Programación Paralela
2020-II
Exámen Final

Prof. José Fiestas

Indicaciones específicas:
Duración: 120 minutos
Número de preguntas: 6
Solo está permitido el uso de material de clase durante el exámen. Ningún otro material.
La entrega se hará por Classroom (pdf o foto)
No necesita enviar códigos solución a los ejercicios, pero no está prohibido

---

La rúbrica en cada pregunta tiene dos puntos escenciales:
- Desarrollo analítico o numérico de los ejercicios (70%)
- Argumentación de resultados obtenidos (30%)

---

1) Dadas dos matrices $n \times n$ distribuidas en un hipercubo de n^3 procesos ($n = 2^q$), de dimensión q. Si los procesos son numerados como (l,i,j) ¿Qué es cierto para el siguiente algoritmo de multiplicación de matrices? Argumente su(s) respuesta(s)

Entrada: $A[i,j]$ en $M_{(j,i,0)}$, $> B[i,j]$ en $M_{(i,0,j)}$
Salida: $C[i,j]$ en $M_{(0,i,j)}$, tal que $C = A \cdot B$
1. $A[i,l] \cdot B[l,j]$ en proceso $M_{(l,i,j)}$, para $0 \leq i,j,l \leq n-1$
2. $D[l,i,j] := A[i,l] \cdot B[l,j]$
3. $D[0,i,j] := \sum_{l=0}^{n-1} D[l,i,j] \rightarrow C[i,j]$

---

a) n^3 procesos, $T(n) = O(log(n))$
b) $T(n) = O(n^3/p \cdot log^2(n))$, para $p \leq n^3$ procesos
c) n^3 procesos, $W(n) = O(n^3 \cdot log(n))$, $T(n) = O(log(n))$
d) $T(n) = O(n^3/p \cdot log^2(n) + log(n))$, para $p \leq n^3$ procesos
(4 pt)

2) Responda a las siguientes preguntas con respecto al pseudocódigo de la suma de prefijos mostrado abajo.
a) Describa el tiempo ($T(n)$) y trabajo ($W(n)$) en cada uno de los siguientes bloques del pseudocódigo
1. condicional
2. primer bucle **for**
3. recursión
4. segundo bucle **for**

y determine $T(n)$ y $W(n)$ total
b) ¿Puede aplicarse un modelo PRAM a este algoritmo?
c) Un algoritmo se considera WT-óptimo cuando $W(n) = T^*(n)$, siendo $T^*(n)$ el algoritmo secuencial óptimo. ¿Es éste algoritmo WT-óptimo ?

Entrada: $x_1, ..., x_n$, $(n = 2^k)$
Salida: $s_1, ..., s_n$, $s_i = x_1 \circ x_1 \circ ... \circ x_i \forall i \in \mathbb{N}$, donde $\circ$ es una operación asociativa

```
begin
  if n = 1 then
    s_1 := x_1
  else
    for i ∈ {1, ..., n/2} pardo
      y_i := x_{2i-1} ∘ x_{2i}
    od
    (z_1, ..., z_{n/2}) := Prefix–Sums(y_1, ..., y_{n/2})
    for i ∈ {1, ..., n} pardo
      if i = 1 then
        s_1 := x_1
      else
        if i mod 2 = 1 then
          s_i := z_{(i-1)/2} ∘ x_i
        else
          s_i := z_{i/2}
        endif
      endif
    od
  endif
end
```

(4 pt)

3) Se utiliza una tarjeta gráfica NVidia Kepler GK180, con 1024 threads per block y hasta 65535 blocks (en 1 dimensión) para modelos astrofísicos de evolución galáctica. (**4 puntos**)

- si cada thread puede procesar una operacion de coma flotante (FLOP), ¿Cuantas estrellas pueden ser simuladas teóricamente en esta tarjeta en forma simultánea, si cada una necesita por iteración una cantidad de FLOPs definida por la fórmula $a_0 + G * (m_1 * m_2)/(x^2 + y^2 + z^2)$?
- ¿Si el tiempo de cálculo por operación es $1x10^{-6}$ segundos, cuál sería el tiempo de procesamiento luego de un millón de iteraciones ? ¿Cual sería el tiempo si el código se ejecutaría en forma secuencial?
- ¿Cual sería una combinación óptima de threads x blocks para declarar el kernel de un modelo de n galaxias de 1000 billones de estrellas cada una? **Fuerza <<blocks_per_grid, threads_per_block, >>**

4) Encuentre dos errores y modifique el código para corregirlos. Argumente sus respuestas

```
[...]
int    i,tid;
double sum = 0.0;
double part;

#pragma omp parallel shared (part, tid) private(sum)
{
  part = 0.0;
  tid = omp_get_thread_num();

  for (i=0; i < 42; i++) {
    part += calcFoo(tid, i);
  }
  #pragma omp atomic
  sum += part;
}

printf("sum: %lf\n", sum);
[...]
```

(**2 puntos**)

5) Encuentre dos errores y modifique el código para corregirlos. Argumente sus respuestas

```
[...]
long localmax;

if (rank != 1) {
  for (int i = 0; i < num_workers-1; i=i+2) {
      MPI_Irecv(&localmax, 1, MPI_LONG, i,  123, MPI_COMM_WORLD, &request);
  }
  MPI_Wait(&request, &status);
  globalmax = findMaxValue(localmax, globalmax);
  doImportantCalculation(globalmax);

} else {
  MPI_Isend(&localmax, 1, MPI_LONG, 1, 123, MPI_COMM_WORLD, &request);
  someFooBarLongCalc();
  MPI_Wait(&request, &status);
}
[...]
```

(**2 puntos**)

6) Realice un análisis de escalabilidad del algoritmo secuencial/paralelo mostrado. I.e. determine la eficiencia y evalúe su variabilidad. Argumente sus respuestas.
Secuencial:

```
len = 0.0;
for (i = 0; i < N; i++) {
  len += sqrt(matrix[i][X] * matrix[i][X] +
              matrix[i][Y] * matrix[i][Y] +
              matrix[i][Z] * matrix[i][Z]);
}
```

Paralelo:

```
len = 0.0;
// each parallel task has its own matrix of size N/p
for (i = 0; i < N/p; i++) {
  len += sqrt(matrix[i][X] * matrix[i][X] +
              matrix[i][Y] * matrix[i][Y] +
              matrix[i][Z] * matrix[i][Z]);
}

if (myID == 0) {
  for (i = 1; i < p; i++) {
    len += receive from i
  }
} else {
  send len to 0
}
```

Considere un problema de tamaño **N** , y cantidad de procesos **p**
(4 pt)