CURSO:
LaTeX y Git aplicado a la investigación científica.

Profesores:

Renato L. Ramirez Rivero
Angel P. Hinojosa Gutiérrez

Cómo afecta el tamaño del problema y el número de procesos a la eficiencia de un algoritmo paralelo.

Marlene E. Vásquez Calero

13 de Junio 2017

Resumen

Se pretende demostrar por qué la eficiencia (E) tiende a aumentar al aumentar el tamaño del problema (N) y a disminuir al aumentar el número de procesos (P).

---

1. Introducción

El aprovechamiento del potencial de los sistemas paralelos requiere disponer de fundamentos de diseño e implementación de software paralelo, tanto para sistemas distribuidos como para sistemas paralelos de memoria compartida.
Ya que existen aplicaciones con elevados requisitos de cómputo que requieren software para plataformas multiprocesador.
Es por ello que se pretende demostrar por qué la eficiencia (E) tiende a aumentar al aumentar el tamaño del problema (N) y a disminuir al aumentar el número de procesos (P).

2. Demostración

2.1. Modelo matemático

Para demostrarlo se va a plantear un ejemplo y se utilizará el del **algoritmo del patrón de nueve puntos** [1] **la matriz tiene el mismo tamaño en todas las direcciones** (NxNxN).
Este algoritmo realiza operaciones sobre una matriz 3D y para computar cada punto hace falta conocer el valor de sus vecinos de arriba y abajo, dos a la izquierda, dos a la derecha, dos hacia delante y dos hacia atrás, además del propio punto (Figura 1).

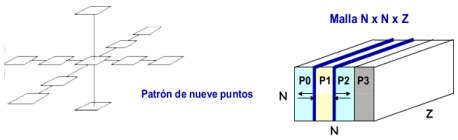

Figura 1: Patrón de nueve puntos, tamaño de la malla y reparto de ésta entre los procesos

El tiempo de un algoritmo secuencial sería (T_1):

$$T_1 = t_c N^3 \tag{1}$$

Cuando se paraleliza, se reparten bloques de la matriz como se muestra en la Figura **??**, por tanto cada proceso necesitará intercambiar $2N^2$ puntos con dos vecinos. Lo que hace que el tiempo de comunicación (T_{comm}) sea:

$$T_{comm} = 2(t_s + t_w 2N^2) \tag{2}$$

Y el tiempo de computación (T_{comp}):

$$T_{comp} = \frac{t_c N^3}{P} \tag{3}$$

Por tanto el tiempo del algoritmo paralelizado (T_p) sería:

$$T_p = T_{comp} + T_{comm} = \frac{t_c N^3}{P} + 2t_s + t_w 4N^2 \tag{4}$$

La ganancia de velocidad (S) se calcula como la división del algoritmo secuencial (T_1) entre el paralelo (T_p):

$$S = \frac{T_1}{T_p} = \frac{t_c N^3}{\frac{t_c N^3}{P} + 2t_s + t_w 4N^2} \quad (5)$$

Y la eficiencia (E), finalmente, se calcularía como la división entre la ganancia de velocidad (S) entre el número de procesos (P):

$$E = \frac{S}{P} = \frac{t_c N^3}{P(\frac{t_c N^3}{P} + 2t_s + t_w 4N^2)} \quad (6)$$

En esta función resultante contamos con cinco variables:

- t_c: tiempo de computación por punto.
- t_s: tiempo de inicialización de comunicación.
- t_w: tiempo de comunicación por palabra.
- N: tamaño de la matriz.
- P: número de procesos.

Las tres primeras no varían, y se va a suponer que $t_c = 1$ y $t_s = t_w = 0.5$. Por lo que vamos a utilizar la siguiente función para obtener los tiempos variando N y P:

$$f(N, P) = \frac{N^3}{N^3 + P + 2PN^2} \quad (7)$$

2.2. Mediciones y resultados

Se toman tiempos con 1, 4, 8, 16 y 32 procesos y con tamaños de 64, 192, 320 y 512. Obteniendo los siguientes resultados:

n	p=1	p=4	p=8	p=16	p=32
64	1.0	0.80	0.57	0.33	0.17
192	1.0	0.92	0.80	0.60	0.38
320	1.0	0.95	0.87	0.71	0.50
512	1.0	0.97	0.91	0.80	0.62

Tabla 1: Datos

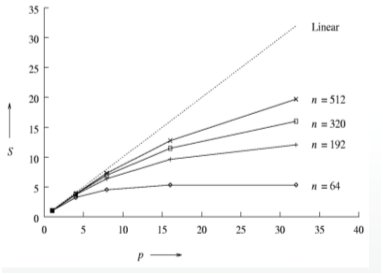

Figura 2: La eficiencia disminuye conforme se aumenta el número de procesos

3. Conclusión

La eficiencia se define como la **fracción de tiempo en el que los procesos realizan un trabajo útil**.

Por ello, es obvio pensar que a más procesos, más tiempo de comunicación y, por tanto, menos tiempo de trabajo útil. Los resultados obtenidos son coherentes respecto a esta afirmación y podemos ver como para todos los tamaños, la eficiencia disminuye.

Al mismo tiempo, se puede presuponer que, de acuerdo a la definición que se da de eficiencia, al aumentar el tamaño del problema más carga de trabajo útil tendrán los procesos y por tanto mayor será el porcentaje de su tiempo dedicado a ello .

Referencias

[1] F. Almeida, D. Gimenez, Jose Miguel Mantas, and A.M. Vidal. *Introducción a la Programación Paralela*. Paraninfo, 2008.

[2] David B. Kirk. *Programming Massively Parallel PProccesors, Second Edition: A Hands-on Approach*. Morgan Kaufmann, 2012.

[3] V. Kumar. *Introduction to Parallel Computing*. Benjamin/Cummings Publishing Company, 2003.