UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL
CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO DO TRABALHO PROGRAMAÇÃO PARALELA DISTRIBUÍDA

Felipe da Rocha Moralles Guterres - 023401

DOURADOS
JUNHO-2016

O trabalho proposto estabelece que faça a busca paralela para um conjunto grande de números desordenados. Dado um arquivo de entrada, deve-se feito a busca por algum elemento x do conjunto de elementos inteiros do arquivo, e que informe em que posições o elemento x se encontra no conjunto. Este programa será testado para 2 e 4 máquinas, assim tendo até 4 unidades de processamento para a busca.

Para testes foram utilizados os computadores dos laboratórios do curso de Ciência da Computação da Universidade Estadual de Mato Grosso do Sul, que possuem o sistema operacional Ubuntu 14.04 LTS, utilizando a linguagem C com a biblioteca MPI devidamente instalada nas máquinas executadas, as máquinas apresentam as seguintes configurações: processador Intel I5-240 6 MB 3.1GH 8Gb de memória RAM.

Para este trabalho foi utilizado a MPI (Message Passing Interface), que é um padrão de troca de mensagens portátil que facilita o desenvolvimento para aplicações paralelas. Para o padrão da MPI, uma aplicação é constituída por um ou mais processos que se comunicam, utilizando funções para envio e recebimento de mensagens entre os processos. É uma biblioteca que tem como objetivo poder fornecer uma eficiente comunicação entres os processos, permitindo implementações em ambientes heterogêneos sem haver a necessidade na alteração no código da implementação.

Utilizando principalmente as funções *send* e *receive* da MPI para realizar a comunicação entre os processos, a função *send* realiza o envio de uma mensagem do ponto de origem para o destino, a função *receive* recebe a mensagem estabelecendo qual foi o processo que enviou. Essas funções são do tipo bloqueantes, este tipo espera até que o *buffer* de recepção contenha a mensagem.

Dado um arquivo de entrada com 17 milhões e um de números aleatórios deve ser realizado a busca por um número inteiro, que é inserido por linha de comando no terminal no momento de execução do programa. O trabalho utiliza três arquivos *SHELL* para executar o algoritmo em paralelo. Primeiro deve-se abrir o terminal entrar na pasta do trabalho, assim o primeiro comando executado será: “bash ssh_configure.sh”, este comando libera o acesso do usuário para as máquinas que contém o IP no arquivo “hostfile”, que está na pasta do trabalho, assim não precisando colocar senha do usuário toda vez que o programa for executado, o que torna o programa mais rápido, em seguida deve-se executado o comando: “bash configure.sh”, nesse comando o programa é compilado utilizando o “Makefile”, que está na pasta do programa, após o executável é transferido para todas as máquinas do hostfile, com permissão do arquivo liberada. E por último deve ser executado o comando ”bash remote.sh X Y” que irá executar o programa, onde X é o número de máquinas utilizadas e Y é número a ser pesquisado.

O arquivo “main.c” contém código fonte do trabalho, o código contém três funções, a função “search” realiza a busca do elemento a ser buscado nos vetores auxiliares, a função a “cont_numbers” faz o cálculo para informar a quantidade de elementos no arquivo de entrada, e a função “break_vector” dividi o vetor inicial na quantidade máquinas utilizadas. Na rotina principal

realiza a contagem de elementos do arquivo de entrada, com a quantidade de elementos é criado um vetor principal que contém todos os elementos do arquivo de entrada, em seguida é realizado o cálculo para saber quantos elementos cada máquina irá conter tanto para a máquina mestre onde foi compilado o programa e as máquinas escravas, a máquina mestre pode conter mais elementos que as demais por causa na diferença da divisão entre as máquinas. Em seguida cada máquina receberá um vetor auxiliar contendo as partes do vetor principal, a máquina mestre começará da posição do vetor principal e as escravas com o resto do vetor principal, assim cada máquina fará a busca no seu conjunto auxiliar, a função de busca retorna um vetor resultante contendo em que posições o elemento a ser buscado se encontra, as máquinas escravas retornam o vetor resultante para a máquina mestre poder salvar em um arquivo.

Para os testes foi utilizado o mesmo código fonte, sem precisar fazer alteração apenas na hora de execução, na qual deve ser informado o número de máquinas. O teste foi realizado três vezes para uma, duas e quatro máquinas, onde obtém-se a média de cada um, analisando a tabela 1 verifica-se a diferença de tempo.

Processamento	P(1)	P(2)	P(4)
Tempo (1)	3,880seg	4,373seg	4,508seg
Tempo (2)	3,877seg	4,359seg	4,510seg
Tempo (3)	3,890seg	4,361seg	4,532seg
Média	3,882seg	4,364seg	4,517seg

Tabela 1: Comparativo de tempo pela quantidade de processos

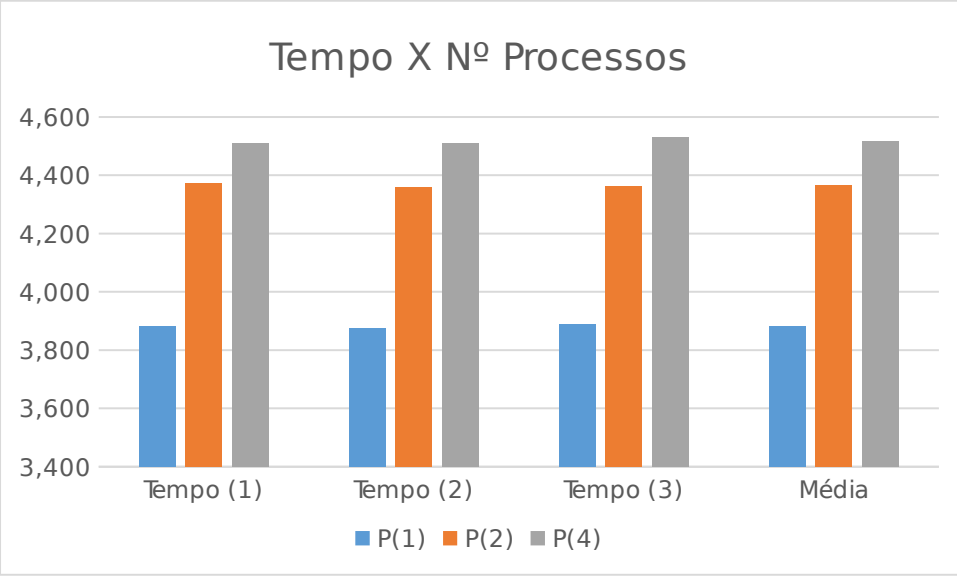

Gráfico 1: Gráfico do tempo pelo número de processos

Utilizando o tempo coletado dos testes será calculado o *speedup* e a eficiência. A eficiência é a razão entre o *speedup* e o número de processadores usado. Como mostra a tabela 2. O *Speedup* será calculado utilizado o tempo médio sequencial em razão do tempo médio paralelo.

Processo	P(2)	P(4)
Speedup	0,889559	0,859557
Eficiência	1,779119	3,438229

Tabela 2: Tabela do Speedup e eficiência

Gráfico 2: Gráfico do Speedup e Eficiência