PROGRAMAÇÃO EM MPI

João Victor de Mesquita Cândido dos Santos **RA:** 102028
Raphael Ribeiro Faria **RA:** 104120
Unidade Curricular: Programação Concorrente e Distribuída
Docente: Dr. Álvaro Luiz Fazenda

Nesta atividade abordou-se dois problemas distintos: um mais simples - cálculo numérico de logaritmo natural, e o problema FTCS para cálculo numérico de condução de calor. Desta vez, portanto, utilizou-se da ferramenta MPI para paralelizar ambos os problemas. Nela, trabalha-se com o sistema de troca de mensagens entre os processos, ou seja, a memória não é mais compartilhada.

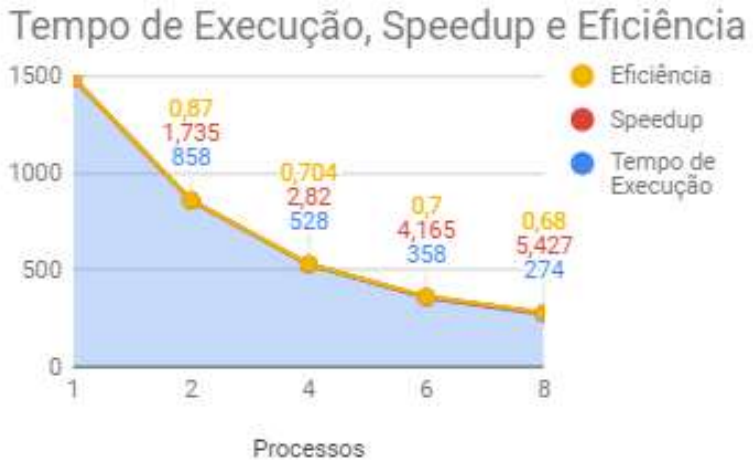
1. Cálculo do Logaritmo Natural

Para cálculo de logaritmo natural de um determinado número, o programa faz uma aproximação por meio de várias somas variando o denominador. Os cálculos podem ser divididos em partes para que os processos possam realizar uma fração do problema e, no final, reduzir todas essas frações em uma só soma final, que será o resultado. Isso é feito pelo MPI utilizando a função MPI_Reduce( ). O processo raiz então imprime o resultado já reduzido na tela. Isso foi feito e testado com variação no número de processos (com 1, 2, 4 e 8). Os resultados obtidos podem ser vistos através da Tabela 1 e pelos gráficos a seguir.

Número de processos	Tempo de Execução	Speedup	Eficiência
1	1486 ms	1	1
2	858 ms	1,735	0,87
4	528 ms	2,82	0,704
6	358 ms	4,165	0,7
8	274 ms	5,427	0,68

Tabela 1 - Cálculo do Logaritmo Natural

Com os dados acima, foi possível montar um gráfico que faz uma análise do número de processos, tempo de execução, *speedup* e eficiência que pode ser visto abaixo.



Ao analisar os dados da Tabela e do Gráfico é possível analisar que há uma diminuição do tempo de execução com o aumento do número de processos. Também é possível analisar que há uma queda significativa do tempo de 1 para 2 processos, além disso a partir de 4 processos a eficiência apresenta uma tendência linear, o que já era esperado. Considerando a dificuldade de manter a eficiência perto de 1 os valores obtidos foram consideravelmente bons.

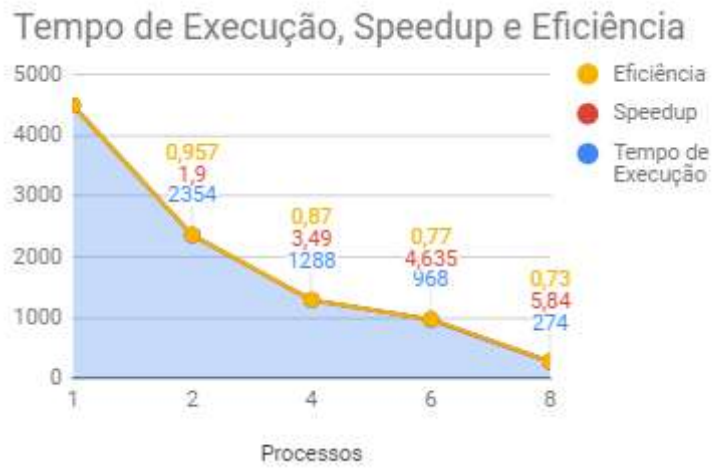
2. FCTS

No problema do FTCS, o vetor u de saída é calculado com base no próprio u de iterações anteriores.Assim, não há possibilidade de atribuir uma iteração a cada processo para não causar um efeito de cascata e por consequência resultando em um programa serial. Assim, foi dividido uma parte do vetor para cada processo, ou seja, o cálculo foi feito dividindo o tamanho do problema pela quantidade de processos. O último processo, deve ficar com o resto, que pode ser um número diferente do que dos outros processos. Assim que determinadas as posições start e first que o processo vai trabalhar no vetor é calculado o u inicial para cada um, e começam-se as iterações. Com o vetor u calculado, cada processo encontra seu máximo local e o índice do mesmo, e todos irão repassar essa informação para o processo raiz (processo 0). O processo raiz por sua vez, junta todas as soluções recebidas à sua própria, e encontra entre todas essas soluções qual é a maior global. No fim, este processo imprime a maior solução e seu índice. Os resultados podem ser vistos na tabela e no gráfico a seguir.

Número de processos	Tempo de Execução	Speedup	Eficiência
1	4486 ms	1	1
2	2354 ms	1,9	0,957
4	1288 ms	3,49	0,87
6	968 ms	4,635	0,77
8	274 ms	5,84	0,73

Tabela 2 - FCTS

Com os dados acima, foi possível montar um gráfico que faz uma análise do número de processos, tempo de execução, *speedup* e eficiência que pode ser visto abaixo.



Analisando o gráfico é perceptível uma queda tempo semelhante ao do primeiro exercício, porém o tempo cai pela metade com o dobro de número de processos. O speedup seguiu uma tendência linear como já era de esperado. Quando se trata de eficiência a mesma se manteve acima dos 70% diferente do primeiro problema.

3. Especificações

Todos os exemplos solicitados tiveram sua execução realizada no cluster do ICT-UNIFESP. Para 1, 2, e 4 processos, teve-se um nó com 1, 2, e 4 tasks, e para 6 e 8, tivemos dois nós com 3 e 4 tasks, respectivamente.