2}$$

Algoritmo ii: Como por cada proceso se realiza un solo envío y al principio se envían n mensajes (1 por cada elemento), entonces la cantidad de mensajes que se envían es (n-1) + n = 2n - 1.

Algoritmo iii: Cada proceso envía uno y recibe uno dependiendo de que si la ronda es par o impar. Además para calcular u ordenar, en este algoritmo se necesitan n rondas en el peor de los casos. Por cada ronda se mandan n mensajes y se reciben n mensajes para que cada proceso (par o impar dependiendo de la ronda) intercambie su valor con otro proceso. Es por todo esto que en este algoritmo se mandan n^2.

c) En cada caso, cuáles mensajes pueden ser enviados en paralelo (asumiendo que existe el hardware apropiado) y cuáles son enviados secuencialmente? Justifique.

Algoritmo i: Cada proceso puede enviar su mensaje de salida en paralelo a otro proceso, siempre y cuando haya recibido el input enviado por su predecesor.

Algoritmo ii: Cualquier proceso puede enviar su mensaje de salida en paralelo a otro proceso, siempre y cuando haya recibido los dos inputs provenientes de sus dos nodos hijos.

Algoritmo iii: Por ronda se mandarán n/2 mensajes porque solo los de ronda par o ronda impar enviarán un mensajes con el dato más grande al mismo tiempo (también podría suceder lo mismo y recibir n/2 simultáneamente). Es por esto que este algoritmo puede enviar $n^2/2$ mensajes en paralelo.

d) Cuál es el tiempo total de ejecución de cada algoritmo? Asuma que cada operación de comparación o de envío de mensaje toma una unidad de tiempo. Justifique.

Algoritmo i: Utilizando el cálculo de la cantidad de mensajes del punto a) y razonando que cada vez que se envía un mensaje se realiza una comparación antes, tenemos que el tiempo de ejecución será igual a la cantidad de mensajes multiplicada por 2 unidades de tiempo (1 comparación + 1 mensaje):

$$2 \sum_{i=1}^{n} i = 2\ \frac{n\ (n+1)}{2} = n\ (n+1) \Rightarrow O(n^2)$$

Algoritmo ii: Por la misma razón que el algoritmo anterior tenemos:

$2(2n-1) = 4n-1 \Rightarrow O(n)$

Algoritmo iii: Cada proceso en cada ronda hace una comparación, envía, recibe y hace una asignación, por eso tenemos que por cada proceso se tarda 4 unidades. Como se necesitan n rondas en el peor de los casos, se realizará 4n unidades de tiempo en total, por lo tanto es de O(n).