PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Curso de Ciência da Computação (Campus de Poços de Caldas)

Busca de valores em uma matriz gerada aleatoriamente

**Gabriel Alvarenga da Cruz**

Fábio Donizetti Borges Faria

**Prof. Dr. João Carlos de M. Morselli Jr**

Poços de Caldas

2017

Busca de valores em uma matriz gerada aleatoriamente

Trabalho apresentado à disciplina de Processamento Paralelo

Curso de Ciência da Computação da Pontifícia Universidade Católica de Minas Gerais (Campus de Poços de Caldas).

Poços de Caldas

2017

Resumo

O trabalho acadêmico em questão utiliza os conceitos abordados em sala de aula e laboratório, bem como, conteúdos extraclasse, adquiridos em literatura, relacionados ao processamento paralelo.

Disponibilizar tarefas em vários computadores, é o grande atrativo da Programação Paralela. Ressaltando que existem algumas vantagens como alto desempenho, solução mais natural para problemas paralelos, maior facilidade de implementação de tolerância a falhas e desenvolvimento de programas modulares.

Sumário

[1 Introdução 1](#_Toc479365464)

[1.1 Considerações Iniciais 1](#_Toc479365465)

[1.2 Objetivos e Metas do Trabalho 1](#_Toc479365466)

[1.3 O Código 1](#_Toc479365467)

[2 Conceitos Introdutórios 2](#_Toc479365468)

[3 Revisão da Literatura 3](#_Toc479365469)

[3.1 Tipos de Programação 3](#_Toc479365470)

[3.2 Granulação das Tarefas 3](#_Toc479365471)

[3.3 Desempenho: Speedup e Eficiência 4](#_Toc479365472)

[3.4 Processamento Paralelo 5](#_Toc479365473)

[4 Desenvolvimento 7](#_Toc479365474)

[4.1 Descrição do Trabalho 7](#_Toc479365475)

[5 Conclusão 11](#_Toc479365476)

[Referências Bibliográficas 12](#_Toc479365477)

# Introdução

Este trabalho descreve conceitos sobre a busca de valores em uma matriz que tem os mesmos gerados de forma aleatória, a partir do relógio de máquina, comparando-os com valores informados pelo usuário. Apresenta a implementação sendo executada sequencialmente e paralelamente em um cluster com uma determinada quantidade de nós, na ocasião foram utilizados 4 nós.

## Considerações Iniciais

O Processamento Paralelo (*parallel processing*), tem como componente a linguagem C que é utilizada para o desenvolvimento de código. O objetivo deste trabalho é detectar se o valor informado pelo usuário esta contido dentro da matriz, sabendo-se que a matriz tem seus valores gerados aleatoriamente através do valor do relógio de maquina, ou seja, os valores se alternam a cada execução.

## Objetivos e Metas do Trabalho

Este trabalho tem como objetivo o desenvolvimento de códigos em linguagem C em paralelo e sequencial, utilizando a biblioteca Linux chamada MPI que tem a funcionalidade da divisão e execução do código em vários nós, para que haja um aumento de poder de processamento na execução das tarefas propostas aos processadores, bem como, demonstrar a quanto se assimilou do conteúdo demonstrado tanto na pratica quanto na teoria.

## O Código

O código gera uma matriz 300x300 populada com dados aleatórios baseados no relógio do sistema, então espera a entrada de dados do usuário e busca se a sequencia existe dentro da matriz. O código sequencial busca por força bruta do começo ao fim da matriz, enquanto que a versão em paralelo divide a matriz pelos nós do cluster para a busca.

# Conceitos Introdutórios

Este trabalho introduz conceitos e termos, bem como, definições necessárias para o entendimento e utilização de processamento paralelo e de clusters em sistemas LINUX. Descreve as ferramentas e bibliotecas que se utiliza para a implementação de sistemas MPI.

# Revisão da Literatura

## Tipos de Programação

* Programação Sequencial: A programação sequencial é definida como várias tarefas sendo executadas uma após a outra, seguindo a sequência de que foi escrita.
* Programação Concorrente: São várias tarefas computacionais sendo executadas concorrentemente (simultaneamente), podendo ser implementadas como programas separados ou como um conjunto de threads.
* Programação Paralela: São processos executados em diferentes processadores, trabalhando para resolver um único problema, vários cálculos ao mesmo tempo.

Neste trabalho utilizaremos a programação sequencial e a programação paralela.

## Granulação das Tarefas

A granulação se refere a forma como as tarefas são paralelizadas, sendo divididas em:

* Granulação Fina: O paralelismo é de baixo nível, tem-se grande número de processos pequenos e simples.

Unidade de Paralelismo: instruções/operações.

* Granulação Média: Paralelismo de nível médio, tendo-se vários processos.

Unidade de Paralelismo: procedimentos/funções/loops.

* Granulação Grossa: Paralelismo de alto nível, usando de poucos processos grandes e complexos.

Unidade de Paralelismo: Processos/Programas.

## Desempenho: Speedup e Eficiência

Uma das principais preocupações da programação paralela é reduzir o tempo de processamento de uma determinada aplicação (speedup) e manter a eficiência do código.

O speedup (Sp) é a relação entre o tempo para se executar um algoritmo em um único processador (T1) e o tempo para executá-los em p processadores (Tp) : Sp = T1/Tp.

A eficiência (Ef) é a relação entre speedup (Sp) e o número de processadores (p) :

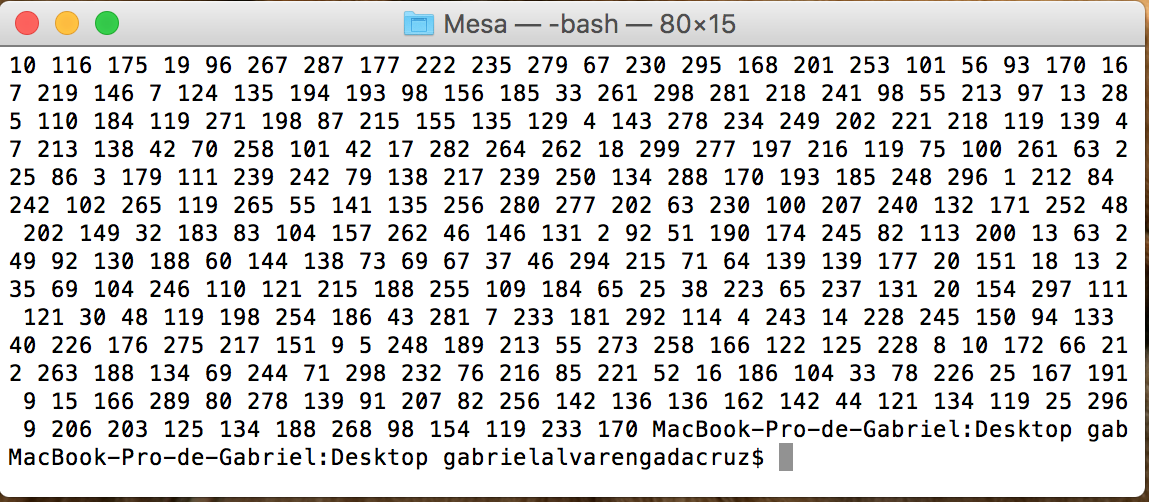
Ef = Sp/p.

No trabalho realizado por nós, através dos testes, fazemos a verificação se de fato ocorreu a população de toda a matriz com valores aleatórios, a partir do valor de relógio da maquina.

O gráfico abaixo mostra o tempo médio utilizado em paralelo e sequencial, durante a execução dos códigos.

Podemos verificar um speedup médio de 1,523. Considerando que utilizamos um cluster com 4 nós de processamento, a eficiência média ficou em torno de 0,380 (Todos os valores foram aproximados para 3 casas decimais).

Os valores aleatórios gerados na nossa primeira execução, foram os seguintes:



## Processamento Paralelo

O processamento paralelo (parallel processing) é uma forma eficiente do processamento da informação com ênfase na exploração de eventos simultâneos na execução de um software. A motivação para o processamento paralelo é a possibilidade de aumentar a capacidade de processamento de uma única máquina. Com a limitação tecnológica da velocidade das máquinas sequenciais, a solução empregada para aumentar o poder de processamento é a utilização de processadores em paralelo. Assim, com o paralelismo torna-se possível ultrapassar as limitações impostas pela utilização de um único processador, tais como frequência de operação, dissipação de potência e outras.

A utilização do paralelismo nos projetos de arquitetura de computadores tem possibilitado um aumento significativo na velocidade de processamento devido à execução simultânea de diversas tarefas. Contudo, os aspectos relacionados ao software paralelo e à paralelização dos programas são essenciais para o desempenho do sistema paralelo.

Muitas pessoas imaginam que máquinas paralelas (parallel machine) são somente os supercomputadores, como os listados no Top500 Supercomputers, mas isso não é verdade, pois uma máquina dual ou quad core já é uma máquina paralela. Contudo, não basta ter uma arquitetura paralela, ou seja com vários processadores, pois o software também tem que estar no formato de execução paralelo.

# Desenvolvimento

## Descrição do Trabalho The mechanics of the message exchange are documented in a Web service [description](http://translate.googleusercontent.com/translate_c?hl=pt-BR&sl=en&tl=pt&u=http://www.w3.org/TR/ws-arch/&rurl=translate.google.com.br&usg=ALkJrhjWhvT-ibzuEgRyTXveew4vbUt8lw#service_description) (WSD).

Utilizando a linguagem C,

* 1. **Código**

CÓDIGO DO PROJETO SEQUENCIAL:

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main (void)

{

int matriz[300][300];

int valorusuario,i,j,contido=0;

printf("\*\*\*\*\*\* Comparador de valores aleátorios de uma matriz com valor informado pelo usuário \*\*\*\*\*\*\n");

printf("\nPor favor informe um valor entre 1 e 300:\n\n");

scanf("%d", &valorusuario);

if (!(valorusuario > 0 && valorusuario < 300)) {

printf("Valor inválido!!!");

exit(0);

}

srand(time(NULL));

for (i = 0; i < 300; i++) {

for (j = 0; j < 300; j++) {

matriz[i][j] = rand () %299+1;

printf("%d ", matriz[i][j]);

}

printf("\n");

}

for (i = 0; i < 300; i++) {

for (j = 0; j < 300; j++) {

if (matriz[i][j] == valorusuario) {

printf("\nO valor esta contido na matriz e esta na posicao [%d][%d]\n", i,j);

contido = 1;

}

}

}

if (contido == 0) {

printf("\nO valor não esta contido na matriz");

}

return(0);

}

--------------------------------------------------------------------------------------------------------------

CÓDIGO DO PROJETO PARALELO:

#include <stdio.h>

#include <string.h>

#include "mpi.h"

main(int argc, char \*argv[])

{

int ret, rank, size, i, tag;

double tempo\_inicial,tempo\_final;

int j, matriz[300][300],contido=0,valorusuario;

MPI\_Status status;

int a=0;

ret = MPI\_Init(&argc, &argv);

ret = MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

ret = MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

tag=100;

tempo\_inicial = MPI\_Wtime();

if (rank == 0)

{

printf("\*\*\*\* Comparador de valores aleatórios de uma matriz com valor informado pelo usuário \*\*\*\*\n");

printf("\nPor favor informe um valor entre 1 e 300:\n\n");

scanf("%d", &valorusuario);

if(!(valorusuario > 0 && valorusuario < 300)){

printf("Valor Inválido !!!");

}

else{

srand(time(NULL));

for(i = 0; i < 30; i++){

for(j=0; j < 30; j++){

matriz[i][j] = rand () %301;

//printf("%d ", matriz[i][j]);// imprime os valores aleatorios da matriz, caso seja necessario verificar

}

//printf("\n");

}

ret = MPI\_Send(&a, 1, MPI\_INT, 1, tag, MPI\_COMM\_WORLD);

ret = MPI\_Send(&a, 1, MPI\_INT, 2, tag, MPI\_COMM\_WORLD);

ret = MPI\_Send(&a, 1, MPI\_INT, 3, tag, MPI\_COMM\_WORLD);

for(i = 0; i < 50; i++){

for(j=0; j < 50; j++){

if(matriz[i][j] == valorusuario){

printf("\nO valor esta contido na matriz e esta posicao [%d][%d]\n", i,j);

contido = 1;

}

}

}

ret = MPI\_Recv(&a, 1, MPI\_INT, 1, tag, MPI\_COMM\_WORLD, &status);

ret = MPI\_Recv(&a, 1, MPI\_INT, 2, tag, MPI\_COMM\_WORLD, &status);

ret = MPI\_Recv(&a, 1, MPI\_INT, 3, tag, MPI\_COMM\_WORLD, &status);

if(contido == 0){

printf("\n\nO valor nao esta contido na matriz\n\n");

}

}

}

else if (rank == 1){

ret = MPI\_Recv(&a, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD, &status);

for(i = 50; i < 100; i++){

for(j=50; j < 100; j++){

if(matriz[i][j] == valorusuario){

printf("\nO valor esta contido na matriz e esta posicao [%d][%d]\n", i,j);

contido = 1;

}

}

}

ret = MPI\_Send(&a, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD);

}

else if (rank == 2){

ret = MPI\_Recv(&a, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD, &status);

for(i = 100; i < 200; i++){

for(j=100; j < 200; j++){

if(matriz[i][j] == valorusuario){

printf("\nO valor esta contido na matriz e esta posicao [%d][%d]\n", i,j);

contido = 1;

}

}

}

ret = MPI\_Send(&a, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD);

}

else if (rank == 3){

ret = MPI\_Recv(&a, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD, &status);

for(i = 200; i < 300; i++){

for(j=200; j < 300; j++){

if(matriz[i][j] == valorusuario){

printf("\nO valor esta contido na matriz e esta posicao [%d][%d]\n", i,j);

contido = 1;

}

}

}

ret = MPI\_Send(&a, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD);

}

tempo\_final = MPI\_Wtime() - tempo\_inicial;

printf("%f ", tempo\_final);

ret = MPI\_Finalize();

}

# Conclusão

Conclui-se que a utilização da tecnologia do processamento paralelo é uma forma de se obter um ganho poderoso e agilidade em processos que teriam uma demora de conclusão, onde isso gera um vasto ganho em tempo e capital investido para que tal funcionalidade seja concluída. Como vimos no projeto descrito, onde o mesmo houve um ganho de tempo que superam as maquinas convencionais com um poder de processamento as vezes mais baixo que um Cluster.

# Referências Bibliográficas

|  |  |  |  |
| --- | --- | --- | --- |
| (EDSON) | | EDSON; O que é o processamento paralelo?. Disponível em : < http://professorferlin.blogspot.com.br/2011/08/o-que-e-processamento-paralelo.html > Acesso em: 06/04/17. | |
| (MORSELLI JUNIOR) | | MORSELLI JUNIOR, J. C. M; Slides de aula sobre Programação Paralela. Disponível em : <https://www.pucpcaldas.br/aularede/pluginfile.php/78285/mod\_resource/content/0/Aulas\_PP.pdf > Acesso em: 06/04/2017. | |