Relatório Projeto 2 – Sistemas Distribuídos

Grupo: Gabriel Henrique da Silva Gava, Gustavo Henrique de Oliveira Villar e Vitoria Zanon Gomes

Professor: Aleardo Manacero Junior

1. Apresentação:

Este relatório tem como o objetivo descrever as atividades desenvolvidas para o Projeto 2 da disciplina de Sistemas Distribuídos, do curso de Bacharelado em Ciência da Computação, assim como os resultados observados. Esse relatório divide-se em quatro seções, incluindo a atual. A segunda seção apresenta o problema a ser resolvido com o projeto em questão, a terceira seção apresenta o desenvolvimento do projeto, incluindo como executá-lo, e a quarta seção traz os resultados observados em relação à execução do projeto, em termos de tempo e precisão de valores, assim como a conclusão após análise dos dados e comparação com o projeto anterior.

2. Descrição do problema:

O problema relativo ao projeto é o mesmo do projeto anterior: o desenvolvimento de um programa, em linguagem C, que calcule a integral da função f(x) = $\sqrt{100^2 - x^2}$ no intervalo [0,100] e utilizando uma estrutura mestre-escravo. A diferença se dá na utilização de MPI (*Message Passing Interface*) ao invés de sockets, para a comunicação entre mestre e escravos, onde o mestre passará os intervalos a serem calculados da integral para os escravos, que farão os cálculos e o retornarão ao mestre, que apresentará o resultado final.

O programa deve oferecer a possibilidade de uso de 1, 2, 4 e 10 escravos, e deve utilizar os intervalos de discretização 0.0001, 0.00001 e 0.000001.

3. Desenvolvimento do projeto:

3.1 Sobre o programa:

O programa foi escrito em linguagem C, seguindo as instruções do problema, gerando um único arquivo intitulado *main.c*

O arquivo possui três funções principais, além da main: *Integral, Master* e *Slave*.

A primeira função é responsável por calcular a integral, recebendo como parâmetro um valor de gap e um de discretização, e retornando o valor da integral naquele intervalo especifico, naquela discretização especifica.

A segunda função implementa as atividades relativas ao mestre, que são o envio de mensagens para todos os slaves com os respectivos intervalos a serem calculados, e o recebimento dos cálculos finalizados por parte dos escravos. Nessa função também é realizada a soma dos cálculos vindos dos escravos, incrementando a cada iteração, de modo que ao final se obtenha o valor total da integral.

A terceira função implementa as atividades relativas aos escravos, que consistem basicamente em receber as instruções de cálculo do mestre, calcular a integral através da chamada da função *Integral* e retornar o resultado para o mestre.

Por fim, na *main*, são instanciadas as variáveis para medição do tempo de execução e para auxiliar na execução do MPI, a exibição dos dados na tela (valor final da integral, número de escravos utilizados e tempo de execução), o controle de quanto durará a comunicação entre mestre e escravos (ou seja, até que se calcule toda a integral) e a definição de quais processos serão escravos e qual será mestre (o critério é que caso o *rank* do processo seja 0, ele será o mestre, e os demais serão escravos).

Vale a pena citar que, na função *Master*, o resultado anterior da integral é sempre salva numa variável denominada *lastTotal*, pois devido a problemas de encerramento dos processos, mesmo após a obtenção do resultado final da integral, um dos processos escravos continua tentando se comunicar com o mestre, o que resulta em um valor *-nan* na variável final. Desse modo, para fins de garantir que se tenha salvo o valor correto ao final da execução do programa, foi necessária a criação dessa variável, uma vez que não conseguimos anular essa comunicação “extra” entre o escravo e o mestre.

3.2 Execução do programa:

Recomenda-se que o programa seja compilado e executado no sistema operacional Linux:

- Para compilar os arquivos: *make makefile compile*
- Para selecionar a quantidade de escravos a ser utilizada:
  - 1 escravo: *make makefile slave1*
  - 2 escravos: *make makefile slave2*
  - 4 escravos: *make makefile slave4*
  - 10 escravos: *make makefile slave10*

4. Resultados observados e Conclusões:

A tabela abaixo apresenta os valores relativos ao tempo de execução (em segundos) de cada cálculo, considerando diferentes quantidades de escravos e os diferentes intervalos de discretização. É necessário citar que

ao invés de 10 escravos, foram utilizados 7 escravos, pois era o valor máximo que os hardwares do grupo conseguiam calcular (8 processos: 7 escravos + 1 mestre):

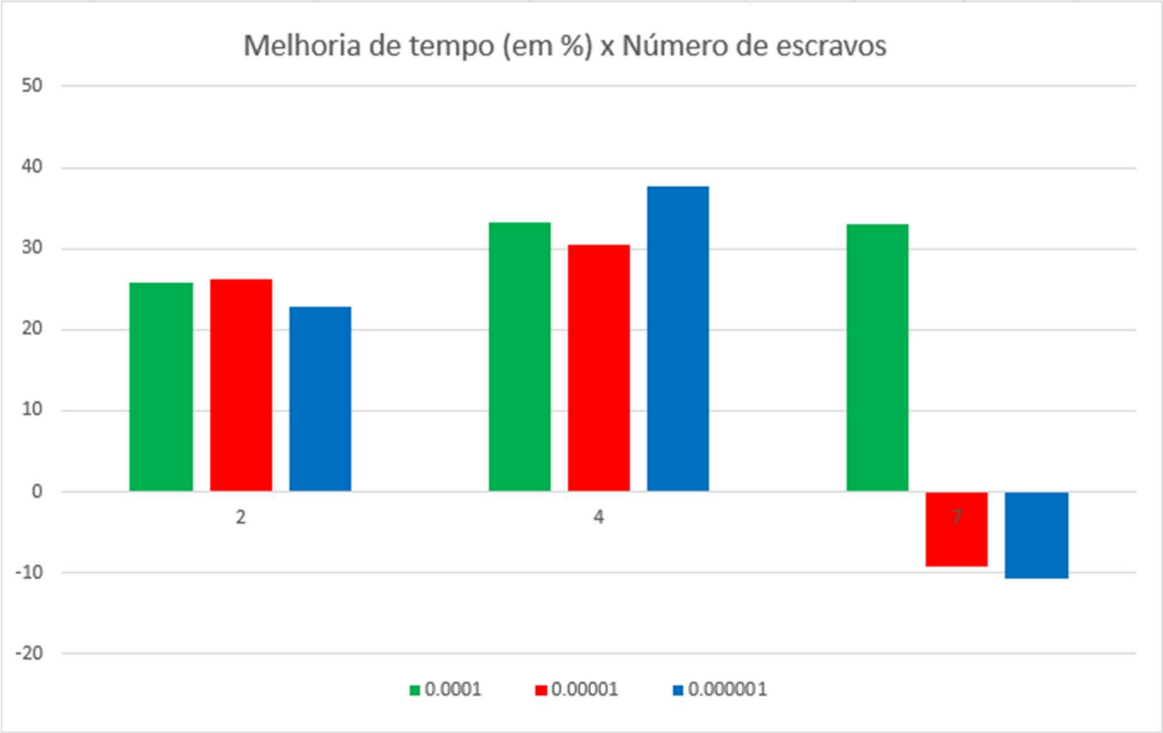
	0.0001	**0.00001**	**0.000001**
1 escravo	0.96	9.47	94.31
2 escravos	0.71	6.99	72.76
4 escravos	0.64	6.59	58.83
7 escravos	0.64	10.33	104.48

Observa-se que conforme o número de escravos aumenta de 1 para 2, e de 2 para 4, o tempo diminui substancialmente. Porém, de 4 para 7, com discretização de 0.0001, o tempo praticamente se mantém, e nos demais valores de discretização o tempo piora. Isso se deve, principalmente, ao uso do máximo de threads possível, que culmina numa disputa por recursos que compromete o tempo de execução, e pelo gerenciamento dessas threads e dos recursos por parte do sistema operacional, que não pode ser controlada e não se tem conhecimento acerca do critério utilizado.

Também podemos observar que conforme o valor de discretização diminui, o tempo aumenta proporcionalmente. Isso é esperado, uma vez que com um menor valor de discretização, mais comunicação entre o mestre e os escravos é demandada para o cálculo da integral, causando impacto no tempo final. Porém, apesar do aumento do tempo, o uso da distribuição de tarefas não é desencorajado, uma vez que os tempos com 2 e 4 escravos são melhores quando comparados ao uso de 1 escravo somente.

Por fim, vale a pena ressaltar que os tempos de execução da tarefa utilizando MPI foram substancialmente menores que utilizando sockets, como foi feito no projeto anterior, o que indica um melhor desempenho desse método, que pode ser atribuído à não necessidade de estabelecimento de comunicação entre mestre e escravos. Assim que os processos são criados, já estão prontos para se comunicarem livremente entre si.

O gráfico a seguir apresenta a melhoria de tempo (em %) quando se aumenta o número de escravos, em ambos valores de discretização. As porcentagens são tiradas quando comparadas aos tempos de execução quando utilizado somente um escravo.



Através desse gráfico, é possível visualizar como o aumento de escravos altera positivamente o tempo de execução (com exceção do caso de uso de 7 escravos, por razões previamente esclarecidas), o que demonstra que a distribuição de tarefas dentro de um sistema é benéfica para o desempenho do mesmo.