Interrogación 1 del curso IMT2112 Algoritmos Paralelos en Computación Científica

Curso: IMT2112 Algoritmos Paralelos en Computación Científica
Universidad: Pontificia Universidad Católica de Chile
Fecha: viernes 27 de septiembre de 2019
Profesor: Elwin van 't Wout

Instrucciones:

- Solo se permite el uso de lápiz;
- No se recorregirán pruebas escritas con lápiz mina;
- Las respuestas podrían ser escritos en castellano o inglés;
- La nota de la interrogación tiene una ponderación de 20 % en la nota final;
- **TIEMPO DE LA PRUEBA: 2 HORAS**.

---

1. En 1965 el científico Roger Moore hizó una proyección sobre el futuro crecimiento de computadores, que hoy en día se conoce como la Ley de Moore.

   a) **[1 pt.]** ¿Cual es la versión original de la Ley de Moore?
   El número de transistores por chip se dobla cada 2 años (o 18 meses).

   b) **[1 pt.]** Hace una década los computadores ya no siguen la versión original de la Ley de Moore. Explica una razón por qué.
   La escala de los transistores es tan pequeño que la física es distinta. Ya no se puede enfriar los chips.

2. **[2 pts.]** Los computadores modernos pueden ser modelados con una arquitectura de Von Neumann. ¿Cuales son las dos características principales de este arquitectura?
   Operaciones son ejecutados en ciclos de fetch-execute-store. Instrucciones se manejan como datos.

3. Supongamos que el computador tiene una velocidad de reloj de 2.5 GHz y un *pipeline* de 6 pasos. Además, todos los datos necesarios ya se encuentren disponible en los registros, incluso $x(0)$.

   a) **[1 pt.]** Ejecutamos la operación $x(n) = x(n-1) + n$ para $n = 1, 2, 3, 4$. ¿Cúantos segundos se require para la instrucción?
   Un ciclo demora 0,4 ns. La instrucción tiene 4 operaciones dependientes, entonces 24 ciclos. En total es 9,6 ns.

   b) **[1 pt.]** Ejecutamos la operación $x(n) = x(\lfloor n/2 \rfloor) + n$ para $n = 1, 2, 3, 4$. ¿Cúantos segundos se require para la instrucción?
   La instrucción tiene 3 operaciones dependientes y 1 independiente que puede ser ejecutado en paralelo, entonces 18 ciclos. En total es 7,2 ns.

4. Supongamos que tenemos una memoria con capacidad de 12 datos y un memoria que tiene capacidad de 4 datos. Los datos $x_0, x_1, \ldots, x_9$ son almacenados en los primeros 10 plazos del memoria grande. Ejecutamos la operación $x_0 = x_6(x_5x_6 + x_9)$ en lo cual los resultados parciales son guardados en el variable x_0.

   a) **[2 pts.]** El mapeo de caché es directo. ¿Cúantos éxitos de caché (*cache hits*) hay? El resultado debe ser respaldado con un dibujo del flujo de datos.
   Dos veces el x_0. El x_6 fuera eliminado por el operador.

   b) **[1 pt.]** El mapeo de caché es asociativo. ¿Cúantos éxitos de caché (*cache hits*) hay?
   Dos veces el x_0 y una vez el x_6.

5. **[1 pt.]** Explica el concepto de concurencia. En tu respuesta, incluye la unidad de concurencia.
   La unidad es bit. Concurencia es la cantidad de datos que está en transmisión entre los partes de la jerarquía de memoria.

6. Tenemos la operación V^TAV con $V \in \mathbb{R}^{n \times k}$.

   a) **[1 pt.]** Calcula la intensidad aritmética.
   El número de datos es $n(k+n)$ y el número de operaciones $2kn(n+k)$. La intensidad aritmética es operaciones por datos: $2k$.

b) **[1 pt.]** Explica si se espera una escalabilidad débil buena.
En la escalabilidad débil el tamaño del problema aumenta con el número de procesadores. En este case, ambos k y n aumentan. Dado que hay, en promedio, $2k$ operaciones por dato, se espera una eficiencia alta porque se puede aprovechar la cantidad grande de operaciones en comparación con el número de datos.

7. **[1 pt.]** Supongamos que un algoritmo tiene un camino crítico de 8 operaciones y un total de 40 operaciones. Según la Ley de Amdahl, ¿cual es el *speedup* paralela máxima? Justifique la respuesta.
La fracción secuencial es 8/40=0,2. El speedup *máximo es 1/0,2=5.*

8. **[1 pt.]** En la taxonomía de Flynn, un procesador puede ser SIMD. Da el nombre completo de SIMD y da un ejemplo de un procesador de este tipo.
Single Instruction Multiple Data. Una tarjeta gráfica o un computador vectorial.

9. Tenemos un tensor $T \in \mathbb{R}^{m\times n\times k}$ y un clúster de p procesadores. La operación $S = T\mathbf{x}$ con $\mathbf{x} \in \mathbb{R}^k$ es definido como $S_{ij} = \sum_{\ell=1}^{k} T_{ij\ell}x_\ell$. El vector $\mathbf{x}$ es siempre particionado sobre el clúster.

   a) **[2 pt.]** El particionamiento es sobre la primera dimensión de T, en partes iguales: $[0, m/p], \ldots, [m(p-1)/p, m]$. Explica como se puede implementar esta multiplicación tensorial en paralelo. En la respuesta, incluye cómo particionas S y describe el algoritmo en términos de operaciones colectivas.
   Primero un allgather *sobre el vector* x. *Después una multiplicación local. Si el particionamiento de S es sobre las columnas, adicionalmente hay un* gather *sobre las columnas.*

   b) **[1 pt.]** Calcula el tiempo de ejecución de la operación. La respuesta puede depender de latencia α, ancho de banda inverso β y tiempo de cálculo inverso γ.
   El allgather *sobre p procesadores de k elementos toma* $\log_2(p)\alpha + (p-1)\frac{k}{p}\beta$ *tiempo y la multiplicación local* $(2mnk/p)\gamma$.

   c) **[1 pt.]** Explica como se puede mejorar la eficiencia paralela.
   El algoritmo es en base un conjunto de multiplicaciones matriz por vector. Por lo tanto, un particionamiento en bloques puede mejorar la escalabilidad.

10. **[2 pts.]** Qué significa una condición de carrera (*race condition*) y como se puede evitar este en OpenMP.
Esta condición significa que el resultado del algoritmo depende del orden de ejecución de los distintos hilos. Se puede resolver este problema con una reestructuración del código, bareras y locks.