quinta-feira, 9 de maio de 2019

CLE - Calculo de Determinantes de Matrizes

A fórmula para calcular o determinante de uma matriz $n x n$ é dada pela expansão de menores e envolve $n!$ termos. Na prática, o número de etapas necessárias para calcular o determinante depende do tamanho de entradas da matriz. Para se utilizar sistemas computacionais, o tempo para calcular a determinante de uma matriz do tipo $n x n$ seria então da ordem de $n!$. Em Ciência da Computação, a complexidade de tempo é a complexidade computacional que descreve o montante de tempo necessário para executar um algoritmo. Normalmente a complexidade de tempo computacional é medida pela contagem estimada do número de operações elementares que são realizadas pelo algoritmo. No caso de um determinante de matriz, usando métodos elementares de resolução, seriam necessárias operações aritméticas da ordem de $O(2^n)$ ou $O(n!)$ onde O é a notação utilizada para expressar a complexidade de tempo em algoritmos. A figura 1, abaixo, demonstra o efeito da complexidade do tempo em algoritmos, conforme a notação O e as funções que expressam essa complexidade:

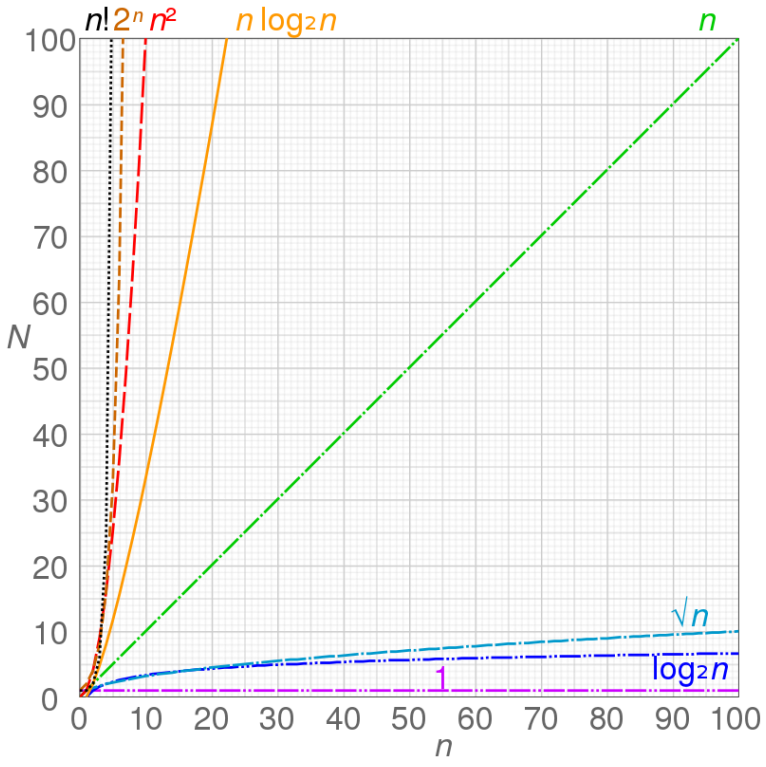

FIG 1 - GRÁFICOS DE FUNÇÕES COMUMENTE USADAS NA ANÁLISE DE ALGORITMOS, MOSTRANDO O NÚMERO DE OPERAÇÕES N VERSUS TAMANHO DE ENTRADA N (MINÚSCULO) PARA CADA FUNÇÃO.

Conforme já destacado, a complexidade do tempo para um algoritmo calcular o determinante de uma matriz nxn, é sensivelmente maior conforme ilustrado na figura 1. Para reduzir essa complexidade de tempo, o determinante de tal matriz também pode ser calculado por **Eliminação de Gauss**.

Passos para aplicar a eliminação de Gauss:

Existem três operações básicas que podem ser aplicadas a qualquer tipo de sistema linear, sem que se altere a solução do mesmo:

1. Trocar duas linhas entre si.
2. Multiplicar todos os elementos de uma linha por uma constante não-nula.
3. Substituir uma linha pela sua soma com um múltiplo de outra.

A figura 2, abaixo exibe uma matriz que passou por esses passos:

$$\begin{pmatrix} 1 & 0 & 3 & -8 \\ 0 & 1 & 3/2 & -3 \\ 0 & 0 & 1 & -4 \end{pmatrix}$$

FIG. 2 - MATRIZ ESCALONADA PELA ELIMINAÇÃO DE GAUSS

Usando estas operações, uma matriz sempre pode ser transformada numa matriz na forma escalonada (forma de escada por linhas) e, posteriormente, ser posta na forma escalonada reduzida. Esta forma final, por sua vez, é única e independente da sequência de operações de linha usadas, sendo mais fácil de resolver que a versão original da matriz. Cabe, também, ressaltar que estas operações elementares são reversíveis, sendo possível retornar ao sistema inicial aplicando a sequência de operações novamente, mas na ordem inversa.

Sabemos como cada operação de linha elementar afeta o determinante de uma matriz e, se acompanharmos as etapas de redução de linha que realizamos no processo de execução da eliminação de Gauss, podemos reconstruir quase imediatamente o determinante da matriz original a partir desses dados. Se a eliminação de Gauss aplicada a uma matriz quadrada A produz uma matriz escalonada de linha B, seja o produto dos escalares pelos quais o determinante foi multiplicado, usando as regras acima. Então o determinante de A é o quociente por d do produto dos elementos da diagonal de B:

$$det(A) = \frac{\pi . diag(B)}{d}$$

O número de passos necessários para reduzir uma matriz com entradas inteiras é n^3 e, portanto, o uso da eliminação de Gauss reduz a complexidade de tempo computacional para calcular o determinante de uma matriz a $O(n^3)$ etapas.

Arquitetura MPI

Para o projeto foi adotado o paradigma de computação *master*/*worker* onde o processo *master* adota o trabalho realizado no *loop* intensivo e o divide em um número de tarefas os depositando em um local. Um ou mais processos, denominados de *workers*, pegam essas tarefas, calculam e retornam o resultado para esse local, conforme ilustrado na figura 3. O processo *master* coleta os resultados na medida em que forem disponibilizados.

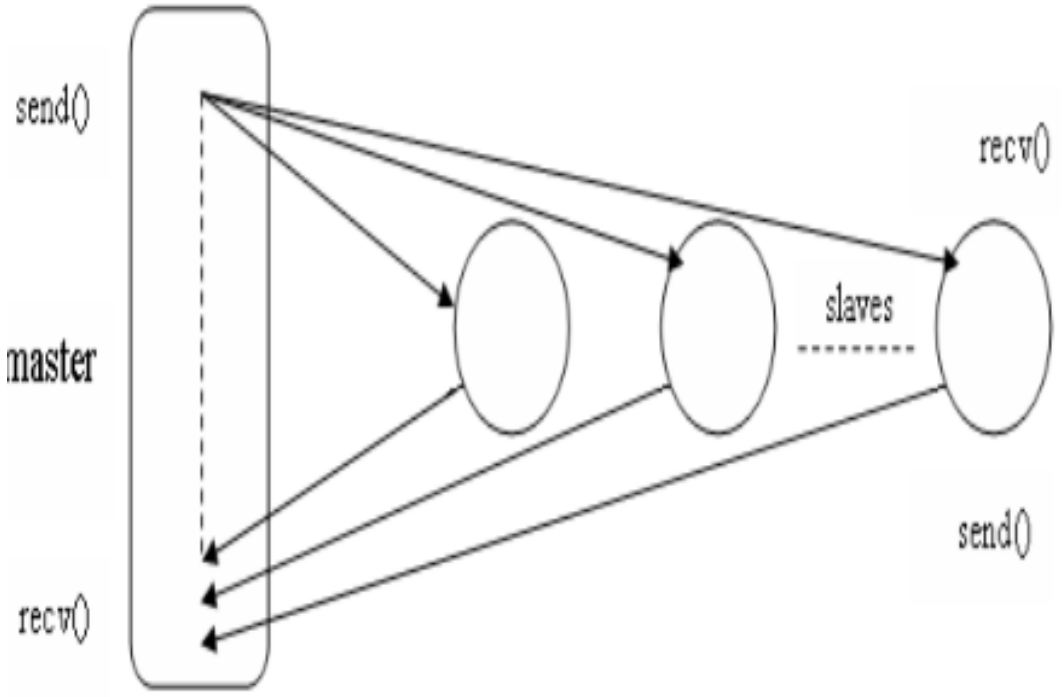

Fig. 3. Master / Slaves Model

Nesse paradigma, então, o *master* reduz as matrizes utilizando a **Eliminação de Gauss**. Para cada tarefa, o *master* envia uma matriz completa para cada *worker* e distribui as linhas das matrizes entre os *workers*. Dessa forma o trabalho de multiplicação das matrizes é computado num formato paralelo.

- Operações em MPI utilizados na solução:

- MPI_INIT :
  - MPI_Init(&argc,&argv);
- MPI_COMM_SIZE :
  - MPI_Comm_size(MPI_COMM_WORLD, &*numtasks*); (numtasks é o valor retornado do mundo MPI, isto é, quantidade de processadores existente no ambiente cluster utilizado.)
- MPI_COMM_RANK :
  - MPI_Comm_rank(MPI_COMM_WORLD, &taskid); (MPI_COMM_WORLD é o comunicador MPI criado pelo MPI_INIT. Permite aos programa MPI chamar as instancias do programa no cluster, *taskid* é o identificador do processador).
- MPI_SEND :
  - MPI_Send(&offset, 1, MPI_INT, dest, mtype, MPI_COMM_WORLD); (*offset* é o endereço de buffer de envio, 1 é o número de elementos do tipo de dados no buffer de envio, MPI_INT é o tipo de dados dos elementos do buffer de envio e de recebimento, *dest* é o rank de destino, mtype é o *tag* da mensagem, essa tag serve para diferenciar entre as mensagens similares.
- MPI_RECV:
  - MPI_Recv(&offset, 1, MPI_INT, source, mtype, MPI_COMM_WORLD, &status);
- MPI_FINALIZE:
  - MPI_Finalize();
- MPI_BARRIER
  - é um bloqueador que espera até que todos os processos no comunicador (MPI_COMM_WORLD) terminem sua tarefa).
- MPI_Type_create_struct
  - Cria um tipo de dados MPI.
- MPI_Type_commit
  - Grava o tipo de dados criado.

Conclusão

Apesar do uso da programação paralela com a exploração de recursos de MPI (*Message Passing Interface*) sobre os processadores do cluster de máquinas dispostos em laboratório, reduzirem o tempo computacional quando comparados às soluções com threads, é perceptível que, com o aumento do número das operações aritméticas de calculo de matrizes, passam a se tornar impraticáveis com esse modelo, isto pode ser verificado pela tendência de piora no tempo de resposta das operações na medida em que o número de operações foram aumentadas a partir da leitura dos arquivos, onde, cada um possui um número maior de operações. Dessa forma, podemos concluir que o tempo computacional para essas operações tende a degradar na medida em que o número de operações aumenta, o que demanda a necessidade de procurar outras alternativas para os casos de operações muito volumosas de cálculos.

Referências Bibliográficas

- determinant, C., Sinick, J., Yuan, Q. and Leeuwen, M. (2019). *Computational complexity of computing the determinant*. [online] Mathematics Stack Exchange. Available at: https://math.stackexchange.com/questions/81529/computational-complexity-of-computing-the-determinant [Accessed 12 May 2019].
- En.wikipedia.org. (2019). *Gaussian elimination*. [online] Available at: https://en.wikipedia.org/wiki/Gaussian_elimination [Accessed 9 May 2019].
- dummies. (2019). *How to Use Gaussian Elimination to Solve Systems of Equations - dummies*. [online] Available at: https://www.dummies.com/education/math/calculus/how-to-use-gaussian-elimination-to-solve-systems-of-equations/ [Accessed 9 May 2019].
- Ijens.org. (2019). [online] Available at: http://www.ijens.org/Vol%2011%20I%2001/119401-5353%20IJECS-IJENS.pdf [Accessed 12 May 2019].