Instituto de Ciências Matemáticas e de Computação

Universidade de São Paulo

Programação Concorrente

Cálculo do Número Pi Aproximado Com Precisão de 10 Milhões de Dígitos

David César Lucas de Souza – 7152973

Luiz Felipe Alves Prado – 7152479

Sibelius Seraphini – 7152340

Sumário

1. Introdução

Na matemática, π é a proporção numérica com origem na relação entre o perímetro de uma circunferência e seu diâmetro (p/d).

Desde a antiguidade, foram encontradas várias aproximações de π para o cálculo da área do círculo. Era comum o uso de valores próximos a 3 por egípcios, hebreus e babilônios.

Desde o advento da computação desenvolveram-se métodos numéricos computacionais para calcular o valor de π. A computação paralela surgiu nesse contexto para aumentar a velocidade e eficiência desse cálculo. Veremos a seguir alguns métodos para o cálculo de π com precisão de 10 milhões de casas decimais, seus algoritmos implementados e executados sequencialmente e paralelamente e speed-up entre eles.

1.1 Gauss-Legendre

O algoritmo de Gauss-Legendre é notável por ser rapidamente convergente (sua precisão dobra a cada iteração). Com 25 interações produz 45 milhões de dígitos corretos do π. Seu inconveniente é o alto uso de memória.

Algoritmo:

1 - Valores iniciais:

$$a_0 = 1 \qquad b_0 = \frac{1}{\sqrt{2}} \qquad t_0 = \frac{1}{4} \qquad p_0 = 1.$$

2 - Repetir as seguintes instruções até que a diferença de an e bn esteja dentro da precisão desejada:

$$\begin{aligned} a_{n+1} &= \frac{a_n + b_n}{2}, \\ b_{n+1} &= \sqrt{a_n b_n}, \\ t_{n+1} &= t_n - p_n(a_n - a_{n+1})^2, \\ p_{n+1} &= 2p_n. \end{aligned}$$

3 - π é aproximadamente:

$$\pi \approx \frac{(a_n + b_n)^2}{4t_n}.$$

1.2 Borwein

O algoritmo de Borwein para o cálculo do π PE um método de convergência quadrática, ou seja, a cada iteração a precisão quadruplica. O algoritmo não é auto-corretivo, ou seja, cada iteração deve ser feita com o número desejado de dígitos de π a ser calculado.

Algoritmo:

1 – Valores iniciais:

$$a_0 = 6 - 4\sqrt{2}$$
$$y_0 = \sqrt{2} - 1$$

2 – Iterar (ak converge quadraticamente para 1/π)

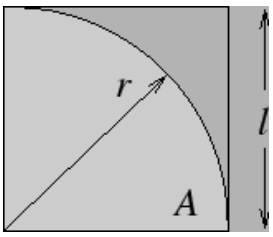
$$y_{k+1} = \frac{1 - (1 - y_k^4)^{1/4}}{1 + (1 - y_k^4)^{1/4}}$$
$$a_{k+1} = a_k(1 + y_{k+1})^4 - 2^{2k+3} y_{k+1}(1 + y_{k+1} + y_{k+1}^2)$$

1.3 Monte Carlo

O algoritmo de Monte Carlo é um método para resolver problemas que se utiliza da estatística. O método consiste em 4 passos básicos que podem se diferir conforme a aplicação: definir um domínio de entradas possíveis, gerar entradas aleatórias distribuídas sobre o domínio, realizar cálculos determinísticos sobre as entradas e agregar os resultados.

No caso do cálculo do π, o algoritmo de Monte Carlo consiste em gerar pares aleatórios (x,y) e verificar quantos estão dentro de um círculo inscrito dentro de um quadrado.

Algoritmo:



Considerando A a área de ¼ de círculo , tem-se que:

$$A = \frac{1}{4} * \pi * r^2$$

Pelo método de Monte Carlo:

$$A = l^2 * \frac{P_{in}}{P_{total}}$$

Unindo as duas equações obtemos:

$$\frac{1}{4} * \pi * r^2 = l^2 * \frac{P_in}{P_total}$$

$$\pi = \frac{l^2 * \frac{P_{in}}{P_{total}}}{\frac{1}{4} * r^2}$$

Considerando-se l = r = 1, obtém-se:

$$\pi = 4 * \frac{P_{in}}{P_{total}}$$

2. Implementações

Para a implementação dos programas foi utilizada a biblioteca GMP para o cálculo com precisão de números grandes, desenvolvida pelo projeto GNU e distribuída sobre a licença GNU LGPL.

2.1 Sequencial

Os programas sequenciais foram baseados nos algoritmos de cada método, utilizando-se das funções da biblioteca GMP para rodar os *loops* até que os valores convirjam para PI.

Todos os métodos rodam até que a diferença entre os valores passados e os novos sejam menores do que a precisão procurada. O valor final é então salvo em um arquivo de texto.

2.2 Paralela

2.2.1 Gauss-Legendre

Foram utilizadas quatro threads, uma para calcular cada parâmetro do algoritmo de Gauss Legendre. Foram utilizadas um total de 8 mutexs para garantir a sincronicidade entre as variáveis.

2.2.2 Borwein

Foram utilizadas duas threads, sendo que cada uma delas calcula uma parte do valor de ak+1. Com a sincronização das duas temos um ganho de tempo paralelizando a variável que será utilizada no cálculo do PI. Como critério de parada, foi utilizado um arquivo nomeado de pi.txt que continha as 10 milhões de casas corretas do número pi. Assim o valor de pi calculado é comparado com as casas do valor correto, até atingir a precisão desejada.

2.2.3 Monte Carlo

A versão paralela do algoritmo de Monte Carlo, usa quatro threads para gerar os pontos e a cada 50 iterações eles são sincronizadas e o valor do PI é calculado

3. Resultados

Para comparação de resultados entre os algoritmos sequenciais e paralelos é feito o cálculo do speedup, que consiste na seguinte fórmula:

$$S_p = \frac{T_1}{T_p}$$

Onde Sp é o valor do speedup, T1 é o tempo de execução do algoritmo sequencial e Tp é o tempo de execução do algoritmo paralelo. Os programas foram executados em uma máquina *Intel® Core™2 Quad CPU Q6600 @ 2.40GHz × 4 Memória: 3.2 GB.* Os dados a seguir são referentes à média de tempo de cinco execuções.

3.1 Gauss-Legendre

O método de Gauss-Legendre paralelo mostrou um speedup de **1.046876253** com as seguintes medições:

Serial	Paralelo	Speedup
300.264 s	286.819 s	1.046876253

3.2 Borwein

O método de Borwein paralelo mostrou um speedup de **1.14296516864** com as seguintes medições:

Serial	Paralelo	Speedup
340.087 s	297.548 s	1.14296516864

3.3 Monte Carlo

O método de Monte Carlo paralelo mostrou um speedup de **2.958478248** com as seguintes medições (valores referentes ao cálculo de 400 pontos):

Serial	Paralelo	Speedup
443.183 s	149.801 s	2.958478248

4. Conclusão

Os três algoritmos testados apresentaram diminuição em tempo de execução quando executados paralelamente porém podemos perceber que no caso do método de Gauss-Legendre o ganho foi muito baixo. Isso se deve ao fato do algoritmo depender fortemente dos dados em suas iterações, impossibilitando um maior paralelismo na execução das threads. O mesmo aconteceu com o algoritmo de Borwein, que obteve um pequeno ganho mas ainda assim teve seu tempo de execução paralelo maior que o tempo de execução paralelo de Gauss-Legendre.

O algoritmo de Monte Carlo teve o maior ganho de tempo de execução devido ao princípio do algoritmo independer de iterações entre os dados, com várias threads executando o “preenchimento” do círculo paralelamente. Outra vantagem desse algoritmo é a possibilidade de aumentar o número de threads conforme o processador onde ele será executado (nesse caso, 4 threads para 4 núcleos de processamento), aumentando ainda mais sua escalabilidade.

5. Read Me

Configuração do Ambiente
O programa faz uso da biblioteca "gmp.h", para utilizá-lo basta seguir os seguintes passos:
Ubuntu, Debian
sudo apt-get install libgmp3-dev
Outras Plataformas:
Realizar o download do pacote em http://gmplib.org/#DOWNLOAD
Descompactar e realizar os seguintes comandos
./configure
make
make check
make install
Compilação dos Programas
Dentro da pasta ..pc2013-grupo09/trabalho1 use o comando make
Execução dos Programas
Caso execute ./arquivo e encontre o seguinte erro:
error while loading shared libraries: libgmp.so.10: cannot open shared object file: No such file or directory
execute o comando ln -s /usr/lib/libgmp.so.3 /usr/lib/libgmp.so.10
Resultado
O arquivo pi.txt possui 10 milhões de dígitos corretos do número pi, usado para parar os algoritmos.
Os programas geram como saida os seguintes arquivos:
it-n.txt: sendo o valor de pi depois de n iterações
execution_time.txt: indica o tempo de execução do algoritmo

6. Referências

Aproximação do Pi pelo método de Monte Carlo – UFRGS - http://www.inf.ufrgs.br/gppd/disc/cmp134/trabs/T2/021/html/index.html

Pi (Métodos de cálculo numérico) – Wikipedia - http://pt.wikipedia.org/wiki/Pi#M.C3.A9todos_de_c.C3.A1lculo_num.C3.A9rico

Borwein's Algorithm – Wikipedia - http://en.wikipedia.org/wiki/Borwein's_algorithm

The World of Pi - http://www.pi314.net/eng/borwein.php

Pi Iterations – Wolfram MathWorld - http://mathworld.wolfram.com/PiIterations.html