

# Suporte ao Paralelismo Multi-Core com FastFlow e TBB em uma Aplicação de Alinhamento de Sequências de DNA

Júnior Löff<sup>1</sup>, Dalvan Griebler<sup>1</sup>, Edans Sandes<sup>2</sup>, Alba Melo<sup>2</sup>, Luiz G. Fernandes<sup>1</sup>

<sup>1</sup> Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)  
Grupo de Modelagem de Aplicações Paralelas (GMAP), Porto Alegre – RS – Brasil

`junior.loff@acad.pucrs.br`

<sup>2</sup>Universidade de Brasília (UnB) – Brasília – DF – Brasil

**Resumo.** *Quando uma sequência biológica é obtida, é comum alinhá-la com outra já estudada para determinar suas características. O desafio é processar este alinhamento em tempo útil. Neste trabalho exploramos o paralelismo em uma aplicação de alinhamento de sequências de DNA utilizando as bibliotecas FastFlow e Intel TBB. Os experimentos mostram que a versão TBB obteve até 4% melhor tempo de execução em comparação à versão original em OpenMP.*

## 1. Introdução

As sequências biológicas podem ser classificadas em: DNA, RNA e sequência de proteínas. Para estes dois últimos, as sequências são menores e atingem no máximo alguns milhares de caracteres. Em contrapartida, as sequências de DNA são bem mais longas, podendo chegar a milhões de nucleotídeos [Sandes et al. 2016]. Na literatura, vários algoritmos foram propostos para a computação de alinhamento de sequências de DNA. Dentre eles podemos citar dois importantes: o algoritmo Needleman-Wunsch (NW) e o algoritmo Smith-Waterman (SW). No trabalho [Sandes and de Melo 2013] foi introduzido o CUDAlign, uma aplicação real para alinhamento de sequências de DNA que implementa otimizações paralelas para GPUs nos algoritmos NW e SW.

Através do trabalho [Sandes et al. 2016], foi identificado que 90% do código do CUDAlign é independente da plataforma e apenas 10% do código é específico para GPUs. Com base no código do CUDAlign, a arquitetura MASA (*Multiplatform Architecture for Sequence Aligners*) [Sandes et al. 2016] foi proposta como uma solução para aplicações de alinhamento em plataformas heterogêneas. Atualmente, o MASA foi implementado em três versões: uma implementação em OpenMP visando o acelerador Intel Phi e duas versões para CPU implementadas em OpenMP e OmpSs.

O objetivo do trabalho é introduzir o suporte ao paralelismo encontrado nas CPUs (multi-core) com FastFlow (FF) e Thread Building Blocks (TBB) e realizar uma análise do desempenho da aplicação MASA paralelizada. Ambas bibliotecas implementam a linguagem padrão C++, por isso divergem das versões já implementadas no MASA, cujas bibliotecas utilizam diretivas de compilação. Através deste trabalho, vários estudos que utilizam o alinhamento de sequências de DNA podem ser beneficiados. Podemos citar alguns, como: (1) Biologia evolutiva, para estudo dos diferentes organismos e suas relações, identificando a origem e descendência das espécies, assim como sua mudança ao longo do tempo (evolução). (2) Medicina, para identificar, diagnosticar e efetuar tratamentos para doenças genéticas. (3) Ciência forense, amplamente aceita para investigação de crimes e resolução de questões cíveis, penais ou administrativas (como por exemplo, o

teste de paternidade). Este trabalho está organizado da seguinte maneira. A seção 2 contém os trabalhos relacionados. A seção 3 apresenta a aplicação MASA e a seção 4 contém os detalhes da implementação paralela da aplicação. Por fim, na seção 5 são discutidos os resultados obtidos, cujas conclusões são descritas na seção 6.

## 2. Trabalhos Relacionados

Na literatura, vários trabalhos que contribuem no estudo das técnicas de alinhamento de sequências de DNA podem ser citados. Em [Rajko and Aluru 2004], foi proposto um algoritmo de alinhamento para CPUs que diminui a complexidade do tempo de execução e do armazenamento sem apresentar trocas. No trabalho [Farrar 2007], foram utilizadas instruções *Single-Instruction Multiple-Data* (SIMD) para paralelizar o algoritmo SW à nível de instruções. Em [Aldinucci et al. 2010], foi continuado o trabalho descrito anteriormente, onde estendeu-se o algoritmo SW para as seguintes bibliotecas paralelas: OpenMP, FastFlow, TBB e Cilk. Este artigo merece uma atenção especial, pois é familiar com a nossa pesquisa. O nosso trabalho difere deste pois iremos utilizar como estudo o MASA, uma aplicação de alinhamento que introduz otimizações sobre os algoritmos SW e NW. Além do mais, pretendemos estender o escopo de memória compartilhada já implementado no MASA com OpenMP e OmpSs (diretivas de compilação), utilizando bibliotecas oriundas da linguagem padrão do C++, como TBB e FastFlow.

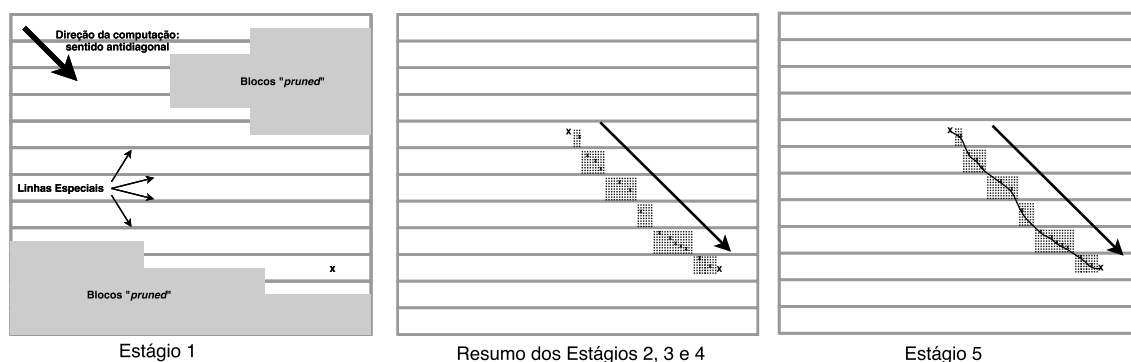
## 3. A Arquitetura MASA

MASA é uma arquitetura flexível e customizável para alinhamento de sequências de DNA em diferentes plataformas de *hardware/software*. Esta arquitetura foi desenvolvida em C++ com o paradigma OO. A computação é distribuída entre seis estágios que dividem o processamento do alinhamento. O primeiro estágio, descrito na Figura 1, representa a computação intensa da aplicação, onde a matriz é processada, o escore do alinhamento ótimo (representado pelo  $x$ ) é calculado e linhas especiais são salvas em memória. Ademais, blocos sem potencial podem ser detidos através da otimização `block pruning`, reduzindo a carga a ser processada. No Segundo, terceiro e quarto estágio a computação é similar, por isso abstraímos o processamento em uma única imagem. Estes estágios representam a computação do aprimoramento/redução da região potencial a conter alinhamentos. Já no quinto estágio, representado pelo último bloco da Figura 1, o alinhamento de DNA é realizado sobre o subconjunto reduzido dos milhões de nucleotídeos das sequências originais. No sexto estágio (opcional), os resultados são visualizados de maneira gráfica ou textual. Um exemplo de alinhamento de DNA é demonstrado a seguir.

```
TGTGG-CGGTCAAAACCC- - -A- - -CTATTTTCTGGATGTAGCGCAGCCTGGT
||||| |||| || |||| | |||||||||||||||||||||||||||||||
TGTGGTTCGGTTAATACCCCTTATTTCTATTTTCTGGATGTAGCGCAGCCTGGT
```

## 4. Projeto MASA-TBB e MASA-FastFlow

A arquitetura MASA facilita a implementação de aplicações de alinhamento de sequências em diversas plataformas. Deste modo, utilizou-se o código original do MASA-OpenMP para construir as implementações MASA-FastFlow e MASA-TBB. Atualmente, as bibliotecas FastFlow e TBB apresentam limitações na abstração do paralelismo em implementações bem estruturadas, como está disposto o MASA. Isto exigiu uma pequena



**Figura 1. Abstração dos estágios de execução do MASA.**

refatoração do código para habilitar o paralelismo. Nos estudos verificamos que a parte mais custosa está no primeiro estágio, onde a matriz de programação dinâmica é calculada. Neste processamento, as tarefas são dependentes, para isto foi utilizado a varredura em direção anti-diagonal (representada pela flecha da Figura 1) onde os dados se comportam independentemente, podendo ser executados em paralelo. Outro desafio foi conseguir obter acesso aos métodos OO após introduzir a biblioteca paralela. Isto acontece porque cada biblioteca cria o seu próprio escopo que é abstraído aos olhos do programador. Para resolver isto, criamos um ponteiro que aponta para a estrutura `this` e o passamos como parâmetro para dentro do escopo paralelo. O esqueleto paralelo utilizado foi um `map` dos blocos gerados após a divisão da matriz com quantidade de blocos igual ao número especificado como parâmetro na função `setPreferredSizes()`.

## 5. Resultados

Os experimentos foram executados em uma máquina equipada com 24 GB de RAM e dois processadores Intel(R) Xeon(R) CPU E5-2620 v3 2.40GHz com 6 núcleos cada e função hyper-threading, totalizando 24 núcleos. O sistema operacional era Ubuntu Server 64 bits com kernel 4.4.0-59-generic. Além disso, utilizamos o compilador GCC 5.4.0 com a otimização `-O3` e as bibliotecas: Thread Building Blocks (4.4 20151115) e FastFlow (r13). Os testes foram repetidos 5 vezes para cada amostra, onde foi utilizado a média aritmética e calculado o desvio padrão que está plotado no gráfico. Foram utilizadas sequências de DNA reais obtidas do National Center for Biotechnology Information (NCBI) referentes à duas bactérias: AE002160.2 (*Chlamydia muridarum* Nigg) e CP000051.1 (*Chlamydia trachomatis* A/HAR-13), ambas com aproximadamente 1 milhão de nucleotídeos. A matriz de programação dinâmica resultante possui 1,12 trilhão de células.

Nos nossos testes, diferente de executar toda a aplicação, focamos no primeiro estágio, que é o mais custoso e possui o maior potencial para usufruir do paralelismo intenso. Os resultados representados pela Figura 2 são referentes à execução do primeiro estágio sem a otimização `block pruning`. Neste gráfico, representa-se a relação de tempo de execução (eixo das ordenadas) por número de *threads* (eixo das abscissas). As versões em OpenMP, FastFlow e Intel TBB estão representadas no gráfico através das abreviações OMP, FF e TBB respectivamente. O desvio padrão ficou abaixo de 2% no pior caso. Pode ser visto que a versão MASA-FastFlow inicia melhor comparado à versão original MASA-OpenMP e perde desempenho logo em seguida. O MASA-TBB é sutilmente mais rápido que o MASA-OpenMP. Estes detalhes podem ser vistos mais

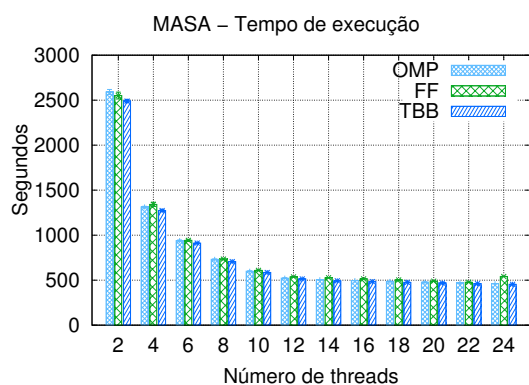


Figura 2. Desempenho do MASA.

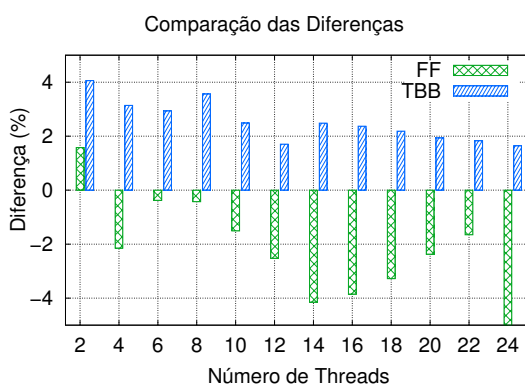


Figura 3. TBB e FastFlow vs OMP.

claramente na Figura 3, que representa a diferença do tempo de execução comparado com a versão original do MASA-OpenMP. Nesta figura fica evidente que a versão MASA-TBB executou em até 4% menos tempo que a versão do MASA-OpenMP. Ao chegar em 24 *threads*, a versão MASA-FastFlow ficou pior com um pico de 14% maior tempo de execução que a versão original. Isto acontece pois o FastFlow utiliza uma *thread* adicional para realizar o escalonamento.

## 6. Conclusões

Este trabalho apresentou a implementação do paralelismo em uma arquitetura de alinhamento de sequências de DNA, denominada MASA. Foram implementadas duas versões paralelas adicionais (MASA-TBB e MASA-FastFlow) e comparadas com a versão original MASA-OpenMP. O MASA-TBB atingiu 10,98 de *speed-up* enquanto que o MASA-OpenMP obteve 10,79. Além do mais, o MASA-TBB obteve até 4% de redução no tempo de execução em relação ao MASA-OpenMP, onde considerando a complexidade de uma aplicação de alinhamento, isto pode representar um considerável tempo de execução adicional. Como trabalhos futuros, pretendemos estender o paralelismo para os outros estágios através de uma análise específica de cada um deles, testar outras otimizações nas bibliotecas já implementadas e executar testes com sequências maiores de DNA.

## Referências

- [Aldinucci et al. 2010] Aldinucci, M., Meneghin, M., and Torquati, M. (2010). Efficient Smith-Waterman on Multi-core with FastFlow. In *PDP*, pages 195–199.
- [Farrar 2007] Farrar, M. (2007). Striped Smith-Waterman speeds database searches six times over other SIMD implementations. In *Bioinformatics*, volume 23, pages 156–161.
- [Rajko and Aluru 2004] Rajko, S. and Aluru, S. (2004). Space and time optimal parallel sequence alignments. *TPDS*, 15(12):1070–1081.
- [Sandes and de Melo 2013] Sandes, E. and de Melo, A. C. M. A. (2013). Retrieving Smith-Waterman Alignments with Optimizations for Megabase Biological Sequences Using GPU. *TPDS*, 24(5):1009–1021.
- [Sandes et al. 2016] Sandes, E., Miranda, G., Martorell, X., Ayguade, E., Teodoro, G., and de Melo, A. C. M. A. (2016). MASA: A Multiplatform Architecture for Sequence Aligners with Block Pruning. *TOPC*, 2(4):28:1–28:31.