Desenvolvimento de uma versão paralela do processo de clusterização aplicado a montagem e comparação de genomas

Micaella Coelho[1], **Carla Osthoff**[1], **Fabrício Vilasbôas**[1]

[1] Laboratório Nacional de Computação Científica(LNCC)– Brasil

{micaella,osthoff,fabricio}@lncc.br

1. Introdução

O desenvolvimento tecnológico na área de bioinformática têm gerado uma grande quantidade de dados a serem processados, por outro lado o advento de tecnologias tais como GPU tem possibilitado um processamento cada vez maior dos dados.

Este trabalho tem como objetivo apresentar técnicas de otimização de desempenho em GPU para o alinhamento e a montagem de genomas. O algoritmo utilizado é o K-mer [1] que consiste na contabilização da frequência de repetição de k nucleotídeos em uma sequencia. Este estudo, apresenta a primeira parte dessa otimização que consiste na implementação do K-mer em GPU, algoritmo este com poucos trabalhos realizados em ambientes paralelos [2].

Neste estudo, foram desenvolvidas duas funções, uma para o cálculo dos índices e outra para o cálculo da frequência, ambas sendo executadas totalmente dentro da GPU.

Os nucleotídeos, quando lidos do arquivo, são convertidos em valores numéricos, sendo esses 0, 1, 2 e 3 mapeados respectivamente para A, C, G e T. Essas sequências lidas são enfileiradas em um único vetor. Para sinalizar o fim de cada sequência, é usada uma identificação caracterizada pelo valor -1. Feito esse mapeamento, todas as sequências são carregadas de uma vez para à memória da GPU e todo o processo será executado dentro da mesma.

Depois do mapeamento e do carregamento dos dados, é feita a conversão dos valor das combinações que estão na base 4 para a base 10. Essas combinações são feitas através do conceito, com overlap, onde atribuí-se como parâmetro o valor de k. Assim é permitido obter um vetor de 4^k posições para cada sequência, onde o índice de cada posição desse vetor é o valor na base 10 das combinações que estão na base 4. As combinações que possuem o sinalizador de terminação como componente são descartadas.

Calculada a combinação e feita a conversão, é possível realizar o cálculo da frequência de repetição de cada combinação. Sabendo que o índice da combinação é mapeado para a posição exata do vetor de combinações, basta somar um na posição do vetor de combinação.

As combinação de k nucleotídeos podem ser calculadas independentemente da outra. Então para cada nucleotídeo de cada sequência é atribuído uma thread, ou seja, todos os índices são calculados ao mesmo tempo. Já para o cálculo das frequências, seria muito complexo identificar a posição exata de onde somar um, dado que as sequências têm tamanho diferentes. Então foi usada a estratégia onde é dada uma sequência à cada thread. Cada thread é responsável por um loop, o qual percorre toda a sequência e soma um na posição correta. Isso é possível porque se sabe a posição do início e o tamanho

de cada sequência e, assim, é conseguido mapear o valor da combinação no vetor de repetição. Também foi adotada essa estratégia para evitar condição de corrida.

Após ser feita a conversão das combinações da base 4 para a base 10, não é mais necessário manter as sequências na memória da GPU. Então, o vetor de sequências é desalocado para alocar o vetor de combinações, possibilitando assim, valores mais consideráveis de k.

2. Resultados obtidos

Para os experimentos foi utilizado uma *workstation* com um processador *Intel Xeon* X5650 @ 2.67GHz, 24GB de memória *RAM* e uma placa gráfica *Tesla K40c* com 12GB de memória e 2880 *CUDA Cores*.

A Figura 1 apresenta a contabilização dos tempos de execução quando executado arquivos de 200MB, 300MB, 400MB e 1,7GB, sendo as duas rotinas envolvidas no processo do K-mer, tanto na versão paralela quanto na serial.

Pode-se observar que a medida em que aumenta o tamanho de k de 2 para 4 o tempo de execução diminui tanto para a versão serial quanto para a versão paralela. Observa-se também que a versão paralela apresenta melhor desempenho para todos os tamanhos de k e para todos os tamanhos dos arquivos analisados, devido à significante diminuição no tempo de execução.

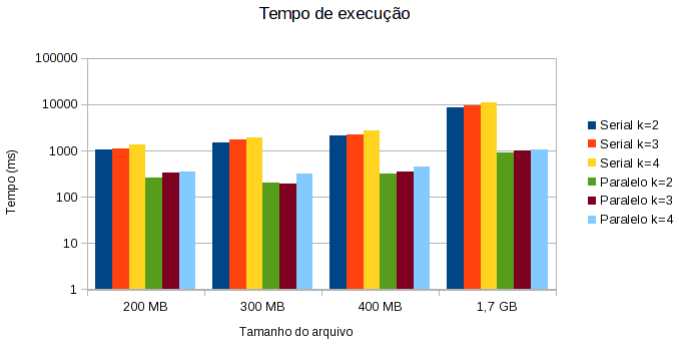

Figura 1. Tempo de execução das duas funções em sua versão serial e paralela para diferentes valores de k.

3. Conclusão

Por meio da implementação dessa estratégia foi possível observar uma melhor performance do algoritmo. Pode-se atribuir esse ganho à grande quantidade de dados e operações sendo realizadas em parelelo, apresentando um melhor resultado em relação à execução serial.

Referências

[1] Nicolas Maillet, Claire Lemaitre, Rayan Chikhi, Dominique Lavenier e Pierre Peterlongo. *Compareads: comparing huge metagenomic experiments*. *BMC Bioinformatics*, Niterói-BRA, 19 out. 2012.

[2] Guillaume Marçais, e Carl Kingsford *A fast, lock-free approach for efficient parallel counting of occurrences of k-mers*. Bioinformatics (2011) 27 (6): 764-770.