Análise de viabilidade de ferramenta para correção híbrida de sequências genômicas em ambiente de memória compartilhada com FPGA

Felipe V. de Almeida[1], **Liria M. Sato**[1], **Edson T. Midorikawa**[1]

[1]Escola Politécnica – Universidade de São Paulo (USP)

{felipe.valencia.almeida,liria.sato,emidorik}@usp.br

Abstract. *Genome analysis comprises extensive researches, that focus on diseases and their treatments. Supporting such activities, researchers handle computational tools to assemble genomes. This work presents a feasibility analysis of a tool for hybrid correction of genomic sequences; a necessary step for genome assembly. An architecture for heterogeneous environments is proposed, and its implementation is made with a CPU and an FPGA board. The results obtained from the theoretical and practical data survey indicate that the implementation with the hardware accelerator has performance gains of up to 19 times over the sequential version, and may increase depending on the communication technology used.*

Resumo. *A análise do genoma compreende pesquisas com amplo escopo, com foco em doenças e em tratamento das mesmas. Em apoio a tais atividades, pesquisadores valem-se de ferramentas computacionais para montagens de genomas. Este trabalho apresenta uma análise de viabilidade de uma ferramenta para correção híbrida de sequências genômicas, etapa esta necessária para a montagem do genoma. É proposta uma arquitetura para ambientes heterogêneos, com implementação feita em CPU e uma placa FPGA. Os resultados obtidos no levantamento dos dados teóricos e práticos apontam que a implementação com o acelerador em hardware possui ganhos de desempenho de até cerca de 19 vezes em relação à versão sequencial, podendo aumentar a depender da tecnologia de comunicação utilizada.*

1. Introdução

A análise do DNA tem como propósito obter informações sobre o funcionamento dos mecanismos internos do organismo em questão. O estudo do DNA então auxilia os pesquisadores em um melhor entendimento dos seres vivos. Este processo era dificultado em determinadas situações devido ao alto custo financeiro e computacional, porém devido a maior disponibilidade de recursos computacionais e à otimização de algoritmos já existentes, estes custos vêm sendo reduzidos nos últimos anos [Goodwin et al. 2016].

O estudo comparativo entre os diferentes genomas só é possível através da montagem de múltiplos genomas de uma mesma espécie. Desta forma, um processo que já possui elevado custo computacional precisa ser executado diversas vezes, aumentando assim o tempo total de execução. A montagem do genoma é um processo que possui um tempo variável, a depender da complexidade do organismo em questão e dos recursos computacionais disponíveis para a montagem, podendo variar de minutos a até meses [Del Angel et al. 2018]. Por isso, e devido ao advento da computação heterogênea

no contexto da computação de alto desempenho, é necessária a criação de ferramentas híbridas, visando obter resultados superiores às ferramentas tradicionais implementadas em software somente.

Para a obtenção do genoma de um organismo, é necessário inicialmente realizar o sequenciamento do seu material genético. As duas maiores fabricantes de sequenciadores são a Illumina e a Pacific Biosciences (PacBio), que criaram tecnologias distintas para tal sequenciamento. O sequenciamento Illumina resulta em pequenos fragmentos contendo em média 100 a 250 bases nitrogenadas e com boa qualidade. Esses fragmentos são denominados na literatura como sequências pequenas (*short reads*).

O sequenciamento PacBio resulta em fragmentos grandes com uma média de 10.000 bases nitrogenadas, porém com baixa qualidade. O fato dos fragmentos serem maiores quando comparados aos Illumina reduz o custo computacional da montagem, devido ao menor número de fragmentos resultantes; porém, a baixa qualidade pode resultar em montagens errôneas, necessitando então uma etapa de correção anterior à montagem. Esses fragmentos são chamados na literatura de sequências grandes/longas (*long reads*).

Existem duas formas de correção para as sequências PacBio; são elas a auto correção e a correção híbrida. A auto correção utiliza os próprios fragmentos para se auto corrigirem, através do consenso entre eles, obtido no alinhamento. A correção híbrida utiliza fragmentos de outra tecnologia com qualidade superior (em geral Illumina), para corrigir os fragmentos PacBio, também através do alinhamento.

Este artigo tem como objetivo apresentar uma análise de viabilidade de uma ferramenta de correção híbrida, implementada em um ambiente heterogêneo de CPU e FPGA. Ele é uma continuação do trabalho apresentado em [Almeida et al. 2019], onde foram exploradas diversas implementações apenas em software de uma versão base do algoritmo de correção utilizando a janela deslizante. Nele foram abordadas as versões *multicore* e para *cluster*, onde observou-se que a utilização do OpenMP junto com o MPI em um *cluster* resulta em melhorias significativas no desempenho. Desta forma, este artigo tem como objetivo confrontar os resultados teóricos aqui obtidos com a aplicação do acelerador em hardware com os resultados anteriores, visando analisar se o emprego de um acelerador pode resultar em ganhos maiores que o aumento da complexidade de um sistema distribuído.

2. Trabalhos Relacionados

Os trabalhos relacionados encontrados na literatura possuem enfoque em ferramentas de montagem de DNA implementadas em software. A empresa Illumina desenvolveu a SPAdes, que é uma ferramenta de montagem para sequências geradas com os seus sequenciadores. É importante ressaltar aqui que a montagem com sequências Illumina em determinadas situações torna-se inviável. Duas destas situações são quando o organismo em questão possui grande quantidade de genoma ou quando seu genoma é composto em boa parte por replicações do material genético. A Canu [Koren et al. 2017] é uma ferramenta baseada em outra ferramenta mais antiga chamada Celera Assembler. Seu propósito é a montagem de sequências PacBio, realizando para isso a auto correção. Seu *pipeline* é dividido em três etapas, sendo elas a correção, o corte de fragmentos destoantes dos demais e a montagem propriamente dita. [Salmela and Rivals 2014] desenvolveram o LorDEC, que é uma ferramenta apenas para a correção híbrida das sequências PacBio. Esta ferra-

menta utiliza-se de grafos montados a partir das sequências Illumina, de forma que sua correção não é perfeita, porém é mais rápida, por não ser necessário o alinhamento com todos os fragmentos, viabilizando seu custo computacional.

Dentre os trabalhos encontrados na literatura com enfoque na aplicação da FPGA, observou-se que o foco dos mesmos é a implementação de técnicas de alinhamento na FPGA. Em [Mahram and Herbordt 2012] é apresentada uma implementação em FPGA de um circuito capaz de realizar um alinhamento múltiplo entre sequências, que é um outro problema explorado pela Genômica. [Varma et al. 2013] apresenta uma ferramenta para montagem de sequências Illumina utilizando a comunicação entre CPU e FPGA com o barramento PCI *Express*. Em [Ramachandran et al. 2015] foi realizada uma comparação entre o circuito desenvolvido na FPGA para melhoria na qualidade das sequências Illumina e o software, mostrando que a implementação em FPGA gera um *speedup* significativo. [Hu and Georgiou 2015] apresenta um circuito auxiliar implementado na FPGA para melhorar o desempenho do processo de montagem das sequências Illumina.

Não foi possível encontrar na literatura um trabalho que analise e/ou sugira uma ferramenta de correção híbrida implementada em FPGA.

3. Arquitetura Proposta

A arquitetura proposta para o sistema consiste em um ambiente computacional conectado com diversos aceleradores de hardware. Nela, cada unidade de execução é composta por uma CPU e um acelerador e é responsável por realizar a correção híbrida em uma parte do arquivo de sequências grandes. Para tal, uma das CPUs será responsável por, além de corrigir, distribuir as sequências grandes para as demais CPUs, à medida que houver a demanda, realizando assim um balançamento de carga.

A seguir são apresentados os pseudocódigos das partes da ferramenta que são executadas em software e em hardware, respectivamente.

Algorithm 1 Algoritmo Software

```
Carrega arquivo de sequências pequenas
Carrega arquivo de sequências grandes
Abre arquivo de saída
while Restam sequências grandes para serem corrigidas do
    Lê um bloco de sequências grandes do arquivo
    Envia bloco de sequências grandes para o acelerador
    while Restam sequências pequenas para corrigir do
        Lê uma sequência pequena do arquivo
        Envia sequência pequena para o acelerador
    Envia sinal de término das sequências pequenas para o acelerador
    Recebe bloco de sequências grandes corrigidas
    Escreve bloco de sequências corrigidas no arquivo de saída
Sincroniza escrita e fecha arquivo de saída
```

Algorithm 2 Algoritmo Hardware

while Existem sequências grandes para corrigir **do**
  Recebe bloco de sequências grandes da CPU
  while Restam sequências pequenas para corrigir as grandes **do**
    Recebe sequência pequena
    Realiza deslocamento da sequência pequena pela sequência grande
    if $similaridademáximaobtida > threshold$ **then**
      Corrige sequência grande com base na sequência pequena
  Envia bloco de sequências grandes corrigidas para a CPU

Dentro do acelerador é implementado um circuito de propósito específico para a correção. O circuito é composto por três blocos, sendo eles o receptor, o transmissor e o corretor. O circuito receptor recebe os pacotes provenientes da CPU e armazena as sequências inteiras em um *buffer* de recepção, através de deslocamentos sucessivos nos pacotes. O circuito corretor é responsável pela correção das sequências grandes. Ele promove o deslizamento das sequências pequenas, armazenadas em um registrador de deslocamento, nas sequências grandes. A comparação para obtenção da similaridade entre as sequências é realizada com o auxílio de portas XNOR. Os valores obtidos nas comparações então são adicionados utilizando-se diversos somadores, com o intuito de se realizar o cálculo da similaridade em apenas 1 ciclo de *clock*.

Assim, cada deslizamento da sequência pequena na sequência grande terá sua similaridade calculada no mesmo ciclo de *clock*. Caso esse valor seja superior ao *threshold* estabelecido, ele é armazenado em um registrador juntamente com o *offset* de deslocamento, controlado a partir de um contador. Após o término do deslizamento, observa-se o conteúdo do registrador de similaridade, para verificar se alguma posição obteve similaridade superior ao *threshold*. Em caso afirmativo, é realizada uma série de deslizamentos sucessivos na sequência pequena, utilizando o valor do *offset* armazenado para voltá-la até a posição de maior similaridade com a sequência grande. Por fim, é realizada a correção naquela posição. Este processo então é repetido para todas as sequências pequenas. Então, a sequência grande corrigida é encaminhada para o circuito transmissor.

O circuito transmissor é responsável por enviar a sequência grande corrigida para a CPU. Ele possui um *buffer* de transmissão, que armazena a sequência proveniente do circuito corretor e a empacota em diversos pacotes.

4. Implementação da Ferramenta

O ambiente de testes para a implementação da ferramenta foi composto por um computador e uma placa FPGA. Que constitui uma unidade de correção da arquitetura proposta, o que não afeta a análise da ferramenta, pois devido ao balanceamento de carga, a ferramenta é escalável com a quantidade de unidades de correção. O computador em questão possui processador intel i7-720QM com *clock* de 1,9 GHz e 4 *cores*. A memória é de 6 GB de RAM e o sistema operacional é o Ubuntu LTS 16.04. Foi utilizado o compilador gcc versão 4.8.3 e *flag* de otimização -O3 para gerar o arquivo executável. A placa FPGA é a DE0-CV.

Para a comunicação entre a CPU e a FPGA foi adotada uma interface serial com

o protocolo RS-232. Sua escolha foi realizada em função da sua maior simplicidade de projeto, além do fato que ele pode ser implementado em qualquer placa FPGA que tenha pinos GPIO (*General Purpose Input/Output*), tendo assim um escopo mais amplo do que outras formas de comunicação como PCI *Express*, Ethernet ou USB.

Visando minimizar os efeitos da baixa velocidade de comunicação, foi adotado um modelo diferente de codificação para cada base nitrogenada, onde a codificação utilizada foi de 3 bits, ao invés de 8 bits no caso do ASCII estendido. Desta forma, é possível enviar um total de 8 bases a cada três pacotes (24 bits), tendo um ganho de desempenho na comunicação de aproximadamente 160%.

Desta forma, foi desenvolvido um código em linguagem C com o OpenMP para ser executado na CPU, sendo este código responsável pela transmissão e a recepção das sequências para a FPGA. Na transmissão ele realiza a leitura do arquivo das sequências grandes e pequenas, codifica-as conforme a codificação apresentada, empacotando-as e escrevendo-as na porta serial. Na recepção os dados recebidos são decodificados e as sequências corrigidas são escritas no arquivo de saída. O processo de codificação/decodificação é realizado através de operações lógicas de deslocamento de bits.

5. Resultados e Análise

Nesta seção serão apresentados os resultados teóricos e práticos obtidos da análise da implementação da ferramenta de correção híbrida, com suas respectivas análises.

5.1. Análise da Comunicação

A comunicação foi realizada com o protocolo RS-232, modo 8O1 e velocidade de 115.200 bits por segundo. Inicialmente foi realizado um teste na comunicação entre a CPU e a FPGA, visando confrontar os resultados práticos com os resultados teóricos. Foram utilizadas duas sequências, sendo uma sequência grande com 1000 bases e uma pequena com 100 bases.

O cálculo teórico é feito considerando o total de bits de cada sequência. Considerando a codificação adotada de 3 bits, temos as seguintes equações para o tempo de recepção e o tempo de transmissão:

$$tempo_{rx} = \frac{n_{bases} * 3}{8} * 11 * \frac{1}{115200} \quad (1)$$

$$tempo_{tx} = \frac{n_{bases} * 3}{8} * 23 * \frac{1}{115200} \quad (2)$$

Desta forma, obtém-se o tempo de 4 ms para a recepção de uma sequência pequena e 7,4 ms para sua transmissão. No caso da sequência grande, seu tempo de recepção é de 36 ms e sua transmissão em 75 ms.

Os tempos práticos foram obtidos através de medições em um osciloscópio Tektronix TBS1052B. A tabela 1 sumariza os resultados obtidos, onde se observa que existe um pequeno desvio percentual entre os resultados teóricos e os práticos.

Sequência (modo)	Tempo teórico (ms)	Tempo prático (ms)	Desvio percentual (%)
Pequena (RX)	4	4,15	3,7
Pequena (TX)	7,4	8,2	10,8
Grande (RX)	36	37,8	0,5
Grande (TX)	75	81,4	8,5

Tabela 1. Resultados da análise de comunicação entre CPU e FPGA

5.2. Análise da Correção

O tempo total necessário para a correção é composto por duas parcelas, que são o tempo de deslizamento da sequência grande na sequência pequena e o tempo da correção (substituição dos bits referentes às bases nitrogenadas).

Como o cálculo da similaridade é realizado em 1 ciclo de *clock*, o tempo de deslizamento é dado pela diferença entre o tamanho da sequência grande e o tamanho da sequência pequena vezes o período do *clock*. Já o tempo de correção é variável, a depender da região em que foi obtida a maior similaridade. Considerando-se o pior caso, seria necessário realizar novamente todo o deslizamento da sequência pequena na grande, para em seguida realizar a correção na região determinada. Considerando que a operação de correção é composta por uma substituição de bits, necessitando de apenas 1 ciclo de *clock* para ser realizada, o cálculo do tempo total de correção na placa FPGA DE0-CV é dado por:

$$tempo_{total} = (tam_{seqgrande} - tam_{seqpequena}) * 20ns * 2 + 20ns \quad (3)$$

Utilizando novamente o exemplo com duas sequências, sendo uma sequência grande com 1000 bases e uma pequena com 100 bases, o tempo necessário para a correção no pior caso é de aproximadamente 36μs. Ou seja, o tempo necessário para a correção é aproximadamente 111 vezes menor que o tempo de recepção de uma sequência pequena. Desta forma, é possível afirmar que o gargalo do circuito está na comunicação entre a CPU e a FPGA, de maneira que nesta implementação a análise do tempo de execução da ferramenta pode ser feita considerando apenas os tempos da comunicação.

5.3. Análise Geral da Ferramenta

Conforme apresentado, a análise do tempo total de execução da ferramenta pode ser realizada apenas usando-se os tempos de comunicação. Desta forma, o tempo de correção de um bloco pode ser obtido pela seguinte equação:

$$t_{bloco} = t_{rx_{bloco}} + t_{rx_{seqpeq}} * n_{seqpeq} + t_{tx_{bloco}} \quad (4)$$

E o tempo total é dado pela equação:

$$t = t_{bloco} * \frac{tam_{arquivo}}{tam_{bloco}} \quad (5)$$

Para analisar a viabilidade da ferramenta para correção híbrida, os resultados de desempenho foram confrontados com os resultados obtidos em [Almeida et al. 2019]. Neste artigo foi apresentada uma análise de desempenho de algumas soluções em software para a implementação do algoritmo de correção híbrida. Foram avaliadas 7 diferentes versões,

sendo que o melhor desempenho foi obtido usando-se uma solução distribuída com nós *multicore*, totalizando 20 *cores* disponíveis no *cluster*.

Considerou-se então um arquivo com 1 MB de sequências grandes e 10 MB de sequências pequenas. Com a codificação utilizada, o tamanho é de 3 Mbits para as sequências grandes e 30 Mbits para as pequenas. No caso das sequências grandes, devido ao tamanho da memória da placa DE0-CV, é possível agrupar o arquivo inteiro em apenas 1 bloco. Nas sequências pequenas foi realizada a aproximação de 100.000 sequências pequenas de 300 bits.

Assim, utilizando-se as Equações (4) e (5), o tempo obtido é de aproximadamente 469 segundos. Este tempo é inferior ao tempo apresentado para a melhor solução em software usando um *cluster* com 20 *cores*. A tabela 2 apresenta os valores dos tempos obtidos para a execução de 3 versões da ferramenta de correção.

Versão	Tempo de execução (s)	Speedup
Software sequencial (1 *core*)	27.532,4	1
Software paralelo e distribuído (*cluster* com 20 *cores*)	1.587,8	17,3
Híbrido (1 *core* + 1 FPGA)	469	58,7

Tabela 2. Comparação dos tempos de execução

Por último, foi realizada também uma análise da escalabilidade da ferramenta de correção híbrida. Este teste visa verificar o comportamento variando os tamanhos dos arquivos de sequências grandes e pequenas. Os dados da implementação usando CPU e FPGA são confrontados com os dados obtidos em [Almeida et al. 2019]. Os resultados obtidos são apresentados no gráficos da figura 1.

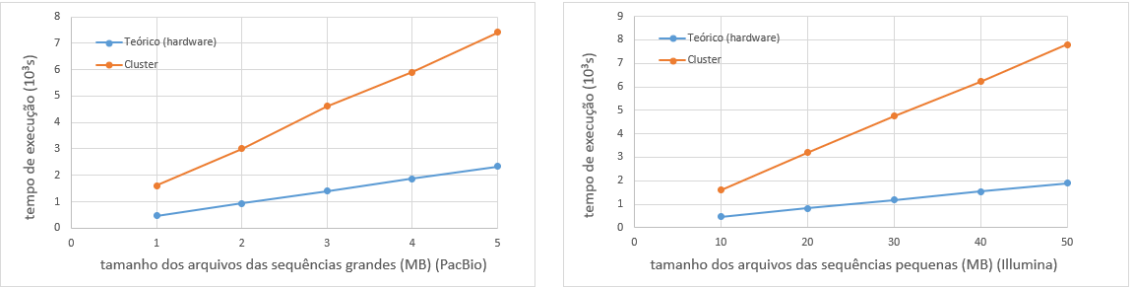

Figura 1. Análise de escalabilidade

Observa-se que a ferramenta é escalável tanto com o tamanho do arquivo das sequências pequenas quanto com o tamanho do arquivo das sequências grandes, e seu desempenho é sempre superior ao melhor caso apresentado no artigo [Almeida et al. 2019].

6. Conclusão e Trabalhos Futuros

O presente artigo teve como objetivo analisar a viabilidade da implementação de uma ferramenta para correção de sequências genômicas em uma placa FPGA. Observou-se que no caso implementado a comunicação torna-se um gargalo no sistema, devido à baixa taxa de comunicação utilizada. Mesmo assim, quando a correção é implementada em apenas uma placa FPGA, esta tem potencial para ser 58 vezes mais rápida que sua implementação em software sequencial. Esse valor deve aumentar com a adição de mais aceleradores.

É estabelecido então, como trabalho futuro, que se realize a implementação completa do circuito com otimizações, visando à utilização de múltiplas placas FPGA em um ambiente de memória compartilhada e distribuída. Espera-se assim atingir um resultado que possa ser confrontado com as outras ferramentas de correção implementadas em software e presentes no estado da arte.

Outro ponto não explorado neste artigo é a utilização de outras formas de comunicação entre CPU e FPGA. Visto que a comunicação é o gargalo da ferramenta, é possível se ter ganhos significativos com a aplicação de tecnologias mais rápidas, como o PCI *Express* por exemplo.

Agradecimentos

Este trabalho foi financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), projeto 132875/2019-5.

Referências

Almeida, F., Sato, L., Midorikawa, E., and Torres, T. (2019). Análise de desempenho de algoritmos para correção híbrida de sequências genômicas em ambiente de memória compartilhada e distribuída. In *Anais do XVIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, Porto Alegre, RS, Brasil. SBC.

Del Angel, V. D., Hjerde, E., Sterck, L., Capella-Gutierrez, S., Notredame, C., Pettersson, O. V., Amselem, J., Bouri, L., Bocs, S., Klopp, C., et al. (2018). Ten steps to get started in genome assembly and annotation. *F1000Research*, 7.

Goodwin, S., McPherson, J. D., and McCombie, W. R. (2016). Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333.

Hu, Y. and Georgiou, P. (2015). A real-time de novo dna sequencing assembly platform based on an fpga implementation. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(2):291–300.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736.

Mahram, A. and Herbordt, M. C. (2012). Fmsa: Fpga-accelerated clustalw-based multiple sequence alignment through pipelined prefiltering. In *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, pages 177–183.

Ramachandran, A., Heo, Y., Hwu, W., Ma, J., and Chen, D. (2015). Fpga accelerated dna error correction. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1371–1376.

Salmela, L. and Rivals, E. (2014). Lordec: accurate and efficient long read error correction. *Bioinformatics*, 30(24):3506–3514.

Varma, B. S. C., Paul, K., Balakrishnan, M., and Lavenier, D. (2013). Fassem: Fpga based acceleration of de novo genome assembly. In *2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 173–176.