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, **\scriptstyle{\pi}** 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 \scriptstyle{\pi} 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 \scriptstyle{\pi}. A computação paralela surgiu nesse contexto para aumentar a velocidade e eficiência desse cálculo. Veremos a seguir alguns métodos para o cálculo de \scriptstyle{\pi} com precisão de 10 milhões de casas decimais, seus algoritmos implementados e executados sequencialmente e paralelamente e speed-up entre eles.

* 1. Gauss-Legendre

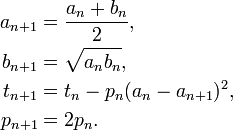
O algoritmo de Gauss-Legendre é notável por ser rapidamente convergente (sua precisão dobra a cada iteração). Com 25 interações produz 45 milhões de dígitos corretos do \scriptstyle{\pi}. Seu inconveniente é o alto uso de memória.

Algoritmo:

**1 - Valores iniciais:**

C:\Users\Luiz Felipe\Desktop\62853e30b3cd9190dcdedeed764bc0f4.png

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



**3 - \scriptstyle{\pi} é aproximadamente:**

C:\Users\Luiz Felipe\Desktop\7fc8f32d47be63c3c7548dc51ab40146.png

* 1. Borwein

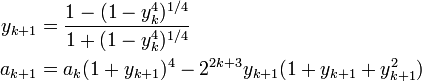
O algoritmo de Borwein para o cálculo do \scriptstyle{\pi} PE um método de convergência quadrática, ou seja, a cada iteração a precisão quadruplica. O algoritmo não é auto-corretivo, ou seja, cada iteração deve ser feita com o número desejado de dígitos de \scriptstyle{\pi} a ser calculado.

Algoritmo:

**1 – Valores iniciais:**

**C:\Users\Luiz Felipe\Downloads\8274770ba0343ebcd18c13745a2ed109.png**

**2 – Iterar (ak converge quadraticamente para 1/\scriptstyle{\pi})**

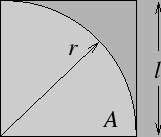
****

* 1. 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 \scriptstyle{\pi}, 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:



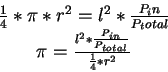
Considerando A a área de ¼ de círculo , tem-se que:

C:\Users\Luiz Felipe\Downloads\img5.png

Pelo método de Monte Carlo:

C:\Users\Luiz Felipe\Downloads\img6.png

Unindo as duas equações obtemos:



Considerando-se l = r = 1, obtém-se:

C:\Users\Luiz Felipe\Downloads\img11.png

1. 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.

* 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.

* 1. Paralela
     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.

* + 1. Borwein
    2. 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

1. Resultados

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

C:\Users\Luiz Felipe\Downloads\a8080d5bd23d7ba57a9cf4fedcefadad.png

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.

* 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 |

* 1. 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 |

* 1. 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 |

1. 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.

1. 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>