

*Uma abordagem MPI para transcrição e tradução de
DNA*

Elihofni Lima
Erick Grilo
Max Fratane

1 Introdução e descrição da aplicação

A transcrição do DNA é o processo através do qual o DNA serve de modelo para a síntese de RNA feita por um ser vivo. Apenas uma cadeia de DNA é usada nesse processo, ativada pela enzima RNA-polimerase. Em uma determinada região da molécula de DNA, ocorre a separação das hélices, onde uma delas forma o RNA através do encadeamento de nucleotídeos complementares. Em suma, essa é a fase responsável por parear as bases nitrogenadas do DNA com as do RNA: A do DNA com U do RNA, T do DNA com A do RNA, C do DNA com G do Rna e G do DNA com C do RNA.

Relembrando alguns conceitos de biologia, A (adenina), C (citossina) T (timina) e G (guanina) são os nucleotídeos que compõem o DNA, e as mesmas, com exceção da timina (que vira U, de uracila, que por sua vez é uma base nitrogenada), compõem o RNA. Cada trinca de 3 dessas bases nitrogenadas é chamada de códon. Um códon codifica um aminoácido, vide Morandini et al. (2013).

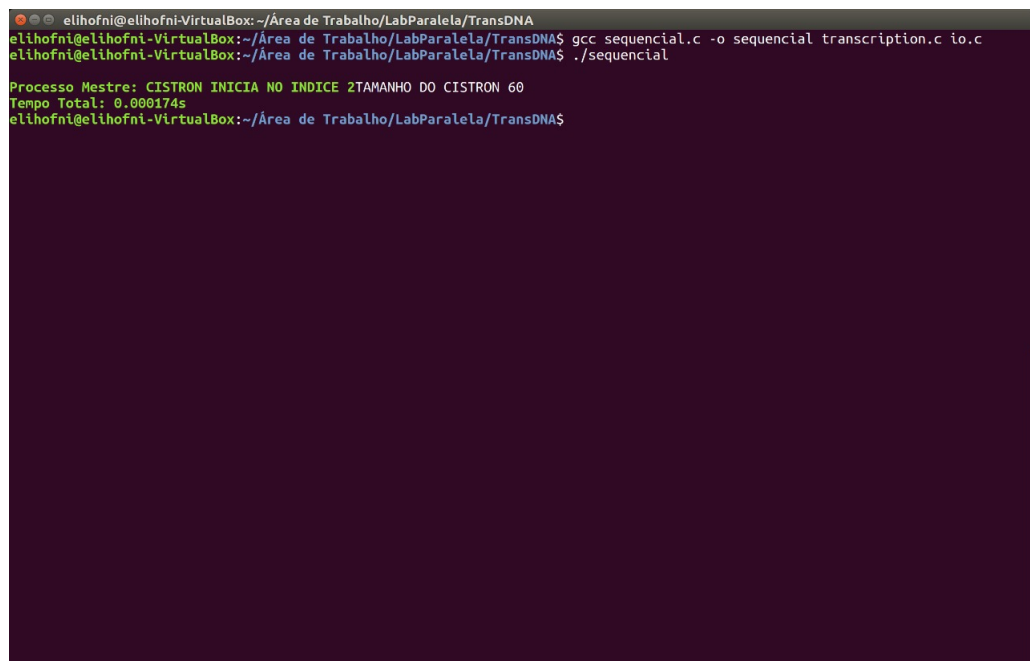
Motivado pelo tamanho que uma cadeia de DNA pode ter (em Venter et al. (2001), no mapeamento do genoma humano, por exemplo, foram encontradas aproximadamente 2.91 bilhões de pares de bases nitrogenadas. Já, em Lodish et al. (n.d.), proteínas encontradas em leveduras possuem uma média de tamanho de códon com 466 nucleotídeos) e pelo processo de transcrição ser uma tarefa repetitiva, a criação de um programa paralelo do tipo SPMD (Simple Program, Multiple Data) aparenta ser uma boa abordagem para a solução desta tarefa.

Em Chibli (2008), por exemplo, é proposta uma abordagem usando MPI para solucionar problemas de comparação de *strings*, incluindo a transcrição de DNA para RNA e a identificação de aminoácidos, enquanto em Kleinjung et al. (2002) e Xue et al. (2014), o padrão MPI é utilizado para outro fim, que é o alinhamento de sequências de DNA (que visa encontrar similaridades no DNA que podem indicar relações evolucionárias entre diferentes indivíduos, ou seja, características semelhantes entre indivíduos de espécies diferentes.)

O objetivo é paralelizar a transcrição e a tradução de DNA, onde a transcrição é o processo responsável por traduzir uma cadeia de DNA para RNA e a tradução consiste em identificar o aminoácido que o códon em questão representa, a partir de códon de RNA (que foram obtidos a partir da transcrição de códon de DNA para RNA). Note que o processo só se inicia efetivamente quando na cadeia original de DNA dado na entrada é identificado um códon (uma cadeia de DNA iniciada pelos códon compostos por ATC, ACT ou ATT, que possui informações para a síntese de uma proteína). Enquanto um códon não é encontrado, não há nenhum processamento efetivo. Quando um códon é encontrado, o processo se inicia. Como estamos considerando nenhuma mutação genética, o códon deve ser múltiplo de 3 para que o algoritmo funcione corretamente. Já, a cadeia original pode ter qualquer tamanho.

2 O método sequencial

O método sequencial consiste em tratar toda a cadeia de entrada como uma única cadeia, oriunda da leitura de um arquivo texto onde se encontra tal cadeia. A partir daí, toda a cadeia do DNA é transcrita para RNA da seguinte forma: primeiramente, é identificada o local correto para iniciar a transcrição da cadeia (onde o códon se encontra); em seguida, a cada três nucleotídeos (um códon), sua transcrição (conversão de DNA para RNA) é feita e a identificação do aminoácido que aquele códon se refere é feita em seguida. Ao término da análise de um códon, o códon DNA original, a sua transcrição para códon RNA e o aminoácido que ele representa é escrito em uma linha do arquivo e esse procedimento se repete por todo o tamanho da cadeia de entrada. Caso o códon não possui 3 nucleotídeos (o tamanho do códon não é múltiplo de 3), tal códon é ignorado.

A terminal window with a dark background and light-colored text. The prompt is 'elihofni@elihofni-VirtualBox: ~/Área de Trabalho/LabParalela/TransDNA'. The user enters 'gcc sequencial.c -o sequencial transcription.c io.c' and then './sequencial'. The output shows 'Processo Mestre: CISTRON INICIA NO INDICE 2TAMANHO DO CISTRON 60' and 'Tempo Total: 0.000174s'.

```
elihofni@elihofni-VirtualBox: ~/Área de Trabalho/LabParalela/TransDNA
elihofni@elihofni-VirtualBox:~/Área de Trabalho/LabParalela/TransDNA$ gcc sequencial.c -o sequencial transcription.c io.c
elihofni@elihofni-VirtualBox:~/Área de Trabalho/LabParalela/TransDNA$ ./sequencial

Processo Mestre: CISTRON INICIA NO INDICE 2TAMANHO DO CISTRON 60
Tempo Total: 0.000174s
elihofni@elihofni-VirtualBox:~/Área de Trabalho/LabParalela/TransDNA$
```

Figura 1: Exemplo de uma execução da aplicação sequencial, indicando onde o códon é encontrado na cadeia original, seu tamanho e o tempo total de execução.

3 O método paralelo

O método paralelo consiste em paralelizar todo o processo de transcrição de DNA em RNA e o de tradução do códon para um aminoácido. Dessa forma, o trabalho é dividido em n tarefas (1 tarefa mestre, $n-1$ tarefas escravo) onde a tarefa mestre é a responsável por repartir a sequência de DNA de entrada para as demais tarefas, ficando com a primeira parte da sequência. Após o envio dessas partes da entrada para cada uma das tarefas escravo, a tarefa mestre faz a transcrição da sua parcela da sequência de DNA para RNA, ao mesmo tempo que as tarefas escravo (após receberem a mensagem que contém os dados da tarefa mestre) também iniciam esse processo. A tarefa mestre então realiza a identificação dos aminoácidos que ela transcreveu e fica aguardando as demais tarefas escravo realizarem a parcela do seu processamento (transcrever e identificar os aminoácidos que lhe foram enviadas).

Quando todas as tarefas escravo terminarem o seu processamento, elas enviam de volta os resultados para a tarefa mestre (os códons transcritos e os aminoácidos identificados), que por sua vez imprime na tela o RNA transcrito e os aminoácidos identificados e escreve em um arquivo tais dados.

```
elihofni@elihofni-VirtualBox: ~/Área de Trabalho/LabParalela/TransDNA
elihofni@elihofni-VirtualBox:~/Área de Trabalho/LabParalela/TransDNA$ mpicc main.c -o dna transcription.c io.c
elihofni@elihofni-VirtualBox:~/Área de Trabalho/LabParalela/TransDNA$ mpirun -np 2 dna

Processo Mestre: INICIOU
Processo Mestre: LEITURA NO ARQUIVO CONCLUIDA
Processo Mestre: CISTRON INICIA NO INDICE 2
Processo Mestre: TAMANHO DA CADEIA LIDA = 63
Processo Mestre: TAMANHO DO CISTRON = 60
Processo Mestre: TOTAL DE CODONS = 20
Processo Mestre: 10 CODONS POR PROCESSO
Processo Mestre: ENVIU CODONS DNA PARA PROCESSO #1
Processo Mestre: TRANSCRICAO PARCIAL CONCLUIDA

Processo #1: INICIOU
Processo #1: RECEBEU PARTE DA CADEIA DE DNA DO MESTRE
Processo #1: TRANSCRICAO PARCIAL CONCLUIDA
Processo #1: IDENTIFICACAO DE AMINOACIDOS PARCIAL CONCLUIDA
Processo Mestre: IDENTIFICACAO DE AMINOACIDOS PARCIAL CONCLUIDA
Processo Mestre: RECEBEU RNAs DO PROCESSO #1
Processo Mestre: TRANSCRICAO TOTAL CONCLUIDA
Processo Mestre: RECEBEU AMINOS DO PROCESSO #1
Processo Mestre: IDENTIFICACAO DE AMINOACIDOS TOTAL CONCLUIDA

.:RESULTADOS:.
DNA  RNA  AMINO
AAA  UUU  Phe
AAC  UUG  Leu
GGC  CCG  Pro
GTA  CAU  His
GCA  CGU  Arg
AAA  UUU  Phe
AAC  UUG  Leu
GGC  CCG  Pro
GTA  CAU  His
GCA  CGU  Arg
AAA  UUU  Phe
AAC  UUG  Leu
```

(a) Início da execução, mostrando o *status* dos processos, tamanho total da cadeia de entrada, tamanho do cistron que será processado, o total de códons, a quantidade de códons enviados para cada processo escravo e parte do resultado final.

```
elihofni@elihofni-VirtualBox: ~/Área de Trabalho/LabParalela/TransDNA
Processo Mestre: ENVIU CODONS DNA PARA PROCESSO #1
Processo Mestre: TRANSCRICAO PARCIAL CONCLUIDA

Processo #1: INICIOU
Processo #1: RECEBEU PARTE DA CADEIA DE DNA DO MESTRE
Processo #1: TRANSCRICAO PARCIAL CONCLUIDA
Processo #1: IDENTIFICACAO DE AMINOACIDOS PARCIAL CONCLUIDA
Processo Mestre: IDENTIFICACAO DE AMINOACIDOS PARCIAL CONCLUIDA
Processo Mestre: RECEBEU RNAs DO PROCESSO #1
Processo Mestre: TRANSCRICAO TOTAL CONCLUIDA
Processo Mestre: RECEBEU AMINOS DO PROCESSO #1
Processo Mestre: IDENTIFICACAO DE AMINOACIDOS TOTAL CONCLUIDA

.:RESULTADOS:.
DNA  RNA  AMINO
AAA  UUU  Phe
AAC  UUG  Leu
GGC  CCG  Pro
GTA  CAU  His
GCA  CGU  Arg
AAA  UUU  Phe
AAC  UUG  Leu
GGC  CCG  Pro
GTA  CAU  His
GCA  CGU  Arg
AAA  UUU  Phe
AAC  UUG  Leu
GGC  CCG  Pro
GTA  CAU  His
GCA  CGU  Arg
Processo #1: ENVIU RESULTADOS PARA O PROCESSO MESTRE
Processo Mestre: ESCRITA DE RESULTADOS NO ARQUIVO CONCLUIDA
Tempo total: 0.001063s
Processo #1: FINALIZOUelihofni@elihofni-VirtualBox:~/Área de Trabalho/LabParalela/TransDNA$
```

(b) Fim da execução, mostrando a confirmação do recebimento dos dados processados pela tarefa mestre, os dados processados (que foram escritos em um arquivo e o tempo total de execução

Figura 2: Exemplo de uma execução da aplicação paralela, no ambiente MPI com dois processos, um mestre e um escravo

4 A implementação paralela

O ambiente escolhido para a paralelização da implementação foi o MPI (Message Passing Interface), um padrão de comunicação de dados em computação

paralela cujo objetivo busca efetuar a troca de mensagens entre processos (ou threads, dependendo da implementação) de forma prática e eficiente.

A nossa implementação consiste no uso do ambiente MPI para paralelizar o processo: Ao iniciar o ambiente MPI, o ambiente de execução paralela é inicializado. Em seguida, as tarefas são criadas e inicializadas no comunicador padrão (MPI_COMM_WORLD) e, então, existem duas rotinas possíveis para serem executadas por cada uma das tarefas, que é a rotina destinada para a tarefa mestre (identificada pelo rank 0) e outra para as demais tarefas.

A tarefa mestre então inicia sua execução: primeiro ela efetua a leitura do arquivo (utilizando a função *ler*, disponível em *io.c*), obtendo a cadeia de DNA inserida como entrada e, em seguida, dividindo a mesma em pedaços menores de sub-cadeias de DNA (por exemplo, para uma cadeia de 60 nucleotídeos, como cada códon possui tamanho 3, nessa etapa a tarefa mestre dividiria a cadeia em 20 sub-cadeias de tamanho 3 cada, utilizando a rotina *split*, encontrada em *transcription.c*) e em seguida, envia as subcadeias para as demais tarefas (mantendo a primeira porção para si) por meio da chamada da função *MPI_Send* (primeiramente enviando a quantidade de códons que a tarefa irá receber, em seguida enviando os códons), que faz o envio bloqueante síncrono de mensagens (isso significa que a tarefa que está enviando retorna a execução assim que a mensagem foi enviada, o que não implica que a mesma foi recebida pela tarefa receptora).

Após o envio dos dados necessários (quantidade de códons e os códons em si), a tarefa mestre começa a efetuar o processamento na parte dos códons que com ela ficou: primeiro, ela divide a sub-cadeia de DNA que restou em outras sub-cadeias de tamanho 3 (o tamanho de um códon) e, para cada uma dessas sub-cadeias, faz a transcrição de DNA para RNA, utilizando a função *transcription* (encontrada em *transcription.c*) e em seguida identificando o aminoácido referente ao códon transcrito, fazendo uso da rotina *transcription* (que pode ser encontrada em *transcription.c*).

Nesse momento, as demais tarefas escravo também estão fazendo o mesmo: após o recebimento das mensagens que foram enviadas pela tarefa mestre (recebimento feito por meio da rotina *MPI_Recv*), cada uma das tarefas escravo divide a cadeia de DNA recebida em sub-cadeias de tamanho 3 (da mesma forma que a mestre), e, para cada sub-cadeia de DNA gerada a partir da cadeia que a tarefa recebeu, ela faz a transcrição de DNA e a tradução do códon para o aminoácido que ele identifica (da mesma forma que a tarefa mestre). Ao término do processo, elas enviam o resultado do processamento (que seria todos os códons transcritos e cada um dos seus respectivos aminoácidos) para a tarefa mestre

A mestre então, para cada tarefa escravo, ela recebe os dados da tarefa escravo (por meio da rotina `MPI_Recv`) e, para cada conjunto de dados recebidos (de cada tarefa), ela vai armazenando as cadeias de RNA transcritos e os seus respectivos aminoácidos. Ao fim da recepção (todas as tarefas escravo concluíram o envio, a tarefa mestre então imprime na tela o resultado do processamento: para cada códon, imprime na tela o códon de DNA original da entrada, a cadeia de RNA transcrita referente à tal códon e o aminoácido que tal cadeia de RNA identifica. Em seguida, escreve o mesmo que ela imprimiu em um arquivo de saída. Ao fim da execução do ambiente MPI, é pego o tempo de finalização (a fim de ver quanto tempo o processamento levou) e o ambiente é finalizado pelo processo main (o mesmo onde o ambiente é inicializado) por meio da rotina `MPI_Finalize`.

5 Descrição dos experimentos computacionais

5.1 Máquina única

A máquina física onde foi testada os códigos é um computador com 8 GB de memória RAM real, processador Intel i7-5500U @ 2.40 GHz, com o S.O Windows 10 64 bits. Os códigos foram efetivamente executados em uma máquina virtual usando VirtualBox 5.1.22, onde a máquina virtual possui o S.O Ubuntu 16.04 LTS 64 bits, 4 GB RAM e 2 núcleos do processador físico (de um total de 4 núcleos) dedicados à ela. Foram executados 5 casos de teste: uma entrada com o tamanho do cóstron de 60 nucleotídeos, uma entrada com tamanho do cóstron de 122 nucleotídeos, uma entrada com tamanho do cóstron de 842 nucleotídeos, uma entrada com tamanho do cóstron de 4437 e uma entrada com cóstron de tamanho 7280 (observe que a quantidade de cóstrons na entrada pode diferir do tamanho da cadeia de DNA de entrada original). A cada bateria de teste, houve uma pausa de três minutos (pois outras atividades que poderiam estar ocorrendo no background poderiam estar interferindo no resultado dos testes). Nessas condições, os seguintes resultados foram obtidos:

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,000161s	0,000263s	0,000170s	0,000198s

Tabela 1: Tabela da execução com entrada de um cóstron com 60 nucleotídeos para a aplicação sequencial

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,000905s	0,000673s	0,001260s	0,000946s
2 processos	0,002815s	0,001322s	0,00170	0,000198s
3 processos	0,009347s	0,012447s	0,002983s	0,008259s
4 processos	0,002717s	0,002732s	0,009044s	0,004831s
5 processos	0,002759s	0,007151s	0,002873s	0,004261s
6 processos	0,002399s	0,006788s	0,003720s	0,004302s
7 processos	0,009775s	0,002534s	0,003134s	0,005147s
8 processos	0,002090s	0,005234s	0,010324s	0,005882s
9 processos	0,006355s	0,005824s	0,007481s	0,006553s
10 processos	0,004734s	0,004441s	0,015296s	0,008157s
20 processos	0,007049s	0,007616s	0,048893s	0,021186s

Tabela 2: Tabela da execução com entrada de um cóstron com 60 nucleotídeos para a aplicação paralela

Para a primeira bateria de testes, note que a aplicação sequencial acabou sendo melhor que as execuções da aplicação paralela: enquanto a execução no ambiente MPI com apenas 1 processo culminou em ser aproximadamente 5 vezes mais lenta que a sequencial, e a execução com 20 processos teve o pior desempenho médio.

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,000270s	0,000319s	0,000279s	0,000289s

Tabela 3: Tabela da execução com entrada de um cístron com 122 nucleotídeos para a aplicação sequencial

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,000560s	0,000673s	0,000735s	0,000656s
2 processos	0,001141s	0,001135s	0,001447s	0,001241s
4 processos	0,006964s	0,006780s	0,007720s	0,007155s
8 processos	0,004822s	0,006210s	0,005909s	0,005647s
16 processos	0,009243s	0,010715s	0,029486s	0,016481s
32 processos	0,045284s	0,047188s	0,132972s	0,075148s

Tabela 4: Tabela da execução com entrada de um cístron com 122 nucleotídeos para a aplicação paralela

Para a segunda bateria de testes, note que a aplicação sequencial novamente acabou sendo melhor que as execuções da aplicação paralela: a execução sequencial obteve uma melhor média frente às execuções paralelas, onde a primeira obteve uma média aproximadamente 2 vezes pior que a sequencial, onde a execução com 32 processos obteve o pior desempenho frente às demais.

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,001204s	0,001251s	0,001167s	0,001207s

Tabela 5: Tabela da execução com entrada de um cístron com 842 nucleotídeos para a aplicação sequencial

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,001647s	0,003152s	0,003888s	0,002895s
2 processos	0,024973s	0,971038s	1,230680s	0,742230s
4 processos	0,016289s	0,086328s	2,050373s	2,152990 s
8 processos	0,012612s	0,137131s	0,549633s	0,233125s
16 processos	0,011125s	0,708151s	2,310141s	1,009806s
30 processos	0,048504s	0,590812s	1,115891s	0,585069s
32 processos	——s	——s	——s	——s
120 processos	1,816374s	8,354731s	6,058427s	5,409844s

Tabela 6: Tabela da execução com entrada de um cístron com 842 nucleotídeos para a aplicação paralela

Na terceira bateria de testes, temos (novamente) que a aplicação sequencial obteve um melhor desempenho frente às execuções da aplicação paralela: o melhor caso da aplicação paralela foi na execução com apenas 1 processo, onde o tempo médio foi 2 vezes pior que o tempo médio da execução da aplicação sequencial, enquanto o pior caso foi a execução com 120 processos, com um tempo médio de execução de 5,409844 segundos (isso é aproximadamente 4.482 vezes pior que a aplicação sequencial). Note que por algum motivo, a execução da aplicação paralela não teve sua execução bem sucedida com 32 tarefas (não sabemos o motivo).

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,004860s	0,007023s	0,005213s	0,005698s

Tabela 7: Tabela da execução com entrada de um códon com 4437 nucleotídeos para a aplicação sequencial

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,017591s	0,015152s	0,015673s	0,016138s
2 processos	0,024995s	0,084425s	0,073754s	0,061058s
4 processos	0,084745s	0,084425s	0,164985s	0,109631s
12 processos	0,037773s	0,008040s	0,008510s	0,018107s
24 processos	0,065613s	0,091352s	0,049874s	0,068946s
48 processos	0,087321s	0,101885s	0,097877s	0,095694s

Tabela 8: Tabela da execução com entrada de um códon com 4437 nucleotídeos para a aplicação paralela

Para esse caso, note que a execução da aplicação sequencial ainda obteve um melhor tempo que as execuções da aplicação paralela.

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,013972s	0,008105s	0,008033s	0,010036s

Tabela 9: Tabela da execução com entrada de um códon com 7280 nucleotídeos para a aplicação sequencial

Nº processos	1ª execução	2ª execução	3ª execução	Média
21 processo	0,869893s	0,574401s	0,0,429978s	0,624757s
23 processos	0,840000s	0,568233s	0,853577s	0,753933s
30 processos	1,146455s	1,087207s	1,182422s	1,138694s
69 processos	2,415901s	1,109173s	1,879926s	1,801666s
79 processos	1,234773s	1,713105s	2,284012s	1,743963s

Tabela 10: Tabela da execução com entrada de um cístron com 7280 nucleotídeos para a aplicação paralela

Para essa bateria de teste, a aplicação sequencial também obteve um melhor desempenho que as execuções sequenciais, onde o melhor resultado da execução paralela foi o com 21 tarefas

Para todos os casos de teste, percebe-se que, em uma única máquina, o desempenho da aplicação paralela no ambiente MPI foi pior que o desempenho da sequencial. Baseados nos testes, somos levados à concluir que a transmissão de mensagens entre os processos pode influenciar drasticamente no desempenho deles, de forma que quanto mais processos tiverem, mais mensagens são trocadas, mais as tarefas têm que lidar com o envio e o recebimento de mensagens, podendo fazer o processamento devido à troca de mensagens crescer exponencialmente.

O tempo de execução também é afetado pelo fato dessa execução paralela ser em uma única máquina, a partir do momento que o número de tarefas for maior que o número de núcleos da máquina, se dá início um processo de "gargalo", onde partes da aplicação paralelizada aguardam um núcleo terminar o processamento de outra parte desta mesma aplicação (uma outra tarefa), causando uma espécie de "sequenciamento da aplicação paralela"

5.2 Várias máquinas

A execução em várias máquinas se deu em um ambiente com 9 máquinas com hardware diferente: 6 máquinas com processador Intel Core i5-6400 CPU @ 2.70GHz, com memória RAM de 8 GB, e 3 máquinas Pentium Dual-Core E5500 @ 2.80 GHz, com memória RAM de 4GB . Infelizmente, por motivos técnicos (a rede do laboratório caiu), não foi possível efetuar os testes referentes à execução do programa paralelo com um cístron com 60 nucleotídeos. Para o caso do processamento paralelo em várias máquinas, foram obtidos os seguintes resultados:

Note que para 16 e para 32 processos, não foi possível efetuar o restante das avaliações devido ao mesmo problema técnico que afetou a realização dos testes com o cístron de tamanho 60.

Note que para 1 e para 2 processos, a avaliação apresentou problemas técnicos.

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,000353s	0,000584s	0,000191s	0,000376s
2 processos	0,000506s	0,000560s	0,000515s	0,000527s
4 processos	0,000814s	0,000853s	0,000904s	0,000857s
8 processos	0,001122s	0,001010s	0,001239s	0,001124s
16 processos	0,001347s	————s	————s	————s
32 processos	————s	————s	————s	————s

Tabela 11: Tabela da execução em várias máquinas com entrada de um cístron com 122 nucleotídeos para a aplicação paralela

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,002840s	0,002407s	0,001622s	0,002290s
2 processos	0,004603s	0,004475s	0,004524s	0,004534s
4 processos	0,004309s	0,003582s	0,004327s	0,004073s
12 processos	0,003560s	0,003453s	0,003140s	0,003384s
24 processos	0,005720s	0,007994s	0,009351s	0,007688s
48 processos	0,009777s	0,009314s	0,009171s	0,009421s

Tabela 12: Tabela da execução em várias máquinas com entrada de um cístron com 842 nucleotídeos para a aplicação paralela

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	0,003618s	0,003518s	0,003553s	0,003563s
2 processos	0,053770s	0,067851s	0,082450s	0,068024s
4 processos	0,112044s	0,104496s	0,117015s	0,111185s
12 processos	0,581061s	0,111685s	0,200701s	0,297816s
24 processos	0,057335s	0,037419s	0,053061s	0,049272s
48 processos	0,061069s	0,016130s	0,026453s	0,034551s

Tabela 13: Tabela da execução em várias máquinas com entrada de um cístron com 4470 nucleotídeos para a aplicação paralela

Nº processos	1ª execução	2ª execução	3ª execução	Média
1 processo	————s	————s	————s	————s
2 processos	————s	————s	————s	————s
21 processos	0,327879s	0,274766s	0,262222s	0,288289s
30 processos	0,351618s	0,262799s	0,288254s	0,300890s
69 processos	0,060148s	0,033368s	0,059113s	0,050876s
79 processos	0,084528s	0,049993s	0,065031s	0,066517s

Tabela 14: Tabela da execução em várias máquinas com entrada de um cístron com 7292 nucleotídeos para a aplicação paralela

6 Conclusões

Concluimos que, para o nosso caso, depois eu termino isso aqui (mencionar que a execução foi lenta pra porra toda em mpi, que apresenta melhora com cadeias muito grandes, e ainda assim só pra números altos de processos, etc.)

Referências

- Chibli, E. A. (2008), *A Multiprocessor Parallel Approach to Bit-parallel Approximate String Matching*.
- Kleijnung, J., Douglas, N. & Heringa, J. (2002), ‘Parallelized multiple alignment’, *Bioinformatics* **18**(9), 1270–1271.
- Lodish, H. B., Matsudaira, A., Kaiser, P., Krieger, C., Scott, M. & Zipurksy, M. (n.d.), ‘Sl & darnell, j.(2004)’, *Molecular Cell Biology. 5th ed. WH Freeman: New York, NY*.
- Morandini, C., Bellinello, L. C. & Constantino, C. (2013), *Coleção Objetivo - Apostila Resumo Biologia 1*.
- Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A. et al. (2001), ‘The sequence of the human genome’, *science* **291**(5507), 1304–1351.
- Xue, Q., Xie, J., Shu, J., Zhang, H., Dai, D., Wu, X. & Zhang, W. (2014), A parallel algorithm for dna sequences alignment based on mpi, *in* ‘Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on’, Vol. 2, IEEE, pp. 786–789.