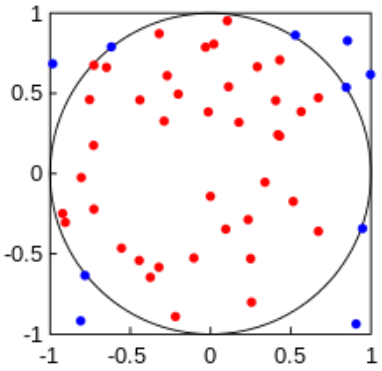
Método de Montecarlo



La técnica de integración por Montecarlo es una técnica usada para la resolución de integrales definidas usando números aleatorios. Para ello, se generan varios números aleatorios, y se evalúa el valor de tales números como x dentro de la función original.

Lo interesante de este método es que computa numéricamente una integral definida. El error en el método de Montecarlo está relacionado de manera inversa con el número de puntos tomados: a más puntos, menor error.

Para probar este método, se utiliza la siguiente función que devuelve el cálculo del número PI:

$$\pi = \int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}}$$

Sin embargo, no se programa directamente la solución con el método de Montecarlo, sino que se codifica una versión modificada compatible con MPI, para poder analizar su rendimiento en función del número de procesadores disponibles. Dicho esto, el código resultante es el siguiente:

```
double randX, pi, insideCircle;

srand(time(NULL) + taskid);

for (int i = taskid; i < THROWS; i += numtasks) {
    randX = rand() / (double) RAND_MAX;
    insideCircle += 1.0 / sqrt(1.0 - randX*randX);
}

if (taskid == MASTER) {
    double insideCircleRecv;
    MPI_Status status;
    for (int i = 1; i < numtasks; i++) {
        MPI_Recv(&insideCircleRecv, 1, MPI_DOUBLE, MPI_ANY_SOURCE, 1,
MPI_COMM_WORLD, &status);
        insideCircle += insideCircleRecv;
    }

    pi = (1 - (-1)) * (float) insideCircle / (float) THROWS;
} else {
    MPI_Send(&insideCircle, 1, MPI_DOUBLE, MASTER, 1, MPI_COMM_WORLD);
}
```

En este caso, THROWS es una constante que define el número de números aleatorios a generar, y `taskid` es el número de tarea (habiendo tantas tareas como procesadores).

Una vez obtenido el valor de PI, se calcula el error con respecto a la siguiente referencia:

$$3.14159265358979323846264338327950288419716939937510582097494 46$$

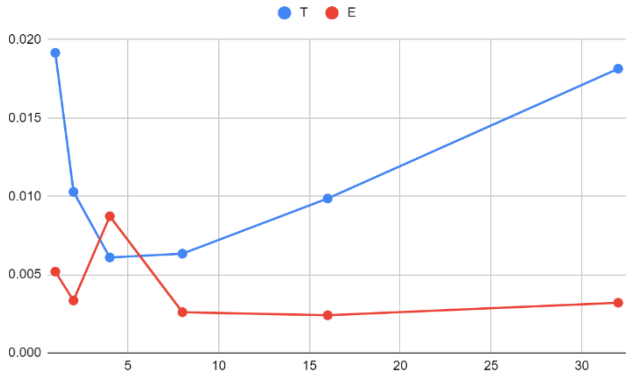
Y se define la calidad del resultado como (siendo T el tiempo de cómputo y E el error):

$$C = \frac{1}{T \cdot E}$$

Se procede entonces a realizar el cómputo con 1, 2, 4, 8, 16 y 32 procesadores. Para cada una de estas opciones, se calcula 5 veces para evitar posibles ruidos, y se toma el valor en la mediana (no se usa la media para así descartar los valores muy desviados debido al ruido).

El resultado se puede ver en la siguiente tabla:

# Cores	T	E	C
1	0.019162	0.005188	10072.43504
2	0.010282	0.003346	29255.59232
4	0.006093	0.008725	18947.94366
8	0.006335	0.002600	55651.73891
16	0.009856	0.002405	43073.50906
32	0.018145	0.003202	14753.40572



Tal como se puede apreciar, el tiempo requerido por el algoritmo se reduce desde 1 a 4 procesadores, pero a partir de 8 comienza a incrementarse lentamente (posiblemente debido a que es necesario hacer más envíos de datos a través de MPI). En cuanto al error, hay un incremento en 4 procesadores, pero tras éste se vuelve a reducir. La calidad, tal como se ve en la escala de colores, es mejor a los 8 procesadores, principalmente debido a que es el tiempo más bajo justo cuando empieza a cambiar la tendencia y a que el error es relativamente bajo.