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

ALEF FARAH

Estudo e otimização dos softwares de bioinformática 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

Neste trabalho foi realizado um estudo e otimização dos problemas de desempenho presentes nos softwares de bioinformática utilizados pelo grupo de pesquisa em genética do Hospital de Clínicas de Porto Alegre (HCPA), empregando para isso técnicas de processamento paralelo. O trabalho focou no software de análise filogenética codeml, do pacote PAML, amplamente utilizado na literatura, e no software SAMtools, usado para chamda de variantes, igualmente popular. Foi desenvolvida uma ferramenta de paralelização de *jobs* do SAMtools, reduzido de vários dias para poucas horas o tempo de análise do grupo de pesquisa, enquanto que no caso do codeml foi realizado uma análise de desempenho de alternativas encontradas em revisão bibliográfica, fornecendo aos pesquisadores uma ferramenta que reduz em mais da metade o tempo de execução em relação ao codeml original.

Palavras-chave: Bioinformática. paralelismo. codeml. PAML. SAMtools.

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

ABSTRACT

In this paper we studied and optimized the performance bottlenecks found in the bioinformatics software used by the genetics research group from Hospital de Clínicas de Porto Alegre (HCPA), employing parallel programming techniques to achieve these goals. The focus of our work was in the phylogenetic analysis software called codeml, from the PAML package, which is widely used in the literature, as well on the SAMtools software, used for variant calling, also popular. We developed a tool for the parallel execution of SAMtools jobs, reducing total execution time for the group's input from various days to a few hours, while in the case of codeml we analyzed the performance of solutions found in a bibliography review, supplying the researches with a tool whose execution time is half that of codeml.

Keywords: Bioinformatics, parallelism, codeml, PAML, SAMtools.

LISTA DE FIGURAS

Figura 2.1 Pilha de chamadas do codeml 16

Figura 4.1 Fluxo de análise original via SAMtools, etapa 1 22
Figura 4.2 Fluxo de análise original via SAMtools, etapa 2 23
Figura 4.3 Captura de tela da interface gráfica “SAMGUI” 26

LISTA DE TABELAS

Tabela 1.1 Taxas de substituição 11

Tabela 2.1 Hardware da máquina “Thor1” 15
Tabela 2.2 Tempos de execução das ferramentas de análise filogenética 15

Tabela 4.1 Tempos de execução do SAMtools 25
Tabela 4.2 Speedup da ferramenta 25

LISTA DE ABREVIATURAS E SIGLAS

HCPA	Hospital de Clínicas de Porto Alegre
PAML	*Phylogenetic Analysis By Maximum Likelihood*
FTCS	*Forward Time Centered Space*
CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
BFGS	Broyden–Fletcher–Goldfarb–Shanno
DNA	Ácido desoxirribonucleico
RNA	Ácido ribonucleico
RAM	*Random access memory*
GB	Giga byte
GHz	Giga hertz
ARC	*Advanced Resource Connector*
PBS	*Population Branch Statistic*

LISTA DE SÍMBOLOS

ω Taxa de substituição

dN Substituições não-sinônimas

dS Substituições sinônimas

F_{ST} Índice de fixação

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Conceitos de bioinformática 10
1.1.1 Análise filogenética 11
1.1.2 Population Branch Statistic 12
2 PACOTE PAML 13
2.1 Trabalhos anteriores 13
2.2 Análise de desempenho 14
2.3 Perfil de execução e estudo de implementação paralela 15
3 PACOTE ANGSD 19
4 PACOTE SAMTOOLS 21
4.1 Fluxo de análise original 21
4.2 Fluxo de análise otimizado 22
4.3 Ambiente e interface 25
REFERÊNCIAS 27

1 INTRODUÇÃO

Neste trabalho foi realizado um estudo e otimização dos problemas de desempenho presentes nos softwares de bioinformática 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.

Nomeadamente foram estudados os softwares codeml, do pacote PAML,(YANG, 2007) amplamente utilizado na literatura para análise filogenética;(MALDONADO et al., 2016) o pacote ANGSD, utilizado para comparação de sequências genéticas de diferentes populações de uma espécie;(KORNELIUSSEN; ALBRECHTSEN; NIELSEN, 2014) e o pacote SAMtools, amplamente utilizado na literatura para manipulação de arquivos de sequências genéticas alinhadas,(DANECEK et al., 2021) e usado pelo grupo como uma alternativa ao software ANGSD.

Cada um desses softwares apresentava problemas de desempenho diferentes dentro do pipeline de análise genética do grupo de pesquisa. Nomeadamente, o PAML e o SAMtools apresentavam tempos de execução proibitivamente elevados, enquanto que o ANGSD apresentava uso de memória proibitivamente alto. No caso dos softwares PAML e ANGSD foram realizadas análises de desempenho dos softwares originais bem como de alternativas encontradas através de um estudo da literatura, enquanto que no caso do pacote SAMtools foi desenvolvida uma ferramenta para execução de *jobs* paralelos para ambientes multiprocessados, bem como uma interface gráfica para uso desse ferramental.

A identificação de softwares alternativos ao PAML através do estudo e análise supracitados resultaram na economia de mais da metade do tempo de análise para os dados de entrada do grupo de pesquisa do HCPA, enquanto que a ferramenta desenvolvida para paralelização de *jobs* do SAMtools reduziu de dias para horas o tempo da análise realizada pelo grupo com essa ferramenta, permitindo que ela fosse utilizada como alternativa ao software ANGSD, cujos problemas de desempenho foram mapeados através de um perfil de execução, mas não foram atacados nesse trabalho, sendo a ferramenta de paralelização do SAMtools a alternativa favorecida.

1.1 Conceitos de bioinformática

A fim de compreender o uso e funcionamento dos softwares que foram objeto de estudo desse trabalho, convém elucidar alguns conceitos de bioinformática. Na seção

1.1.1 é apresentado o conceito de análise filogenética, objeto de estudo dos usuários do pacote PAML, enquanto que na seção 1.1.2 é exposto o conceito de análise de variações genéticas de diferentes populações de uma espécie através do método *Population Branch Statistic* (PBS), para o qual os pacotes ANGSD e SAMtools são empregados.

1.1.1 Análise filogenética

A análise filogenética, propósito do PAML, compreende o estudo da evolução de um ou mais organismos e suas características. Nesta sessão são brevemente apresentados alguns conceitos de análise filogenética relevantes ao entendimento dos softwares utilizados e seu comportamento.

A informação genética presente no DNA dos seres vivos 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 são 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 substituições 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)

Em síntese, o propósito de softwares de análise filogenética como o pacote PAML é detectar eventos de seleção positiva nas diferentes linhagens da árvore filogenética de uma espécie. Para isso, softwares como o PAML utilizam modelos estatísticos de máxima verossimilhança para comparar a hipótese de ocorrência desses eventos contra a hipótese nula de seleção neutra ou negativa.(MORETTI et al., 2012)

1.1.2 Population Branch Statistic

Um dos objetos de estudo da bioinformática é a compreensão dos genes envolvidos em variações fenotípicas observadas entre diferentes populações de uma espécie, em particular da espécie humana.(JIANG; ASSIS, 2019) Para isso, uma possibilidade é o estudo a nível molecular das variações genéticas entre indivíduos de diferentes populações. A identificação dessas variações é denominada "chamada de variantes" ou *SNV calling*, o pacote SAMtools providenciando ferramental para realização desse proceso.(PIROOZNIA et al., 2014) Existem diversos métodos para identificação e análise dessas variações, um desses métodos sendo o *Population Branch Statistic* ou PBS,(JIANG; ASSIS, 2019) para o qual pode ser empregado o software ANGSD.

Uma métrica de diferença na estrutura genética entre duas populações de uma espécie é o índice de fixação ou F_{ST}, baseado, em suma, na variança da frequência alélica das duas populações. Genes com um F_{ST} alto são potenciais alvos de seleção natural.(YI et al., 2010) Como o F_{ST} é uma métrica de comparação de pares, ele não é capaz de identificar a direção das mudanças, nem pode ser utilizado como única métrica para identificar eventos de seleção natural. O PBS consiste de um método estatístico que utiliza comparações em pares do F_{ST} entre três populações distintas para quantificar as diferenças entre suas sequências genéticas. Genes com valor PBS alto indicam seleção positiva.(JIANG; ASSIS, 2019) Uma descrição mais aprofundada do método foge ao escopo desse trabalho, e pode ser encontrada em (YI et al., 2010).

Os pesquisadores do HCPA utilizam os pacotes ANGSD e SAMtools para realizar etapas da análise de variações fenotípicas entre populações humanas utilizando o método PBS.

2 PACOTE PAML

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.

Neste trabalho foi realizada uma revisão bibliográfica a respeito do software em questão e de alternativas a ele; foram realizadas análises de seu desempenho através de perfis de execução, identificando as rotinas responsáveis pela maior parte do tempo de execução; os garagalos de desempenho foram estudados através de análise do código fonte e do manual da ferramenta, em seguida foi realizada revisão bibliográfica a respeito dos métodos numéricos empregados, visando sua otimização. Foram implementadas estratégias de paralelização de tais métodos e realizadas novas análises de desempenho e de corretude do software paralelizado. Por fim, foram realizadas análises de desempenho de softwares alternativas encontrados no estudo da literatura.

2.1 Trabalhos anteriores

Em (MORETTI et al., 2012) os autores exploram o fato de múltiplas execuções do codeml serem independentes, o que torna a aplicação embaraçosamente paralela para múltiplas entradas. Nesse trabalho os autores visam atender às necessidades do grupo de pesquisa que mantém o banco de dados Selectome, em que múltiplas instâncias do codeml precisam ser executadas, uma para cada arquivo de entrada mantido pelo banco de dados. Para isso, os autores obtam por um paralelismo de *jobs*, desenvolvendo uma ferramenta voltada para execução em cluster, mais especificamente visando ambientes com o *middleware* Advanced Resource Connector (ARC), utilizado no grid aos quais os autores possuíam acesso. A ferramenta, chamada gcodeml, é desenvolvida na linguagem de programação Python, utilizando a biblioteca GC3Pie, que providencia *bindings* para controle e execução de *jobs* ARC. O caso de uso dos pesquisadores do HCPA envolve um arquivo de entrada único, e o ambiente de execução aos quais os pesquisadores possuem

acesso não é de um cluster, portanto essa solução não foi estudada mais a fundo nesse trabalho.

Em (MALDONADO et al., 2016) é implementado uma ferramenta para execução paralela de múltiplos *jobs* do codeml em uma única máquina (em CPU).

Em (SCHABAUER et al., 2012) os autores otimizam e organizam software original (codeml), mantendo o formato de entrada e saída bem como os métodos numéricos utilizados. Em mais detalhes, os autores substituem implementações ingênuas de métodos numéricos por aquelas de bibliotecas amplamente utilizadas na literatura como BLAS e LAPACK, além de manipularem equações matriciais a fim de permitir o uso de implementações mais eficientes mas dependentes de propriedades como a simetria das matrizes envolvidas na operação. A implementação é sequencial. O novo software, nomeado slimcodeml, desempenhou quase dez vezes melhor que o original para os dados de entrada dos autores, em uma máquina com processador Intel Xeon W3540 de 2.93 GHz. O fato da entrada e saída serem os mesmos, bem como os métodos numéricos, e o bom desempenho em ambiente semelhante ao utilizado pelos pesquisadores do HCPA fizeram dessa ferramenta um dos focos de avaliação de desempenho desse trabalho.

Em (VALLE et al., 2014) os mesmos autores do slimcodeml escrevem um novo software, com base de código completamente independente do codeml original, fornecendo uma implementação paralela em CPU. Essa implementação, chamada fastcodeml, possui formatos de entrada e saíde diferentes do software original, e implementa apenas um sub-conjunto de seus métodos. O propósito dos autores nesse trabalho foi explorar estratégias de paralelização e otimização para o problema de análise filogenética de forma geral. Não possuindo as mesmas entradas e saídas do software original nem implementando todos seus métodos, essa solução não é útil aos pesquisadores do HCPA, mas serve como referência para o desenvolvimento de novos softwares de análise filogenética.

Nesse trabalho foram exploradas as soluções supracitadas, com ênfase na análise do desempenho da ferramenta desenvolvida por (SCHABAUER et al., 2012), por ser a que melhor atendia aos requisitos dos pesquisadores do HCPA.

2.2 Análise de desempenho

A primeira aplicação a ser testada foi o slimcodeml.(SCHABAUER et al., 2012) Utilizando os dados de entrada fornecidos por pesquisadores do grupo, obteve-se uma redução no tempo total de execução de 16h38m para 5h24m, ou 67,53%, em relação ao

codeml original. Os testes foram realizados em um ambiente controlado, de uso exclusivo dos autores, em uma máquina com o hardware descrito na tabela 2.1, doravante “Thor1”.

Tabela 2.1 – Hardware da máquina “Thor1”

Componente	*Especificação*
Modelo CPU	Intel Core i5-9400
Cores CPU	6 núcleos
Clock CPU	2.90 GHz (base), 4.5 GHz (turbo)
Cache CPU	9M
RAM	64 GB

Fonte: Os autores

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

A tabela 2.2 apresenta uma comparação dos tempos de execução de cada uma das ferramentas estudadas para os arquivos de entrada fornecidos pelos pesquisadores do HCPA, no ambiente de execução descrito na tabela 2.1.

Tabela 2.2 – Tempos de execução das ferramentas de análise filogenética

Ferramenta	*Tempo de execução*
codeml (PAML)	16h38m
slimcodeml	5h24m
codeml (LMAP)	TODO

Fonte: Os autores

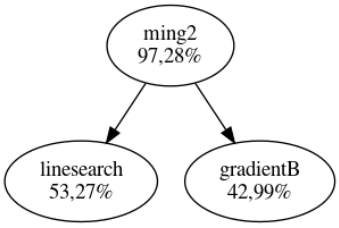
2.3 Perfil de execução e estudo de implementação paralela

Além do estudo de implementações presentes na literatura, 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, objetivando-se identificar gargalos de desempenho e explorar soluções alternativas às presentes na literatura.

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 será descrito abaixo. Uma abordagem para GPU foi estudada, mas igualmente descartada em favor das soluções existentes.

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

Figura 2.1 – Pilha de chamadas do codeml



Fonte: Os Autores

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 2.1 fornece uma 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)

O cálculo do gradiente é aproximado utilizando diferenças finitas para obter as derivadas parciais de primeiro grau. O codeml permite utilizar diferenças finitias progressivas, centradas, ou regressivas. A paralelização se dá sob todas variáveis da função cujo gradiente está sendo obtido. O número de threads foi definido pelo mínimo entre o número de núcleos de processamento disponíveis e o número de variáveis na função (iterações no laço). Foi utilizado um escalonador estático com tamanho do bloco igual ao número de variáveis sob o número de threads, uma vez que a carga de trabalho é homogênea (as variáveis são todas da mesma função). O algoritmo encontra-se reproduzido abaixo, onde p é o número de processadores disponíveis, t o número de threads a ser

usado, c o tamanho do bloco, e n o número de variáveis.

$t \leftarrow \max(1, \min(p, n))$

$c \leftarrow \max(1, \frac{n}{t})$

for $i \leftarrow 0, n$ **in parallel do**

if centrada **then**

$\frac{\partial f}{\partial x_i} \leftarrow \frac{f(x+h)-f(x-h)}{2h}$

else if progressiva **then**

$\frac{\partial f}{\partial x_i} \leftarrow \frac{f(x+h)-f(x)}{h}$

else

$\frac{\partial f}{\partial x_i} \leftarrow \frac{f(x)-f(x-h)}{h}$

end if

end for

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, apesar da implementação estar correta, a função cujo grandiente está sendo calculado é implementada de forma não *thread-safe*, gerando condições de corrida que inviabilizam a simples paralelização da rotina em questão.

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. Foi estudada então a implementação do slimcodeml, que re-organiza amplas seções do código fonte, na esperança de que tais dependências pudessem ter sido removidas, possibilitando a implementação paralela do cálculo do gradiente. Todavia, apesar de melhor organizado, o slimcodeml ainda apresenta o mesmo compartilhamento de memória e uso de variáveis globais encontrados na implementação original, inviabilizando essa estratégia de paralelização nesse software.

A partir daí foram voltadas as atenções para *LineSearch2*, mas não foi encontrado na literatura implementações paralelas para o método descrito em (WOLFE, 1978), o algoritmo não é de paralelização trivial, e funções auxiliares utilizadas em seu cálculo apresentavam os mesmos problemas de compartilhamento de memória encontrados nas outras rotinas estudadas, tanto na implementação original como no slimcodeml, portanto a paralelização dessa rotina 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), cujos resultados demonstram que a implementação não performa bem para entradas pequenas. Foi realizado então um estudo do uso pelo codeml desse algoritmo, realizando para isso pequenas adaptações no código para aumentar a verbosidade dos *logs* da aplicação, e foi observado que o codeml trabalha com uma entrada fixa de tamanho 61 para esse algoritmo – o número códons que traduzem para aminoácidos – tamanho esse considerado pequeno com base no estudo supracitado. Dessa forma, conclui-se que uma paralelização em GPU seria ineficiente. Além disso, a paralelização desse algoritmo é complexa e de difícil adaptação ao caso do codeml. 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, optou-se por fornecer aos pesquisadores as ferramentas encontradas através de estudo da literatura. Em particular, os pesquisadores mostraram-se satisfeitos com o uso do slimcodeml como uma alternativa à implementação original do codeml.

3 PACOTE ANGSD

Outra aplicação utilizada pelo grupo de pesquisa em genética é o pacote ANGSD, software para análise genética de diferentes populações de uma espécie.(KORNELIUSSEN; ALBRECHTSEN; NIELSEN, 2014) Os pesquisadores utilizam esse software para determinar se a genética de uma população influencia em alguma característica específica de seus indivíduos, através de um teste chamado *Population Branch Statistic* (PBS), caracterizado pela comparação da frequência de ocorrência de determinados alelos entre pares de indivíduos de diferentes populações.(YI et al., 2010)

A análise em questão consiste de duas etapas, a primeira utilizando o binário angsd, a aplicação principal do pacote ANGSD, e a segunda etapa utilizando a ferramenta realSFS, um utilitário fornecido pelo pacote. A primeira etapa leva cerca de um dia para cada arquivo de entrada e porventura apresenta uso elevado de memória, enquanto que a segunda etapa, que recebe como entrada a saída da etapa anterior, nunca termina a execução devido a uso de memória proibitivamente elevado, para os dados de entrada dos pesquisadores do HCPA.

A entrada da primeira etapa consiste em arquivos nos formatos BAM ou CRAM, respectivamente uma representação binária de sequências genéticas alinhadas e seu equivalente com compressão, um arquivo de entrada para cada indivíduo de cada população sob análise. O angsd então utiliza a biblioteca htslib(BONFIELD et al., 2021) para manipulação desses arquivos.

Uma primeira observação realizada através de reuniões com os pesquisadores é que esses dados de entrada são obtidos sempre 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) e que estava sendo realizada uma etapa de conversão de CRAM para BAM para uso do angsd. Essa conversão é realizada utilizando o software SAMtools, descrito em mais detalhes na seção 4, e leva algumas horas para cada arquivo de entrada. Todavia, como observado anterioremente, o angsd trabalha também com o formato CRAM, fato esse observado através do estudo do manual da ferramenta. Sendo assim, a etapa de conversão é desnecessária. Essa observação já economizou algumas horas de execução aos pesquisadores.

Eliminada essa etapa de pré-processamento dos dados, realizou-se um estudo do software a fim de elucidar seu funcionamento, para posteriormente realizar um perfil de seu uso de memória com a ferramenta massif, do software callgrind.(WEIDENDORFER,

2008)

Inicialmente realizou-se um estudo do utilitário realSFS, que apresentava o principal problema de desempenho. Através de um estudo do código fonte foi observado que o utilitário implementa dois otimizadores, um legado utilizando BFGS (o mesmo algoritmo usado no pacote PAML) e o padrão utilizando "Electromagnetism-like Mechanism", um método estocástico de otimização não-linear que utiliza mecanismos de atração e repulsão para mover "partículas" na direção ótima.(WANG; ZENG; SONG, 2010) Foi observado ainda que, independente do otimizador utilizado, a alocação de memória é a mesma. Fatores que influenciam no uso de memória incluem o número de threads e o número de síteos considerados. Em ambos os casos há um *trade-off* - ao reduzir o número de threads o tempo de execução aumenta, e ao reduzir o número de síteos a confiabilidade dos resultados diminui.(KORNELIUSSEN, 2016) A redução de ambos é indesejável, visto que o tempo de execução já é muito elevado e a confiabilidade dos resultados é fundamental.

Foi traçado então um perfil de memória da aplicação utilizando callgrind, visando elucidar as causas do uso elevado de memória. Esse perfil revelou que a etapa de otimização supracitada é o principal responsável pelo uso proibitivamente alto de recursos, e não foram encontradas oportunidades de melhoria que não afetassem o tempo de execução ou a confiabilidade dos resultados.

Foi realizada uma reunião com os pesquisadores, em que foi observado que é possível realizar a mesma análise feita com o angsd (PBS) utilizando as ferramentas SAMtools e bcftools, construídas em cima da biblioteca htslib (tal qual o angsd), e que já eram utilizadas na etapa de pré-processamento dos dados, seguida de uma análise estatística dos resultados. Todavia, o pipeline de análise utilizando tais ferramentas apresentava tempos proibitivamente lentos. Na seção 4 é descrito um estudo desse pipeline e o posterior desenvolvimento de uma ferramenta que otimiza e paraleliza sua execução, resolvendo o problema de desempenho supracitado. Tal estudo foi realizado em paralelo ao estudo do angsd descrito acima e, uma vez que os resultados foram se mostrando mais frutíferos, o foco dese trabalho passou a ser esse ferramental, a análise pelo angsd sendo substituída pela análise pelo SAMtools por parte dos pesquisadores.

4 PACOTE SAMTOOLS

Conforme mencionado na seção 3, uma aplicação utilizada pelo 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. Dentro outras funcionalidades dessas ferramentas destaca-se a chamada de variantes, processo de comparação de 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 realiza a chamada de variantes, tendo tais arquivos como entrada, produzindo como saída 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)

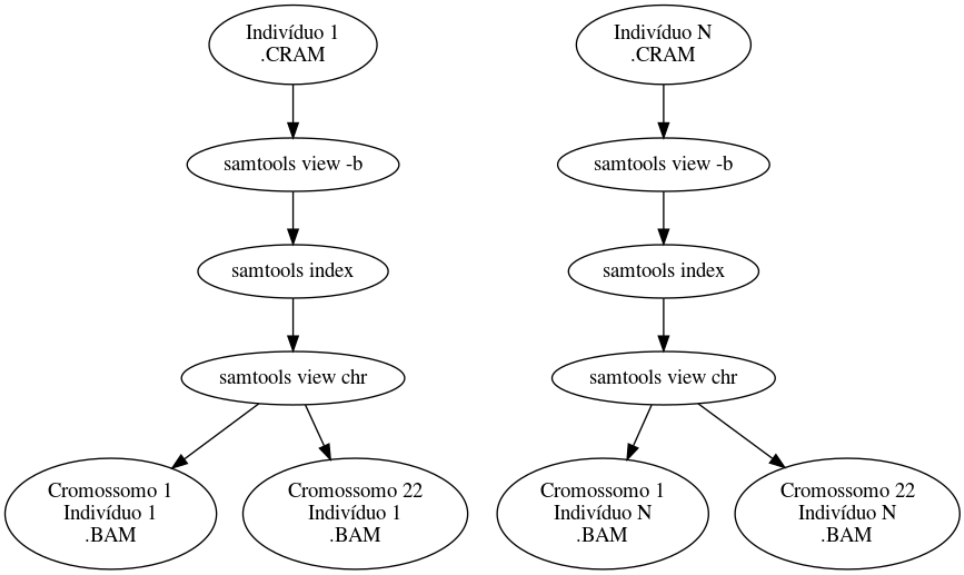
Nesse trabalho foi estutdado, otimizado, e paralelizado o pipeline de análise do grupo de pesquisa do HCPA utilizando o SAMtools e o bcftools, reduzindo o tempo de execução de vários dias para poucas horas, e desenvolvida uma interface gráfica para execução do pipeline otimizado.

4.1 Fluxo de análise original

O fluxo 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

Figura 4.1 – Fluxo de análise original via SAMtools, etapa 1



Fonte: Os Autores

pesquisa encontra-se reproduzida na figura 4.1.

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.

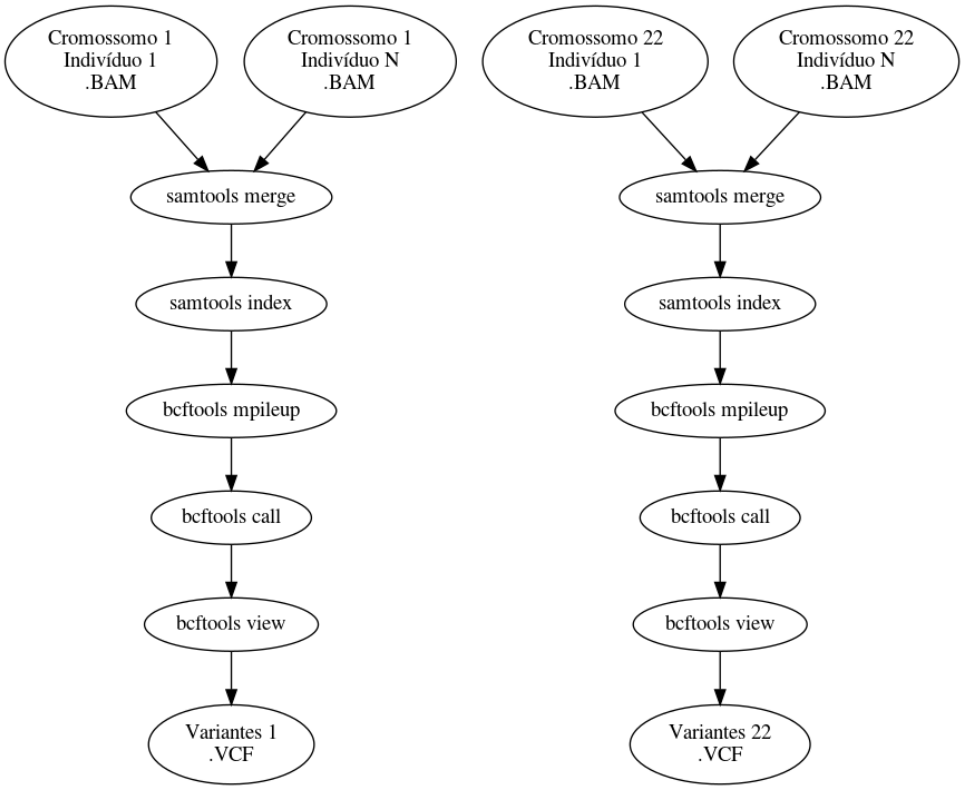
Por fim, o comando *bcftools view* é executado para converter a saída do formato BCF para VCF. Independente do número de arquivos de entrada, a saída é sempre 22 arquivos. Essa etapa encontra-se reproduzida na figura 4.2.

4.2 Fluxo de análise otimizado

Através de um estudo da literatura e do manual das ferramentas utilizadas percebeu-se oportunidade de melhorias no fluxo de análise utilizado pelos pesquisadores. Em particular, em,(DANECEK et al., 2021) os autores observam que o uso do comando *SAMtools view -b*, para conversão de CRAM para BAM, apesar de frequentemente presente em guias na internet é normalmente desnecessário, uma vez que as ferramentas possuem suporte a CRAM.

Foram realizados testes de desempenho de cada uma das etapas do fluxo original de análise, e a conversão de CRAM para BAM era particularmente custosa. Observado

Figura 4.2 – Fluxo de análise original via SAMtools, etapa 2



Fonte: Os Autores

isso, foi testado e comprovado que todo restante da análise poderia ser executado sem essa etapa, observando o suporte para CRAM e comparando as saídas do fluxo otimizado com o fluxo original.

Além disso, foi observado que ambas etapas eram embaraçosamente paralelas, consistindo de passos independentes para cada arquivo de entrada. Como os pesquisadores do HCPA possuem acesso a máquinas multiprocessadas, foi realizada uma paralelização de ambas etapas utilizado GNU Parallel(TANGE et al., 2011). A escolha dessa ferramenta se deu pela facilidade de uso, disponibilidade nos ambientes usados pelos pesquisadores, e suporte a ajuste automático do número de *jobs* paralelos ao número de núcleos de processamento disponíveis.

Foi desenvolvida uma ferramenta para execução do fluxo de análise otimizado e paralelizado. Um pseudo-código da ferramenta encontra-se reproduzido abaixo.

$C \leftarrow$ input CRAMs
$n \leftarrow$ input length
for $j \leftarrow 0, n$ **in parallel do**
    SAMtools index C(j)
    for $i \leftarrow 1, 22$ **do**
        $R(i, j) \leftarrow$ SAMtools view $C(j)$ chr i
    end for
end for
for $i \leftarrow 1, 22$ **in parallel do**
    $M(i) \leftarrow$ SAMtools merge $R(i, j)$ **for j in 0,n**
    SAMtools index $M(i)$
    bcftools mpileup $M(i)$
    bcftools call $M(i)$
    bcftools view $M(i)$
end for

As otimizações e a paralelização reduziram drasticamente o tempo total de execução da análise. Na tabela 4.1 encontram-se reproduzidos os tempos de execução utilizando o pipeline original e o otimizado. Os testes foram realizados na máquina Thor1, descrita na tabela 2.1, utilizando todos núcleos de processamento disponíveis.

Além disso, foram realizados testes de desempenho para a entrada de 58GB utilizando um número diferente de núcleos de processamento, a fim de estimar o *speedup* bem como a quantidade de tempo economizado pela paralelização, o resto sendo atribuído às

Tabela 4.1 – Tempos de execução do SAMtools

Tamanho da entrada	*Tempo original*	*Tempo otimizado*
58 GB (2 CRAMs)	TODO	TODO
124 GB (4 CRAMs)	TODO	TODO
232 GB (8 CRAMs)	TODO	TODO

Fonte: Os autores

otimizações no fluxo de análise. Os resultados encontram-se reproduzidos na tabela 4.2.

Tabela 4.2 – Speedup da ferramenta

Número de núcleos	*Tempo de execução*	*Speedup*
1 núcleo	TODO	TODO
2 núcleos	TODO	TODO
4 núcleos	TODO	TODO
6 núcleos	TODO	TODO

Fonte: Os autores

4.3 Ambiente e interface

Tanto o SAMtools e o bcftools como a ferramenta desenvolvida pelos autores requer um ambiente com uma série de softwares pré-instalados e em versões específicas, o que porventura gerava dificuldades aos pesquisadores do HCPA.

A fim de superar tais dificuldades, bem como disponibilizar um ambiente para reprodução fidedigna dos resultados obtidos nesse trabalho, decidiu-se por utilizar o software Docker, de virtualização a nível de sistema operacional, que fornece uma série de vantagens para pesquisa reprodutível.(BOETTIGER, 2015) A imagem criada está disponível em repositório público.(FARAH, 2021)

Além disso, foi desenvolvida uma interface gráfica para o uso do SAMtools e bcftools para chamada de variantes. O intuito é facilitar o uso do ferramental para profissionais da biologia que porventura não estejam familiariezados com interfaces por linha de comando.

A interface é executada na máquina local do usuário, que pode configurar uma máquina remota para execução do pipeline de análise. Além da execução do pipeline, a interface automatiza uma série de outros passos realizados pelos pesquisadores, como obtenção dos dados e manipulação dos arquivos de resultado. A interface permite ainda realizar todas etapas em segundo plano, verificar seu progresso, e interrompê-las.

Para desenvolvimento da interface obtou-se por utilizar a linguagem de progra-

Figura 4.3 – Captura de tela da interface gráfica “SAMGUI”

Fonte: Os Autores

mação Python, amplamente utilizada para computação científica.(OLIPHANT, 2007) A escolha da linguagem se deu principalmente pelo sua disponibilidade em múltiplos sistemas operacionais, visto que os usuários (pesquisadores do HCPA) manifestaram interesse em utilizar a ferramenta nos sistemas operacionais Windows, Linux, e Mac, e há suporte tanto de interpretadores da linguagem como das bibliotecas utilizadas para todas essas plataforams.(OLIPHANT, 2007) Outro fator que influenciou a escolha foi a agilidade no desenvolvimento que a linguagem proporciona.(OLIPHANT, 2007)

Foi utilizada a biblioteca *guietta* para criação da interface gráfica,(PUGLISI, 2020) um *wrapper* em cima da biblioteca *PySide2*, um *binding* para Python da biblioteca multiplataforma Qt.(LOGANATHAN, 2013) A escolha da bilbioteca vem pela sua facilidade de uso e pela disponibilidade multiplataforma. A figura 4.3 apresenta uma captura de tela da interface gráfica desenvolvida.

REFERÊNCIAS

BOETTIGER, C. An introduction to docker for reproducible research. **ACM SIGOPS Operating Systems Review**, ACM New York, NY, USA, v. 49, n. 1, p. 71–79, 2015.

BONFIELD, J. K. et al. Htslib: C library for reading/writing high-throughput sequencing data. **Gigascience**, Oxford University Press, v. 10, n. 2, p. giab007, 2021.

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.

FARAH, A. **An Ubuntu image with all packages necessary for running SAMtools and BCFtools and goodies**. 2021. <https://hub.docker.com/repository/docker/afarah1/ubuntu-samtools>.

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

JIANG, X.; ASSIS, R. Population-specific sequence and expression differentiation in europeans. **bioRxiv**, Cold Spring Harbor Laboratory, p. 749499, 2019.

KORNELIUSSEN, T. **RealSFS - angsd**. [S.l.], 2016.

KORNELIUSSEN, T. S.; ALBRECHTSEN, A.; NIELSEN, R. Angsd: analysis of next generation sequencing data. **BMC bioinformatics**, BioMed Central, v. 15, n. 1, p. 1–13, 2014.

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

LOGANATHAN, V. **PySide GUI application development**. [S.l.]: Packt Publishing, 2013.

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.

OLIPHANT, T. E. Python for scientific computing. **Computing in science & engineering**, IEEE, v. 9, n. 3, p. 10–20, 2007.

PIROOZNIA, M. et al. Validation and assessment of variant calling pipelines for next-generation sequencing. **Human genomics**, Springer, v. 8, n. 1, p. 1–10, 2014.

PUGLISI, A. T. **guietta 0.6.1 documentation**. [S.l.], 2020.

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.

TANGE, O. et al. Gnu parallel-the command-line power tool. **The USENIX Magazine**, usenix. org, v. 36, n. 1, p. 42–47, 2011.

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.

WANG, Q.; ZENG, J.; SONG, W. A new electromagnetism-like algorithm with chaos optimization. In: **2010 International Conference on Computational Aspects of Social Networks**. [S.l.: s.n.], 2010. p. 535–538.

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.

YI, X. et al. Sequencing of 50 human exomes reveals adaptation to high altitude. **science**, American Association for the Advancement of Science, v. 329, n. 5987, p. 75–78, 2010.