Computación Paralela y Distribuída

Exámen Parcial
Pregrado
2023-II
Profesor: José Fiestas
1.01

Indicaciones específicas:

- Esta evaluación contiene 10 páginas (incluyendo esta página) con 4 preguntas.
- 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)

1. (5 puntos)
   Las siguientes gráficas representan el speedup vs. CPUs y el tiempo de ejecución en paralelo vs. CPUs de dos modelos FFT (Fast Fourier Transform)

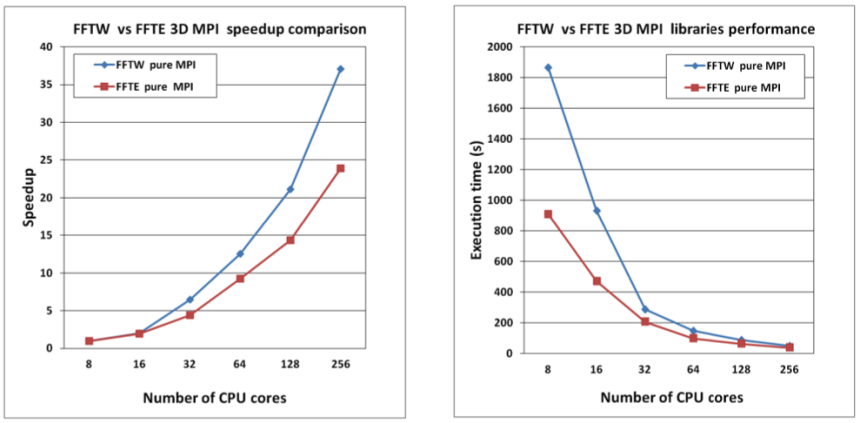

Figura 1: Izquierda: speedup vs. CPUs. Derecha: Tiempo de ejecución vs. CPUs , para N = 256^3. De *Sunderland et al., An Analysis of FFT Performance in PRACE Application Codes, 2012*

a) Estime la cantidad óptima de procesos utilizando la grafica ¿Dónde ubicaría la curva de tiempos de los mismos modelos pero con $n = 1024^3$? Puede hacer un dibujo aproximado de las curvas pedidas **(1 pt)**
Respuesta: p_{opt} se ubica en aproximadamente 256 CPU cores, como se observa en la gráfica. La curva de tiempos para N mayor se movería arriba a la derecha, siendo p_{opt} en ese caso mayor a 256.

b) Se obtiene un tiempo de cómputo en paralelo $T_{comp} = O(nlog(n)/p)$. Defina el tiempo de ejecución y speedup teóricos, considerando un tiempo de comunicación $T_{comm} = O(\frac{n}{p}log(p))$. FFT secuencial es $O(nlog(n))$**(1 pt)**
Respuesta: $T_{ejec} = O(nlog(n)/p + n/plog(p))$. Ya que este problema tiene una complejidad secuencial O($nlog(n)$), el speedup $S = \frac{nlog(n)}{nlog(n)/p+n/plog(p)}$

c) Determine la granularidad G del problema ¿Qué tipo de granularidad se muestra en la gráfica? **(1 pt)**
Respuesta: $G = \frac{T_{comp}}{T_{comm}} = \frac{nlog(n)/p}{n/plog(p)} = \frac{log(n)}{log(p)}$, es decir, para p pequeño (por debajo de p_{opt}) y n constante, como en la gráfica, observamos granularidad gruesa. No se observa granularidad fina en la gráfica

d) Determine la relación entre n y p para lograr un speedup ideal. Según esta relación, ¿cuantos procesos se necesitan si se duplica la cantidad de elementos? **(1 pt)**
Respuesta: Obtenemos un sppedup $S = \frac{nlog(n)}{nlog(n)/p+n/plog(p)} = p\frac{log(n)}{log(p)}$. El speedup ideal es O(p), se logra bajo la condición de que $log(n) \propto log(p)$

e) ¿Que tipo de escalabilidad se observa en las gráficas mostradas? ¿Cual es la condición de eficiencia constante? **(1 pt)**

Respuesta: Se observa escalabilidad fuerte, ya que n es constante. la condición de eficiencia constante es la misma que de speedup ideal: $log(n) \propto log(p)$

Argumente sus respuestas, la mitad del puntaje por pregunta corresponde a la justificación del resultado

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 (`2 pt`)	Errores mínimos en el método que no afectan el resultado (`1.5pts`)	Errores en el método que afectan el resultado (`0.5 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)
El siguiente codigo implementa comunicación punto a punto en un conjunto de procesos

```
int id = 0, nproc = 0;
int sendValue = 0, receivedValue = 0;
MPI_Status status;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &id);
MPI_Comm_size(MPI_COMM_WORLD, &nproc);

if (nproc > 1) {
    sendValue = id;
    if ( id%2!=0 ) {
        MPI_Send(&sendValue, 1, MPI_INT, id-1, 1,
           MPI_COMM_WORLD);
        MPI_Recv(&receivedValue, 1, MPI_INT, id-1, 2,
           MPI_COMM_WORLD, &status);
    } else {
        MPI_Recv(&receivedValue, 1, MPI_INT, id+1, 1,
           MPI_COMM_WORLD, &status);
        MPI_Send(&sendValue, 1, MPI_INT, id+1, 2,
           MPI_COMM_WORLD);
    }
    printf("Proceso %d de %d envia %d y recibe %d\n",id,
        nproc, sendValue, receivedValue);
} else if (!id) {
    printf("\nUtilice este codigo para mas de un proceso
       .\n\n");
}

MPI_Finalize();
```

a) Describa el funcionamiento del código al ejecutar sobre un numero **par** e **impar** de procesos **(1.5 pt)**
Respuesta: Se realiza un envío/recibo de un entero entre pares de procesos. Es decir, procesos 0 y 1 intercambian el mismo dato (su identificador), procesos 2 y 3 hacen lo mismo, etc. Si la cantidad de procesos es impar, aparece un error de ejecución

b) ¿Genera este código un bloqueo mutuo entre procesos? Considere los casos de un total de procesos par e impar **(1.5 pt)**
No aparece bloqueo entre procesos. El error en comunicación para procesos impares es porque no tiene su procesos par para completar la comunicación.

c) Describa si existirá algún cambio en la ejecución en paralelo en caso de usar un envío sincrónico (Ssend) o buffered (Bsend) **(1 pt)**

el funcionamiento es el mismo ya que la comunicación se restringe a pares de procesos

d) ¿Qué sucede si modificamos todos los identificadores de proceso (tag) por MPI_ANY_TAG? Es decir, se recibe el mensaje sin restricción de tag **(1 pt)**
Las operaciones de comunicación ya no estarán definidas entre pares de procesos y habrían errores de comunicación

Argumente sus respuestas, la mitad del puntaje por pregunta corresponde a la justificación del resultado. Es suficiente analizar el código dado sin necesidad de compilarlo, pero también puede hacerlo.

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 (`2 pt`)	Errores mínimos en el método que no afectan el resultado (`1.5pts`)	Errores en el método que afectan el resultado (`0.5 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)
Genere un PRAM que reproduzca el siguiente patrón de comunicación:

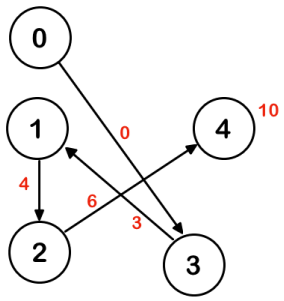

Figura 2: Ejemplo de comunicación random con p=5. El mensaje está en rojo.

El maestro genera un rank aleatorio y envía su nombre (0) a este proceso. Luego, éste último proceso añadirá su rank al dato recibido, y generará otro rank aleatorio para enviar este resultado. De esta manera cada proceso envía un mensaje a otro proceso.

Cada proceso no puede enviar un mensaje a si mismo ni al maestro. Procesos que ya han recibido un mensaje, no recibirán otro. El último proceso no envía mensajes.

Permita un mínimo de 3 y un máximo de 50 procesos.

a) Decida que directivas de comunicación debe usar y agrégelas al PRAM. No necesita la directiva completa pero si los argumentos relevantes, como qué se envía y a donde se envía. E.g. *send M[i] to j* **(1.5 pt)**
Respuesta:
Entrada: M[i]=0, con $1 < i < p$, contando con p procesos.
Salida: mensaje final M[p] en el último proceso p

```
1. k=p
    for i=1 to p
        M[i]=0 // almacena el mensaje a enviar
    if(nproc<3 or nproc>50) abort() // total de procesos
       debe ser entre 3 y 50
    time0=time()
2. for i = 1 to p pardo {
    if(rank==1)
            id_next= random entre 2 y p
            M[i] += rank
             send  M[i] to id_next
            id_prev= rank
    else{
        if (i!=j && M[i]==0) {
            receive  M[i] from id_prev
             id_next= random entre 2 y p
             M[i] += rank
```

```
                send  M[i] to id_next
                id_prev=rank
                }  // if
        }  // while
        } // for
3.  if (p==1){
        time=time() - time_0
        print(M[p])
}
```

b) Discuta sobre la posibilidad de usar MPI_ANY_SOURCE (el proceso origen no está definido, puede ser cualquiera) en las directivas de comunicación **(1 pt)**
Respuesta: MPI_ANY_SOURCE es posible ya que la comunicación es siempre única entre un par de procesos

c) Formule la complejidad de comunicación del algoritmo generado, utilizando las dependencias en latencia y ancho de banda vistas en clase **(1.5 pt)**
Respuesta: Leyendo el PRAM, las operaciones son de comuncación punto a punto, $T_{comm} = (\alpha + x\beta)$, donde x es el mensaje que se envía, es decir 1. Esta operacion se repite p veces, por lo que la complejidad final es $T_{comm} = p(\alpha + \beta)$

d) ¿Considera necesario usar comunicación colectiva en este problema? **(1 pt)**
Respuesta: Se podria usar un BCast para informar a todos los procesos que ya enviaron pero se logra el PRAM sin ello

Opcional (1 pt extra): el mensaje consistirá en un array/vector del tamaño del mensaje en el caso anterior. Es decir, en el ejemplo, el proceso 0 inicia con un array/vector vacío y el proceso 4 recibirá un array de 10 elementos
Respuesta: Esta operación se repite p veces, por lo que la complejidad final es $T_{comm} = p(\alpha + p\beta)$

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 (2 pt)	Errores mínimos en el método que no afectan el resultado (1.5pts)	Errores en el método que afectan el resultado (0.5 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)
En clase vimos el problema de la multiplicación matriz-vector. La complejidad teórica, en el caso del particionamiento por filas es

$$T_p = t_c \cdot (n^2/p) + t_s \cdot log(p) + n \cdot t_w$$

y en el caso del particionamiento por bloques

$$T_p = t_c \cdot (n^2/p) + 2t_s \cdot log(\sqrt{p}) + t_w \cdot (n/\sqrt{p})$$

donde t_c es el tiempo de multiplicación/suma, t_w el ancho de banda, y t_s el tiempo de latencia.
Determine en cada caso (filas, bloques):

a) La dimensión de los subdominios de la matriz que recibe cada proceso según la fórmula de T_p. Grafique el particionamiento para mayor entendimiento **(1.5 pts)**
Respuesta:
a) $n/p * n = n^2/p$, **b)** $n/\sqrt{p} * n/\sqrt{p} = n^2/p$

b) La condición de escalabilidad entre n y p para cada caso **(1.5 pts)**
Respuesta:
a) $n \propto p$
b) $n \propto \sqrt{plog(\sqrt{p})}$

c) ¿Cuál de los métodos muestra una granularidad mayor? Comente las ventajas/desventajas de éste método**(1 pt)**
Respuesta:
a) $G = \frac{n^2}{p(n+log(p))}$
b) $G = \frac{n^2}{p(n/\sqrt{p}+log(\sqrt{p}))}$
el particionamiento por bloques (b) muestra mayor granularidad (denominador es más pequeño), lo que es una ventaja al tener un overhead (comunicacion) menor

d) ¿Cuál de los métodos considera más escalable? **(1 pt)**
Respuesta: como calculado en a)
a) $n \propto p$
b) $n \propto \sqrt{plog(\sqrt{p})}$
El segundo caso es mas escalable ya que cuando n crece se requieren menos procesos para mantener la eficiencia constante

Argumente sus respuestas, la mitad del puntaje por pregunta corresponde a la justificación del resultado.

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 (2 pt)	Errores mínimos en el método que no afectan el resultado (1.5pts)	Errores en el método que afectan el resultado (0.5 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).