Concurrencia y Paralelismo - Examen teórico de promoción - 16/07/2021

1. ***(0.2 puntos) Conceptos generales:*** Escribir una definición de *programación concurrente* y *programación paralela*, diferenciando ambos conceptos. Definir lo que es la sincronización entre procesos y explicar cuáles son los dos tipos de sincronización en que se clasifica. Dar un ejemplo (explicarlo con palabras, no con código) de un problema donde se usen ambos tipos de sincronización.

2. ***(0.2 puntos) Problema de la sección Crítica:*** Explicar en qué consiste este problema. Nombrar y explicar las propiedades que debe cumplir una solución a este problema. Explicar (con palabras, no código) en forma sintética 3 soluciones a este problema usando sólo variables compartidas con la técnica *Busy Waiting*, y que cumplan TODAS las propiedades requeridas en este problema.

3. ***(0.25 puntos) Monitores:*** Indicar la forma en que se realiza la comunicación y la sincronización con esta herramienta. Explicar la diferencia entre los protocolos de sincronización en monitores: *signal and wait (S&W)* y *signal and continue (S&C)*. Implementar un monitor que funcione correctamente con el protocolo *S&W* y no con *S&C* (además de implementar el monitor enuncie que debería resolver).

4. ***(0.25 puntos) Librería OpenMP:*** Indicar y explicar las principales características de la librería. Detallas las diferentes formas que maneja el constructor *Parallel For* para distribuir las iteraciones del for entre los hilos.

5. ***(0.3 puntos) Librería Pthreads:*** Indicar y explicar cómo funcionan las herramientas que tiene la librería Pthreads para manejar la sincronización por exclusión mutua y por condición, y como se relacionan entre ellas. Suponga la siguiente porción de código, ¿cómo se asegura que el *buffer* se accede con exclusión mutua, suponiendo que *hayElemento* se inicializa en 0?

```
void *productor(void *datos) {
  tipo_element elem;

  while (true) {
     generar_elemento(elem);
     pthread_mutex_lock (&mutex);
     while (hayElemento = = 1)
        pthread_cond_wait (&vacio, &mutex);
     buffer = elem;
     hayElemento = 1;
     pthread_cond_signal (&lleno);
     pthread_mutex_unlock (&mutex);
  }
}
```

```
void *consumidor(void *datos) {
  tipo_element elem;

  while (true) {
     pthread_mutex_lock (&mutex);
     while (hayElemento = = 0)
        pthread_cond_wait (&lleno, &mutex);
     elem= Buffer;
     hayElemento = 0;
     pthread_cond_signal (&vacio);
     pthread_mutex_unlock (&mutex);
     procesar_elemento(elem);
  }
}
```

6. ***(0.3 puntos) Librería MPI:*** Explicar la importancia de las comunicaciones colectivas en MPI y cómo funcionan para lograr un buen rendimiento. Suponga que existen 8 procesos (cada uno en un procesador diferente), y el que tiene *rank=0* debe enviar un número entero a todos los otros por medio de MPI_Bcast; mostrar gráficamente y explicar los pasos que realiza el Bcast para que todos los procesos reciban el valor.

7. ***(0.3 puntos) Métricas en Sistemas Paralelos:*** Sea la siguiente solución al problema del producto de matrices de *nxn* con ***P*** procesos en paralelo con variables compartidas. Suponga *n =640* y cada procesador capaz de ejecutar un proceso.

```
process worker [ w = 1 to P] {
  int primera = (w-1)*(n/P) + 1;
  int ultima = primera + (n/P) -1;

 for [i = primera to ultima]
  { for [j = 1 to n]
     { c[i,j] = 0;
       for [Z = 1 to n]
          c[i,j] = c[i,j] + (a[i,Z]*b[Z,j]);
     }
  }
}
```

a. Calcular cuántas asignaciones, sumas y productos se hacen secuencialmente (caso en que P=1); y cuántas se realizan en cada procesador en la solución paralela con P=8.
b. Dados que los procesadores de P1 y P2 son idénticos, con tiempos de asignación 2, de suma 4 y de producto 8; los procesadores P3 y P4 son el doble de potentes (tiempos 1, 2 y 4 para asignaciones, sumas y productos respectivamente); y el resto de los procesadores (de P5 a P8) son la mitad de potente que P1 (tiempos 4, 8 y 16 para asignaciones, sumas y productos respectivamente). Calcular cuánto tarda el programa paralelo y el secuencial.
c. Realizar paso a paso el cálculo del valor del speedup y la eficiencia.
d. Modificar el código para lograr una mejor eficiencia. Calcular la nueva eficiencia y comparar con la calculada en (c).

NOTA: para hacer los cálculos sólo tenga en cuenta las operaciones realizadas en la instrucción dentro del for Z.

8. ***(0.2 puntos) Paradigmas:*** Explicar que es un paradigma y para cuál es su utilidad. Explicar las características principales de los paradigmas "Master/Worker", "Dividir y Conquistar", SPMD y Pipeline. Mencionar para cada uno de estos paradigmas un ejemplo donde podría utilizarse.

Examen teórico de promoción **17/02/2021**

1. ***(0.1 puntos) Conceptos generales:*** Escribir una definición de *programación concurrente* y *programación paralela*. Diferencie ambos conceptos. Determinar el objetivo de la programación paralela.

2. ***(0.1 puntos) Conceptos generales:*** Definir sincronización entre procesos. Explicar cuáles son los dos tipos de sincronización en que se clasifica.

3. ***(0.1 puntos) Conceptos generales:*** Definir comunicación entre procesos. Explicar cuáles son los dos tipos de comunicación en que se clasifica.

4. ***(0.2 puntos) Problema de la sección Crítica:*** Explicar en qué consiste este problema. Nombrar y explicar las propiedades que debe cumplir una solución a este problema. Explicar en forma sintética 3 soluciones de tipo Busy Waiting a este problema (usando sólo variables compartidas).

5. ***(0.1 puntos) Sentencias Await:*** Explicar para que sirve y cómo funcionan las sentencias AWAIT en sus diferentes formas. Dar un ejemplo con cada una de ellas.

6. ***(0.1 puntos) Monitores:*** Indicar la forma en que se realiza la comunicación y la sincronización con esta herramienta.

7. ***(0.3 puntos) Librería Pthreads:*** Indicar y explicar cómo funcionan las herramientas que tiene la librería Pthreads para manejar la sincronización por exclusión mutua y por condición, y como se relacionan entre ellas. Suponga la siguiente porción de código, ¿cómo se asegura que el *buffer* se accede con exclusión mutua, suponiendo que *hayElemento* se inicializa en 0?

```
void *productor(void *datos) {
  tipo_element elem;

  while (true) {
     generar_elemento(elem);
     pthread_mutex_lock(&mutex);
     while (hayElemento = = 1)
        pthread_cond_wait (&vacio, &mutex);
     buffer = elem;
     hayElemento = 1;
     pthread_cond_signal(&lleno);
     pthread_mutex_unlock(&mutex);
  }
}
```

```
void *consumidor(void *datos) {
   tipo_element elem;

   while (true) {
      pthread_mutex_lock(&mutex);
      while (hayElemento = = 0)
         pthread_cond_wait (&lleno, &mutex);
      elem= Buffer;
      hayElemento = 0;
      pthread_cond_signal(&vacio);
      pthread_mutex_unlock(&mutex);
      procesar_elemento(elem);
   }
}
```

8. ***(0.3 puntos) Librería MPI:*** Explicar la importancia de las comunicaciones colectivas en MPI y cómo funcionan para lograr un buen rendimiento. Suponga que existen 8 procesos (cada uno en un procesador diferente), y el que tiene *rank=0* debe enviar un número entero a todos los otros por medio de MPI_Bcast; mostrar gráficamente y explicar los pasos que realiza el Bcast para que todos los procesos reciban el valor.

9. ***(0.4 puntos) Métricas en Sistemas Paralelos:*** Sea la siguiente solución al problema del producto de matrices de *nxn* con ***P*** procesos en paralelo con variables compartidas. Suponga *n =400* y cada procesador capaz de ejecutar un proceso.

```
process worker [ w = 1 to P] {
  int primera = (w-1)*(n/P) + 1;
  int ultima = primera + (n/P) -1;

 for [i = primera to ultima]
   { for [j = 1 to n]
      { c[i,j] = 0;
        for [Z = 1 to n]
           c[i,j] = c[i,j] + (a[i,Z]*b[Z,j]);
      }
   }
}
```

a. Calcular cuántas asignaciones, sumas y productos se hacen secuencialmente (caso en que P=1); y cuántas se realizan en cada procesador en la solución paralela con P=8.
b. Si los procesadores P1 y P2 son idénticos, con tiempos de asignación 1, de suma 2 y de producto 4, y si el resto de los procesadores (P3 a P8) son la mitad de potente (tiempos de asignación 2, suma 4 y producto 8), calcular cuánto tarda el proceso total concurrente y el secuencial.
c. Realizar paso a paso el cálculo del valor del speedup y la eficiencia.
d. Modificar el código para lograr una mejor eficiencia. Calcular la nueva eficiencia y comparar con la calculada en (c).

NOTA: para hacer los cálculos sólo tenga en cuenta las operaciones realizadas en la instrucción dentro del for Z.

10. ***(0.3 puntos) Paradigmas:*** Suponga que una imagen se encuentra representada por una matriz A(nxn), y que el valor de cada pixel es un número entero que es mantenido por un proceso distinto (es decir, el valor del pixel i,j está en el proceso P(i,j)). Cada proceso puede comunicarse sólo con sus vecinos izquierdo, derecho, arriba y abajo (los procesos de las esquinas tienen solo 2 vecinos, y los otros en los bordes de la grilla tienen 3 vecinos). Escriba (pseudocodigo) un *algoritmo heartbeat* que calcule el máximo y el mínimo valor de los pixels de la imagen. Al terminar el programa, cada proceso debe conocer ambos valores.