Pensando en paralelo

Joel A. Trejo-Sánchez, Miguel A. Uh-Zapata, Francisco J. Hernández-López

Centro de Investigación en Matemáticas

{joel.trejo,angeluh,fcoj23}@cimat.mx

July 10, 2018

Bosquejo del curso

- Programación multi-procesador (10 de julio)
- Programación con paso de mensajes MPI (11 de julio)
- Programación en GPU (12 de julio)
- Sesión de problemas abiertos (13 de julio)

Programación multi-procesador

En general vamos a utilizar el modelo de memoria compartida.
Trabajaremos principalmente con OpenMP.

- Instalación y configuración
- Introducción a la programación concurrente en memoria compartida
- Dos ejemplos triviales de programación concurrente
- Programación de coloreado en gráficas

Linux

- Compilador gcc, g++
- Ya se incluyen las librerías OMP
- Compilar agregando la libreria omp (-fopenmp)

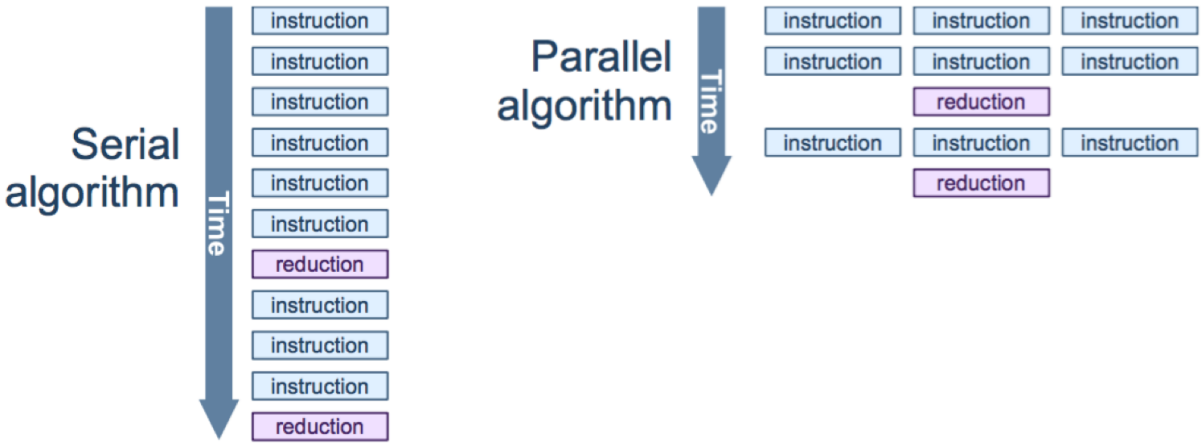
Windows

- CodeBlocks
- MinGW
- Configurar MinGW en CodeBlocks
  - Settings – Compiler – Other setting Optios –fopenmp
  - Settings– Compiler–Linker Settings – C:/MinGW/bin/libgomp-1.dll

Servidor K20 CIMAT-Mérida

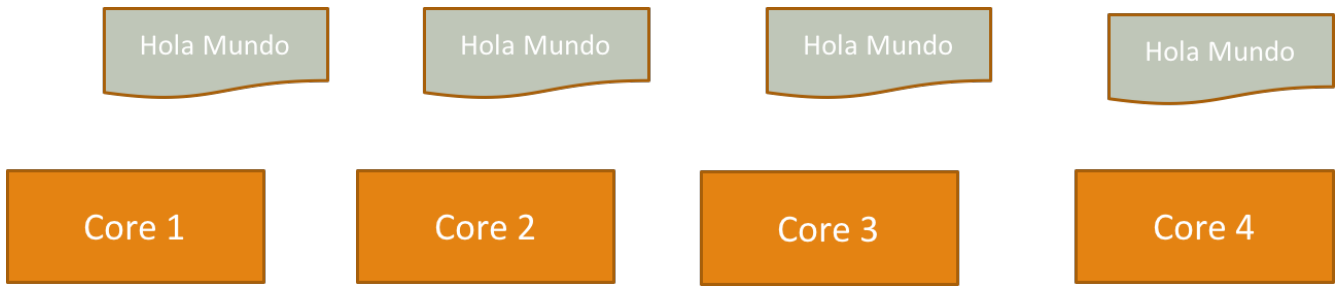
- verano2018@10.16.120.200
- cimat2018
- scp programa.cpp verano2018@10.16.120.200:/home/usuario

Introducción



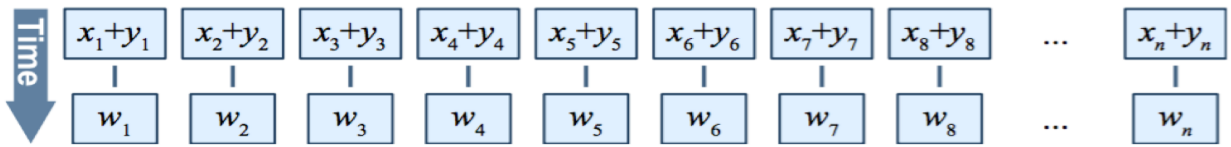
Hola Mundo

En este programa veremos como hacer el clásico "Hola Mundo" pero de forma concurrente.
Puede ver los ejemplos en https://github.com/trejoel/VeranoCIMAT2018



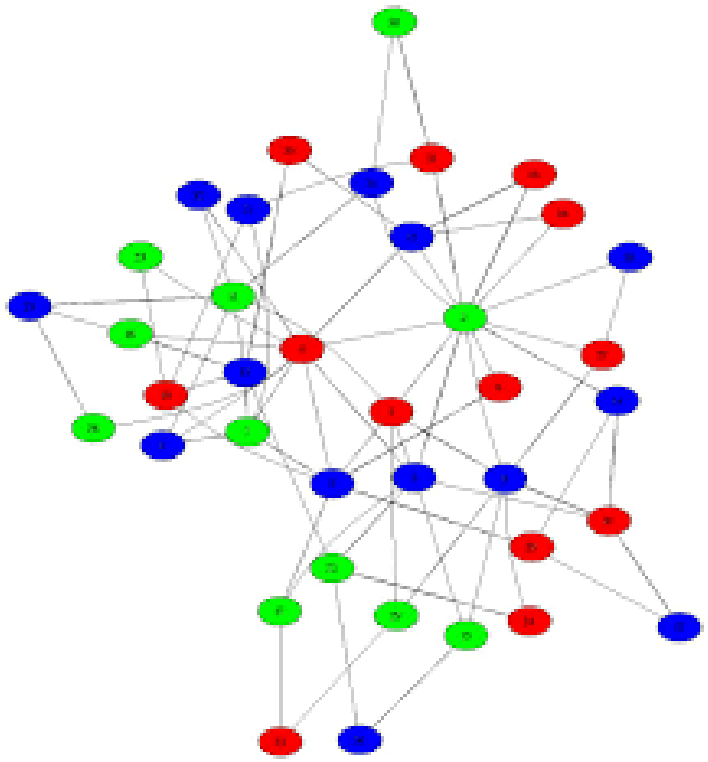
Suma de vectores

- Dados dos vectores $X, Y \in R^n$, la suma $W = X + Y$ se resuelve sumando cada elemento del vector X con su correspondiente elemento del vector Y. En otras palabras $w_i = x_i + y_i$, $\forall i \in 1, \ldots, n$ para cada $w_i \in W$
- La forma más fácil de paralelizar es cuando no existe dependencia entre los diferentes datos a paralelizar



Coloreado de una gráfica

- El problema de coloreado de una gráfica, consiste en que dada una gráfica $G = (V, E)$, encontrar el mínimo número de colores en que se puede colorear una gráfica, de forma tal que dos vértices vecinos tengan necesariamente colores distintos.



Coloreado de una gráfica

- El problema de coloreado de una gráfica, consiste en que dada una gráfica $G = (V, E)$, encontrar el mínimo número de colores en que se puede colorear una gráfica, de forma tal que dos vértices vecinos tengan necesariamente colores distintos.
- El problema de coloreado de una gráfica pertenece a la clase de problemas $\mathcal{NP}-$difícil

Algoritmo secuencial voraz para encontrar un "buen" coloreado en una gráfica

Algorithm 1 Algoritmo secuencial para el coloreado de una gráfica $G = (V, E)$

1: **for** $i = 1$ to $|V|$ **do**
2: Asignar al vértice v_i un color NULL
3: **end for**
4: **for** $i = 1$ to $|V|$ **do**
5: Sea c_{min} el menor valor posible tal que $c_{min} \notin \{c_1, c_2, \ldots, c_k\}$, donde $c_j, j = 1 \ldots k$ es el conjunto de colores de los vecinos de v_i
6: Asignar al vértice v_i el color c_{min}
7: **end for**

Algoritmo secuencial voraz para encontrar un "buen" coloreado en una gráfica

- El algoritmo no garantiza un coloreado mínimo, pero si un buen coloreado $O(\Delta + 1)$
- Complejidad lineal $O(\Delta n)$

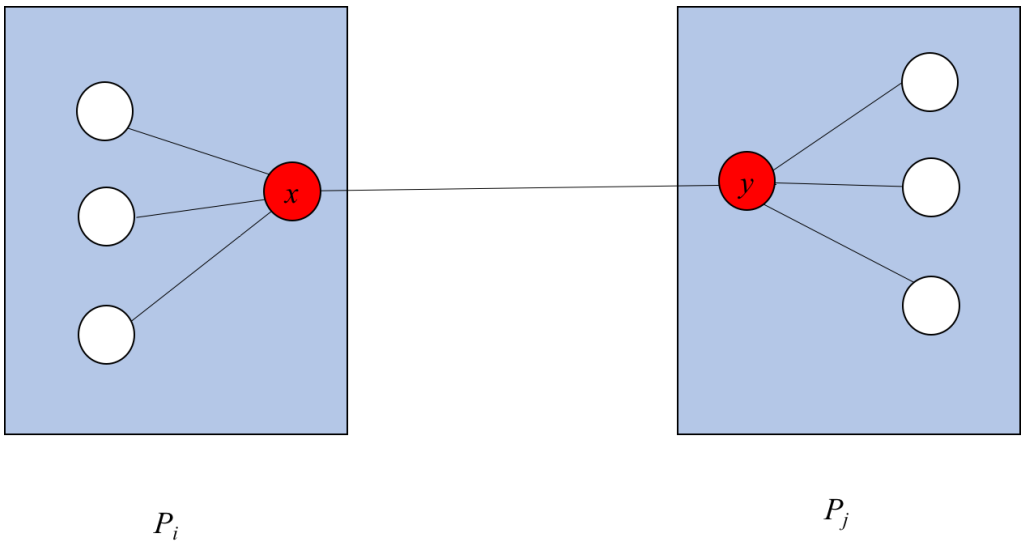
Algoritmo **paralelo** voraz para encontrar un "buen" coloreado en una gráfica

Algorithm 2 Algoritmo paralelo para el coloreado de una gráfica $G = (V, E)$

1: Particionar V en p bloques $V_1, V_2, \ldots, V_p$, donde $\lfloor \frac{|V|}{p} \rfloor \leq |V_i| \leq \lceil \frac{|V|}{p} \rceil$
2: **for** $i = 1$ to p in parallel **do**
3: **for** cada $v_j \in V_i$ **do**
4: Sea c_{min} el menor valor posible tal que $c_{min} \notin \{c_1, c_2, \ldots, c_k\}$, donde $c_m, m = 1 \ldots k$ es el conjunto de colores de los vecinos de v_j
5: Asignar al vértice v_j el color c_{min}
6: **end for**
7: **end for**

Algoritmo **paralelo** voraz para encontrar un "buen" coloreado en una gráfica

- Conflicto en dos bloques



Algoritmo **paralelo** voraz para encontrar un "buen" coloreado en una gráfica (segunda fase)

Algorithm 3 Algoritmo paralelo para el coloreado de una gráfica $G = (V, E)$

1: **for** $i = 1$ to p in parallel **do**
2: **for** cada $v_j \in V_i$ **do**
3: **for** cada vecino $u \in N(V_j)$ **do**
4: **if** $color(v_j) = color(u)$ **then**
5: Agrega $\min\{u, v_j\}$ en la tabla de conflictos
6: **end if**
7: **end for**
8: **end for**
9: **end for**

Algoritmo **paralelo** voraz para encontrar un "buen" coloreado en una gráfica (tercera fase)

Algorithm 4 Tercera fase, colorear secuencialmente los colores en la tabla de conflictos

1: **for** cada v_j en la tabla de conflictos **do**
2: Sea c_{min} el menor valor posible tal que $c_{min} \notin \{c_1, c_2, \ldots, c_k\}$, donde $c_m, m = 1 \ldots k$ es el conjunto de colores de los vecinos de v_j
3: Asignar al vértice v_j el color c_{min}
4: **end for**

Preguntas

joel.trejo@cimat.mx