Otimização do espaço de memória das estruturas internas de um contabilizador de frequência de repetição de *k-mers*

Matheus P. Ferreira[1], **Fabio T. Ishikawa**[1], **Fabrício G. Vilasbôas**[1], **Calebe P. Bianchini**[1]

[1]Faculdade de Computação de Informática – Universidade Presbiteriana Mackenzie (UPM)
São Paulo, Brasil

Resumo. *Nesse artigo apresentamos o CFRK-MC, um contabilizador da frequência de repetição de k-mers otimizado para ambientes de memória compartilhada baseado no CFRK. O CFRK-MC apresentou uma redução de* $11,5\times$ *no tempo total de execução em relação a versão original, apresentando também uma redução consistente de ocupação de memória.*

1. Introdução

A análise de amostras de metagenoma é uma das áreas da biologia que mais se destacaram nos últimos anos devido ao surgimento dos sequenciadores NGS [Head et al. 2014]. Com isso, o acesso aos dados genômicos de amostras de ambiente, ou metagonoma, ficaram mais fáceis. No entanto, o grande volume de dados gerado por esses sequenciadores tornaram necessário a aplicação de técnicas de processamento de alto desempenho para a viabilidade das análises [Zerbino and Birney 2008].

A análise de um material genético é uma das principais tarefas que devem ser tratadas pelos métodos computacionais. Dentre as técnicas utilizas para esse tipo de análise está o *k-mer*. O *k-mer* é utilizado em genômica computacional para extrair todas as possíveis combinações de *k* nucleotídios em uma *read*, que é uma representação linear simbólica de uma molécula de DNA, ou RNA ou de aminoácidos. Essa técnica pode ser utilizada, por exemplo, para montagem de metagenoma [Zerbino and Birney 2008], para análise qualitativa de amostras [Onate et al. 2015], para alinhamento de sequencias [Chen et al. 2015], dentre outros usos. Uma outra técnica útil associada ao *k-mer* é a contabilização da frequência de repetições como parte da análise de dados [Zhang et al. 2014].

Nesse artigo, apresentamos os resultados da aplicação de uma técnica para a redução e otimização do uso de memória principal (RAM) em uma versão do algoritmo do *k-mer* para sistemas computacionais de memória compartilhada.

2. CFRK-MC

Considerando os algoritmos de contabilização de *k-mer*, o algoritmo referência é o *Jellyfish* [Marçais and Kingsford 2011] que usa tabelas *hash* com resolução de colisão. Apesar de existirem também outros algoritmos, nenhum deles se propõem a analisar especificamente metagenomas, como o CFRK.

O algoritmo CFRK proposto originalmente [Vilasbôas 2017] representa internamente os nucleotídios de forma numérica, transformando as sequências para a base 4. Existe uma marcação para a separação dos dados armazenados que, neste caso é simbolizado por -1. Porém, para que a contabilização da frequência *k-mer* seja realizada, uma

estrutura adicional é criada de tamanho V. Esta estrutura ocupa uma quantidade exponencial de memória, conforme mostra a equação Equação 1, pois reserva um espaço na memória principal para todas as possíveis combinações dos *k-mer* para cada sequência.

$$V_{original} = Sequencias \times 4^k \tag{1}$$

Os resultados coletados neste artigo são provenientes de uma mudança na estrutura de dados utilizada para a contabilização de *k-mers*. Segundo [Morgado 1991], há uma relação discreta entre o valor de *k* e o tamanho *L* da sequência analisada. Esta relação define que há um número limite de subsequências de tamanho *k* distintas contidas em uma sequência de tamanho *L*. Usando como exemplo uma sequência hipotética de tamanho $L = 4$ e $k = 3$, a relação discreta leva em consideração os elementos da subsequência anterior para construção de uma nova subsequencia, ou seja, a primeira subsequencia $3 - mer_0$ seria definida como (n_0, n_1, n_2); a segunda subsequencia $3 - mer_1$ seria definida como (n_1, n_2, n_3); e assim, sucessivamente, evidenciando o número de combinação reduzida para cada subsequência gerada.

Dessa forma, é possível limitar o tamanho ocupado pela estrutura de armazenamento de frequência, conforme indica a Equação 2, trocando um método de alocação exponencial para linear.

$$V_{experimental} = L - k + 1 \tag{2}$$

A Equação 2 apresenta ainda uma relação inversa entre *k* e *L*, isso porque quanto maior o valor de *k* menos sequências distintas poderão ser representadas no tamanho limitado *L* da sequência. Isso significa que, quanto maior for o valor de *k*, menor será a quantidade de espaço reservado na estrutura.

3. Experimentos

O ambiente dos experimentos possui um Processador Intel(R) Core(TM) i5-6500 com 4 núcleos físicos e 4 núcleos lógicos, 8 GB de RAM com *CentOS 8* e *kernel* versão *4.18.0-147.8.1.el8_1.x86_64*. Foi utilizado também o *gcc* versão 8.3.1 com *flags* de otimização *-O3*. Para os experimentos, foi utilizado um arquivo de 8.952 *Kilobytes* de tamanho, com 24.282 sequências.

Os resultados de medição da memória ocupada podem ser vistos na Figura 1. Essas medidas foram coletadas diretamente do sistema operacional (*ps*), sem nenhum tratamento estatístico. Nessa figura, a solução original (BASE) cresce exponencialmente, seguindo a definição feita na Equação 1. Para $k = 8$, o equipamento de teste teve 77% de sua memória principal comprometida, sendo inviável aumentar o valor de k a partir desse ponto para novas coletas de dados, ou seja, a quantidade de memória necessária para $k > 8$ ultrapassa a capacidade atual do equipamento. Para a nova solução (PROPOSTA), a quantidade de memória se mantém praticamente constante, sinalizando uma tendência de diminuição de consumo de memória, conforme apresentado na Equação 2.

Além do consumo de memória, percebeu-se também que existe uma diferença de tempo de execução da solução original (BASE) em relação ao tempo de execução da nova

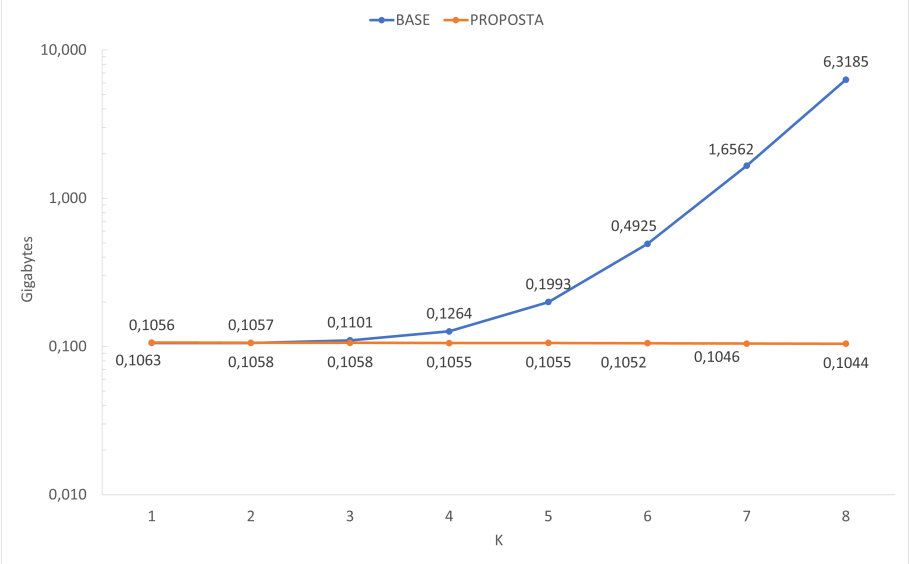

Figura 1. Gráfico comparativo entre soluções de uso de memória por *k*

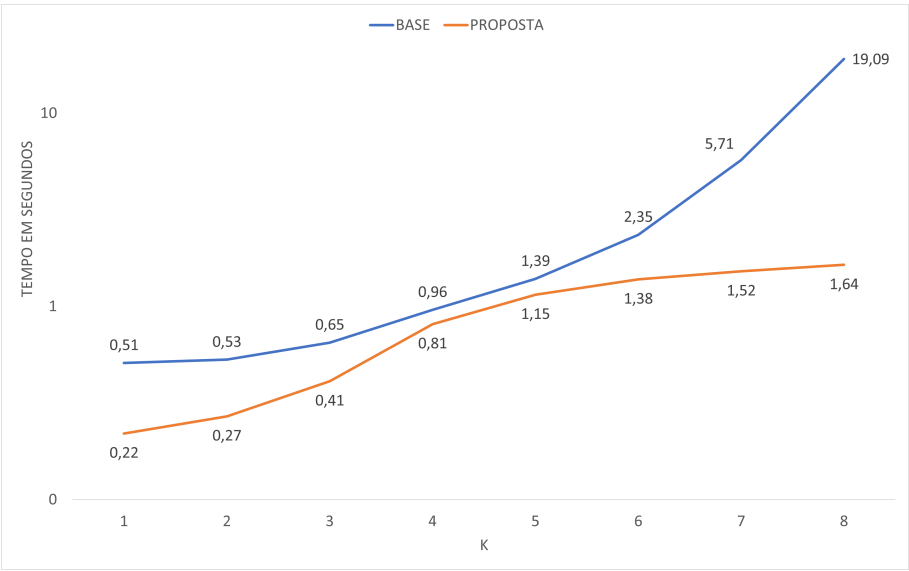

Figura 2. Gráfico comparativo de tempo de execução por *k*

solução (PROPOSTA), conforme mostra a Figura 2. O tempo de execução da solução PROPOSTA para $k = 8$ chega a ser $11,5\times$ mais rápida que a solução original (BASE).

4. Conclusão

A redução de espaço de memória reservado e a consequente melhoria da execução do algoritmo de *k-mer* foi alcançado. Esse objetivo proposto pode ser avaliado e verificado por meio dos experimentos realizados em um computador de memória compartilhada comum. Essa melhoria também permitirá a melhoria da execução do *k-mer*, pois será possível executá-lo com valores maiores de *k*, já que essa limitação era rapidamente alcançada devido ao aumento exponencial de reserva de memória.

Ainda serão verificados, por instrumentação, os motivos da melhoria do tempo

de execução - pois existem operações de entrada e saída para arquivos que merecem ser analisados.

Os autores também esperam melhorar ainda mais o desempenho das soluções, não só devido à redução do consumo de memória, mas também preparando novas soluções (tanto da BASE quanto da PROPOSTA) para usufruir dos múltiplos processadores existentes. Estas soluções utilização tanto OpenMP quanto OpenACC, estendendo o trabalho de [Vilasbôas 2017].

Agradecimentos

Os autores agradecem ao MackCloud[1], Centro Multidisciplinar de Computação Científica e Nuvem da Universidade Presbiteriana Mackenzie, ao MackPesquisa, ao CNPq, à FAPESP pelo apoio financeiro.

Referências

Chen, Y., Ye, W., Zhang, Y., and Xu, Y. (2015). High speed BLASTN: an accelerated MegaBLAST search tool. *Nucleic Acids Research*, 43(16):7762–7768.

Head, S. R., Komori, H. K., LaMere, S. A., Whisenant, T., Van Nieuwerburgh, F., Salomon, D. R., and Ordoukhanian, P. (2014). Library construction for next-generation sequencing: overviews and challenges. *Biotechniques*, 56(2):61–77.

Marçais, G. and Kingsford, C. (2011). A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770.

Morgado, A. C. d. Q. (1991). Análise combinatória e probabilidade. *Sociedade Brasileira de Matemática*.

Onate, F. P., Batto, J.-M., Juste, C., Fadlallah, J., Fougeroux, C., Gouas, D., Pons, N., Kennedy, S., Levenez, F., Dore, J., et al. (2015). Quality control of microbiota metagenomics by k-mer analysis. *BMC genomics*, 16(1):1–10.

Vilasbôas, F. G. (2017). Método computacional baseado em gpu para contabilização de k-mers aplicado a metagenomas. Master's thesis, Laboratório Nacional de Computação Científica, Brasil.

Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829.

Zhang, Q., Pell, J., Canino-Koning, R., Howe, A. C., and Brown, C. T. (2014). These are not the k-mers you are looking for: Efficient online k-mer counting using a probabilistic data structure. *PLOS ONE*, 9(7):1–13.

[1] `https://mackcloud.mackenzie.br`