Computación Paralela y Distribuída

Exámen Parcial
Pregrado
2022-II
Profesor: José Fiestas
1.01

Indicaciones específicas:

- Esta evaluación contiene 12 páginas (incluyendo esta página) con 4 preguntas. El total de puntos son 20.
- El tiempo límite para la evaluación es 120 minutos.
- El exámen deberá ser respondido en un solo archivo pdf. Si es foto pueden ser varios archivos
- En el examen no se pide desarrollar un código completo en paralelo, pero no está prohibido. En caso de hacerlo, puede entregarlo anexo a la solución del problema
- Deberá subir estos archivos directamente a https://www.gradescope.com
- Se permite consultar el material de clases y bibliografía del curso

Competencias:

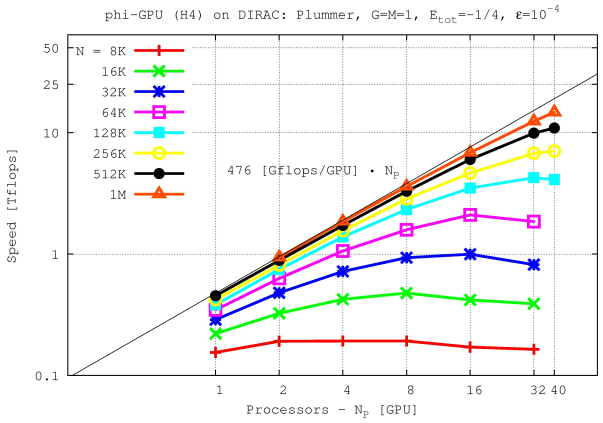
- Aplica conocimientos de computación apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa. (nivel 3)
- Resuelve problemas de computación y otras disciplinas relevantes en el dominio (nivel 3)
- Analiza y valora el impacto local y global de la computación sobre las personas, las organizaciones y la sociedad (nivel 3)
- Reconoce la necesidad del aprendizaje autónomo (nivel 2)

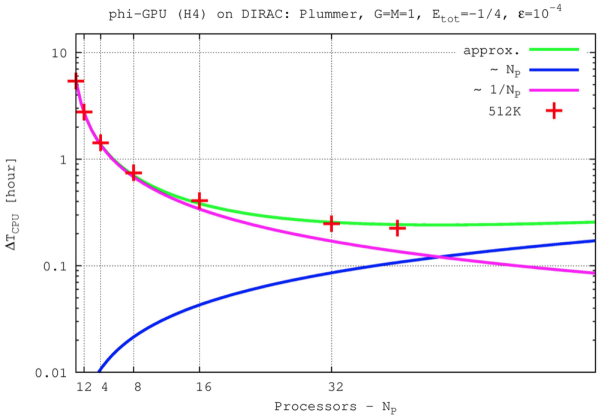
Calificación:

Tabla de puntos (sólo para uso del professor)

Question	Points	Score
1	5	
2	5	
3	5	
4	5	
Total:	20	

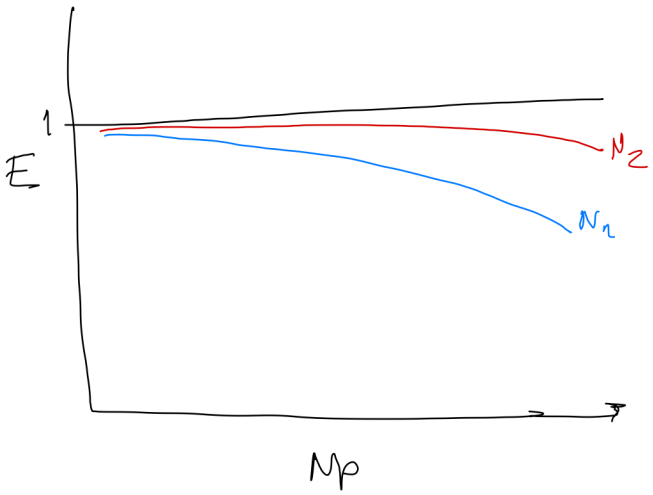
1. (5 points)
   Las siguientes gráficas representan velocidad en FLOPs vs. número de nodos de un código de N-cuerpos para N=512K, así como el tiempo de ejecución en horas vs. número de nodos.





   - Defina el tiempo de ejecución del problema de N-cuerpos, considerando el tiempo de cómputo y de comunicación. Éste último esta dado por $T_{comm} = O(p(\alpha + x\beta))$, donde **p** es la cantidad de procesos, α es la latencia, β es el ancho de banda, y **x** es el tamaño del mensaje enviado/recibido **(1 pt)**
     Respuesta: Este problema tiene una complejidad secuencial O(n^2) y teórica O(n^2/p), (curva magenta), que se ve modificada por el tiempo de comunicación, que es $O(p(\alpha + x\beta))$, lo que se puede observa en el gráfico de la derecha (curva azul), donde x=N/P, ya que cada proceso debe enviar su dara completa al resto de procesos para completar el calculo.
   - Encuentre una expresión para el speedup y compare los resultados con las gráficas mostradas ¿Qué factores generan una diferencia entre la data experimental y la teórica? Argumente sus respuestas.**(1 pt) Respuesta:** $S = \frac{O(n^2)}{O(n^2/p+p(\alpha+n/p\beta))}$, donde se puede apreciar la proporcionalidad con p (caso ideal), además del overhead (comunicacion entre procesos) en el segundo término del denominador
   - Construya una expresión y un gráfico para la eficiencia y describa el análisis de strong/weak scaling para este problema.**(1.5 pts)**
     Respuesta: $E = \frac{O(n^2/p)}{O(n^2/p+p(\alpha+n/p\beta))} = \frac{O(1)}{O(1+p^2/n^2(\alpha+n/p\beta))}$, lo que indica que un análisis de weak scaling permitirá mantener E=O(1) , con la condicion $n \propto p$
   - Ahora considere una optimización en la complejidad de comunicación, tal que $T_{comm} = O(log(p)(\alpha + x\beta))$¿Cómo cambian los resultados anteriores en este caso? **(1.5 pts)**
     Respuesta: $S = \frac{O(n^2)}{O(n^2/p+log(p)(\alpha+n/p\beta))}$, $E = \frac{O(n^2/p)}{O(n^2/p+log(p)(\alpha+n/p\beta))} = \frac{O(1)}{O(1+log(p)/n}$, lo que indica que un análisis de weak scaling permitirá mantener E=O(1) , con la condicion $n \propto log(p)$

   La rúbrica para esta pregunta es:



Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Método o algoritmo	Describe al algoritmo de solución del problema planteado en forma adecuada (2 pts)	Algoritmo con algunos errores que no afectan el resultado (1.5 pts).	Algoritmo con errores que afectan minimamente el resultado (0.5 pt).	Algoritmo con errores, que afectan significativamente el resultado (0 pts)
Resultados	Solución correcta usando un método adecuado (1 pt)	Errores mínimos en el método que no afectan el resultado (0.6 pts)	Errores en el método que afectan el resultado (0.3 pts)	No aplica el método ni llega a la solución correcta (0 pts).
Optimización	Solución original y optimizada (1 pt)	Solución parcialmente optimizada (0.6 pts)	Solución original pero no optimizada (0.3 pts)	Resultado encontrado no está optimizado (0 pts).

2. (5 points)
Dadas las siguientes llamadas a funciones desde el main

```
int main(){
double a,b,c,d,e,x,y,z,w;
calculo_1(a,b);
calculo_2(b,b);
calculo_3(c,d);
calculo_4(d,d);
e=a+b*c+d;
calculo_5(x,a/e,b/e);
calculo_6(y,b/e);
calculo_7(z,c/e);
w=x+y+z;
}
```

El primer argumento de todas las funciones usadas es de salida y el resto de argumentos son argumentos de entrada. Por ejemplo, calculo_1(a,b) es una función que a partir de la variable b modifica la variable a.

- Describa una forma de paralelizar el código, utilizando directivas MPI **(2 pts)**

```
/* primera fase */
  if (rank==0)
    a = calculo_1(a,b);
  else if (rank==1)
    b = calculo_2(b,b);
  else  if (rank==2) /* rank==2 */
    c = calculo_3(c,d);
   else /* rank==3 */
    d = calculo_4(d,d);
    MPI_Bcast(a,1,MPI_DOUBLE,0,MPI_COMM_WORLD);
    MPI_Bcast(b,1,MPI_DOUBLE,1,MPI_COMM_WORLD);
    MPI_Bcast(c,1,MPI_DOUBLE,2,MPI_COMM_WORLD);
    MPI_Bcast(d,1,MPI_DOUBLE,3,MPI_COMM_WORLD);
  e=a+b*c+d;
  /* segunda fase */
  if (rank==0)
    x = calculo_5(x,a/e,b/e);
  else if (rank==1)
    x = calculo_6(y,b/e);
  else /* rank==2 */
    x = calculo_7(z,c/e);
  MPI_Reduce(&x, &w, 1, MPI_DOUBLE, MPI_SUM, 0,
     MPI_COMM_WORLD);
```

- Describa una forma de paralelizar el código, utilizando directivas OMP **(2 pts)**

```
/* primera fase */
#pragma omp parallel sections num_threads(2) shared(a,b,c
   ,d,e)
{
    #pragma omp section
            a = calculo_1(a,b);
            b = calculo_2(b,b);
    #pragma omp section
            c = calculo_3(c,d);
            d = calculo_4(d,d);
        }
  e=a+b*c+d;
#pragma omp parallel sections num_threads(3) shared(a,b,c
   ,d,e)
{
  /* segunda fase */
    #pragma omp section
    x = calculo_5(x,a/e,b/e);
    #pragma omp section
    y = calculo_6(y,b/e);
    #pragma omp section
    z = calculo_7(z,c/e);
    }
  w=x+y+z;
```

- Escoja el método más eficiente. Argumente su respuesta.**(1 pt)**
  Respuesta: al tratarse de variables de tamaño limitado, el segundo método, que evita la comunicación entre procesos, es el de menor tiempo de ejecución, por lo tanto el más eficiente.

Argumente cada una de sus respuestas

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Método o algoritmo	Describe al algoritmo de solución del problema planteado en forma adecuada (2 pts)	Algoritmo con algunos errores que no afectan el resultado (1.5 pts).	Algoritmo con errores que afectan minimamente el resultado (0.5 pt).	Algoritmo con errores, que afectan significativamente el resultado (0 pts)
Resultados	Solución correcta usando un método adecuado (1 pt)	Errores mínimos en el método que no afectan el resultado (0.6 pts)	Errores en el método que afectan el resultado (0.3 pts)	No aplica el método ni llega a la solución correcta (0 pts).
Optimización	Solución original y optimizada (1 pt)	Solución parcialmente optimizada (0.6 pts)	Solución original pero no optimizada (0.3 pts)	Resultado encontrado no está optimizado (0 pts).

3. (5 points)
3.1) En la siguiente función, se busca un valor en un array. Describa las directivas necesarias OMP para paralelizar la función de la manera más eficiente posible. La búsqueda debe culminar tan pronto como se encuentre el elemento buscado **(2.5 pts)**
Input: array **A** de **n** elementos y valor de busqueda **v**
Output: booleano **true** (encontrado) o **false** (no encontrado)
Procedimiento **busqueda**:

```
int busqueda(int A, int n, int v)
          {
             int encontrado=0, i=0;
             while (!encontrado && i<n) {
                if (*(A+i)==v) encontrado=1;
                i++;
            }
   return encontrado;
}
```

Respuesta:

```
int busqueda(int A[], int n, int v)
{
         int encontrado=0;
         int i, salto;
         #pragma omp parallel private(i)
         {
            i = omp_get_thread_num();
            salto = omp_get_num_threads();
            while (!encontrado && i<n) {
            if (A[i]==v) {
             encontrado=1;
             #pragma omp flush // actualiza valor de
                encontrado para todos los hilos
             }
             i += salto;
         }
     }
return encontrado;
}
```

3.2) Dada la siguiente función:
Input: array A de n elementos
Output: array B
Procedimiento **trigonometrica**:

```
void trigonometrica(int n, double A[], double B[])
{
            int i;
            for (i=0; i<n; i++) {
                B[i]=sin²(A[i]);
    }
}
```

Describa las directivas necesarias OMP para paralelizar la función, tal que se indique el número de cada hilo y cuántas iteraciones ha procesado. Estime la cantidad de FLOPs necesarios para resolver el problema en paralelo **(2.5 pts) Solución:**

```
void trigonometrica(int n, double A[], double B[])
     {
        int i, cont;
        #pragma omp parallel private(cont)
        {
            cont=0;
            #pragma omp for
            for (i=0; i<n; i++) {
                B[i]=sin(A[i])*sin(A[i]);
                cont++;
        }
            printf("Hilo %d: %d iteraciones procesadas\n",
                      omp_get_thread_num(), cont);
} }
     }
```

La cantidad de FLOPs es n*(2k+1), donde k es la cantidad de FLOP de la operacion **sin()**. Dato adicional: la operacion sin() se puede expandir en una serie puede estimar en una serie como $\sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}(x-0)^k = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + ...$, lo que resulta (hasta 4 terminos) en k= 16

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Método o algoritmo	Describe al algoritmo de solución del problema planteado en forma adecuada (2 pts)	Algoritmo con algunos errores que no afectan el resultado (1.5 pts).	Algoritmo con errores que afectan minimamente el resultado (0.5 pt).	Algoritmo con errores, que afectan significativamente el resultado (0 pts)
Resultados	Solución correcta usando un método adecuado (1 pt)	Errores mínimos en el método que no afectan el resultado (0.6 pts)	Errores en el método que afectan el resultado (0.3 pts)	No aplica el método ni llega a la solución correcta (0 pts).
Optimización	Solución original y optimizada (1 pt)	Solución parcialmente optimizada (0.6 pts)	Solución original pero no optimizada (0.3 pts)	Resultado encontrado no está optimizado (0 pts).

4. (5 points)
4.1) Describa las directivas MPI necesarias para implementar la función shift left (desplazamiento a la izquierda) de una cadena de bits distribuída en p procesos. x << y desplaza **y** posiciones el entero original **x**. **(2.5 pts)**
Por ejemplo, para 3 procesos, x=119, y=1 :

P_0	P_1	P_2
[0 0 1]	[1 1 0]	[1 1 1]

x<<y, sería

P_0	P_1	P_2
[0 1 1]	[1 0 1]	[1 1 0]

su valor decimal, 238
Solución:

```
void desplazar(double vloc[], int mb)
     {
        int p, rank, i,first,last;

        MPI_Comm_size(MPI_COMM_WORLD, &p);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    if(rank==p-1) vloc[mb-1]=0;
        /* Desplazamiento local */
        last = vloc[mb-1];
        first=vloc[0];
        for (i=mb-1; i>0; i--)
           vloc[i-1] = vloc[i];
        /* Comunicacion con los vecinos */
        if(rank!=0)
                MPI_Sendrecv(&first, 1, MPI_INT, rank-1, 0,&
                   last, 1, MPI_INT, rank, 0,
            MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
     }
```

4.2) Describa las directivas MPI necesarias para, dada una matriz A de N × M números reales e índices r (entre 0 y N-1), s (entre 0 y M-1)haga que la fila r y la columna s de la matriz se comuniquen desde el maestro al resto de procesos (sin comunicar ningún otro elemento de la matriz). Debe generar un tipo MPI para optimizar la comunicación.
(2.5 pts) Solución:

```
void bcast_fila_col(double A[N][N], int k)
     {
```

```
    MPI_Datatype colu;
    MPI_Type_vector(N, 1, N, MPI_DOUBLE, &colu);
    MPI_Type_commit(&colu);
    /* Envio de la fila */
    MPI_Bcast(&A[k][0], N, MPI_DOUBLE, 0, MPI_COMM_WORLD)
       ;
    /* Envio de la columna */
    MPI_Bcast(&A[0][k], 1, colu, 0, MPI_COMM_WORLD);
    MPI_Type_free(&colu);
}
    }
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Método o algoritmo	Describe al algoritmo de solución del problema planteado en forma adecuada (2 pts)	Algoritmo con algunos errores que no afectan el resultado (1.5 pts).	Algoritmo con errores que afectan minimamente el resultado (0.5 pt).	Algoritmo con errores, que afectan significativamente el resultado (0 pts)
Resultados	Solución correcta usando un método adecuado (1 pt)	Errores mínimos en el método que no afectan el resultado (0.6 pts)	Errores en el método que afectan el resultado (0.3 pts)	No aplica el método ni llega a la solución correcta (0 pts).
Optimización	Solución original y optimizada (1 pt)	Solución parcialmente optimizada (0.6 pts)	Solución original pero no optimizada (0.3 pts)	Resultado encontrado no está optimizado (0 pts).