Computación Paralela y Distribuída

Exámen Parcial
Pregrado
2023-I
Profesor: José Fiestas
1.01

Indicaciones específicas:

- Esta evaluación contiene 10 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. Cualquier fuente externa debe ser citada y se corregirá, según el enunciado, lo resuelto por el alumno.

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 puntos)
Las siguientes gráficas representan el speedup vs. nodos y la eficiencia vs. nodos de un código de N-cuerpos que utiliza una libreria especialmente diseñada para problemas de concurrencia y paralelismo en C++ (HPX)

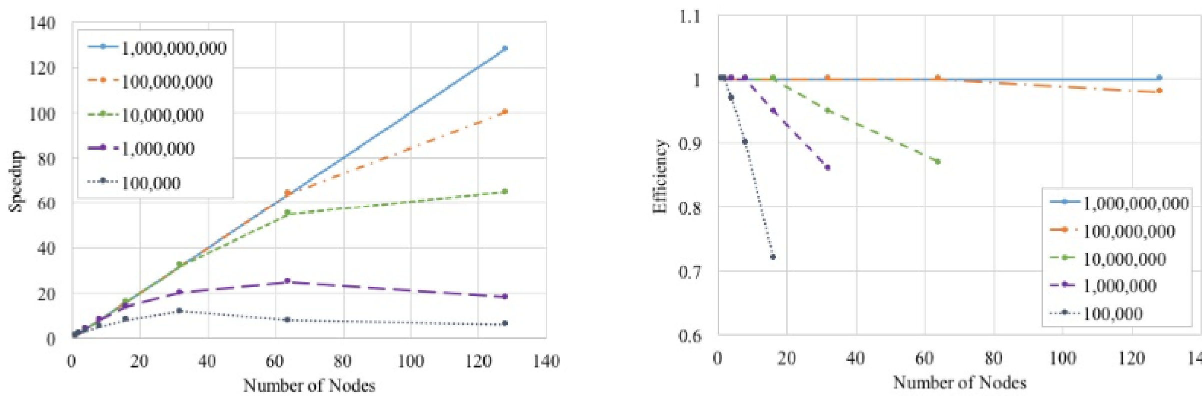

Figura 1: Izquierda: speedup vs. nodos. Derecha: eficiencia vs. nodos , para distinto N

a) Se obtiene con HPX un tiempo de cómputo $T_{comp} = O(log(1/\epsilon)N/p)$, con una tolerancia ϵ. Defina el tiempo de ejecución y speedup teóricos del problema de N-cuerpos, considerando un tiempo de comunicación $T_{comm} = O(log(p))$**(1 pt)**
Respuesta: $T_{ejec} = O(log(1/\epsilon)N/p + log(p))$. Ya que este problema tiene una complejidad secuencial O(n^2), el speedup $S = \frac{N^2}{log(1/\epsilon)N/p+log(p)}$

b) Encuentre una expresión para la eficiencia y compare cualitativamente los resultados con la gráfica mostrada de eficiencia ¿Qué factores generan una diferencia entre la data experimental y la teórica? Argumente sus respuestas.**(1 pt) Respuesta:** $E = \frac{O(N^2/p)}{O(N/p+log(p))} = \frac{1}{1/N+plog(p)/N^2}$. Manteniendo N constante, la curva experimental corresponde a la curva teorica, luego de aplicar factores de proporcionalidad

c) ¿En que casos se obtiene un speedup ideal? ¿Cómo se comporta la eficiencia en este caso?**(1 pt)**
Respuesta: $S = O(p)$, cuando $n \propto \sqrt{plog(p)}$. Para la eficiencia, en el denominador hay dos términos que derivan en la misma relación

d) Comente sobre el tipo de escalabilidad del problema utilizando las gráficas. Note que la curva con el mayor valor de N se superpone a la curva ideal **(1 pt)**
Respuesta: Se observa escalabilidad fuerte en la curva de mayor N (celeste) en ambas gráficas. Un analisis de escalabilidad debil, requiere conservar la relacion $n \propto \sqrt{plog(p)}$

e) ¿Cambia el resultado si $T_{comm} = O(p)$? **(1 pt)**
Respuesta: En este caso, en el denominador ámbos términos derivan en $n \propto p$. Esto cambia el comportamiento de las curvas de speedup y eficiencia. Ya no corresponderia teoria a experimento, lo que permite concluir que la comunicacion en este problema está optimizada.

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 puntos)
Dada la siguiente función que determina los valores de un array por debajo de la desviación estandar

```
int calculo(float A[], int n){
    int i, cont=0;
    float promedio=0, suma_cuadrados=0;
    for (i=0;i<n;i++)
        promedio += A[i];
    promedio = promedio/n;
    for (i=0;i<n;i++)
        suma_cuadrados+=pow(A[i]-promedio,2);
    desviacion=suma_cuadrados/(n-1);
    desviacion= sqrt(desviacion);
    for (i=0;i<n;i++)
        if (A[i]<desviacion)
            cont++;
    return cont;
    }
```

a) Formule las directivas MPI de comunicación necesarias para paralelizar la función calculo(). Especifique todos sus argumentos. Asuma que el arreglo A esta almacenado inicialmente en el maestro. Considere $n = 2^k$, $p = 2^r$; $k, r \in \mathbb{N}$ **(3 pts)**

```
int calculo(float A[], int n) {
int i, cont, cont_loc=0, p;
float promedio, suma_loc=0;
float* Aloc;
MPI_Comm_size(MPI_COMM_WORLD, &p);
Aloc = (float*) malloc(n/p*sizeof(float));
MPI_Scatter(A, n/p, MPI_FLOAT, Aloc, n/p, MPI_FLOAT, 0,
   MPI_COMM_WORLD);
for (i=0;i<n/p;i++) suma_loc += Aloc[i];
MPI_Allreduce(&suma_loc, &promedio, 1, MPI_FLOAT,
   MPI_SUM, MPI_COMM_WORLD);
promedio = promedio/n;
for (i=0;i<n/p;i++) suma_cuadrados_loc += pow(Aloc[i]-
   promedio,2);
MPI_Allreduce(&suma_cuadrados_loc, &suma_cuadrados, 1,
   MPI_FLOAT, MPI_SUM, MPI_COMM_WORLD);
desviacion=suma_cuadrados/n-1;
desviacion= sqrt(desviacion);
for (i=0;i<n/p;i++)
    if (Aloc[i] > desviacion)  cont_loc++;
MPI_Reduce(&cont_loc, &cont, 1, MPI_INT, MPI_SUM, 0,
   MPI_COMM_WORLD);
```

```
    free(vloc);
    return cont;
}
```

---

b) Halle una expresión para $T_p(n,p)$, $S(n,p)$, así como el límite del speedup y eficiencia cuando n tiende a infinito. Considere tiempos de comunicacion colectiva $O((p-1)(t_s + xt_b))$, donde t_s es el tiempo de latencia, t_b es el tiempo de envío de un byte, y x es el tamaño del mensaje enviado. ¿Considera el algoritmo escalable? Argumente sus respuestas **(2 pts)**

Scatter: el proceso 0 envía un mensaje de n/p elementos a cada uno de los demás procesos. Por tanto:

$$(p-1)(t_s + \frac{n}{p}t_w)$$

Reduce: el proceso 0 recibe un mensaje de un elemento de cada uno de los demás procesos,y suma los elementos, es decir:

$$(p-1)(t_s + t_w) + (p-1)$$

Allreduce: se puede realizar mediante una operación reduce sobre el proceso 0, seguida de un broadcast del resultado. Para el broadcast, suponemos que el proceso 0 envía un mensaje de un elemento a cada uno de los demás procesos.

$$2(p-1)(t_s + t_w) + 2(p-1)$$

Bucles de calculo

$$\sum_{i=0}^{n/p-1} 1 + \sum_{i=0}^{n/p-1} 2 + \sum_{i=0}^{n/p-1} 2 = \frac{5n}{p}$$

El tiempo de ejecución paralelo es la suma de lo anterior, es decir:

$$T(n,p) = O(4pt_s + (n+3p)t_w + 2p + 5n/p)$$

Por otra parte, el tiempo secuencial es:

$$T(n) = \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{n-1} 2 + \sum_{i=0}^{n-1} 2 = 5n$$

De donde tenemos

$$S(n,p) = \frac{5n}{4pt_s + (n+3p)t_w + 2p + 5n/p}$$

$$lim_{n\to\infty} S(n,p) = \frac{5}{t_w + 5/p}$$

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`).

3. (5 puntos)

3.1) Enviar un vector de tamaño n de un proceso a otro demora $t(n) = (\alpha + n \cdot \beta)$, donde α de la latencia, y β el ancho de banda.

a) Determine $t_p(p, q)$ al calcular

$$s = \sum_{j=1}^{q \cdot 2^N} y_j$$

$y_j \in \mathbb{R}^n$, $j = 1, ...q \cdot 2^N$

Con $p = 2^N$ procesos.

Considere que la suma de n números tiene un tiempo secuencial $n \cdot t_a$ (donde t_a es el tiempo empleado en hacer una suma)**(1 pt)**

Solución: $T_s = nt_a$, $T_p = qt_a + p(\alpha + \beta) + pt_a$

qt_a es el cálculo por proceso, $p(\alpha + \beta)$ es la comunicación de las sumas parciales al maestro, pt_a es la suma de los resultados parciales en el maestro.

En esta solución no se incluye el particionamiento, que sería $p(\alpha + q\beta)$, ya que el enunciado se refiere en especifico a la sumatoria, pero también podría incluirse.

b) Determine la eficiencia E(p, q) **(1 pt)**

Solución:

$S = \frac{nt_a}{qt_a + p(\alpha+\beta)t_a + pt_a}$ $E = \frac{n/2^n t_a}{qt_a + p(\alpha+\beta) + pt_a} = \frac{1}{1 + p/q(1+\alpha+\beta)}$

c) ¿En que casos se obtiene una eficiencia $E \geq 0{,}5$? ¿Se comprueba la relación obtenida? $E = \frac{1}{1+p/q(1+\alpha+\beta)} > 0{,}5$, o $p < q/(1 + \alpha + \beta)$

Determine E para $\alpha = 1000$, $\beta = 10$, $N = 6$, q=10,100,1000. **(1 pt)**

Para q=10, E=0.016

Para q=100, E=0.622

Para q=1000, E=0.993

Se comprueba la relación obtenida

3.2) Proponga dos formas de paralelizar la siguiente función y escoja la forma más eficiente. Argumente su respuesta **(2 pts)**

```
int bucles_for(int** a, int** b, int m, int n)
{
    int i,j,k,temp,s=0;
    for (i=0; i<m; i++) {
        for (j=0; j<n; j++) {
            temp=0;
            s += a[i][j];
            for (k=0; k<n; k++) {
                temp += a[i][k] * a[k][j];
            }
            b[i][j] = temp;
        }
    }
    return s;
```

```
}
```

Solución:
Primer bucle: #pragma omp parallel for reduction(+:s) private(j,k,temp)
Segundo bucle: #pragma omp parallel for reduction(+:s) private(k,temp)
Tercer bucle: #pragma omp parallel for reduction(+:aux)
La forma más eficiente consiste en paralelizar el bucle más externo, pues se produce una menor sobrecarga debida a la activación y desactivación de hilos, y también se reducen los tiempos de espera debidos a la sincronización implícita al final de la directiva.

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 puntos)
Considere la expresión para el tiempo en paralelo del algoritmo de quicksort:
$$T_p = O(\frac{n}{p}log\frac{n}{p}) + O(\frac{n}{p}log(p))$$

a) Si el tiempo secuencial es O(n log(n))¿Cuál de estos términos representa el tiempo de cómputo en paralelo? ¿Cuál representa el tiempo de comunicación en paralelo? Complete la expresión para T_p con los factores que describen el tiempo de comparación local (t_c), la latencia (α) y el tiempo de envío de un byte (β). **(1.5 pt)**
$$T_p = t_c\frac{n}{p}log\frac{n}{p} + (\beta\frac{n}{p} + \alpha)log(p))$$

b) Quicksort en paralelo cuenta con un tercer término debido a la recursividad del algoritmo $O(log^2(p)(\alpha + \beta))$. Considérelo para encontrar una expresión para la eficiencia **(1.5 pt)**
$$T_p = t_c\frac{n}{p}log\frac{n}{p} + (\beta\frac{n}{p} + \alpha)log(p) + log^2(p)(\alpha + \beta)$$
Calcule $E = O(\frac{T_s}{pT_p})$

c) Determine la condición de isoeficiencia (eficiencia constante)¿Cómo afectan a esta condición los valores de t_c, α y β ? Considere valores de β entre 1 y 10 , y de α entre 1 y 100. Mantenga t_c constante (e.g. =1) ¿A qué conclusión puede llegar con respecto a la escalabilidad del quicksort? **(2 pt)**
Aqui se obtienen varias condiciones, según la ecuación anterior:
$\alpha plog(p) \propto t_c nlog(n)$, $(\alpha+\beta)plog^2(p) \propto t_c nlog(n)$, $\beta log(p) \propto t_c log(n)$
La eficiencia crece con α y β, al ser t_c constante.
El término dominante es $\beta log(p) \propto log(n)$

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 mínimamente 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`).