UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ALEF FARAH

Estudo e otimização dos softwares de análise filogenética do Hospital de Clínicas de Porto Alegre

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Claudio Fernando Resin Geyer
Co-orientador: Prof. Dr. Julio Cesar Santos Anjos

Porto Alegre
2021

RESUMO

A análise filogenética compreende o estudo da evolução dos organismos e suas características. Neste trabalho foi realizado um estudo e otimização dos problemas de desempenho presentes nos softwares de análise filogenética utilizados pelo grupo de pesquisa em genética do Hospital de Clínicas de Porto Alegre (HCPA), empregando para isso técnicas de processamento paralelo. O tempo total da análise realizada pelo grupo foi reduzido de X para Y.

Palavras-chave: Análise filogenética. otimização. paralelismo.

Study and optimization of the phylogenetic analysis software used by Hospital de Clínicas de Porto Alegre

ABSTRACT

Phylogenetic analysis comprises the study of the evolution of living beings and their characteristics. In this paper we studied and optimized the performance bottlenecks present in the phylogenetic analysis software used by the genetics research group from the Hospital de Clínicas de Porto Alegre, using parallel programming techniques. The analysis total execution time was reduced from X to Y.

Keywords: Phylogenetic analysis. Optimization. Parallelism..

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HCPA	Hospital de Clínicas de Porto Alegre
PAML	*Phylogenetic Analysis By Maximum Likelihood*
CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
BFGS	Broyden–Fletcher–Goldfarb–Shanno
DNA	Ácido desoxirribonucleico
RNA	Ácido ribonucleico

LISTA DE SÍMBOLOS

ω Taxa de substituição

SUMÁRIO

1 INTRODUÇÃO

Neste trabalho foi realizado um estudo e otimização dos problemas de desempenho presentes nos softwares de análise filogenética utilizados pelo grupo de pesquisa em genética do Hospital de Clínicas de Porto Alegre (HCPA), empregando para isso técnicas de processamento paralelo e distribuído sempre que viável.

A análise realizada pelo grupo emprega softwares diversos, que apresentavam tempos de execução proibitivamente lentos - a análise de um único gene levava até dois dias, enquanto o grupo pretende analisar centenas de genes - além de uso de memória proibitivamente elevado em certas etapas da análise. Neste trabalho foram estudados os softwares utilizados pelo grupo, realizando para tal revisão bibliográfica sobre o tema e análises de desempenho aprofundadas. Além disso, foi estudada a possibilidade de paralelização dos algoritmos empregados visando a redução do seu tempo de execução, e, sempre que possível, foi implementada tal paralelização, seguida de novas análises, tanto da corretude da nova implementação como das melhorias de desempenho obtidas.

Por fim, as otimizações resultantes deste trabalho nos diversos softwares que compõem a análise filogenética foram reunidas em uma ferramenta para execução de ponta a ponta da análise. Com isso, obteve-se uma redução significativa do tempo total de execução da análise, que de vários dias passou a levar algumas horas.

1.1 Análise filogenética

A análise filogenética, cujos softwares associados foram objeto de estudo desse trabalho, compreende o estudo da evolução de um ou mais organismos e suas características. Os pesquisadores do HCPA realizam tal estudo evolutivo a nível molecular, através da comparação da informação genética de diferentes linhagens ou populações de um ser.

A informação genética presente no DNA dos seres vivos, objeto de estudo do grupo de pesquisa do HCPA, é utilizada no processo de síntese proteica, onde os códons (tripla de nucleotídeos) presentes no RNA mensageiro, gerado a partir do DNA, determinam a síntese de aminoácidos específicos, componentes das proteínas, macromoléculas fundamentais à vida.

Existem quatro tipos de nucleotídeos no código genético, formados por adenina (A), guanina (G), uracila (U), e citosina (C). Dessa forma, existem $4^3 = 64$ códons distintos, dos quais três sâo códons de terminação, que indicam o fim da etapa de tradução na

síntese protéica, enquanto os outros 61 códons traduzem para um aminoácido específico.

Apenas 20 aminoácidos compõem as proteínas em seres vivos. Sendo assim, a maioria dos aminoácidos é traduzido por mais de um códon. Em outras palavras, a substituição de certos códons no DNA não produz alteração nos aminoácidos gerados na síntese proteica.

As substituições que não geram alterações na síntese proteica são chamadas de sinônimas ou silenciosas, enquanto as que modificam os aminoácidos gerados são não-sinônimas. Acredita-se que as substituições sinônimas sejam mais comuns e não sofram tanta pressão seletiva. A taxa de substituiçõs sinônimas e não sinônimas $\omega = \frac{dN}{dS}$ é uma medida de seleção natual, conforme a tabela abaixo.(YANG; NIELSEN, 2002)

Tabela 1.1 – Taxas de substituição

Taxa	*Seleção*
$\omega = 1$	Seleção neutra
$\omega < 1$	Seleção negativa ou purificadora
$\omega > 1$	Seleção positiva

Fonte: (YANG; NIELSEN, 2002)

Existem diversos modelos de análise dessas substituições em diferentes linhagens de um indivíduo, e variados softwares que os implementam. Esse trabalho foca nos modelos e softwares de análise filogenética utilizados pelo grupo do HCPA.

1.2 Software utilizado e abordagens selecionadas

O *Phylogenetic Analysis by Maximum Likelihood* (PAML) é um pacote de software com ferrametas para análise filogenética utilizando métodos estatísticos de máxima verossimilhança.(YANG, 2007) Dentre outras ferramentas disponíveis no pacote se destaca o codeml, amplamente utilizado na literatura.(MALDONADO et al., 2016)

Apesar de estatisticamente robusto,(MALDONADO et al., 2016) o codeml possui implementação ingênua de métodos numéricos computacionalmente custosos.(YANG, 2020) Para o caso de uso do grupo de pesquisa em genética do HCPA, o tempo de execução é de vários dias.

Na literatura encontra-se implementações diversas que buscam melhorar o desempenho do codeml. Em (MORETTI et al., 2012) os autores fornecem uma solução para cluster; em (MALDONADO et al., 2016) é implementado uma ferramenta para execução paralela de múltiplos *jobs* do codeml em uma única máquina (em CPU); em (SCHA-

BAUER et al., 2012) os autores re-escrevem o software melhorando sua organização e otimizando-o, substituindo implementações ingênuas por aquelas de bibliotecas de métodos numéricos como BLAS e LAPACK, e em (VALLE et al., 2014) os mesmos autores fornecem uma implementação paralela (em CPU) para um sub-conjunto de métodos do seu software.

Nesse trabalho foram exploradas todas as soluções supracitadas com exceção da abordagem para cluster, visto que os pesquisadores do HCPA vem utilizando um *setup* com apenas uma máquina, além da solução em questão ser customizada para um cluster específico ao qual os autores desse trabalho não possuem acesso.

Além disso, nesse trabalho o codeml foi perfilado utilizando as ferramentas callgrind e kcachegrind,(WEIDENDORFER, 2008) para os dados de entrada do caso de uso do grupo de pesquisa do HCPA. Com base no resultado desse perfil de execução, foram implementadas e testadas algumas abordagens de paralelismo em CPU das rotinas que consumiam o maior tempo de execução do codeml, mas tal abordagem foi posteriormente descartada em favor das soluções encontradas na literatura, conforme descrito em 1.2.1. Uma abordagem para GPU foi estudada, mas igualmente descartada em favor das soluções existentes, conforme descrito em 1.2.1.

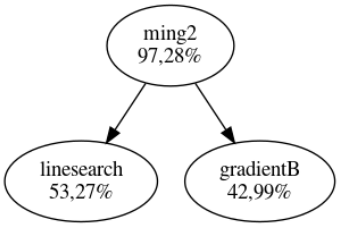
1.2.1 Perfil de execução e estudo de implementação paralela

A fim de determinar os gargalos de desempenho do codeml para o caso de uso do grupo de pesquisa do HCPA, foi traçado um perfil de execução da aplicação com os dados de entrada fornecidos pelos pesquisadores do HCPA utilizando as ferramentas callgrind e kcachegrind.(WEIDENDORFER, 2008)

O perfil de execução revelou que $97,28\%$ do tempo de execução da aplicação era dispendido na rotina *ming2*. Um estudo do código fonte revela que ela implementa o algoritmo de Broyden–Fletcher–Goldfarb–Shanno (BFGS), um método numérico para resolução de problemas de otimização. Tal método possui, no caso do codeml, duas sub-rotinas responsáveis por quase a totalidade de seu tempo de execução: *gradientB* e *LineSearch2*.

A rotina *gradientB*, responsável por $42,99\%$ do tempo total da aplicação, implementa o cálculo do gradiente via diferenças finitas, enquanto *LineSearch2*, responsável por $53,27\%$ do tempo de execução, implementa um método numérico de busca linear utilizando interpolação quadrática, descrito em (WOLFE, 1978). A figura 1.1 fornece uma

Figura 1.1 – Pilha de chamadas do codeml



Fonte: Os Autores

visualização da pilha de chamadas em questão.

Foi considerada uma implementação paralela para todos os métodos numéricos supracitados. Em um primeiro momento o cálculo do gradiente foi paralelizado utilizando OpenMP, um modelo de programação paralela para sistemas com múltiplos processadores com memória compartilhada.(CHANDRA et al., 2001) A escolha se justifica pela implementação ser trivial.

A fim de garantir a corretude da implementação paralela, foram desenvolvidos testes unitários para a função, utilizando dados de entrada arbitrários e comparando os resultados da implementação sequencial e paralela. Os testes revelaram que a implementação paralela gerava resultados diferentes da sequencial. Um estudo da implementação revelou que ela está correta, mas a função cujo grandiente está sendo calculado é implementada de forma não *thread-safe*, gerando condições de corrida.

Foi considerado a re-implementação das funções sendo diferenciadas, todavia, o uso extensivo de variáveis globais e memória compartilhada na implementação original do codeml demonstrou-se um grande obstáculo para essa abordagem, que por isso foi abandonada.

A partir daí foram voltadas as atenções para *LineSearch2*, mas não foi encontrado na literatura implementações paralelas para o método descrito em (WOLFE, 1978), e o algoritmo não é de paralelização trivial, portando tal abordagem também foi abandonada.

Por fim, as atenções foram voltadas para o próprio *ming2*, cujo método numérico (BFGS) é iterativo. Foi realizada uma revisão bibliográfica, que revelou implementações paralelas desse algoritmo para GPU em (FEI et al., 2014). Todavia, além de complexa e de difícil adaptação para o caso do codeml, o algoritmo não performa bem para entradas pequenas,(FEI et al., 2014). que é o caso do codeml que trabalha com um tamanho fixo de 61 (o número códons que traduzem para aminoácidos). Dessa forma, essa abordagem também foi descartada.

Esgotadas todas possibilidades de paralelização das rotinas responsáveis por quase a totalidade do tempo de execução do codeml para o caso de uso dos pesquisadores do HCPA, partiu-se para um estudo das soluções presentes na literatura.

1.2.2 Estudo do desempenho de soluções da literatura

A primeira aplicação a ser testada foi o slimcodeml,(SCHABAUER et al., 2012) visto que o fastcodeml(VALLE et al., 2014) não implementa os modelos necessários à análise do grupo do HCPA. Utilizando os dados de entrada fornecidos por pesquisadores do grupo, obteve-se uma redução no tempo total de execução de 16h38m para 5h24m, uma redução de 67,53% no tempo de execução. Os testes foram realizados em um ambiente controlado, de uso exclusivo dos autores, em uma máquina TODO.

Em um segundo momento foi testado a aplicação TODO,(MALDONADO et al., 2016)

1.3 samtools

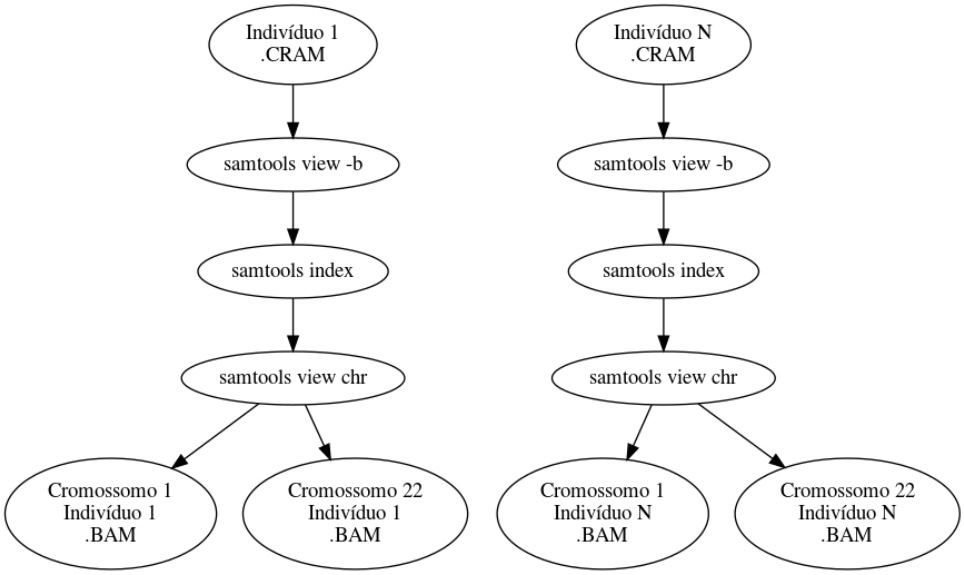
Outra aplicação utilizada no pipeline de análise do grupo de pesquisa em genética é o pacote samtools,(LI et al., 2009) aplicação amplamente utilizada na literatura para análise de sequências genéticas.(DANECEK et al., 2021) O samtoools e a ferramenta bcftools que o acompanha são construídos em cima da biblioteca htslib, dos mesmos autores, que permite a leitura e manipulação de arquivos de arquivos em formatos diversos representando sequências genéticas alinhadas.(DANECEK et al., 2021)

Em mais detalhes, o formato SAM traz uma representação em texto-plano das sequências genéticas alinhadas, o BAM é seu equivalente binário, e o CRAM é uma versão binária com compressão. Enquanto o samtools manipula esses arquivos, o bcftools manipula os arquivos no formato VCF, que armazena informação de variantes genéticas em texto-plano, e no formato BCF, seu equivalente binário.(DANECEK et al., 2021)

Os dados de sequenciamento genético utilizados como entrada no pipeline de análise do grupo de pesquisa do HCPA são obtidos no formato CRAM do projeto 1000 Genomes, um esforço colaborativo internacional que visa sequenciar o genoma humano e disponibilizar a informação publicamente.(VIA; GIGNOUX; BURCHARD, 2010)

Nesse trabalho foi estutdado, otimizado, e paralelizado o pipeline de análise do

Figura 1.2 – Etapa 1: Original



Fonte: Os Autores

grupo de pesquisa do HCPA utilizando o samtools e o bcftools, reduzindo o tempo de execução de vários dias para poucas horas.

1.3.1 Estudo e otimização

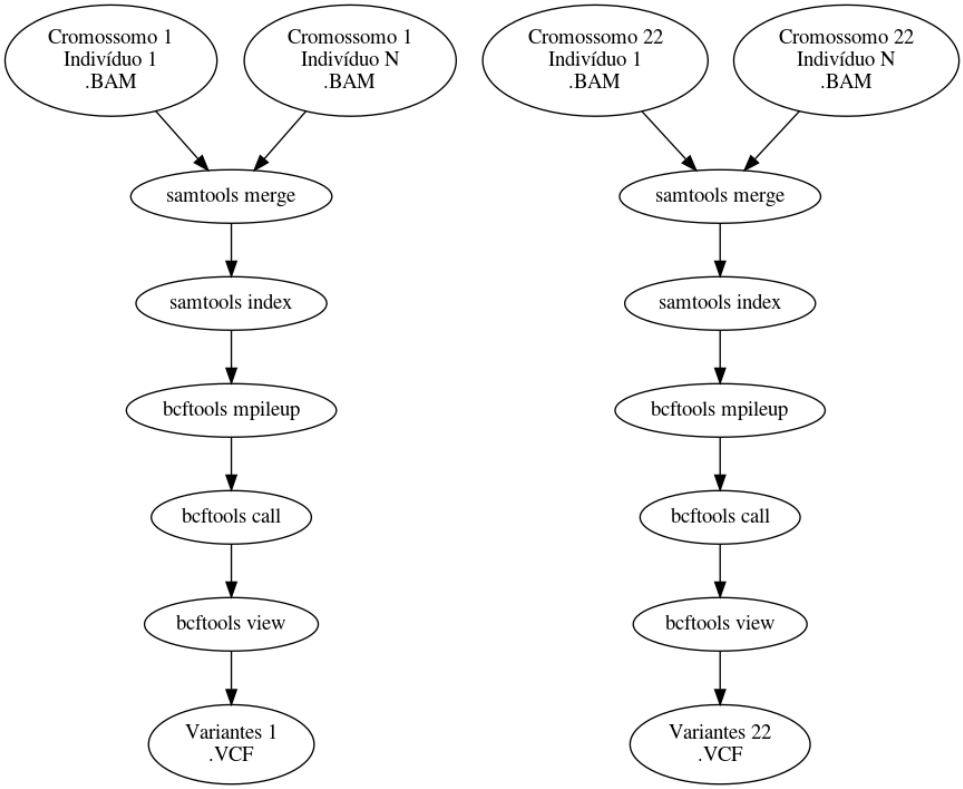
O pipeline de análise original fornecido pelos pesquisadores pode ser dividido em duas etapas. Na primeira etapa é executada uma conversão de formato de CRAM para BAM utilizando o comando *samtools view -b*, seguida de uma rotina de indexação dos arquivos utilizando *samtools index*.

Uma vez convertidos e indexados, cada arquivo de entrada é separado em 22 arquivos de saída, um para cada par de cromossomos autossômicos humanos, utilizando o comando *samtools view chr*. Essa etapa na sua forma originalmente usada pelo grupo de pesquisa encontra-se reproduzida na figura 1.2.

Numa segunda etapa, para cada cromossomo são aglutinados os respectivos arquivos de todos indivíduos, utilizando o comando *samtools merge*. Posteriormente esses arquivos são indexados, e os comandos *bcftools mpileup* e *bcftools call* são invocados para executar a chamada de variantes.

Por fim, o comando *bcftools view* é executado para converter a saída do formato BCF para VCF. Independente do número de arquivos de entrada, a saída é sempre 22

Figura 1.3 – Etapa 2: Original



Fonte: Os Autores

arquivos. Essa etapa encontra-se reproduzida na figura 1.3.

1.3.2 Paralelização

TODO

REFERÊNCIAS

CHANDRA, R. et al. **Parallel programming in OpenMP**. [S.l.]: Morgan kaufmann, 2001.

DANECEK, P. et al. Twelve years of samtools and bcftools. **Gigascience**, Oxford University Press, v. 10, n. 2, p. giab008, 2021.

FEI, Y. et al. Parallel l-bfgs-b algorithm on gpu. **Computers & graphics**, Elsevier, v. 40, p. 1–9, 2014.

LI, H. et al. The sequence alignment/map format and samtools. **Bioinformatics**, Oxford University Press, v. 25, n. 16, p. 2078–2079, 2009.

MALDONADO, E. et al. Lmap: lightweight multigene analyses in paml. **BMC bioinformatics**, BioMed Central, v. 17, n. 1, p. 1–11, 2016.

MORETTI, S. et al. gcodeml: a grid-enabled tool for detecting positive selection in biological evolution. In: **HealthGrid**. [S.l.: s.n.], 2012. p. 59–68.

SCHABAUER, H. et al. Slimcodeml: an optimized version of codeml for the branch-site model. In: IEEE. **2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum**. [S.l.], 2012. p. 706–714.

VALLE, M. et al. Optimization strategies for fast detection of positive selection on phylogenetic trees. **Bioinformatics**, Oxford University Press, v. 30, n. 8, p. 1129–1137, 2014.

VIA, M.; GIGNOUX, C.; BURCHARD, E. G. The 1000 genomes project: new opportunities for research and social challenges. **Genome medicine**, BioMed Central, v. 2, n. 1, p. 1–3, 2010.

WEIDENDORFER, J. Sequential performance analysis with callgrind and kcachegrind. In: **Tools for High Performance Computing**. [S.l.]: Springer, 2008. p. 93–113.

WOLFE, M. A. **Numerical methods for unconstrained optimization: an introduction**. [S.l.]: Van Nostrand Reinhold, 1978.

YANG, Z. Paml 4: phylogenetic analysis by maximum likelihood. **Molecular biology and evolution**, Oxford University Press, v. 24, n. 8, p. 1586–1591, 2007.

YANG, Z. **PAML: Phylogenetic Analysis by Maximum Likelihood**. [S.l.], 2020.

YANG, Z.; NIELSEN, R. Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. **Molecular biology and evolution**, Oxford University Press, v. 19, n. 6, p. 908–917, 2002.