Análise da Integral Numérica de Monte Carlo

Método Experimental Utilizando MPI com Python

Felipe Alberto Capati
Dept. Inteligência Artificial do Centro Universitário FEI
São Bernardo do Campo, São Paulo
felipecapati@fei.edu.br

***Abstract*—This project aims to introduce the Monte Carlo mathematical method, exemplify the integral calculation using it and one of the High Performance Computing (HPC) techniques called Message Passing Interface (MPI), along with a validation methodology that analyzes the processing time required to calculate the integral applied in a prototyped model using Python with mpi4py.**

Keywords—Monte Carlo Integration, Monte Carlo Analysis, Python, mpi4py, MPI, HPC.

I. INTRODUÇÃO

Este projeto tem como objetivo introduzir o método matemático de Monte Carlo, exemplificar o cálculo da integral utilizando o mesmo e uma das técnicas de *High Performance Computing* (HPC) chamada de *Message Passing Interface* (MPI), juntamente com uma metodologia de validação que analisa o tempo de processamento necessário para o cálculo da integral aplicada em um modelo prototipado utilizando Python com mpi4py.

II. FUNDAMENTAÇÃO TEÓRICA

A. Monte Carlo

O algoritmo de Monte Carlo é um método numérico utilizado para solucionar problemas matemáticos de difícil solução em uma gama diversa de áreas do conhecimento indo de física atômica a métodos financeiros, probabilísticos e cálculo de integral numérica. Em contrapartida a robustez do algoritmo, tem-se seu custo computacional lento [$O(N^{-1/2})$] [1].

B. Integral de Monte Carlo

De acordo com [1], temos que a integral de uma função pode ser expressa como a média da função expectativa / avaliação em local aleatório. Simplificando o sistema e considerando uma integral unidimensional, temos:

$$I[f] = \int_0^1 f(x)\,\mathrm{d}x = \bar{f}. \tag{1}$$

Seja x uma variável uniformemente distribuída em um intervalo unitário.

$$I[f] = E[f(x)]. \tag{2}$$

A equação da quadratura de Monte Carlo é baseada na probabilidade de uma variável distribuída uniformemente coincidir com a área sob o gráfico ou não, na qual para uma distribuição normal de N variáveis em um array {xn} podemos explicitar como:

$$I_N[f] = \frac{1}{N}\sum_{n=1}^{N} f(x_n). \tag{3}$$

Na qual para um número de interações infinitas, temos:

$$\lim_{N\to\infty} I_N[f] \to I[f]. \tag{4}$$

Com base na equação (4) podemos inferir que para um número infinito de interações a integral de Monte Carlo converge. Na qual temos a função de erro e erro médio quadrático (RMSE), respectivamente:

$$\epsilon_N[f] = I[f] - I_N[f] \tag{5}$$

$$E[\epsilon_N[f]^2]^{1/2}. \tag{6}$$

Aplicando o *Central Limit Theorem* (CLT) [2] citado em [1] na qual descreve o tamanho e as propriedades estatísticas dos erros de integração de Monte Carlo, pode-se escrever o erro como:

$$\sigma[f] = \left(\int_{I^d} (f(x) - I[f])^2\,\mathrm{d}x\right)^{1/2}. \tag{7}$$

Similar ao cálculo da integral unidimensional, é possível fazer o cálculo da integral de volume partindo do mesmo princípio, com distinções que terá mais limites de integração (x, y, z) e que o ponto tem que estar contido dentro do espaço determinado pela função, na qual podemos simplificar como:

$$\iiint f(x,y,z)dxdydz \cong \frac{V_{integração}}{N}\sum_{n=1}^{N} f(x_n, y_n, z_n) \tag{8}$$

III. METODOLOGIA

Foi utilizado o Python, juntamente com o *mpi4py* e uma máquina virtual Linux Ubuntu (VirtualBox) com 4 núcleos virtuais (cada nó era um núcleo) para prototipar o modelo, na qual a função que calcula a integral utiliza de um gerador aleatório responsável por gerar N pontos distribuídos

uniformemente sobre os limites de integração. A seguir tem-se um pseudocódigo da modelagem.

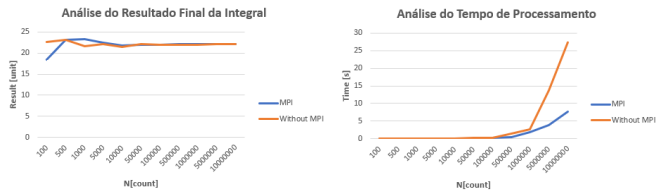
```
Function PartOfSumOfIntegral (N:int, limOfIntegration:int[]):
    Loop (0 até N):
        x = GetRandomValue(limOfIntegration);
        soma = soma + GetResultFunction(x);
    End Loop
    return soma;
Main():
    requisition = (NumSlave+1)/N;
    If MasterMPI:
        SendRequisitionToSlave(requisition);
        sum = PartOfSumOfIntegral(requisition, limOfIntegration);
        all_sum = GetSumOfSlaves() + sum;
        integral = VolIntegration*all_sum/N;
        print(integral);
    ElseIf SlaveMPI:
        GetRequisitionToMaster(requisition);
        sum = PartOfSumOfIntegral(requisition, limOfIntegral);
        SendToMaster(sum);
    EndIf
```

Para exemplificação da integral de Monte Carlo, foi calculada a integral (9), na qual (9) é uma integral de volume definida de um toroide e seus resultados serão discutidos melhor em IV.

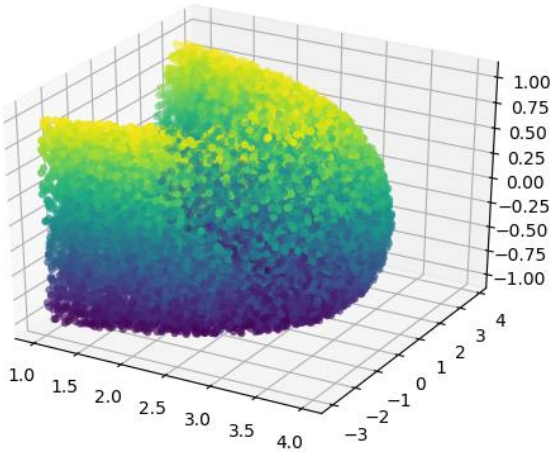
$$\begin{aligned} & z^2 + \left(\sqrt[2]{x^2+y^2} - 3\right)^2 \leq 1 \\ & x > 1 \\ & y \geq -3 \end{aligned} \qquad (9)$$

IV. Resultados

Baseando-se na metodologia proposta em III foram gerados os gráficos a seguir que dizem respeito ao cálculo da integral utilizando (4), aplicando o pseudocódigo apresentado em III, ao tempo de processamento necessário dada uma quantidade N de iterações e ao ganho de relativo ao tempo de processamento quando utiliza-se MPI:



Em que para (9) temos o volume exemplificado para N pontos na figura a seguir:



V. Conclusão

Dado os resultados vistos em IV podemos inferir que o Método de Integração de Monte Carlo é funcional, converge ao valor real de (9) com cerca 30.000 iterações e que o tempo de processamento tem um comportamento crescente proporcional a N tanto para o cálculo da integral utilizando MPI, quanto para integral sem o MPI.

Para essa análise 30.000 iterações foram suficientes, porém vale ressaltar que o valor de N é função do problema proposto, ou seja, varia para cada aplicação do modelo.

O MPI proporcionou um aumento significativo para o tempo de processamento (3,5x) quando utiliza-se 4 nós de processamento, sendo que é possível escalar esse valor para ***M*** máquinas ligadas em rede, o que daria um aumento muito mais significado do que o apresentado nesse projeto.

Agradecimentos

Agradecimentos especiais a CAPES e ao Centro Universitário FEI por financiar o mestrado que está em curso; ao professor Reinaldo Bianchi por proporcionar visões sobre o mundo acadêmico e orientar trabalhos científicos com o objetivo de lapidar os conhecimentos abordados em sala; aos meus pais e a minha família que sempre me apoiaram em meio a dificuldades.

Referências

[1] R. E. Caflisch, “Monte Carlo and quasi-Monte Carlo methods” Mathematics Department, UCLA, Los Angeles, CA, USA, 1998.

[2] W. Fellen, An Introduction to Probability Theory and its Application, vol. 1. Wiley, 1971.