Relatório da Atividade Prática 3: Programação Multithread

Laila Mota[1], **Matheus Hofstede**[1], **Alberto Neto**[1], **Rafael Nagy**[1]

[1]Instituto de Matemática e Estatística – Departamento de Ciência da Computação
Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

1. Considerações gerais

O objetivo deste projeto é a realização da implementação de vários programas de forma a fortalecer os aspectos teóricos e práticos relacionados à programação paralela, a partir da utilização de múltiplas threads, além disso, entender as primitivas básicas pra multiplicação de threads do padrão POSIX Threads mais detalhado adiante, dessa forma analisando de que forma o paralelismo impacta no desempenho dos programas.

Como os membros da equipe já são familiarizados com lógica de programação, com a linguagem C e com a utilização do git, optou-se por utilizar um repositório git para gestão, controle de versão e trabalho colaborativo. Não foram necessárias grandes revisões em outros aspectos, com a exceção da biblioteca pthreads.h.

A equipe realizou um estudo sobre a biblioteca e suas funções previamente para a correta implementação nos programas solicitados pela atividade.

O repositório supracitado está disponível através do link https://github.com/hofstede-matheus/so-ufba-lab-03-multi-thread, onde é possível encontrar as orientações e comandos de execução e os relatórios desta atividade.

2. Estatística Multithread - estatistica.c

O objetivo do programa é o cálculo de diversos valores estatísticos para uma lista de números.

Para tanto, foi criado o programa estatistica.c que recebe um vetor de inteiros e realiza em threads diferentes o cálculo de média, valor mínimo e valor máximo. Os argumentos são passados como texto e transformados em número pela função atoi().

Para tornar possível a criação e manipulação de threads foi feita a inclusão da biblioteca pthread.h, onde estão definidas mais de 60 funções para manipulação de threads estabelecido pelo IEEE como padrão POSIX threads (IEEE_1003.1c), ou Pthreads.

A função pthread_create foi utilizada para inicializar as threads, recebendo em seu argumento um endereço de um dado pthread_t * threads. O endereço da função é passado como parâmetro (medio, menor, maior), essa função tem como retorno um valor void *. A função pode receber um atributo para configuração do thread. No caso do exercício, foi passado o argumento NULL, indicando que o thread será configurado conforme o padrão.

A função pthread_join recebe como argumentos a estrutura de controle do thread, do tipo pthread_t, que será finalizado e o valor NULL para o valor de retorno do thread. A função pthread_exit é utilizada para finalizar as threads.

As funções medio, maior e menor, são executadas conforme esperado.

3. Primos Multithread - primos.c

A programação para este execício utilizou a mesma biblioteca pthread.h de forma análoga ao exercício anterior. A função ePrimo calcula os primos menores ou iguais ao numero indicado.

4. Cálculo Multithread de π - calcpi.c

A mesma biblioteca pthread.h foi utilizada para a solução deste exercício.

O número π é gerado através de sucessivas somas e subtrações, utilizando a série de Leibniz para o cálculo de π.

Para tanto foi definida a variável qtdTotalDeTermos como 1000000000 e, conforme indicado, foram utilizadas variáveis double e long (long) para os cálculos.

Ao final é exibida a mensagem abaixo:

> "Foram utilizadas <long double/qtdThreads> threads para calcular <long long double/qtdTotalDeTermos> termos de PI, com <long float 3 casas decimais de precisão/tempoExecucaoTotal> segundos de execução."

5. Multiplicação Multithread de matrizes - mulmatriz.c

A mesma biblioteca pthread.h foi utilizada para a solução deste exercício.

O objetivo era a realização da multiplicação de duas matrizes $C_{m*n} = A_{m*p} * B_{p*n}$, usando até $N = m * n$ threads simultâneas.

Conforme indicado, foram utilizadas variáveis double para execução dos cálculos.

Ao final é exibida a mensagem abaixo:

> "Foram utilizadas <decimal/numThreads> threads na multiplicação da matriz 1000x1000, com <long float 3 casas decimais de precisão/tempoExecucaoTotal> segundos de execução."

Anexo A

6. Instruções de compilação

Para a compilação dos códigos, foi utilizado um makefile que contém as instruções de como compilar os programas. Abaixo seguem instruções suscintas de sua utilização, conforme consta no arquivo README.md do repositório git.

> Clone o projeto, e no diretório raiz, execute em uma linha de comando:
> make estatistica (para criação do executável do programa estatística.c)
> make primos (para criação do executável do programa primos.c)
> make pi (para criação do executável do programa calcpi.c)
> make multmatriz (para criação do executável do programa multmatriz.c)
> Os arquivos executáveis gerados estarão na pasta build/ e podem ser executados através do comando ./nomeDoPrograma através do terminal.

Para execução dos códigos de forma individual deixamos abaixo exemplos das instruções para cada programa.

estatistica.c

O programa deve ser executado desta maneira:

```
gcc estatistica.c -std=c99 -Wall -o estatistica -lm -lpthread
./estatistica 90 81 78 95 79 75 8
```

primos.c

O programa deve ser executado desta maneira:

```
gcc primos.c -std=c99 -Wall -o primos -lm -lpthread
./primos <numero>
```

Onde <numero> é o limite para mostrar os numeros primos, menores ou igual a ele.

calcpi.c O programa deve ser executado desta maneira:

```
gcc calcpi.c -std=c99 -Wall -o calcpi -lm -lpthread
./calcpi
```

multmatriz.c

O programa deve ser executado desta maneira:

```
gcc multmatriz.c -std=c99 -Wall -o mulmatriz -lm -lpthread
./multmatriz MatrizA MatrizB
```

Anexo B

Os threads implementados são preemptivas ou cooperativas? Explique sua resposta.
Na maioria dos casos, ela é cooperativa pois uma complementa a outra, tornando o cálculo mais rápido, principalmente quando são utilizados dois núcleos. Entretanto, observamos que o aumento no número de threads não quer dizer necessariamente que teremos um cálculo mais rápido. Por exemplo, na máquina Monoprocessada observamos que ao aumentar a quantidades de threads, a execução parece que ”se perde”um pouco, tornando mais complexa essa junção, sendo então não muito vantajoso o uso de muitas threads na máquina Monoprocessada.
Que modelo de threads o sistema operacional que você usou implementa (N:1, 1:1 ou N:M)? Como isso pode ser deduzido a partir dos experimentos?
N:1, ou seja, N threads do processo mapeadas em uma thread de núcleo, em ambos os casos (mono e dual processada), com a diferença que, em máquinas dual processadas, as threads são divididas igualmente entre os núcleos.

Abaixo, seguem os gráficos gerados a partir da execução dos programas e as tabelas com tempo de execução e coeficiente de variação.

Tabela 1: Execução do programa de Cálculo de π em máquina Mono processada

Threads	T1 (s)	T2 (s)	T3 (s)	T4 (s)	T5 (s)	T.Med. (s)	C. de Variação (%)
1	2,909	3,233	3,121	2,948	3,047	3,0516	3,8443
2	3,185	3,211	3,06	3,216	2,953	3,125	3,2972
4	3,267	3,004	3,101	2,891	3,059	3,0644	4,0267
8	3,174	2,983	2,993	3,141	2,996	3,0574	2,6986
16	2,92	3,052	3,06	3,168	3,091	3,0582	2,627
32	2,973	3,059	3,079	2,86	3,068	3,0078	2,7565

Tabela 2: Execução do programa de Cálculo de π em máquina Dual Procesada

Threads	T1 (s)	T2 (s)	T3 (s)	T4 (s)	T5 (s)	T.Med. (s)	C. de Variação (%)
1	2,903	2,797	2,814	2,834	2,878	2,8452	1,3916
2	1,699	1,522	1,602	1,552	1,651	1,6052	4,0044
4	1,5	1,464	1,575	1,524	1,516	1,5158	2,3795
8	1,538	1,424	1,629	1,557	1,51	1,5316	4,3522
16	1,548	1,501	1,61	1,486	1,552	1,5394	2,8379
32	1,543	1,467	1,43	1,482	1,484	1,4812	2,4622

Tabela 3: Execução do programa de Cálc. de Matriz em máquina Mono Procesada

Threads	T1 (s)	T2 (s)	T3 (s)	T4 (s)	T5 (s)	T.Med. (s)	C. de Variação (%)
1	6,802	6,905	6,601	6,765	6,792	6,773	1,4507
2	7,023	6,398	6,233	6,356	6,455	6,493	4,2331
4	7,182	6,875	6,886	6,658	6,638	6,8478	2,8759
8	7,071	7	6,904	6,59	7,374	6,9878	3,6287
16	7,208	6,349	6,737	6,911	7,033	6,8476	4,2782
32	6,953	6,398	6,783	6,514	6,341	6,5978	3,5438

Tabela 4: Execução do programa de Cálc. de Matriz em máquina Dual Procesada

Threads	T1 (s)	T2 (s)	T3 (s)	T4 (s)	T5 (s)	T.Med. (s)	C. de Variação (%)
1	6,334	6,458	6,622	6,349	6,172	6,387	2,3301
2	3,487	3,476	3,64	3,5	3,411	3,5028	2,145
4	3,368	3,219	3,398	3,445	3,357	3,3574	2,2522
8	3,72	3,805	3,585	3,39	3,41	3,582	4,5912
16	3,718	3,489	3,385	3,331	3,226	3,4298	4,8764
32	3,132	3,264	3,204	3,31	3,4	3,262	2,7968

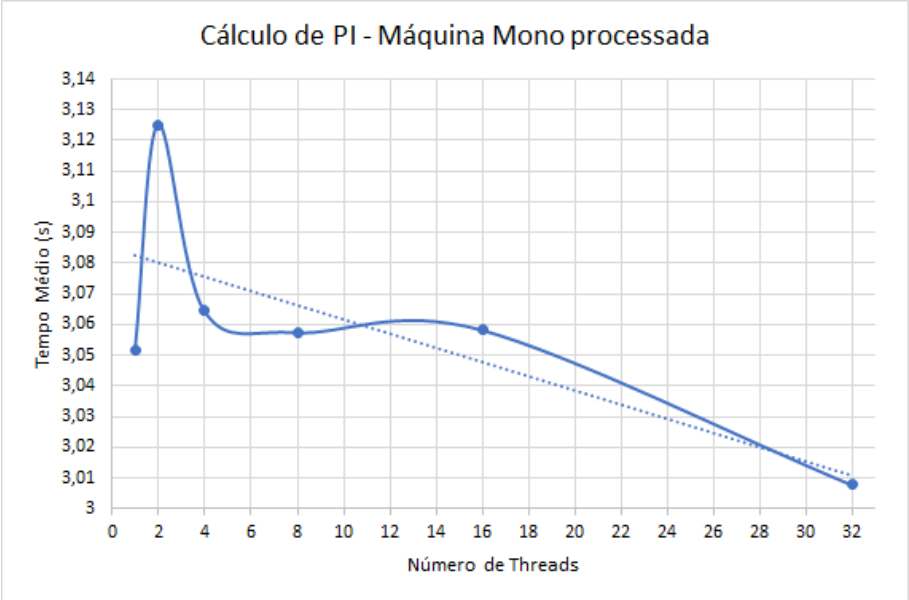

Figura 1: Calculo de π em máquina Mono processada. Tempo x Threads

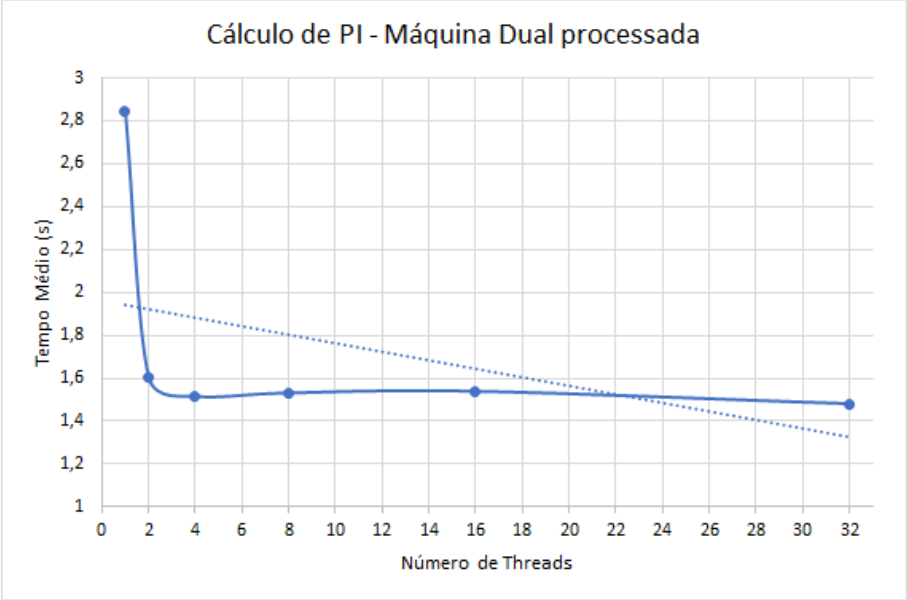

Figura 2: Calculo de π em máquina Dual processada. Tempo x Threads

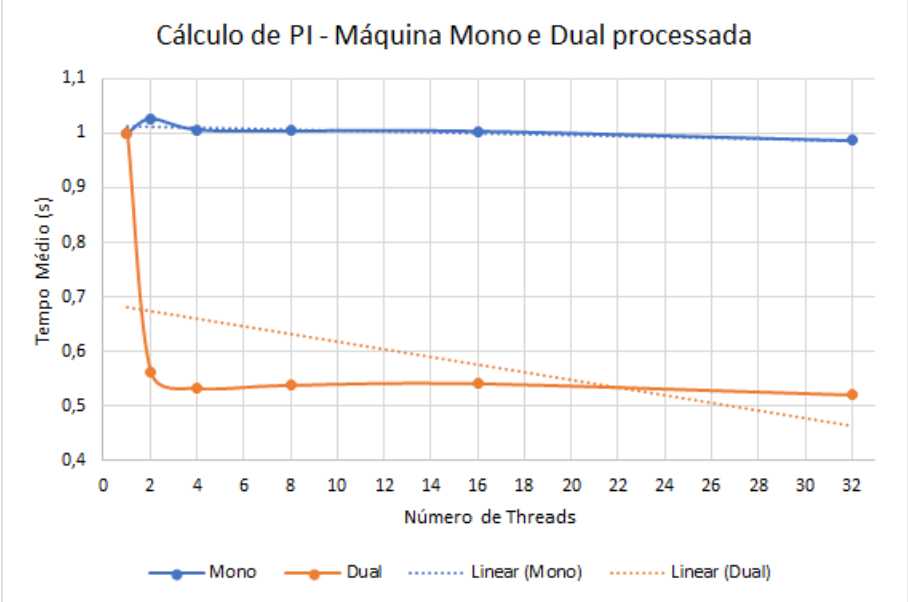

Figura 3: Comparativo do cálculo de π em máquina Mono e Dual processada. Tempo x Threads

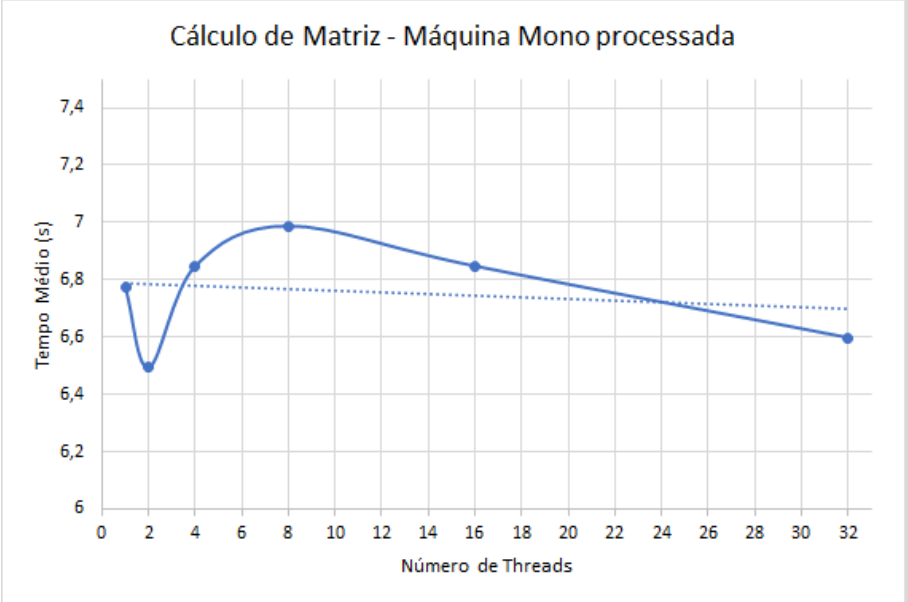

Figura 4: Calculo de Matriz em máquina Mono processada. Tempo x Threads

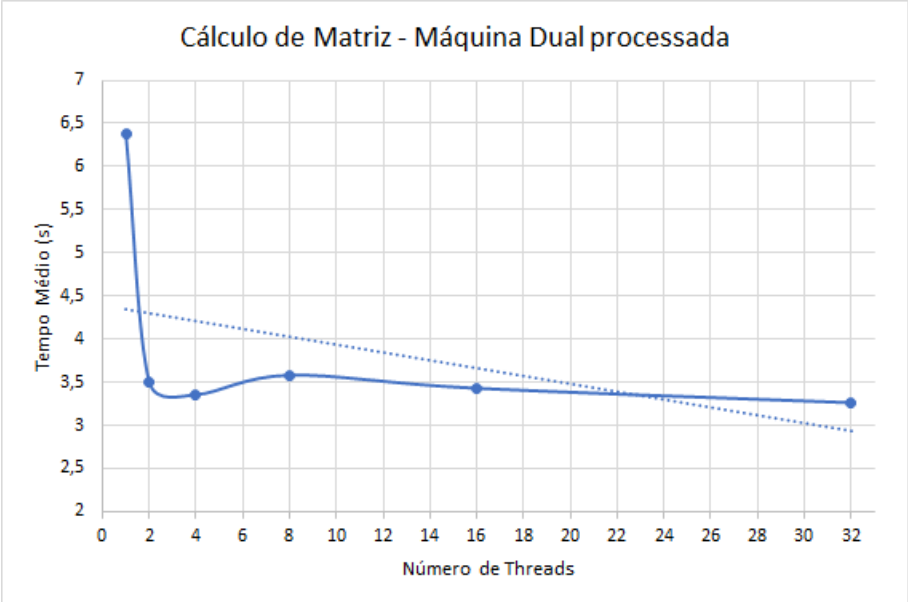

Figura 5: Calculo de Matriz em máquina Dual processada. Tempo x Threads

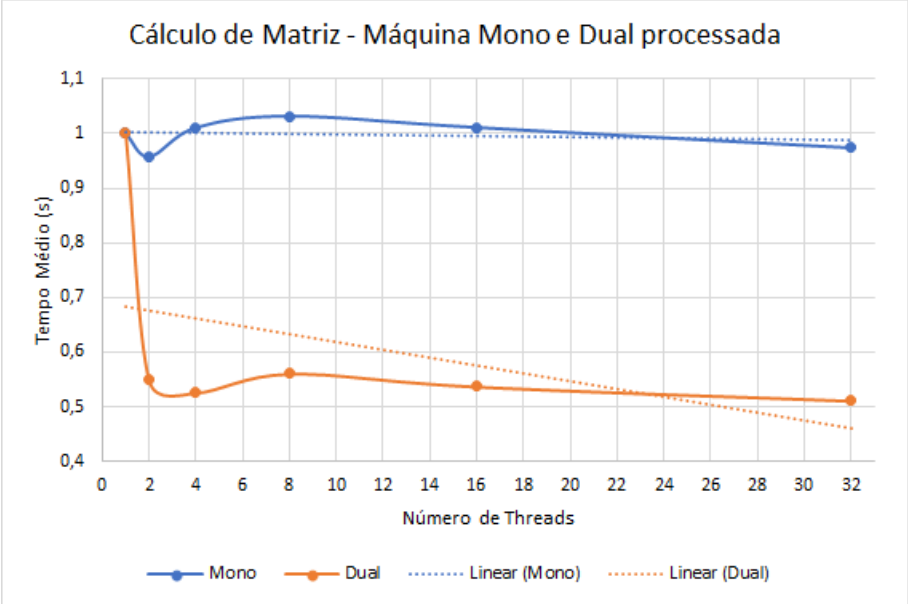

Figura 6: Comparativo do cálculo de Matriz em máquina Mono e Dual processada. Tempo x Threads