

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE
COMPUTAÇÃO

SSC0143
PROGRAMAÇÃO CONCORRENTE - TURMA B

Palíndromos

PROFESSOR DR. JULIO ESTRELLA

Grupo 06:

Bruno Junqueira Adami

Lucas Junqueira Adami

Lucas Lobosque

Números USP:

6878762

6792496

6792645

1 de maio de 2012

1 Introdução

O problema proposto é o de desenvolver versões de um programa paralelo para realizar as tarefas de determinar a ocorrência de palíndromos em dois textos especificados. Além disso, no texto maior, uma vez encontrado o palíndromo, é preciso determinar se a soma dos números correspondentes ao mapeamento do código ASCII de cada caracter da palavra é um número primo. Para calcular se o número é primo, o algoritmo de Crivo de Erastótenes deve ser utilizado. As bibliotecas OpenMP e MPI foram utilizadas para realizar o trabalho paralelo. Para realizar o desenvolvimento da proposta, o projeto foi separado em três partes:

- Execução do algoritmo do crivo.
- Leitura dos arquivos de entrada.
- Cálculo dos palíndromos.

2 O projeto OpenMP

Neste projeto, a paralelização do código foi feita nos loops do programa através das chamadas dos macros da biblioteca. O programa segue um fluxo contínuo e possui blocos de código executados em paralelo. Inicialmente, o cálculo do crivo é feito. Após esse passo, os arquivos são lidos e as palavras lidas são entregues ao verificador de palíndromos.

2.1 O parser

TODO.

2.2 Cálculo dos palíndromos

O algoritmo de cálculo dos palíndromos é simples. Sua complexidade é $O(n)$, pois baseia-se em apenas um loop para verificar a palavra. Além disso, ele já soma os valores ASCII dos caracteres para responder se a soma total é um número primo. Para a paralelização do algoritmo, o macro citado no código 1 foi utilizado. Neste passo, a decomposição de dados foi utilizada. O vetor que contém a palavra é dividido em sessões que serão computadas pela OpenMP através da chamada do macro. Cada partição executa o loop com suas iterações correspondentes em paralelo. As definições `PALINDROME_N_THREADS` e

```
1 #pragma omp parallel for num_threads(PALINDROME_N_THREADS)
2   schedule(dynamic, PALINDROME_BLOCK_SIZE) reduction(+:sum)
3   reduction(&&:palindrome)
```

Código 1: Macro que paraleliza o algoritmo do palíndromo

`PALINDROME_BLOCK_SIZE` são passadas ao programa através do arquivo `makefile`. A primeira diz quantas threads serão utilizadas na paralelização do loop. A segunda, quantas iterações cada thread irá realizar. A palavra `dynamic` define que não haverá uma ordem na distribuição das iterações para os loops.

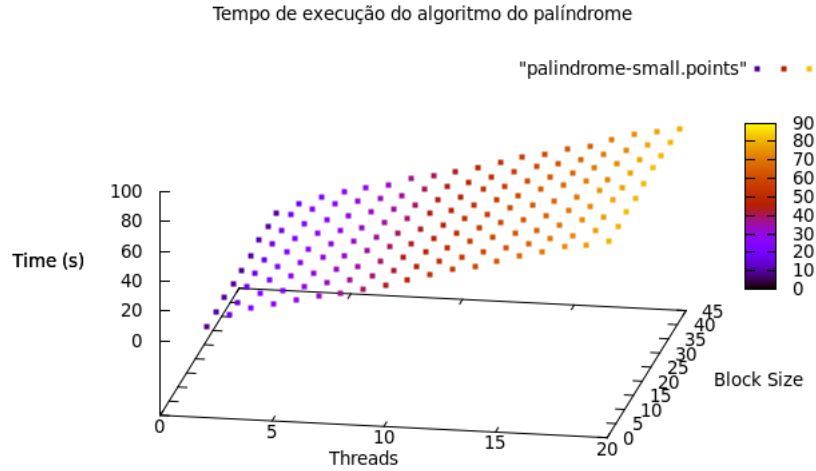


Figura 1: Resultados do algoritmo do palíndromo para o arquivo menor

Neste macro também estão definidas as reduções da soma para verificar a primalidade e da validade do palíndromo.

Alguns testes foram aplicados isoladamente do resto do sistema para testar a performance do algoritmo e suas diferentes configurações. Uma leitura cega foi feita dos dois arquivos de entrada e os resultados obtidos estão presentes na figura 1 e na figura 2. Os valores do números de threads e o tamanho do bloco que cada thread executa foram baseados em valores estimados. Um cálculo do tamanho médio das palavras provenientes da leitura cega apontou que para o arquivo menor, essa média valia 40 e para o maior, 4.

2.3 O algoritmo do crivo

TODO.

3 O projeto MPI

O projeto MPI foi desenvolvido seguindo a estrutura da figura 3. Todos os processos do programa têm suas chamadas principais executadas no início do programa. Existem três tipos de processos.

O primeiro é o processo principal, também chamado de mestre principal. O mestre principal é responsável por controlar os processos do segundo tipo, chamados de processos mestres. Ele é responsável por parar os processos. Há quatro processos do segundo tipo, o mestre crivo, o mestre palíndromo e os mestres parser maior e menor.

Os mestres parser maior e menor são os processos responsáveis pela leitura dos arquivos. Neste projeto, diferentemente do projeto OpenMP, ambos os arquivos são processados em uma só execução. Cada mestre desse tipo lê um arquivo e manda mensagens ao mestre palíndromo para que ele execute seu

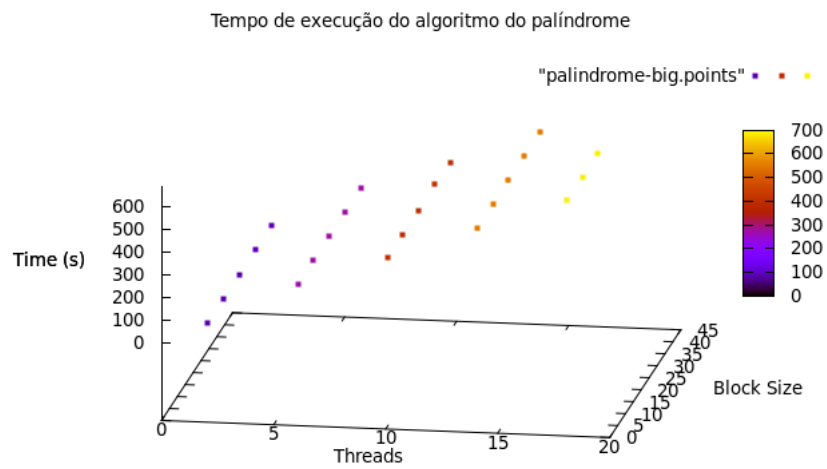


Figura 2: Resultados do algoritmo do palíndromo para o arquivo maior

papel de encontrar palíndromos e números primos. TODO: falar mais.

O mestre palíndromo coordena vários processos escravos designados a ele, os escravos palíndromos, que são do terceiro tipo. Basicamente, o mestre ordena que seus escravos executem requisições dos parsers e ele armazena as respostas para fornecê-las no final da execução.

TODO: falar do mestre crivo.

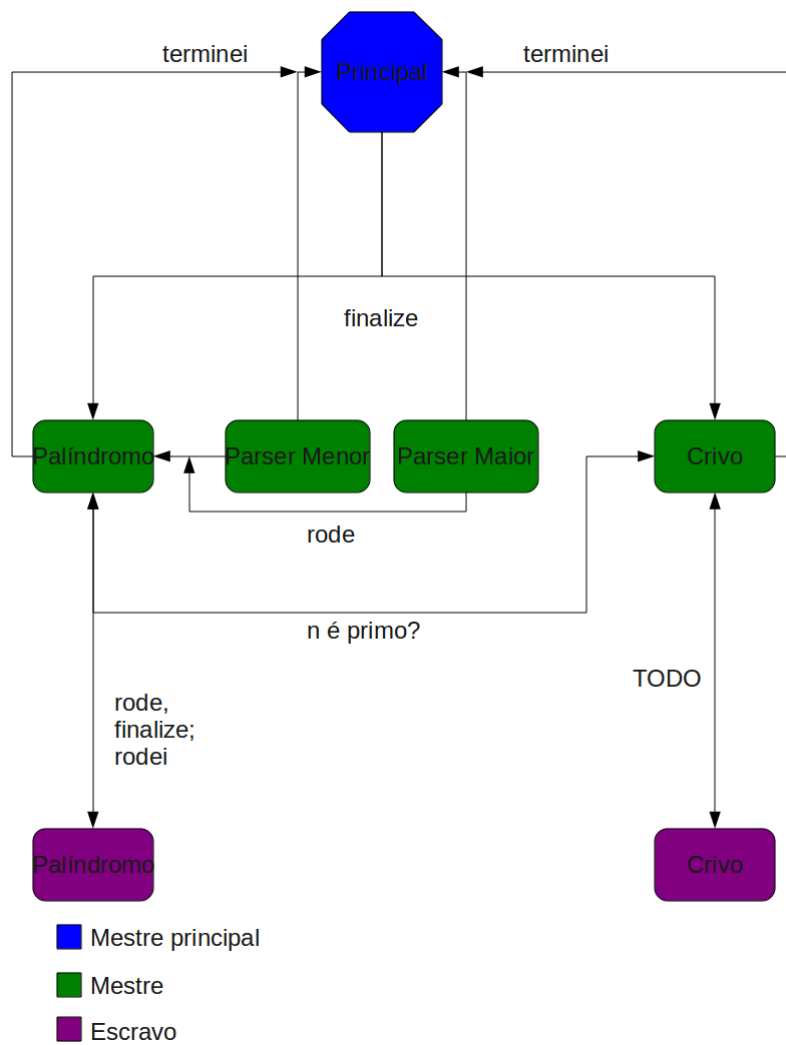


Figura 3: A estrutura do projeto MPI

Referências

[OpenMP] <http://bisqwit.iki.fi/story/howto/openmp/>

[OpenMP] <http://openmp.org/wp/>

[Gnuplot] <http://www.duke.edu/~hpgavin/gnuplot.html>