Aplicação de Métodos Numéricos para o Cálculo do Número π e Simulação do Modelo financeiro de Black Scholes

Leandro Giusti Mugnaini[1](10260351), Matheus Borges Kamla[1](10277015)

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
São Carlos – SP – Brasil

leandromugnaini@usp.br, matheuskamla@usp.br

Resumo. *Este relatório é referente ao trabalho realizado na disciplina SSC0641 - Sistemas Operacionais I, ministrada pelo Prof. Dr. Julio Cezar Estrella. O trabalho tem como objetivo analisar a diferença de desempenho de diferentes programas escritos na linguagem C, quando executados de forma sequencial e de forma paralela (utilizando a biblioteca Pthreads). Serão analisados três métodos numéricos para para calcular o número π, através dos métodos de Gauss-Legendre, Borwein e Monte Carlo. Em seguida, objetiva-se realizar uma Simulação da área de economia, utilizando o Modelo de Black-Scholes, para descrever o fenômeno da precificação de derivativos.*

1. Introdução

Usualmente utiliza-se duas formas de execução de códigos: de forma sequencial e de forma paralela. Por mais que um algoritmo escrito de forma sequencial supra uma grande gama de necessidades, há alguns usos em que a programação paralela se torna necessária. Em aplicações de *deep learning*, *machine learning* e computação gráfica a computação paralela pode reduzir o tempo de processamento que seria de dias em forma sequencial, para alguns minutos (ou horas). Existem diversas bibliotecas que tornam possível a execução de algoritmos de forma paralela, as mais comuns são a POSIX Threads e OpenMP. Nesse trabalho utilizou-se a biblioteca POSIX Threads (chamada frequentemente de Pthreads), para programação em linguagem C.

Para comparar o desempenho de códigos sequenciais e paralelos, utilizaremos quatro algoritmos: 3 métodos numéricos para calcular o número π e um modelo econômico. Os métodos numéricos são: Gauss-Legendre, Borwein e Monte-Carlo. Já o modelo econômico a ser simulado é o de Black-Scholes.

2. Metodologia

O computador utilizado é um modelo Dell, com processador i7-6500u operando a 2.50GHz, 16GB de memória principal e placa gráfica NVIDIA GeForce 930M. O sistema operacional utilizado é o Ubuntu 16.04 e a interface utilizada foi o Microsoft Visual Studio. Os algoritmos foram escritos na linguagem C.

Para execução dos códigos, utilizou-se a biblioteca GMP para manipulação de grandes números e a biblioteca pthreads para realizar o paralelismo.

Nos algoritmos de Gauss-Legendre e de Borwein, foi utilizado um número de iterações igual a 10^5. Não foi possível utilizar um número maior pois a partir de 10^6 iterações já

ocorria uma falha de segmentação. Utilizou-se arquivos de entrada e saída para medir o tempo de execução utilizando o comando: ”/usr/bin/time -f ”%e”./nomedoprograma <entrada.txt> saida.txt”.

No algoritmo de Monte Carlo, foi possível utilizar 10^9 iterações. Mediu-se o tempo de execução da mesma forma que os anteriores.

Já na simulação de *Black-Scholes* foi utilizado um número fixo de iterações, presente no arquivo de texto fornecido pelo professor. Optou-se por não utilizar a biblioteca GMP nesse algoritmo, pois no passo mais custoso do algoritmo é necessário realizar um cálculo exponencial com expoente não inteiro, o que não é suportado pela biblioteca GMP (cálculo explícito na explicação do algoritmo).

Para obter uma métrica confiável do tempo de execução dos algoritmos, realizou-se a média aritmética do tempo de 5 execuções.

3. Os algoritmos

3.1. Gauss-Legendre sequencial

O algoritmo de Gauss-Legendre leva em conta 4 valores iniciais, e a partir desses valores, iterações são realizadas para aproximar o valor do número π. Os valores iniciais são os seguintes:

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

Já as iterações são as seguintes:

$$a_{n+1} = \frac{a_n+b+n}{2}, \quad b_{n+1} = \sqrt{a_n.b_n}$$

$$t_{n+a} = t_n - p_n(a_n - a_{n+1})^2, \quad p_{n+1} = 2p_n$$

O cálculo do número π será dado então por:

$$\pi = \frac{a_{n+1}+b_{n+1}}{4t_{n+1}}$$

No algoritmo em C foram reproduzidos os passos acima, tornando possível o cálculo do número π.

3.2. Gauss-Legendre paralelo

No algoritmo de gauss paralelo é realizado três threads, uma para calcular o termo a, uma para o termo b e uma última para calcular o termo p e o t. Foi utilizado de uma váriavel de controle e de uma variável mutex para que a primeira thread a ser realizada na interação fosse sempre a thread do termo a para depois calcular a do b (que depende do a) e depois passar para última thread calculando o termo p (que não depende de nada) para depois calcular o termo t (que depende de todos os outros termos).

3.3. Borwein sequencial

Consideramos para esse trabalho a Fórmula BBP para o cálculo do π. BBP possui as iniciais dos sobrenomes dos nomes dos criadores da fórmula, sendo eles David Harold Bailey, Peter Borwein e Simon Plouffe. A fórmula é a seguinte:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right)$$

3.4. Borwein paralelo

No algoritmo Borwein paralelo, utilizou-se diferentes threads independentes para as etapas do somatório, e em seguida criou-se uma thread que depende dos resultados das anteriores para realizar a multiplicação. Para que a thread responsável pela multiplicação não seja executada antes das outras, utilizou-se variáveis de controle que exerciam essa função.

3.5. Monte-Carlo sequencial

O algoritmo de Monte-Carlo consiste em utilizar um segmento circular de raio 1 (com $\frac{1}{4}$ do tamanho da circunferência completa), inscrito em um quadrado de lado 1. Utilizando um x e um y gerado aleatoriamente entre 0 e 1, podemos verificar se o ponto está dentro ou fora do círculo, através da equação da distância de um ponto à origem, definida abaixo:

$$d = x^2 + y^2$$

Se $d \leq 1$, o ponto está inscrito na circunferência, caso contrário, está inscrito somente no quadrado. O algoritmo consiste então em dividir o número de pontos que caíram dentro do segmento circular pelo número total de pontos sorteados. Esse valor tende a $\frac{\pi}{4}$ quando o número de pontos tende ao infinito. Uma figura ilustrativa está apresentada abaixo:

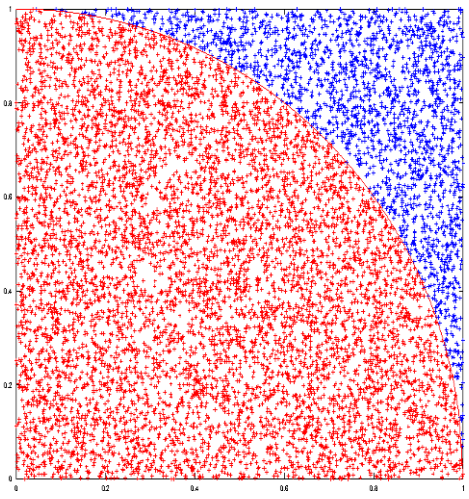

Figura 1. Figura mostrando o funcionamento do algoritmo

Os pontos vermelhos estão dentro do círculo, enquanto os azuis estão fora. Para calcular o valor de π, basta então multiplicar por 4 a razão dos pontos que caíram dentro pelos que caíram fora para obter um valor aproximado de π.

3.6. Monte-Carlo paralelo

No algoritmo de Monte-Carlos são utilizadas 10 threads para paralelizar o processo de geração dos pontos, sendo que cada thread vai realizar um loop de número de interações/ número de threads. Sendo assim cada thread retorna o número de pontos acertados, que serão utilizados para o cálculo do pi.

3.7. Black-Scholes sequencial

O algoritmo de Black-Scholes serve para simular por meio de cálculos o fenômeno financeiro da precificação de derivativos. Usando variáveis de entrada do arquivo txt fornecido e o pseudocódigo da especificação do trabalho, podemos criar um algoritmo na linguagem C, tornando possível a simulação. O código utiliza conceitos estatísticos, como média aritmética, desvio padrão e intervalo de confiança para simular o fenômeno.

Em um dos passos do algoritmo, é realizado o seguinte cálculo exponencial:

$$t = S.\exp\left(r - \tfrac{1}{2}\sigma^2.T + \sigma\sqrt{T}.randomNumber()\right)$$

Por conta do expoente não se tratar de um número inteiro e a biblioteca GMP não possuir uma função para realizar esse cálculo, optou-se por fazer o código sem utilizar a biblioteca, já que os valores das contas seriam truncados, perdendo o sentido do uso da biblioteca.

3.8. Black-Scholes paralelo

No algoritmo de Black-Scholes paralelo, criou-se um vetor de threads que são responsáveis por realizar o cálculo exponencial, que é a parte mais custosa do algoritmo. Optou-se por não utilizar threads nas outras etapas, pois todas elas dependiam totalmente do resultado anterior, portanto o paralelismo não seria útil. Os demais passos foram feitos seguindo o código sequencial.

4. Resultados

O tempo médio de execução juntamente com as respectivas médias de 5 execuções estão apresentados nas tabelas abaixo. A primeira tabela representa o tempo para os algoritmos sequenciais, enquanto a segunda tabela representa para os algoritmos paralelizados.

Execuções	Gauss-Legendre	Borwein	Monte-Carlo	Black-Scholes
1	47.38	123.07	227.26	0.05
2	47.31	121.69	228.2	0.04
3	47.15	123.24	234.46	0.03
4	47.39	121.89	224.24	0.04
5	47.36	123.12	226.87	0.05
Média	47.31	122.60	228.20	0.042

Execuções	Gauss-Legendre	Borwein	Monte-Carlo	Black-Scholes
1	47.38	139.04	130.28	0.20
2	47.31	135.15	133.04	0.18
3	47.15	122.47	132.82	0.19
4	47.39	137.22	131.43	0.17
5	47.36	125.34	132.80	0.20
Média	47.31	136.20	132.07	0.18

5. Conclusão

Analisando os tempos médios de execução dos algoritmos, podemos perceber que o paralelismo não é eficiente em todos os casos. Em algoritmos de natureza sequencial, ou que possuem passos que são dependentes de passos anteriores, a execução paralela se mostrou igual (ou até pior) do que a execução sequencial.

6. Referências

https://pt.wikipedia.org/wiki/F%C3%B3rmula_BBP

https://pt.wikipedia.org/wiki/Algoritmo_de_Gauss-Legendre

https://www.blogcyberini.com/2018/09/calculando-o-valor-de-pi-via-metodo-de-monte-carlo.html