Pre-examen de Fundamentos de Sistemas Informáticos

Mariano Rico & Borja Gil

Febrero 2018

Este documento es un ejemplo de los exámenes de prácticas que tendremos en Fundamentos de Sistemas Informáticos.

En este ejercicio vamos a poner en práctica lo aprendido en las primeras semanas de prácticas. Concretamente, con las funciones del sistema UNIX[1] relacionadas con `fork()` que permiten que un proceso cree otros procesos. Aunque no haya comunicación entre los procesos (más allá de la copia inicial de memoria que recibe un hijo de su padre) se pueden resolver bastantes problemas de paralelización.

[1] Bueno, ya sabes, donde dice UNIX debe decir "GNU is not UNIX"☺

Ejercicio

Planteamiento del problema

MONTECARLO, ADEMÁS DE SER FAMOSA por su circuito de Fórmula 1 y su casino, también es una familia de métodos matemáticos que usan números aleatorios. Uno de estos métodos permite calcular el número pi (π).

En un cuadrado de lado 1 se puede inscribir un cuarto de círculo como muestra la figura 1 (cículo en rojo). Aleatoriamente ponemos puntos dentro del cuadrado y contamos cuántos están dentro del cuarto de círculo (distancia al centro ≤ 1). Siendo 1 el lado del cuadrado, su área es 1, y el área del cuarto del círculo es $\pi/4$.

El cociente entre el área del cuarto del círculo y el área del cuadrado es $\pi/4$, así que estimamos π calculando el número de puntos dentro del cuarto de círculo, dividido por el número de puntos dentro del cuadrado (puntos totales), y multiplicando por 4.

Más información en esta entrada de Wikipedia, donde es especialmente interesante la animación.

Cuantos más puntos usemos, más precisión tendremos en el cálculo de π, como muestra la figura 2. No es un gran método para calcular decimales de π (los hay mucho mejores), pero tiene la ventaja de que es paralelizable de forma perfecta.

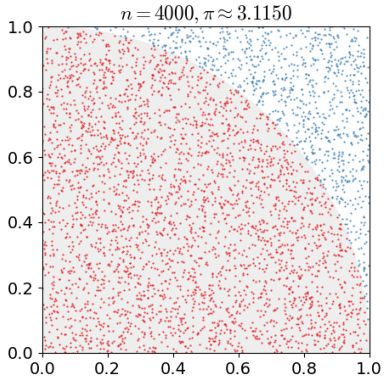

Figura 1: Ejecución con 4K puntos.

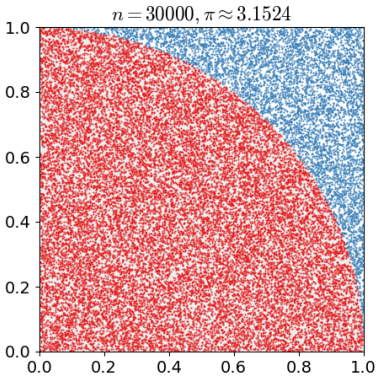

Figura 2: Ejecución con 30K puntos.

El programa

El programa tendrá como argumento el número de procesos que trabajarán en paralelo así como el número de puntos totales que usará cada proceso. El programa lanzará los procesos, quedando a

la espera de su finalización. Cada proceso calculará el número de puntos que han quedado dentro del cuarto de círculo y escribirá el resultado en un fichero. Cuando todos los procesos hayan terminado, el programa principal leerá los ficheros, calculará la suma de puntos dentro del cuarto de círculo, y calculará π usando el método descrito. También mostrará el error (diferencia con 3,14159265358979...).

Recomendaciones

- No te calientes la cabeza con `long` ni con `long long`. Este método converge a π muy lentamente, así que con `int` es suficiente.
- usa `random()` para generar números aleatorios, y `srand()` para cambiar la semilla.
- Usa `fgets()` para leer de fichero. Se supone que tienes soltura con las funciones `fopen()`, `fprintf()` y `fclose()`. Para eliminar ficheros se puede usar `unlink()`.
- En `math.h` se definen `M_PI` y `RAND_MAX`.

El estilo

Este documento simula el estilo de documento creado por Edward R. Tufte [2] [3] mediante el paquete Tufte LaTeX creado por Bil Kleb, Bill Wood, y Kevin Godby bajo licencia Apache 2.0.

[2] Edward R. Tufte. *Beautiful Evidence*. Graphics Press, LLC, first edition, May 2006. ISBN 0-9613921-7-7

[3] Edward R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990. ISBN 0-9613921-1-8

Referencias

Edward R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990. ISBN 0-9613921-1-8.

Edward R. Tufte. *Beautiful Evidence*. Graphics Press, LLC, first edition, May 2006. ISBN 0-9613921-7-7.