

Computação Paralela e Distribuída

Ano lectivo 2023-24

Rascunho - Será reorganizado e actualizado

Tema #4: Programação em Sistemas de Memória Distribuída
Título: Introdução ao MPI (continuação)

João José da Costa
joao.costa@isptec.co.ao

Coordenação de Engenharia Informática
Departamento de Engenharias e Tecnologias
Instituto Superior Politécnico de Tecnologias e Ciências

MPI

Message Passing Interface

Exemplo #2 – A peneira de Eratóstenes

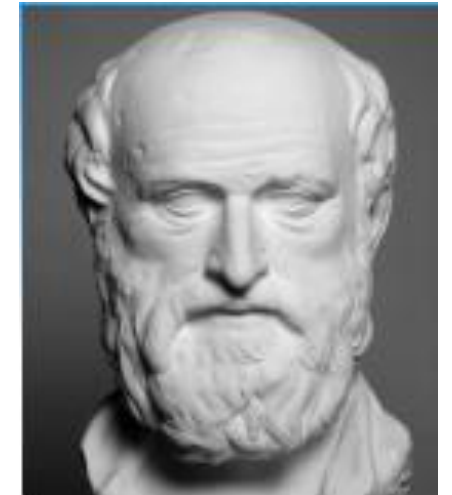
Tópicos

- A peneira de Eratóstenes
- Opções de decomposição de dados
- Desenvolvimento e análise do algoritmo paralelo
- Benchmarking
- Optimizações

A peneira de Eratóstenes

Algoritmo de Eratóstenes

- Algoritmo para encontrar todos os primos até n , proposto pelo matemático grego Eratóstenes (276-194 ac).
- Algoritmo
 1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$
 2. $K=2$
 3. Repete
 1. Marcar todos os múltiplos de k entre $k*k$ e n
 2. $K =$ menor número não marcado $> k$ até $k*k > n$
 4. Os números não marcados são primos



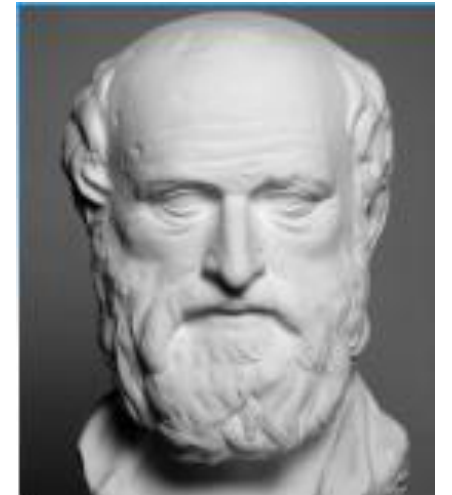
Qual é a complexidade em notação assintótica ????

Caracterizar o tempo de execução do algoritmo para tamanhos arbitrários das entradas

A peneira de Eratóstenes

Algoritmo de Eratóstenes

- Algoritmo para encontrar todos os primos até n , proposto pelo matemático grego Eratóstenes (276-194 ac).
- Algoritmo
 1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$
 2. $K=2$
 3. Repete
 1. Marcar todos os múltiplos de k entre $k*k$ e n
 2. $K =$ menor número não marcado $> k$ até $k*k > n$
 4. Os números não marcados são primos



Qual é a complexidade????

A peneira de Eratóstenes

Algoritmo de Eratóstenes – Exemplo com $n = 60$

- Algoritmo
 1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$
 2. $K=2$
 3. Repete
 1. Marcar todos os múltiplos de k entre $k*k$ e n
 2. $K =$ menor número não marcado $> k$ até $k*k > n$
 4. Os números não marcados são primos

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

A peneira de Eratóstenes

Algoritmo de Eratóstenes – Exemplo com $n = 60$

- Algoritmo

1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$

2. $K=2$

3. Repete

1. Marcar todos os múltiplos de k entre $k*k$ e n

2. $K =$ menor número não marcado $> k$ até $k*k > n$

4. Os números não marcados são primos

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

A peneira de Eratóstenes

Algoritmo de Eratóstenes – Exemplo com $n = 60$

- Algoritmo

1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$

2. $K=2$

3. Repete

1. Marcar todos os múltiplos de k entre $k*k$ e n

2. $K =$ menor número não marcado $> k$ até $k*k > n$

4. Os números não marcados são primos

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

A peneira de Eratóstenes

Algoritmo de Eratóstenes – Exemplo com $n = 60$

- Algoritmo

1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$

2. $K=2$

3. Repete

1. Marcar todos os múltiplos de k entre $k*k$ e n

2. $K =$ menor número não marcado $> k$ até $k*k > n$

4. Os números não marcados são primos

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

A peneira de Eratóstenes

Algoritmo de Eratóstenes – Exemplo com $n = 60$

- Algoritmo
 1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$
 2. $K=2$
 3. Repete
 1. Marcar todos os múltiplos de k entre $k*k$ e n
 2. $K =$ menor número não marcado $> k$ até $k*k > n$
 4. Os números não marcados são primos

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

A peneira de Eratóstenes

Algoritmo de Eratóstenes – Exemplo com $n = 60$

- Algoritmo

1. Criar uma lista de números naturais não marcados $2, 3, \dots, n$

2. $K=2$

3. Repete

1. Marcar todos os múltiplos de k entre $k*k$ e n

2. $K =$ menor número não marcado $> k$ até $k*k > n$

4. Os números não marcados são primos

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

A peneira de Eratóstenes

Aplicação da Metodologia de Foster

- Particionamento
 - Torna o teste de cada elemento da lista a tarefa primitiva.
- Comunicação
 - É necessário o envio do valor da peneira, k , para todas as tarefas
- Aglomeração + Mapeamento
 - Qual é a melhor forma de particionar a lista???
 - Decomposição de dados intercalados (cíclicos)
 - Fácil de determinar o “proprietário” de cada índice
 - Leva ao desequilíbrio de carga para este problema
 - Decomposição de dados em bloco
 - Equilibra cargas
 - Mas complicado determinar o proprietário se n não for um múltiplo de p .

A peneira de Eratóstenes

Decomposição de bloco

- Qual é a melhor forma de distribuir n elementos a p tarefas?
 - Algumas tarefas recebem $\left\lceil \frac{n}{p} \right\rceil$ elementos, outras recebem $\left\lfloor \frac{n}{p} \right\rfloor$.
- Qual tarefas recebem qual tamanho?
 - **Uma abordagem:** agrupar blocos maiores em tarefas de índice mais baixo.
 - Seja $r = n \% p$
 - Se $r = 0$, atribuir $\frac{n}{p}$ elementos por tarefa
 - Caso contrário
 - As primeiras r tarefas recebem $\left\lceil \frac{n}{p} \right\rceil$ e as restantes recebem $\left\lfloor \frac{n}{p} \right\rfloor$

A peneira de Eratóstenes

Decomposição de bloco

- Qual é a gama de elementos que cada tarefa recebe ?
 - Primeiro elemento da tarefa i : $i \left\lfloor \frac{n}{p} \right\rfloor + \min(i, r)$
 - Último elemento da tarefa i : $(i + 1) \left\lfloor \frac{n}{p} \right\rfloor + \min(i + 1, r) - 1$
 - Tarefa dono do elemento j : $\max \left(\left\lfloor \frac{j}{\left\lfloor \frac{n}{p} \right\rfloor} \right\rfloor, \left\lfloor \frac{j-r}{\left\lfloor \frac{n}{p} \right\rfloor} \right\rfloor \right)$

A peneira de Eratóstenes

Decomposição de bloco

- Abordagem agrupada: agrupar blocos maiores em tarefas de índice mais baixo.
 - Primeiro elemento da tarefa i : $i \left\lfloor \frac{n}{p} \right\rfloor + \min(i, r)$
 - Último elemento da tarefa i : $(i + 1) \left\lfloor \frac{n}{p} \right\rfloor + \min(i + 1, r) - 1$
 - Tarefa dono do elemento j : $\max \left(\left\lfloor \frac{j}{\left\lfloor \frac{n}{p} \right\rfloor} \right\rfloor, \left\lfloor \frac{j-r}{\left\lfloor \frac{n}{p} \right\rfloor} \right\rfloor \right)$
- Abordagem distribuída: distribui blocos maiores uniformemente.
 - Primeiro elemento da tarefa i : $\left\lfloor i \frac{n}{p} \right\rfloor$
 - Último elemento da tarefa i : $\left\lfloor (i + 1) \frac{n}{p} \right\rfloor - 1$
 - Tarefa dona do elemento j : $\left\lfloor \frac{p(j+1)-1}{n} \right\rfloor$

A peneira de Eratóstenes

Decomposição de bloco

- Abordagem agrupada: agrupar blocos maiores em tarefas de índice mais baixo.

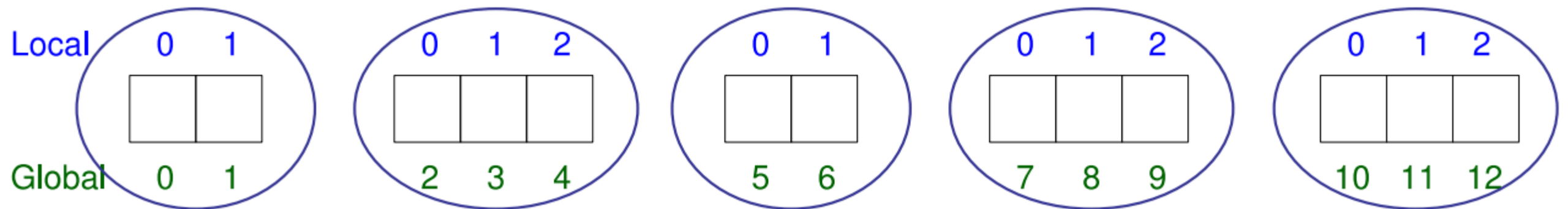


- Abordagem distribuída: distribui blocos maiores uniformemente.



A peneira de Eratóstenes

Decomposição de bloco – índices locais vs globais



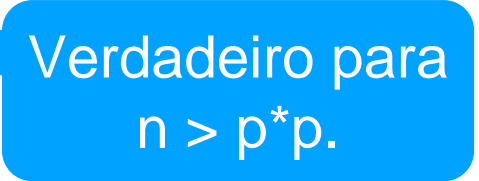
- Programa sequencial

```
for (i = 0; i < n; i++) {  
    ...  
}
```
- Programa Paralelo

```
t = BLOCO_TAMANHO(id, p, n);  
for (i = 0; i < t; i++) {  
    gi = i + BLOCO_INICIO(id, p, n);  
}
```

A peneira de Eratóstenes

Implementação paralela

- Utilize decomposição de bloco distribuída
 - Primeira tarefa responsável por selecionar o próximo primo a peneirar
 - Realiza o broadcast do novo primo a peneirar no fim de cada iteração
 - A primeira tarefa tem $\left\lfloor \frac{n}{p} \right\rfloor$ elementos, garanta que inclua o maior primo utilizado para peneirar, \sqrt{n}
 - O programa retorna o número de primos até n .
- 
- Verdadeiro para $n > p \cdot p$.

A peneira de Eratóstenes

Implementação paralela

1. Criar a lista de números naturais não marcados $2, 3, 4, \dots, n$

- Cada processo cria sua parte da lista

2. $K = 2$

- Todos os processos realizam isso

3. Repetir

- Marcar todos os múltiplos de k entre k^2 e n
 - Cada processo marca sua parte da lista
- $K =$ ao menor número não marcado $> K$ até que $k^2 > n$
 - Processa apenas 0 e transmite-o

4. Os números não marcados são primos

- Redução para computar o número de primos

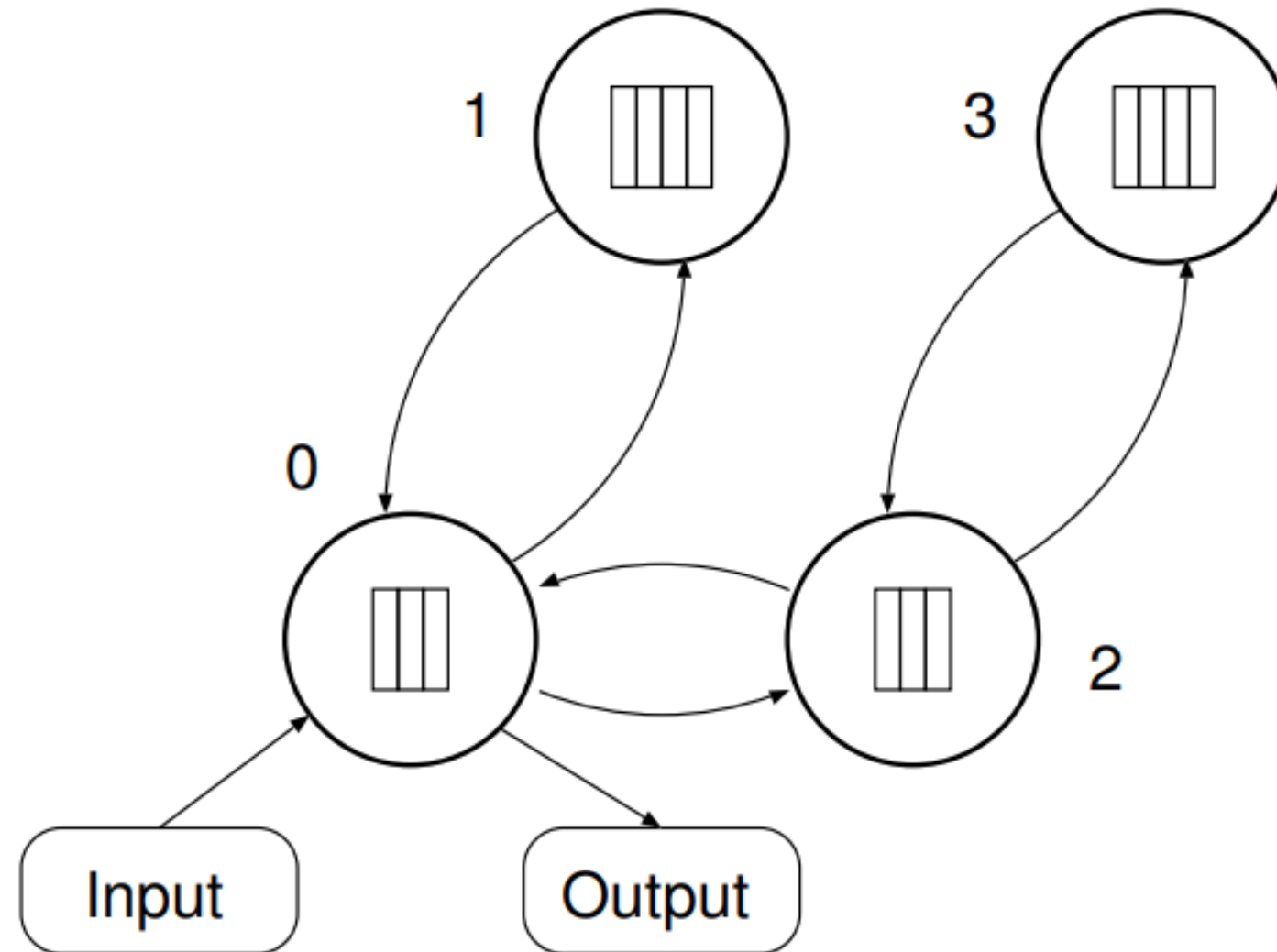
A peneira de Eratóstenes

Implementação paralela – Rotina MPI_Bcast

```
int MPI_Bcast (  
    void *buf, /* endereço do 1º elemento */  
    int count, /* # elementos para broadcast */  
    MPI_Datatype type, /* Tipo do elemento */  
    int root, /* ID do processo root */  
    MPI_Comm comm) /* Comunicador */  
  
MPI_Bcast (&k, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

A peneira de Eratóstenes

Implementação paralela – Grafo da tarefa/canal



Tarefas organizadas em uma **árvore binária** para operações de broadcast e de redução.

A peneira de Eratóstenes

Análise do algoritmo paralelo

- Seja α o tempo para marcar uma célula
- Tempo de execução serial: $\alpha n \log \log n$
- Tempo computacional do programa paralelo: $\frac{\alpha n \log \log n}{p}$
- Número de broadcast: $\frac{\sqrt{n}}{\log \sqrt{n}}$
- Tempo de broadcast: $\beta [\log n]$
- Tempo de redução: $\beta [\log n]$
- Tempo de execução paralela esperado:

Teorema de número primos.

$$\alpha \frac{n \log \log n}{p} + \beta \frac{\sqrt{n} [\log p]}{\log \sqrt{n}} + \beta [\log p]$$

A peneira de Eratóstenes

Código C com MPI (1 / 4)

```
#include <mpi.h>
#include <math.h>
#include <stdio.h>
#define BLOCO_INICIO(id,p,n) ((id)*(n)/(p))
#define BLOCO_FIM(id,p,n) (BLOCO_INICIO((id)+1,p,n)-1)
#define BLOCO_TAMANHO(id,p,n) (BLOCO_FIM(id,p,n) -
    BLOCO_INICIO(id,p,n)+1)
int main (int argc, char *argv[])
{
    ...
    MPI_Init (&argc, &argv);
    MPI_Barrier(MPI_COMM_WORLD);
    tempo = -MPI_Wtime();
    MPI_Comm_rank (MPI_COMM_WORLD, &id);
    MPI_Comm_size (MPI_COMM_WORLD, &p);
    if (argc != 2) {
        if (!id) printf ("Linha de comando: %s <m>\n", argv[0]);
        MPI_Finalize(); exit (1);
    }
```

A peneira de Eratóstenes

Código C com MPI (2 / 4)

```
n = atoi(argv[1]);
vi = 2 + BLOCO_INICIO(id,p,n-1);
vf = 2 + BLOCO_FIM(id,p,n-1);
t = BLOCO_TAMANHO(id,p,n-1);
pt = (n-1)/p;
if ((2 + pt) < (int) sqrt((double) n)) {
    if (!id) printf ("Muitos processos.\n");
    MPI_Finalize(); exit (1);
}
marcado = (char *) malloc (t);
if (marcado == NULL) {
    printf ("Não consegue alocar memória\n");
    MPI_Finalize();
    exit (1);
}
for (i = 0; i < t; i++) marcado[i] = 0;
```


A peneira de Eratóstenes

Código C com MPI (3 / 4)

```
if (!id) indice = 0;
primo = 2;
do {
    if (primo * primo > vi)
        primeiro = primo * primo - vi;
    else {
        if (!(vi % primo))
            primeiro = 0;
        else
            primeiro = primo - (vi % primo);
    }
    for (i = primeiro; i < t; i += primo) marcado[i] = 1;
    if (!id) {
        while (marcado[++indice]);
        primo = indice + 2;
    }
    MPI_Bcast (&primo, 1, MPI_INT, 0, MPI_COMM_WORLD);
} while (primo * primo <= n);
```

A peneira de Eratóstenes

Código C com MPI (4 / 4)

```
conta = 0;
for (i = 0; i < t; i++)
    if (!marcado[i]) conta++;
MPI_Reduce (&conta, &gconta, 1, MPI_INT, MPI_SUM,
    0, MPI_COMM_WORLD);
tempo += MPI_Wtime();
if (!id) {
    printf ("%d primos são menor ou igual a %d\n",
        gconta, n);
    printf ("Tempo total: %10.6f\n", tempo);
}
MPI_Finalize ();
return 0;
}
```

A peneira de Eratóstenes

Benchmarking

- Tempo de execução paralela esperado:

$$\alpha \frac{n \log \log n}{p} + \beta \frac{\sqrt{n} \lceil \log p \rceil}{\log \sqrt{n}} + \beta \lceil \log p \rceil$$

- Estimativa experimental de α , com $n = 10^8$, tempo de execução 24.9s

$$\alpha = \frac{24.9}{10^8 \log \log 10^8} = 52.64ns$$

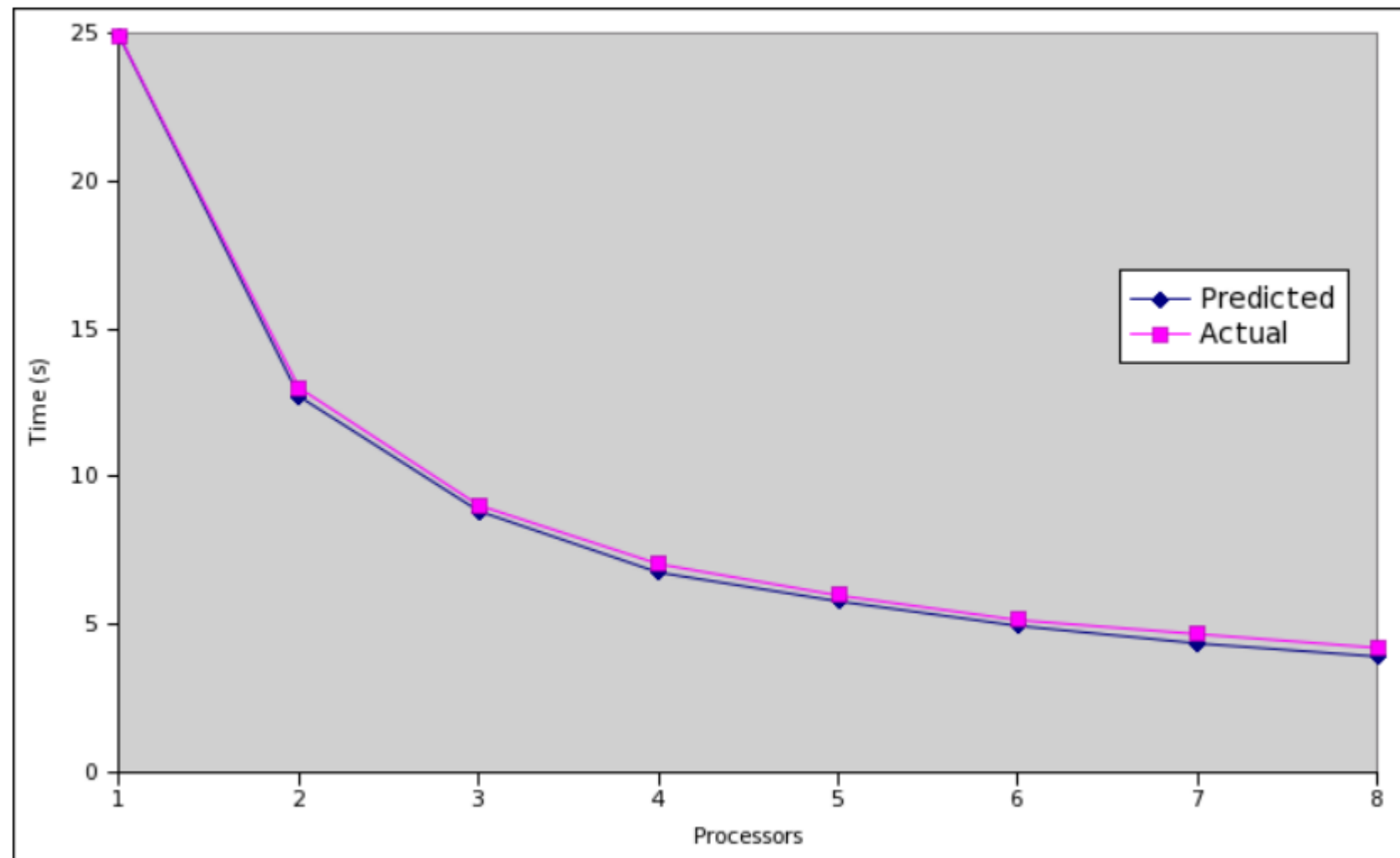
- A estimativa experimental de β , com $n = 10^8$, tempo de execução foi medido e comparado com a fórmula acima, para 2,3,...,8 processadores.

$$\beta = 250\mu s \quad \gg (\sqrt{n} / \log \sqrt{n}) \beta \lceil \log p \rceil > \alpha \sqrt{n} \log \log \sqrt{n}$$

A peneira de Eratóstenes

Resultados experimentais

p	Estimado	Actual
1	24.9	24.9
2	12.7	13.0
3	8.8	9.0
4	6.8	6.0
5	5.8	6.0
6	5.0	5.2
7	4.4	4.7
8	3.9	4.2



A peneira de Eratóstenes

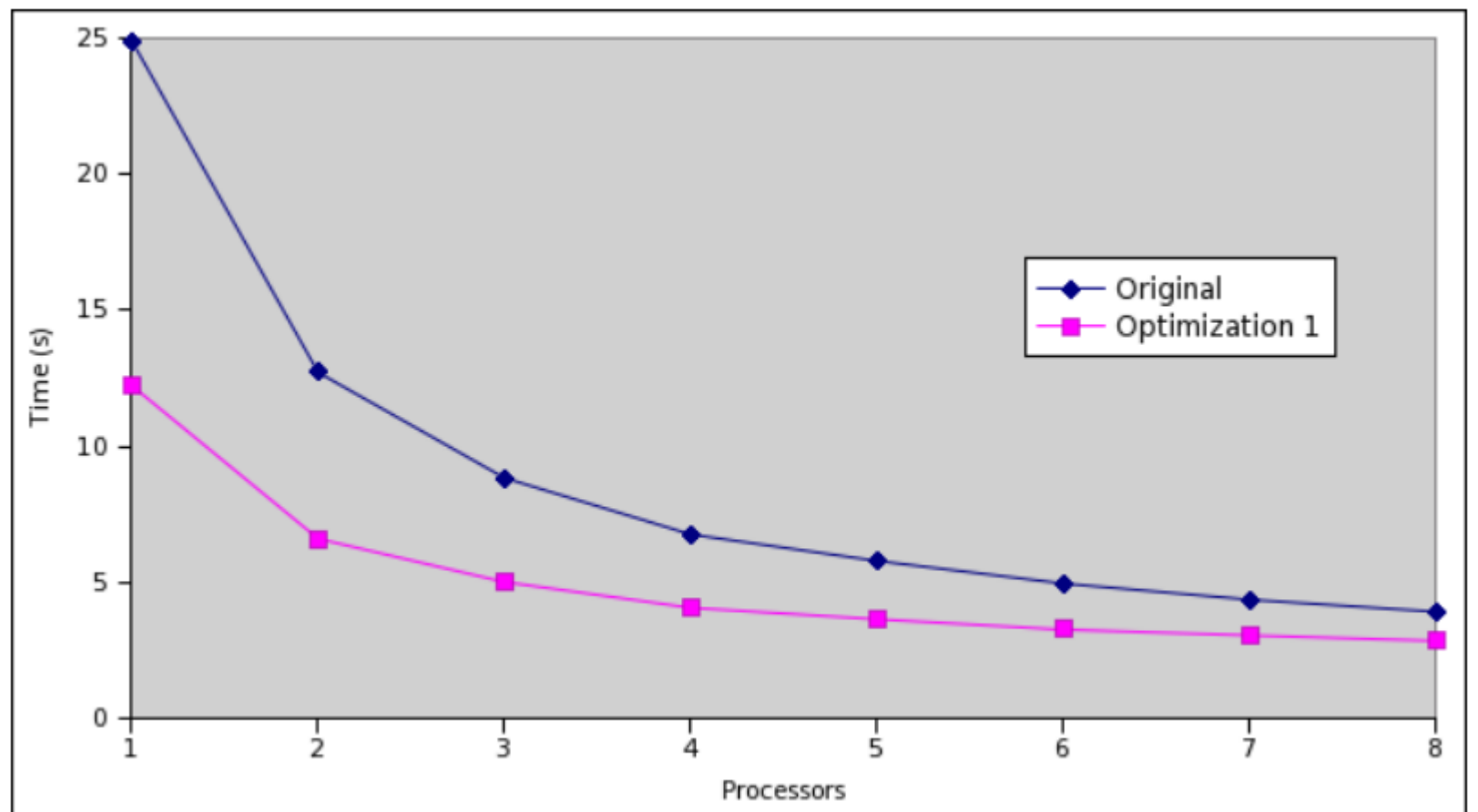
Melhoria ao programa I

- Todas as outras marcas estão em um número par -> **elimina-lhes!**

Novo tempo de execução paralela estimado

$$\alpha \frac{n \log \log n}{2p} + \beta \frac{\sqrt{n}[\log p]}{\log \sqrt{n}} + \beta[\log p]$$

p	Original	Optimizado
1	24.9	12.2
2	13.0	6.6
3	9.0	5.0
4	6.0	4.1
5	6.0	3.7
6	5.2	3.3
7	4.7	3.1
8	4.2	2.9



A peneira de Eratóstenes

Melhoria ao programa II

- Minimizar a comunicação
 - Duplicar a comunicação da peneira em todas as tarefas para evitar o broadcast
 - Trocar o tempo de comunicação pelo tempo de execução.

Substitui $\beta \frac{\sqrt{n}[\log p]}{\log \sqrt{n}}$ por $\alpha \sqrt{n} \log \log \sqrt{n}$

- Novo tempo de execução paralelo estimado

$$\alpha \left(\frac{n \log \log n}{2p} + \sqrt{n} \log \log \sqrt{n} \right) + \beta [\log p]$$

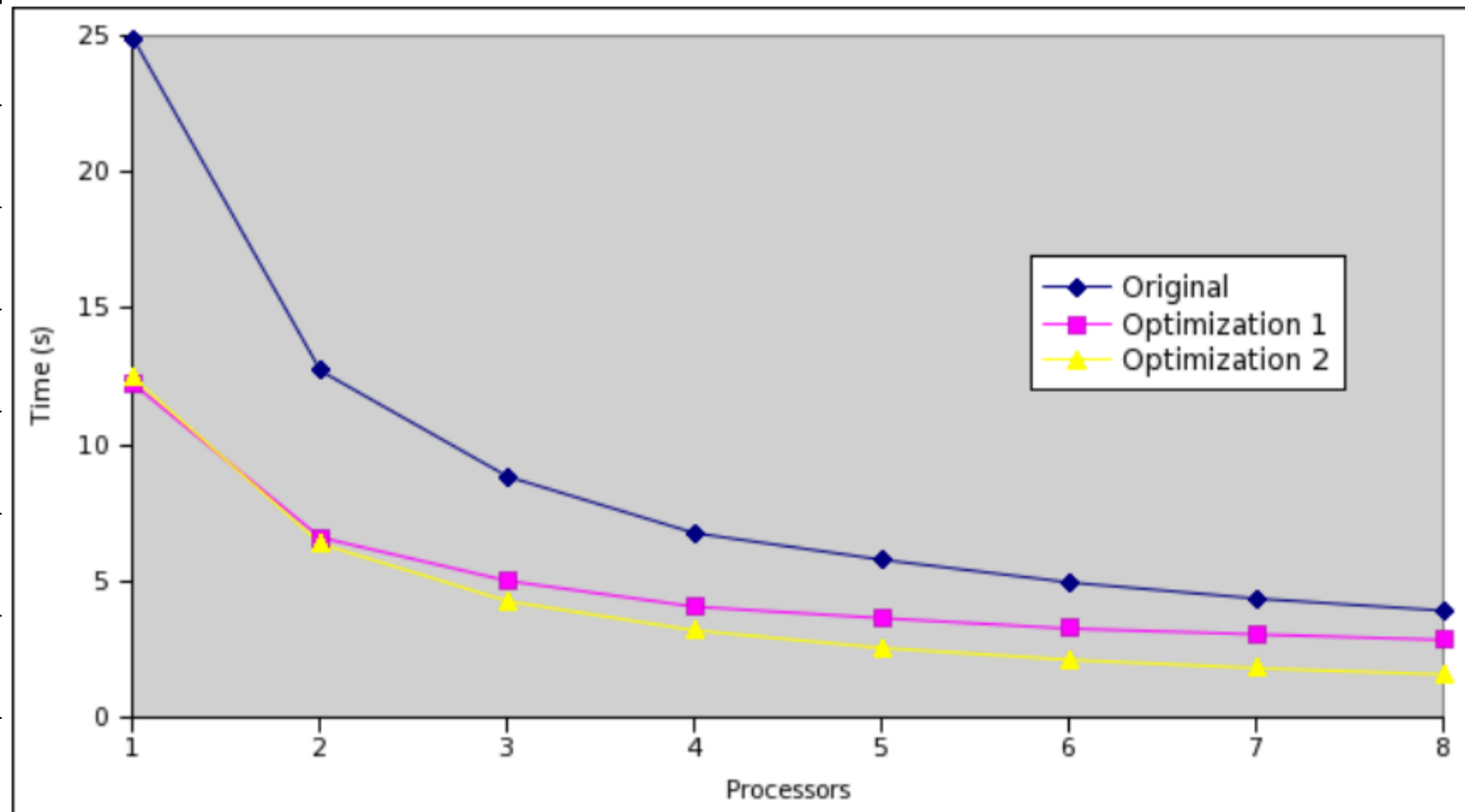
A peneira de Eratóstenes

Melhoria ao programa II

- Tempo de execução depois da segunda melhoria

p	Original	Optimizado
---	----------	------------

1	24.9	12.5
2	13.0	6.4
3	9.0	4.3
4	6.0	3.2
5	6.0	2.6
6	5.2	2.1
7	4.7	1.8
8	4.2	1.6



A peneira de Eratóstenes

Melhoria ao programa III

- Padrão de acesso a memória????

3	5	7	9
11	13	15	17
19	21	23	25
27	29	31	33
35	37	39	41
43	45	47	49
51	53	55	57
59	61	63	65
67	69	71	73
75	77	79	81

19	21	23	25
35	37	39	41
43	45	47	49
51	53	55	57
59	61	63	65
75	77	79	81

43	45	47	49
59	61	63	65
75	77	79	81

A peneira de Eratóstenes

Melhoria ao programa III

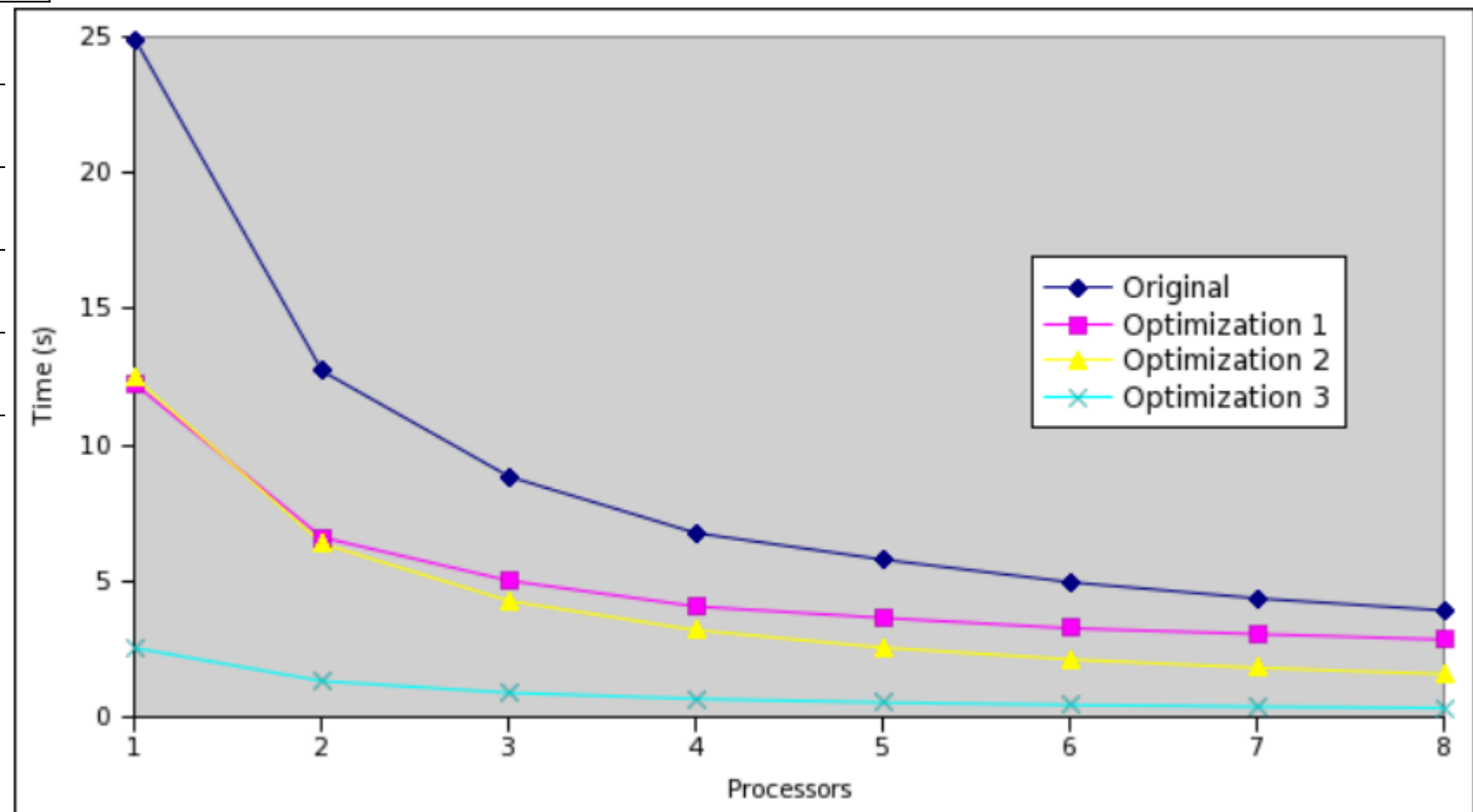
- Solução: Reorganizar os ciclos
 - Marcar cada elemento da matriz para todas as peneiras entre 3 e \sqrt{n}

3	5	7	9
11	13	15	17
19	21	23	25
27	29	31	33
35	37	39	41
43	45	47	49
51	53	55	57
59	61	63	65
67	69	71	73
75	77	79	81

A peneira de Eratóstenes

Melhoria ao programa III

p	Original	Opt.1	Opt.2	Opt.3
1	24.9	12.2	12.5	2.5
2	13.0	6.6	6.4	1.3
3	9.0	5.0	4.3	0.9
4	7.1	4.1	3.2	0.7
5	6.9	3.7	2.6	0.5
6	5.2	3.3	2.1	0.5
7	4.7	3.1	1.8	0.4
8	4.2	2.9	1.6	0.3



Revisão

- A peneira de Eratóstenes
- Opções de decomposição de dados
- Desenvolvimento e análise do algoritmo paralelo
- Benchmarking
- Optimizações

Referências

- Consultar a pasta “References” dentro da pasta do tema.

Bom trabalho!!!!