UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

EVALDO BEZERRA DA COSTA

AVALIAÇÃO DE DESEMPENHO DE MONTADORES PARA SEQUENCIAMENTO DE DNA

Rio de Janeiro
2014

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

EVALDO BEZERRA DA COSTA

AVALIAÇÃO DE DESEMPENHO DE MONTADORES PARA SEQUENCIAMENTO DE DNA

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Orientador: Marcello Goulart Teixeira
Co-orientador: Gabriel P. Silva

Rio de Janeiro
2014

C837 Costa, Evaldo Bezerra da

Avaliação de desempenho de montadores para sequenciamento de DNA / Evaldo Bezerra da Costa. – 2014.

66 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2014.

Orientador: Marcello Goulart Teixeira; Co-orientador: Gabriel P. Silva.

1. DNA. 2. Sequenciamento. 3. Montadores. 4. Allpaths-LG. 5. SOAPdenovo2. 6. Velvet. 7. Multithreads. – Teses. I. Teixeira, Marcello Goulart (Orient.). II. Silva, Gabriel P. (Co-orient.). III. Universidade Federal do Rio de Janeiro. IV. Título

CDD:

EVALDO BEZERRA DA COSTA

Avaliação de desempenho de montadores para sequenciamento de DNA

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Aprovado em: Rio de Janeiro, _____ de _____ de _____.

_____

Prof. Dr. Marcello Goulart Teixeira (Orientador)

_____

Prof. Dr. Gabriel P. Silva (Co-orientador)

_____

Prof. Dr. Carlos Eduardo Guerra Schrago - UFRJ

_____

Profa. Dra. Lúcia M. A. Drummond - UFF

_____

Prof. Dr. Mitre Costa Dourado - PPGI-UFRJ

Rio de Janeiro
2014

"Feliz a nação que tem o Senhor por seu Deus, e o povo que ele escolheu para sua herança."
Salmos 32:12

AGRADECIMENTOS

Ao Senhor meu Deus, que sempre está ao meu lado e me fortalece durante todos os momentos da minha vida.

A minha doce Rosalice, mulher que escolhi para viver ao meu lado por toda a vida. Pelas horas que não pude estar ao seu lado, dedicando-me aos meus estudos. Ao meu anjo Maria Eduarda, amor da minha vida.

Ao professor Marcello Goulart Teixeira, por acreditar em meu potencial e pela oportunidade que me foi dada.

Meu agradecimento especial ao professor Gabriel P. Silva. Mais que uma relação de aluno e professor, tenho grande satisfação em tê-lo como amigo.

Ao professor Carlos Eduardo Guerra Schrago, do Departamento de Genética da Universidade Federal do Rio de Janeiro, por prover os dados do genoma do primata Muriqui utilizado neste estudo.

À Microway Inc., por disponibilizar toda a infraestrutura necessária para a execução dos testes apresentados neste estudo.

A todos os professores do PPGI, os quais tive a hora de conhecer e dos quais tive o privilégio de ter sido aluno.

Aos colegas que conheci durante o curso e que me ajudaram durante todo este desafio.

RESUMO

Costa, Evaldo Bezerra da. **Avaliação de desempenho de montadores para sequenciamento de DNA**. 2014. 66 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2014.

Com o crescimento constante da quantidade de dados gerados a partir do sequenciamento de grandes genomas de DNA, um dos grandes desafios atualmente relacionados aos montadores de sequenciamento são o tempo e a quantidade de recursos computacionais utilizados para a execução desse processamento. Atualmente, são utilizados vários montadores, que buscam reduzir esses parâmetros com o uso da programação paralela baseada em multitarefa para o processamento dessas informações.

Nesta dissertação, serão apresentados os estudos de avaliação de desempenho dos montadores Allpaths-LG, SOAPdenovo2 e o Velvet. Dois tipos de dados serão utilizados para a execução dos testes: o genoma de novo , gerado a partir do sequenciamento do primata Muriqui, e o genoma do cromossomo humano 14 como referência. Os testes de desempenho dos montadores e a avaliação dos algoritmos serão executados em um servidor multitarefa, e, a partir dos resultados obtidos, definido qual dos montadores obteve o melhor desempenho.

Palavras-chave: DNA, sequenciamento, montadores, Allpaths-LG, SOAPdenovo2, Velvet, multithreads.

ABSTRACT

Performance evaluation of assemblers for DNA sequencing

Costa, Evaldo Bezerra da. **Avaliação de desempenho de montadores para sequenciamento de DNA**. 2014. 66 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2014.

With the constant growth of amount of data generated from the sequencing of large DNA genomes, one of the major challenges currently related to sequencing assemblers are the amount of time and computational resources used to perform this processing. Currently several assemblers seeking to reduce these parameters using parallel programming based multitasking for processing such information is used.

This dissertation studies evaluating the performance of assemblers Allpaths-LG, SOAPdenovo2 and Velvet will be presented. Two types of data are used for the tests. The genome de novo generated from the sequencing of the primate Muriqui and genome human chromosome 14 as a reference. Performance tests of assemblers and evaluation algorithms are executed in a multitasking server and from the results obtained, we can dene what the assemblers had the best performance.

Keywords: DNA, sequencing, assemblers, Allpaths-LG, SOAPdenovo2,Velvet, multithreads.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Quando teve início o processamento de sequenciamento de DNA, a quantidade de dados obtidos tinha um crescimento lento em função das tecnologias utilizadas para o sequenciamento. Com a descoberta e o aperfeiçoamento de novas técnicas de sequenciamento, a quantidade de dados cresceu rapidamente e passou a exigir servidores com maior poder de processamento e armazenamentos de dados.

Após o processo de sequenciamento do DNA, é aplicado outro processo chamado de montagem, que é a reconstrução do genoma a partir dos dados gerados . Existem vários programas desenvolvidos para a montagem dessas sequências, os quais também necessitam de grande poder de processamento e armazenamento.

Dependendo do tamanho da sequência a ser montada, esse processo pode durar dias para ser executado, aumentando-se o risco de ocorrerem falhas durante a execução e as informações processadas serem perdidas. Com os avanços das arquiteturas computacionais, que passaram a ter maior poder de processamento, memória e armazenamento de dados, os programas de montagem procuram utilizar de forma mais eficiente esses recursos.

1.1 Motivação

Dois grandes desafios atualmente relacionados aos montadores de sequenciamento de DNA são o tempo e a quantidade de recursos computacionais utilizados para a execução do processamento. Atualmente, existem vários montadores que buscam reduzir esses parâmetros utilizando a programação paralela para o proces-

samento dessas informações.

Cada vez mais servidores com alto poder computacional são utilizados para o processamento de grandes quantidades de dados. Essas informações têm o processamento cada vez mais eficiente em relação ao tempo de execução.

1.2 Proposta

Este trabalho tem como objetivo realizar um estudo de análise de desempenho utilizando três montadores de sequenciamento de DNA. Fazem parte do estudo os montadores com maior utilização em Bioinformática. Os métodos utilizados para avaliar os montadores são baseados no desempenho e utilização dos recursos computacionais disponíveis e não na qualidade obtida das montagens. Em relação à qualidade obtida das montagens, existem outros estudos que abordam somente esse assunto.

Para a execução dos testes, foram utilizados o genoma do cromossomo humano 14 e o genoma de novo do primata Muriqui. Como método utilizado para a execução dos testes, cada genoma foi dividido em grupos e executados em cada montador, para definir qual obtém o melhor desempenho.

1.3 Organização da dissertação

Esta dissertação está dividida em capítulos. No capítulo 2, são apresentados alguns conceitos de Biologia e Bioinformática. No capítulo 3, é feita uma descrição dos processos de sequenciamento e montagem. No capítulo 4, é feita a descrição dos montadores utilizados no estudo. O capítulo 5 descreve os dados e os procedimentos

utilizados nos testes. O capítulo 6 descreve os resultados obtidos, e finalmente, o capítulo 7, a conclusão do estudo.

2 INTRODUÇÃO À BIOLOGIA E À BIOINFORMÁTICA

Este capítulo tem por objetivo definir alguns dos conceitos básicos de Biologia e Bioinformática necessários para o estudo proposto.

2.1 DNA

Os primeiros relatos da descoberta do DNA se deram em 1869 por Johann Friedrich Miescher, nascido em Basel, na Suíça. Formou-se em medicina, mas, devido a um problema de audição que teve na infância, tinha dificuldades no exercício da medicina. Por causa desse problema, em 1868, Miescher dedicou-se à pesquisa.

Miescher obteve as células para seus experimentos a partir de curativos cirúrgicos, que eram coletados de uma clínica nas proximidades de seu laboratório. O material encontrado nos curativos foi utilizado como base para os seus estudos. Ao analisar o núcleo de células oriundas dos glóbulos brancos (DAHM (2005)), Miescher notou um material ácido do qual ainda não havia nenhuma referência. O material identificado possuía uma grande quantidade de fosfato e nitrogênio. Como não existia nenhuma referência ao material identificado, ele recebeu o nome de nucleína. A Figura 2.1 mostra o laboratório onde Miescher realizou os seus estudos e fez a descoberta do DNA.

Em 1889, o alemão Richard Altmann, através de alguns experimentos, constatou que o material encontrado por Miescher no núcleo das células, que recebera o nome de nucleína, era um ácido. Após essa confirmação, o material identificado ini-

Figura 2.1: O laboratório usado por Miescher para a descoberta do DNA (DAHM (2005)).

cialmente passou a se chamar ácido nucleico. Na Figura 2.2 podemos ver o processo de identificação do DNA no núcleo da célula.

As células são divididas em dois tipos: as procariotas e eucariotas.

- Células procariotas: são células que não possuem núcleo em seu interior. Essas células são desprovidas de mitocôndrias, plastídios, complexo de Golgi, retículo endoplasmático e, sobretudo, cariomembrana, o que faz com que o DNA fique disperso no citoplasma.

- Células eucariotas: possuem membrana nuclear individualizada e vários tipos de organelas. A maioria dos animais e plantas a que estamos habituados são dotados desse tipo de células.

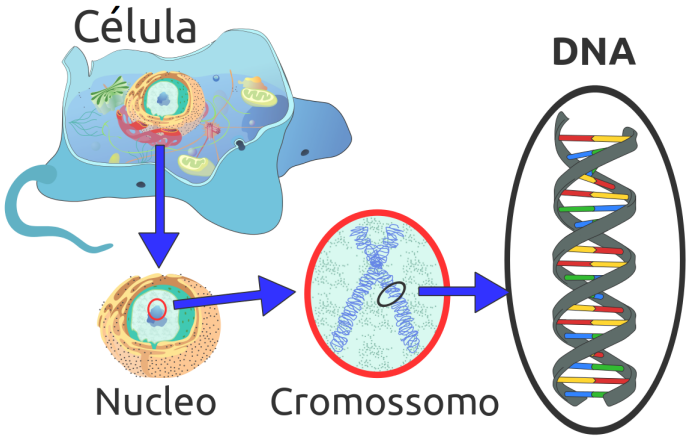


Figura 2.2: Identificação do material de DNA na célula.

2.2 Estrutura do DNA

A estrutura do DNA é composta por três partes: fosfato, pentose (açúcar), chamada de desoxirribose, e uma base nitrogenada (GRIFFITHS et al. (2008)). A base nitrogenada do DNA é formada por quatro tipos de nucleotídeos: adenina (A), citosina (C), guanina (G) e timina (T). Os nucleotídeos são referenciados pelas letras iniciais de seus nomes.

Esses nucleotídeos são separados em dois grupos conforme suas características. Purina, que contém os nucleotídeos adenina e guanina; pirimidina, que contém os nucleotídeos citosina e timina. Nas Figuras 2.3 e 2.4 é vista a estrutura dos nucleotídeos.

Erwin Chargaff realizou experimentos com o DNA de vários organismos. Com base nesses estudos, estabeleceu regras sobre os tipos de nucleotídeos encontrados no DNA. A primeira regra de Chargaff demonstra que na estrutura de DNA o número de nucleotídeos de guanina é igual ao número de nucleotídeos de citosina, e

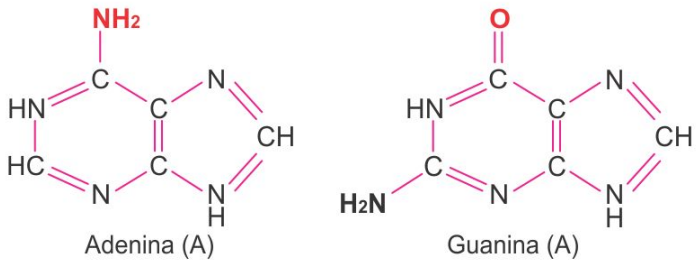


Figura 2.3: Estrutura dos nucleotídeos do grupo purina.

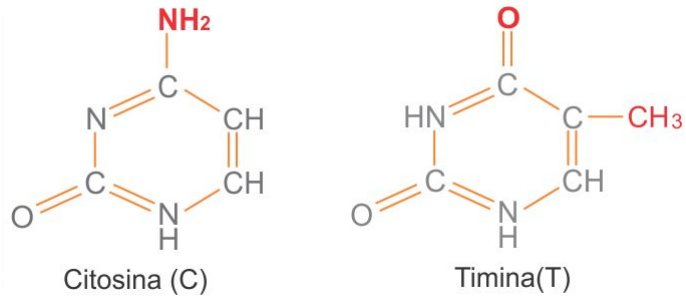


Figura 2.4: Estrutura dos nucleotídeos do grupo pirimidina.

o número de nucleotídeos de adenina é igual ao número de nucleotídeos de timina. A segunda regra de Chargaff é que a composição do DNA varia de acordo com a espécie, principalmente na quantidade de nucleotídeos A, C, G e T. Os estudos desenvolvidos por Chargaff ajudaram Watson e Crick na descoberta da dupla hélice do DNA.

As bases nitrogenadas do DNA seguem um padrão de pareamento de ligação entre os nucleotídeos. A ordem de pareamento entre os nucleotídeos são guanina e citosina ou vice-versa, adenina e timina ou vice-versa.

A estrutura da molécula de DNA foi descoberta em 1953 pelo norte-americano James Watson e pelo britânico Francis Crick (WATSON ; CRICK (1953)). No

modelo proposto por Watson e Crick, a molécula de DNA é constituída por duas cadeias dispostas em hélice ao redor de um eixo imaginário. As duas cadeias se mantêm unidas por pontes de hidrogênio, que se estabelecem entre pares de bases específicas, conforme Figura 2.5.

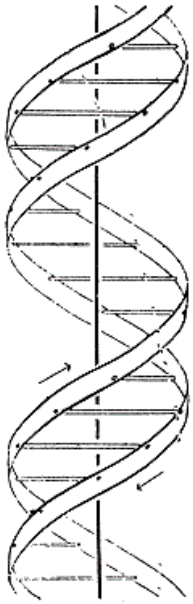


Figura 2.5: Modelo de dupla hélice proposta (WATSON ; CRICK (1953)).

2.3 PGH - Projeto Genoma Humano

Uma das mais importantes descobertas nos últimos anos na área de genética foi devido ao PGH (Projeto Genoma Humano) (VENTER et al. (2001)). O projeto genoma humano foi um esforço concentrado de vários países com o objetivo de mapear todos os genes humanos, e, através desse mapeamento, identificar todos os nucleotídeos do conjunto de 23 pares de cromossomos humanos. Antes de se iniciarem os estudos para o sequenciamento do genoma humano, alguns outros projetos ajudaram a criar a base de estudos para o projeto, tais como:

- o sequenciamento dos vírus bacterianos;
- o programa para criar um mapa genético humano para tornar possível localizar genes de doenças de função desconhecida com base apenas em seus padrões de herança;
- os programas para criar mapas físicos de clones.

O Projeto Genoma Humano foi discutido e proposto pela primeira vez em 1985. Em 1990, o PGH foi oficialmente iniciado nos Estados Unidos por James D. Watson, que iniciou a pesquisa em conjunto com cientistas de outros centros de pesquisas espalhados em vários países (LANDER et al. (2001)). Esse mapeamento tem como um dos principais objetivos a identificação de possíveis doenças, suas causas e como preveni-las. Com essas informações também será possível aprimorar ou criar novos tratamentos e desenvolver medicamentos com maior eficiência.

Em 1999, foram anunciados os primeiros resultados das pesquisas do PGH (IHGSC (2004)). Em 2003, foi feito o anúncio do término do projeto de genoma humano, que foi concluído com sucesso, tendo totalizado 100% do mapeamento. A partir daí, vários países começaram a desenvolver pesquisas baseadas nos resultados obtidos.

2.4 Bioinformática

O uso de recursos computacionais em Biologia começou a ser utilizado nos anos 70, quando já se era possível obter sequências de genomas (HOGEWEG (2011)). A partir desse período, o uso de recursos computacionais passou a ser de muita importância no estudo da Biologia.

Com a grande quantidade de dados gerados de genomas, essas informações precisaram de programas que pudessem manipular esses dados. A Bioinformática é uma ciência que une os dados e as informações fornecidos pela Biologia e os programas desenvolvidos na computação, para que juntas consigam interpretar e processar esses dados.

Atualmente, a Bioinformática é uma das áreas da ciência que mais se desenvolve, porque as quantidades de dados estão cada vez maiores e precisam de programas computacionais com maior poder de processamento.

3 SEQUENCIAMENTO E MONTAGEM

Este capítulo tem por objetivo definir o conceito de sequenciamento e suas técnicas aplicadas nesse processo. Também será definido o processo de montagem e como ele ocorre utilizando genoma por referência e genoma de novo.

3.1 Sequenciamento

Todas as informações que constituem um ser vivo estão presentes no DNA. O sequenciamento de DNA é um processo que determina a ordem dos nucleotídeos em uma amostra. Nesse processo é feita a identificação de todas as informações contidas no DNA e as ordens dos nucleotídeos.

Existem alguns métodos utilizados para o sequenciamento do DNA. Neste trabalho serão abordados os métodos de sequenciamento Sanger, Pirosequenciamento e o método de Short Reads Sequencing.

3.1.1 Sequenciamento Sanger

O método tradicional de sequenciamento de DNA foi proposto pelo inglês Frederick Sanger em 1975 (SANGER ; COULSON (1975)). Consiste na adição de nucleotídeos modificados, chamados desoxirribose nucleotídeos, que impedem o crescimento de um fragmento de DNA em replicação pela DNA polimerase (enzimas, responsáveis pela reação química que dá origem aos polímeros das novas fitas de DNA) após sua adição.

A síntese de DNA é iniciada por um RNA iniciador (pequeno fragmento de cadeia simples de ácido nucleico), sendo que depois a DNA polimerase continua ao longo da sequência molde, incorporando os nucleotídeos e sintetizando a fita nova. Nessa técnica existem dois tipos de nucleotídeos: os normais, que recebem o nome de deoxi: dATP, dGTP, dCTP e dTTP; e os dideoxinucleotídeos, que não têm o grupo hidroxila no carbono: ddATP, ddGTP, ddCTP e ddTTP, e são marcados com um material fluorescente

A polimerização do DNA prossegue com a inclusão dos nucleotídeos "normais" até que a DNA polimerase insere um dideoxinucleotídeo, o que interrompe a polimerização. Ao final desse processo, observam-se fragmentos de tamanhos diferentes. Essa reação é então colocada em um gel poroso, no qual, após a eletroforese, observa-se a migração dos fragmentos. Os fragmentos menores migram mais rapidamente. A Figura 3.1 mostra o processo de sequenciamento usando o método de Sanger.

No sequenciamento manual, os dideoxinucleotídeos são marcados com radiação e não com compostos fluorescentes, por isso uma radiografia é necessária para a leitura da sequência do DNA. No sequenciamento automático, o sequenciador identifica os fragmentos pela incidência de laser sobre os dideoxinucleotídeos fluorescentes. Ao final, o sequenciador exibe um gráfico no qual cada nucleotídeo tem uma cor.

3.1.2 Pirosequenciamento

Pirosequenciamento é uma técnica de sequenciamento de DNA com base na detecção de pirofosfato libertado (PPi) durante a síntese de DNA (RONAGHI (2001)). Em uma cascata de reações enzimáticas, uma luz visível é gerada, proporcionalmente ao número de nucleotídeos incorporados, como visto na Figura 3.2. A

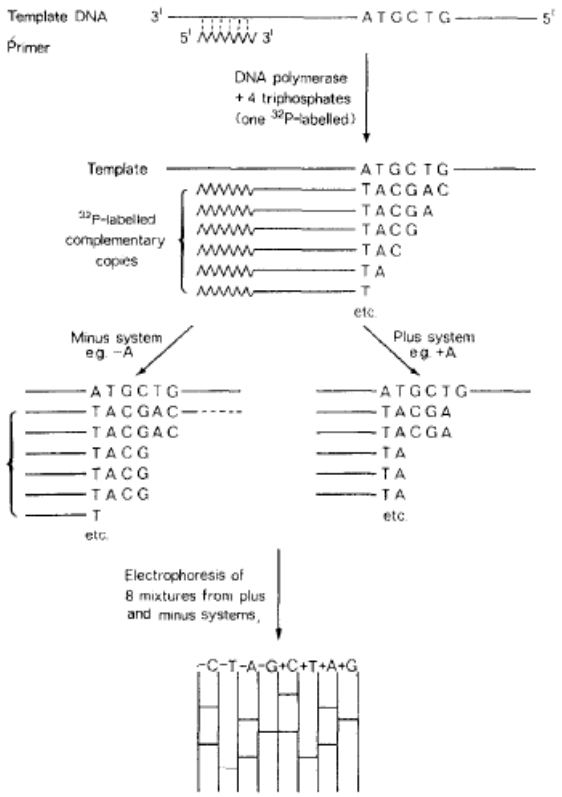


Figura 3.1: O princípio do método de Sanger (SANGER ; COULSON (1975)).

cascata começa com uma reação de polimerização de ácido nucleico, em que PPi inorgânico é liberado como resultado de nucleotídeos incorporados pela polimerase.

O PPi liberado é posteriormente convertido em ATP pela sulfurilase de ATP, que proporciona a energia para a luciferase oxidar luciferina e gerar luz. Uma vez que o nucleotídeo acrescentado seja conhecido, a sequência do molde pode ser determinada.

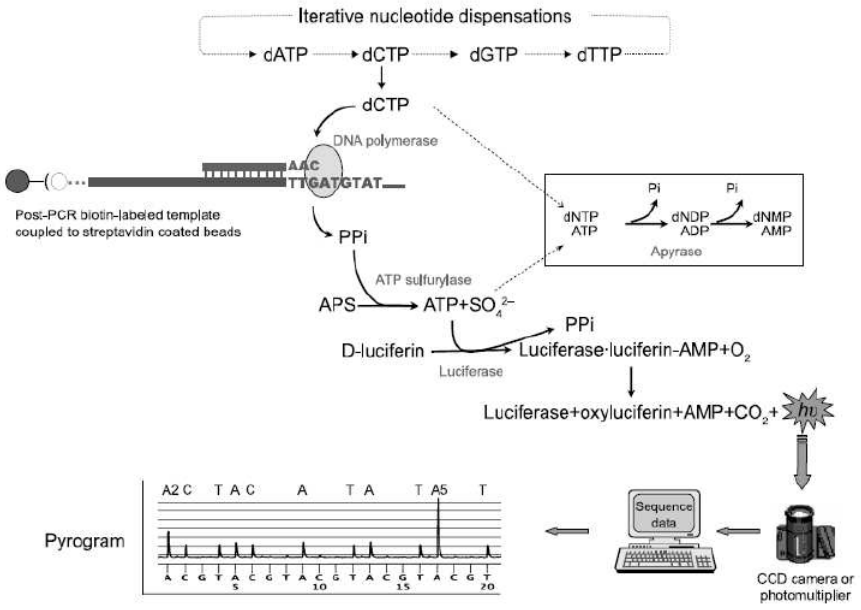


Figura 3.2: O princípio geral do método de pirosequenciamento (FAKRUDDIN et al. (2012)).

3.1.3 Short Reads Sequencing

O método é semelhante à sequência de Sanger, mas usa dNTP modificados contendo um terminador que bloqueia a polimerização adicional, portanto, apenas uma única base pode ser adicionada por uma enzima polimerase de DNA para cada cópia de cadeia crescente. A reação do sequenciamento é realizada simultaneamente em um número muito grande de moléculas diferentes espalhadas sobre uma superfície sólida.

O terminador também contém um marcador fluorescente, que pode ser detectado por uma câmera. Apenas uma única cor fluorescente é utilizada, de modo que cada uma das quatro bases deve ser adicionada em separado de um ciclo de síntese de DNA e de imagem. Em seguida à adição dos quatro dNTPs para os moldes, as imagens são registradas, e os terminadores são removidos. Essa reação química é

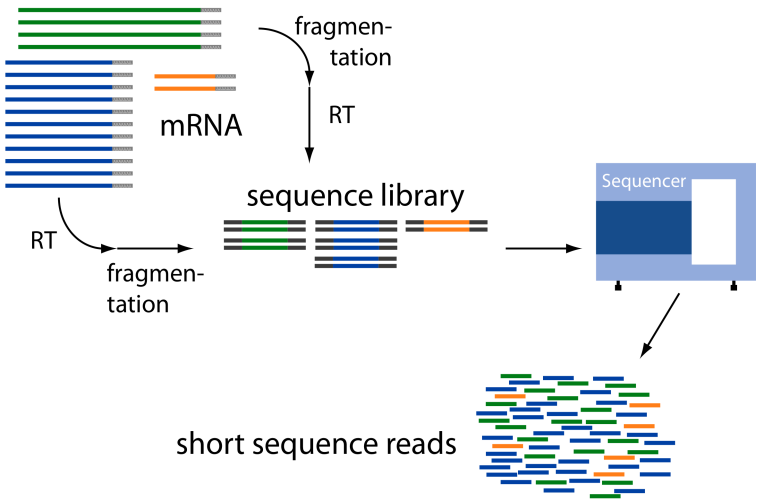


Figura 3.3: O princípio geral do método de Short Reads Sequencing (FAKRUDDIN et al. (2012)).

chamada de "terminadores reversíveis". Finalmente, mais quatro ciclos de adições de dNTP são iniciados. Como bases simples são adicionadas a todos os modelos de um modo uniforme, o processo de sequenciamento produz um conjunto de sequência de DNA de comprimento uniforme.

3.2 Tipos de arquivos

Durante o processo de sequenciamento do DNA, são gerados arquivos que contêm as informações dos nucleotídeos. Esses arquivos gerados possuem formatos diferentes e são utilizados de acordo com o programa de montagem. Os principais formatos de arquivos são fasta, fastq e sff.

3.2.1 Arquivo fasta

O formato fasta é baseado em texto para representar uma sequência de nucleotídeos, utilizando-se uma sequência de letras. Os arquivos no formato fasta são mais fáceis de manipular e de analisar as sequências contidas neles usando-se ferramentas de processamento de texto e linguagens de script. As linguagens de programação mais utilizadas para esse processo são Python, Ruby e Perl.

3.2.2 Arquivo fastq

Já o fastq é um formato baseado em texto para armazenar uma sequência de nucleotídeos e seus índices de qualidade correspondentes. Esse formato foi desenvolvido pelo Wellcome Trust Sanger Institute, com o objetivo de melhorar a qualidade do arquivo Fasta. Atualmente, tornou-se o padrão para o armazenamento de informações de sequenciamento de DNA. A seguir, um exemplo do arquivo fastq:

```
@HWI-ST1054:100:d0abmacxx:7:1208:13194:181739 1:N:0:CGATGT
AAAGTGCCATAGAGTTGTACAAGCAGGAGAGATACATGTGGATANNNNC
+
B?BDBDEFHGHHHJGIGIJJIIJJIIGIEGHHJJJGGIIE*00?FHGJJHDFEFFDE@
```

O arquivo fastq possuí em seu cabeçalho informações referentes à geração do arquivo, as quais são especificadas no início do arquivo, em suas quatro linhas iniciais, conforme descrição seguinte.

Linha 1 = "@"+ identificador de sequência
Linha 2 = sequência

Linha 3 = "+"e é opcionalmente seguido pelo mesmo identificador de sequência
Linha 4 = Representa os valores atribuídos à qualidade da sequência descrita na linha 2

3.2.3 Arquivo sff

Outro tipo é o sff, que é um formato de arquivo binário usado para codificar os resultados do pirosequenciamento da 454 Life Sciences plataforma de sequenciamento de alto rendimento. Arquivos sff podem ser manipulados utilizando-se o programa DNA Baser SFF Workbench (Figura 3.3), ou convertidos para o formato fastq.

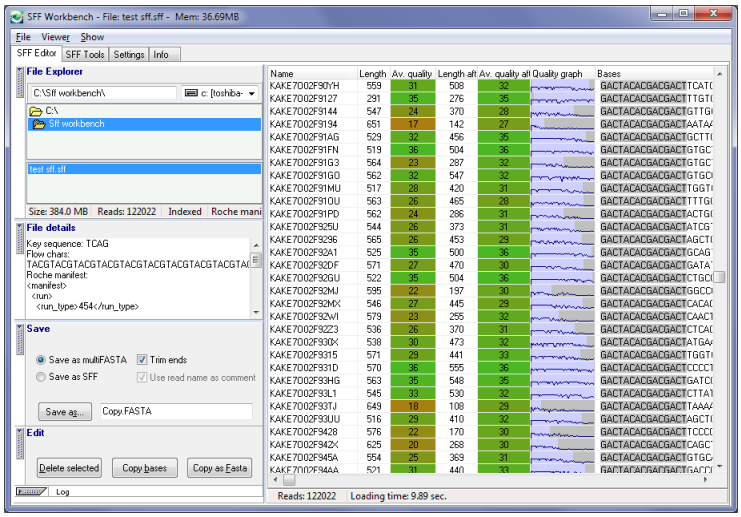


Figura 3.4: Tela inicial do programa DNA Baser SFF Workbench.

3.2.4 Convertendo arquivos

Alguns arquivos são sequenciados em formatos que não podem ser lidos por alguns montadores. Neste caso, é necessário converter esses arquivos para o formato

que possa ser lido pelo montador.

Os arquivos utilizados neste estudo estão no formato fastq. Como alguns dos montadores utilizados não trabalham com esse formato, foi necessário o uso de um programa para fazer a conversão dos arquivos: o FastqToCA.

3.3 Montagem

O processo de montagem de um genoma de DNA é feito utilizando-se fragmentos de DNA que foram sequenciados através de uma das técnicas de sequenciamento descritas. O objetivo da montagem é reconstruir o genoma original a partir de sua sequência.

Existem dois tipos de processo de montagem: por referência e a de novo. No processo de montagem por referência, a sequência é comparada a uma já existente, com isso a utilização de recursos computacionais é menor. Quando não se tem um genoma de referência para ser utilizado durante o processo de montagem, é necessário realizar a montagem de novo. No processo de montagem de novo, como não se utiliza um genoma como referência, a utilização de recurso computacional é muito alto. Para se obter bons resultados com o sequenciamento de novo, é necessário o uso de grande quantidade de processamento e memória. A Figura 3.5 mostra como ocorrem os dois tipos de processo de montagem, por referência e de novo.

Para o melhor entendimento do processo de montagem de um genoma de DNA, é preciso entender alguns termos usados no processo de montagem. Durante o processo de sequenciamento, o genoma é dividido em fragmentos que contêm os nucleotídeos representados através das letras A, C, G e T; seus fragmentos são

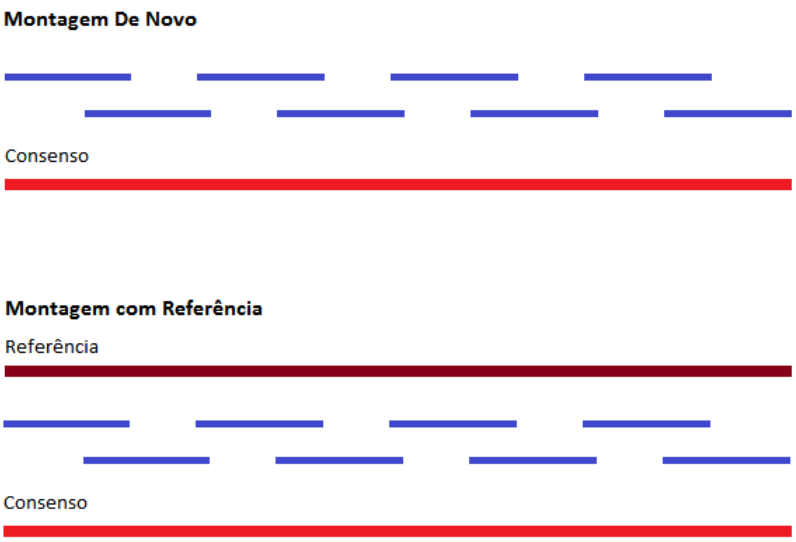


Figura 3.5: Montagem de genoma por referência e genoma de novo.

chamados de reads. Um read tem em sua forma os quatro nucleotídeos que podem estar em qualquer ordem e até mesmo possuir nucleotídeos repetidos. Durante o processo de montagem, os fragmentos são unidos, a essa união damos o nome de contig.

Por sua vez, os Gaps ocorrem quando os fragmentos sequenciados não são identificados por nenhum nucleotídeo durante o processo de montagem, ou seja, quando ocorre uma descontinuidade de uma das duas cadeias, devido à perda de um ou mais nucleotídeos. Já os scaffolds são compostos de contigs e gaps.

E, por fim, a cobertura. A cobertura é a relação entre a quantidade total de fragmentos sequenciados pelo tamanho do genoma utilizado. Isto significa dizer em média a quantidade de vezes que cada base de um genoma foi sequenciada.

A Figura 3.6 mostra como é executado o processo de montagem de um genoma, desde a leitura dos reads até a construção final do genoma. A qualidade da

montagem do genoma depende da qualidade dos dados sequenciados; se os dados tiveram uma boa qualidade durante o processo de sequenciamento, a montagem também pode gerar uma boa qualidade.

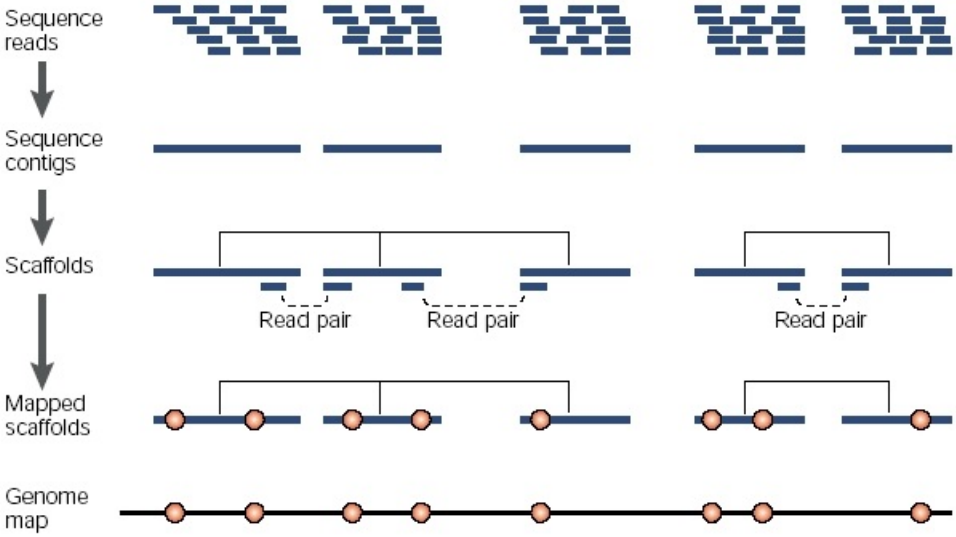


Figura 3.6: Processo de montagem de genoma de DNA.

3.4 Modelo de Lander-Waterman

Em 1988, Eric Lander e Michael Waterman publicaram um artigo que verificava um dos problemas durante o processo de montagem de DNA, o problema de cobertura (LANDER ; WATERMAN (1988)). Eles desenvolveram um conjunto de dados que são utilizados como teoria padrão para o sequenciamento de genomas de grande escala. Esse modelo também foi utilizado durante o Projeto Genoma Humano e continua a desempenhar um papel importante no sequenciamento de DNA.

No modelo proposto por Lander e Waterman, alguns parâmetros são necessários para a utilização dos cálculos. Esses parâmetros variam de acordo com as características de cada genoma. Segue-se a descrição dos parâmetros utilizados no

modelo de Lander-Waterman.

G = tamanho do genoma
L = tamanho das leituras
N = número de leituras
T = mínimo de sobrepsição entre leituras
c = cobertura (NL/G)
σ = 1 - T/L

Um dos resultados mais utilizados desse modelo é o número esperado de contigs, dado o número de fragmentos sequenciados. Para esse cálculo é utilizada a Equação 3.1.

$$E(\#contigs) = Ne^{-c\sigma} \tag{3.1}$$

4 MONTADORES

A maioria dos novos programas de montagem pode ser classificada como NGS (Next-Generation Sequencing). Com esse método, é possível sequenciar DNA em grande escala e reduzir os custos associados com leituras de curto comprimento (MILLER ; KOREN ; SUTTON (2010)).

Para o processo de montagem de genoma de novo, são utilizados três métodos: o método guloso, consenso layout de sobreposição (OLC) e de método de Bruijn. Neste estudo será abordado somente o método de Bruijn, em função dos montadores que serão utilizados. A Figura 4.1 mostra como os montadores são classificados de acordo com o método utilizado durante o processo de montagem. Como pode ser observado, os montadores utilizados neste estudo estão classificados no grupo que usa o método de Bruijn.

4.1 Grafo De Bruijn

Uma das principais características dos montadores de curta leitura que os diferenciam de montadores tradicionais é o uso do grafo de Bruijn para o processo de comparação e montagem de leituras por não utilizar muito recurso computacional. O conceito de grafo de Bruijn veio de teoria dos grafos e é utilizado em várias áreas. Em Bioinformática, é amplamente utilizado pelos montadores de curta leitura. A abordagem Euleriana para o uso no processo de montagem foi descrito por Pevzner em 2001 (PEVZNER ; TANG ; WATERMAN (2001)).

Uma definição generalizada de grafo de Bruijn se baseia em um conjunto de

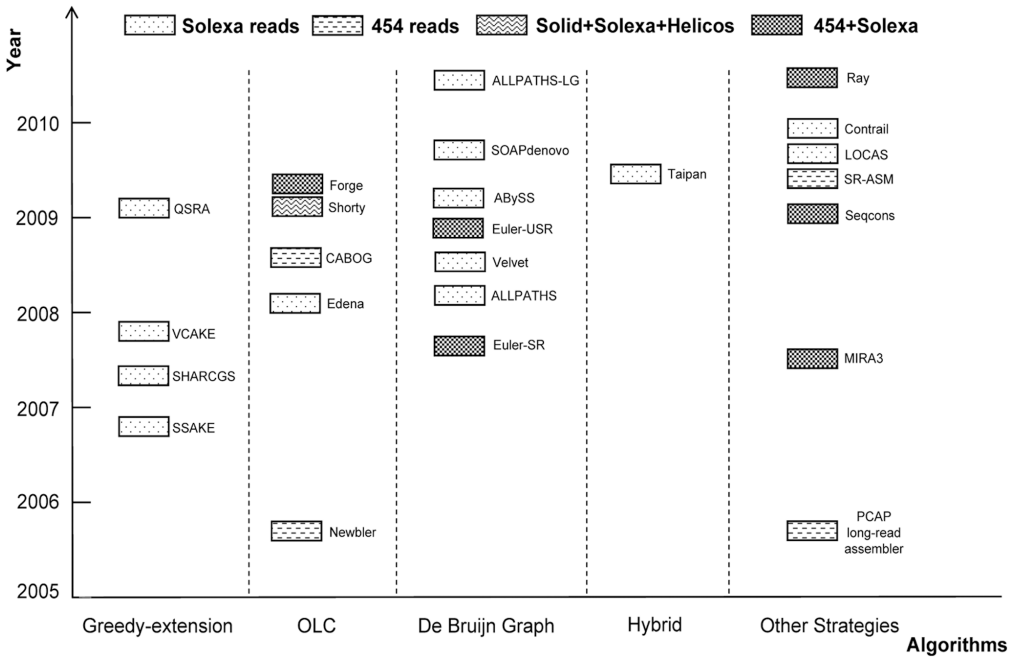


Figura 4.1: Visão geral dos montadores de novo de curtas leituras (ZHANG et al. (2011)).

símbolos de M e N dimensões. Para aplicações de Bioinformática, esses símbolos estão relacionados com os nucleotídeos. O conjunto de símbolos consiste em quatro caracteres ('A', 'C', 'T', 'G') e a dimensão é equivalente ao tamanho k-mer escolhido.

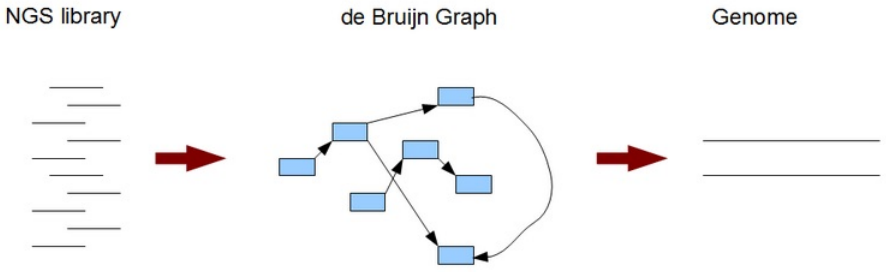


Figura 4.2: Representação esquemática do grafo De Bruijn.

A Figura 4.2 mostra como o processo é executado. Os algoritmos de montagem de genoma baseados no grafo de Bruijn trabalham em duas etapas. No primeiro

passo, os arquivos de curta leitura são divididos em tamanhos menores, que são comuns e se repetem frequentemente, com tamanho predefinido k. A essas pequenas partes chamamos de k-mers. Então, o grafo de Bruijn é construído a partir desses pequenos pedaços. No passo seguinte, o genoma é obtido a partir do grafo de Brujin.

Um grafo de Bruijn pode ser construído para qualquer sequência. O primeiro passo é definir o tamanho do k-mer. O próximo passo é dividir a sequência pelos k-mer definidos. Em seguida, um grafo orientado é construído ligando pares de k-mers com sobreposição entre os primeiros k-1 nucleotídeos e os últimos k-1 nucleotídeos. O sentido da seta vai do k-mer, cujo k-1 último nucleotídeo está sobreposto, para o k-mer, cujo primeiro k-1 nucleotídeo é sobreposto.

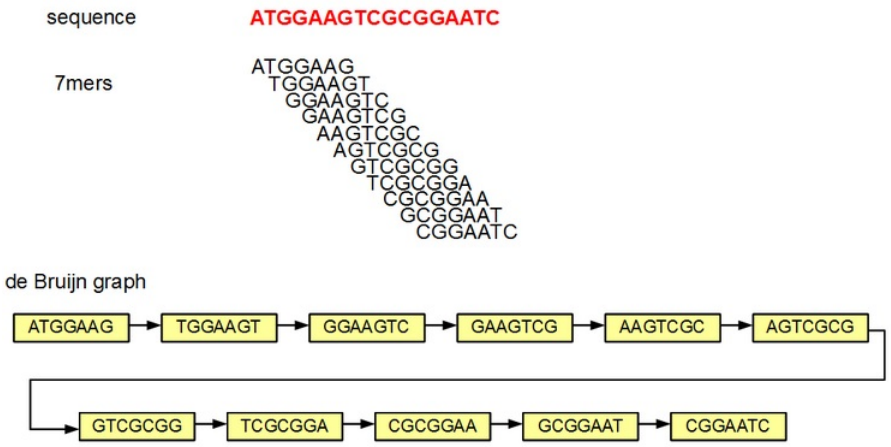


Figura 4.3: Representação da aplicação do grafo De Bruijn.

Na Figura 4.3 é construído o grafo de Bruijn dividindo a sequência ATGGAAGTCGCGGAATC em k-mers (k = 7), em seguida, através da construção de um grafo com os 7-mers como nós. As arestas são desenhadas entre pares de nós de tal maneira que os nós conectados têm sobreposições de 6 (= k-1) nucleotídeos. No exemplo, apenas as adjacentes 7-mers a partir da sequência original foram ligadas no gráfico.

4.2 Allpaths-LG

É um montador da nova geração, que possibilita a montagem do genoma com alta qualidade (BUTLER et al. (2008)). A principal diferença entre os demais montadores utilizados neste estudo é que o processo de execução não acontece de forma linear. Os gráficos gerados com o resultado final do Allpaths-LG mantêm as mesmas características dos outros montadores. Durante o seu processo de execução, o montador Allpaths-LG não corrige erros de leitura. O montador Allpaths-LG tem duas características principais:

- Todas as sequências de dados utilizados no processo de leitura são cobertas em conjunto.
- As sequências são separadas em regiões e, em seguida, são montadas separadamente umas das outras.

O montador Allpaths-LG é desenvolvido e mantido pelo BROAD Institute.

4.3 SOAPdenovo2

SOAPdenovo2 é usado para fazer a montagem de curta leitura e foi desenvolvido para trabalhar com Illumina GA. SOAPdenovo2 é uma nova versão que tem um algoritmo mais eficiente e uma melhor utilização da memória para a construção do grafo De Bruijn (LUO et al. (2012)). O SOAPdenovo2 é mantido e desenvolvido pelo BGI (Beijing Genomics Institute).

SOAPdenovo2 também permite reconstruir grandes genomas, além de os pro-

blemas de montagem com sequências repetidas ainda existirem. A nova versão do módulo GapCloser mudou e melhorou os gaps dos arquivos sequenciados. A versão anterior do GapCloser considerava em cada ciclo iterativo apenas as leituras que poderiam ser alinhadas em ciclo. Esse método poderia fazer uma seleção incorreta em locais incompatíveis com as informações insuficientes para a distinção, devido à alta semelhança entre as sequências repetitivas. O novo algoritmo do SOAPdenovo2 torna o montador muito mais eficiente em relação à versão anterior.

4.4 Velvet

O montador Velvet (ZERBINO ; BIRNEY (2008)) trabalha com grafo de Bruijn sem as informações de perda para criar o gráfico. O Velvet tem um algoritmo para corrigir os erros e remove repetições para criação do arquivo de sequência.

O Velvet usa sequências de curta leitura, e, quando um erro é encontrado, ele é removido para produzir contigs com melhor qualidade. A partir da versão 1.1, o Vetvet passou a trabalhar em ambiente multitarefa.

Em 2008, Simon Gladman e Torsten Seemann desenvolveram um programa chamado VelvetOptimiser, que ajuda o Velvet durante o processo de montagem, procurando definir os melhores valores para os principais parâmetros do Velvet, tais como k, -exp_cov e -cov_cutoff.

O processo de montagem utilizando o Velvet e VelvetOptimiser é executado em duas etapas: na primeira, o comando velveth converte os arquivos sequenciados para k-mers usando uma tabela hash. Na segunda etapa, o comando velvetg monta sobreposição k-mers em contigs através do grafo De Bruijn. O programa VelvetOptimiser é utilizado para automatizar os parâmetros dos comandos velveth e velvetg

e gerar um conjunto de dados com melhor resultado.

Na Tabela 4.1 é descrita a versão dos montadores utilizados neste estudo.

Tabela 4.1: Versão dos montadores.

Montador	Versão	Local para baixar
Allpaths-LG	46923	http://www.broadinstitute.org/software/allpaths-lg/blog/
SOAPdenovo2	2.04	http://soap.genomics.org.cn/
Velvet	1.2.10	https://www.ebi.ac.uk/ zerbino/velvet/

5 TESTES

Os programas para montagem das sequências de DNA foram desenvolvidos para se executar utilizando múltiplos processadores e memória compartilhada. Os testes foram realizados em um servidor com dois processadores Intel Xeon E5-2650 (2; 00 GHz, 8 núcleos cada, 20 MB de cache), com 64 GB de memória. Uma placa Intel Xeon Phi 7120P (1,238 GHz, 61 núcleos, 30,5 MB de cache L2, memória de 16 GB).

Todos os montadores de DNA foram compilados utilizando o conjunto de programas Intel C++ Compiler Suite XE 13.1 Update 2 para obter o melhor desempenho. Os arquivos usados nos testes foram armazenados em disco local de alta velocidade do tipo SSD (Solid State Drive). O sistema operacional utilizado foi o Linux CentOS Linux de 64 bits versão 6.3.

5.1 Intel Xeon Phi

Arquitetura do Intel MIC (Many Integrated Core) é voltada para a computação de alto desempenho (HPC), que utiliza grandes demandas de dados para processamentos paralelos em uma grande variedade de áreas, como a Física Computacional, Química, Biologia e a área financeira (INTEL (2013)). Hoje essas demandas de trabalho são executadas em grandes clusters de computação. O coprocessador é suportado por um ambiente de desenvolvimento que inclui diversos produtos, como compiladores, diversas bibliotecas, ferramentas de tuning e depuradores. A Tabela 5.1 mostra as características do Intel Xeon Phi 7120P.

Tabela 5.1: Especificações técnicas do Intel Xeon Phi 7120P.

Intel Xeon Phi 7120P	
Modelo do processador	7120P
Número de núcleos	61
Velocidade do relógio	1,238 GHz
Frequência turbo máxima	1,333 GHz
Cache L2	30,5 MB
Conjunto de instruções	64 bit
Tamanho máximo de memória	16 GB
Operações de ponto flutuante por segundo (pico teórico)	1,208 TeraFLOPS

Uma característica do Intel Xeon Phi é que cada core é capaz de executar quatro threads simultâneas, isto significa que, usando o modelo 7120P, temos um total de 244 threads rodando ao mesmo tempo. Na Figura 5.1 é visto o diagrama de blocos do Intel Xeon Phi. O coprocessador Intel Xeon Phi está ligado a um processador Intel Xeon, também conhecido como o "hospedeiro"(HEBENSTREIT (2013)). Uma vez que o coprocessador Intel Xeon Phi executa o sistema operacional Linux, o usuário pode acessar o coprocessador como sistema independente. Assim, qualquer usuário pode conectar-se ao coprocessador através de um shell seguro e executar diretamente trabalhos individuais ou submeter trabalhos em lotes para ele.

O coprocessador também suporta aplicações heterogêneas, em que uma parte do aplicativo é executada no servidor, enquanto uma parte é executada no coprocessador. Vários coprocessadores Intel Xeon Phi podem ser instalados em um único servidor. Dentro de um único sistema, os coprocessadores podem se comunicar uns com os outros através das interconexões PCIe sem qualquer intervenção do servidor. Da mesma forma, os coprocessadores também podem se comunicar através da rede.

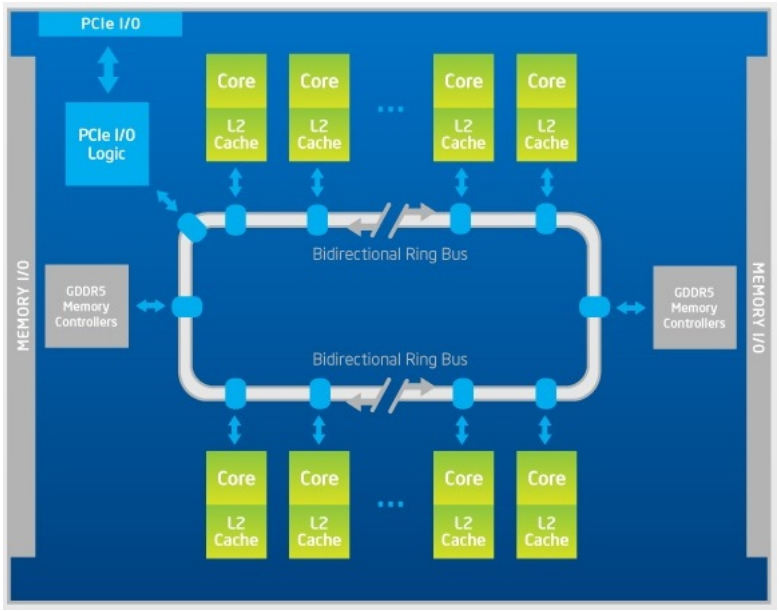


Figura 5.1: Diagrama de blocos do Intel Xeon Phi ((INTEL, 2013)).

5.2 Dados utilizados

Para a execução dos testes foram utilizados dois tipos de dados. Um genoma já conhecido como referência para os resultados e um genoma de novo. O genoma utilizado como referência foi o cromossomo humano 14, por ser amplamente usado em outros estudos.

Os dados do genoma de novo foram fornecidos pelo Instituto de Biologia da UFRJ, a partir do sequenciamento do primata Muriqui. Como se trata de um genoma de novo, os seus resultados estão em processo de estudo.

5.2.1 Cromossomo humano 14

Cromossomo 14 é um dos cinco acrocêntricos do genoma humano, isto é, quando a região mais condensada do cromossomo está mais próxima das extremidades do que do centro. Os outros cromossomos acrocêntricos humanos são os 13, 15, 21 e 22. A sequência completa do cromossomo humano 14 compreende 87.410.661 pares de bases, o que representa um total próximo de 100% (HEILIG et al. (2003)).

O cromossomo humano 14 possui um comprimento de leitura de 101 bp. Arquivos utilizados: frag_1.fastq (tamanho: 3,9 GB), frag_2.fastq (tamanho: 3,9 GB), longjump_1.fastq (tamanho: 249 MB), longjump_2.fastq (tamanho: 249 MB), shortjump_1.fastq (tamanho: 2,5 GB) e shortjump_2.fastq (tamanho: 2,5 GB). Todos os arquivos estão sem correção de erros. A Tabela 5.2 mostra os arquivos do cromossomo humano 14.

Tabela 5.2: Tamanho dos dados.

Conjunto de dados	
frag_1.fastq	3,9 GB
frag_2.fastq	3,9 GB
longjump_1.fastq	249 MB
longjump_2.fastq	249 MB
shortjump_1.fastq	2,5 GB
shortjump_2.fastq	2,5 GB

5.2.2 Genoma de novo Muriqui

O Muriqui (Brachyteles spp.), um dos primatas endêmicos encontrados na Mata Atlântica (CHAVES et al. (2011)), é classificado na subfamília Atelinae (Primates, Platyrrhini), juntamente com três outros gêneros. Arbórea é encontrada exclusivamente na Amazônia e América Central (Ateles, Lagothrix, e Oreonax). O

gênero Brachyteles inclui, provavelmente, o maior primata neotropical, que atinge mais de um metro de comprimento. Inicialmente considerado monotípico, duas espécies são agora reconhecidas: B. hypoxanthus, o Muriqui do norte, e B. arachnoides, o Muriqui do sul.

Durante o processo de sequenciamento, o genoma do primata Muriqui foi separado em quatro conjuntos de dados chamados de Muriqui8, Muriqui9, Muriqui10 e Muriqui12. O genoma do primata Muriqui possui 60 cromossomos e estão espalhados randomicamente entre os quatro conjuntos de dados. Cada conjunto de dados contém os 60 cromossomos. Quando cada conjunto de dados é montado separadamente, o resultado é uma baixa cobertura.

Cada conjunto de dados consiste em vários arquivos. O conjunto de dados Muriqui8 consiste de quatro arquivos com 1,7 GB de tamanho. O Muriqui9 consiste de oito arquivos com um tamanho total de 6,24 GB. O conjunto de dados do Muriqui10 é composto por vinte arquivos com 16,6 GB de tamanho, e, por fim, os dados do Muriqui12, que consistem de 14 arquivos com um tamanho total de 10,8 GB. As informações dos conjuntos de dados podem ser vistos na Tabela 5.3.

Tabela 5.3: Tamanho dos conjuntos de dados.

Conjunto de dados	
Muriqui8	1,7 GB
Muriqui9	6,24 GB
Muriqui10	16,6 GB
Muriqui12	10,8 GB

5.3 Allpaths-LG

Para a execução do montador, é necessária uma configuração mínima de recursos computacionais. A memória mínima exigida é de 16 GB, e o sistema operacional uma distribuição Linux de 64 bits. O Allpaths-LG utiliza em seu processo de execução dois arquivos chamados de in_groups.csv e in_libs.csv. Esses arquivos contêm as informações referentes aos parâmetros do conjunto de dados que serão utilizados durante o processo de montagem do genoma.

Na Tabela 5.4 são vistos os parâmetros de configuração do arquivo in_groups.csv.

Tabela 5.4: Parâmetros de configuração do arquivo in_groups.csv.

group_name	a UNIQUE nickname for this specific data set
library_name	the library to which the data set belongs
file_name	the absolute path to the data file. Wildcards '*' and '?' are accepted (but not in the extension) when specifying multiple files as in the case of two paired or multiple unpaired fastq or fasta files. Supported extensions are: '.bam', '.fasta', '.fa', '.fastq', '.fq', '.fastq.gz', and '.fq.gz', all caseinsensitive. For '.fasta' and '.fa' it is expected that corresponding '.quala' and '.qa' files exist, respectively

A Tabela 5.5 mostra as informações necessárias para a configuração do arquivo in_libs.csv.

5.4 SOAPdenovo2

Para executar o montador SOAdenovo2, a configuração mínima exigida é de 5 GB de memória física e uma distribuição padrão do Linux de 64 bits. Para execução

Tabela 5.5: Parâmetros de configuração do arquivo in_libs.csv.

library_name	matches the same field in in_groups.csv
project_name	a string naming the project
organism_name	the organism
type	fragment, jumping, EcoP15, etc. This field is only informative
paired	0: Unpaired reads; 1: paired reads
frag_size	average number of bases in the fragments (only defined for FRAGMENT libraries)
frag_stddev	estimated standard deviation of the fragments sizes (only defined for FRAGMENT libraries)
insert_size	average number of bases in the inserts (only defined for JUMPING libraries; if larger than 20 kb, the library is considered to be a LONG JUMPING library)
insert_stddev	estimated standard deviation of the inserts sizes (only defined for JUMPING libraries)
read_orientation	inward or outward. Outward oriented reads will be reversed
genomic_start	index of the FIRST genomic base in the reads. If non-zero, all the bases before genomic_start will be trimmed out
genomic_end	index of the LAST genomic base in the reads. If non-zero, all the bases after genomic_end will be trimmed out

do SOAPdenovo2, é necessário criar um arquivo de configuração, aqui chamado de soapdenovo.conf. Esse arquivo contém os parâmetros que serão utilizados durante o processo de montagem, como pode ser visto na Tabela 5.6. Outro parâmetro muito importante é k. Nos testes de montagem, foi utilizado o valor de 63 para o parâmetro k.

Tabela 5.6: Arquivo de configuração soapdenovo.conf.

```
mal read length
max_rd_len=100
[LIB]
# average insert size
avg_ins=200
# if sequence needs to be reversed
reverse_seq=0
# in which part(s) the reads are used
asm_flags=3
# use only first 100 bps of each read
rd_len_cutoff=100
# in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection
# (at least 3 for short insert size)
pair_num_cutoff=3
# minimum aligned length to contigs for a reliable read
location
# (at least 32 for short insert size)
map_len=32
# a pair of fastq file, read 1 file should always be followed
by read 2 file
q1=fastq1_read_1.fq
q2=fastq1_read_2.fq
# another pair of fastq file, read 1 file should always be
followed by read 2 file
q1=fastq2_read_1.fq
q2=fastq2_read_2.fq
```

5.5 Velvet

Para executar o montador Velvet, a configuração mínima exigida é de 12 GB de memória física, e o sistema operacional Linux de 64 bits. O Velvet consiste de dois programas que são usados para gerar os resultados: velveth e velvetg. Velveth

lê os arquivos de sequências que foram passados como entradas de dados e cria dois arquivos de saída chamados Roadmaps e Sequences, que serão utilizados pelo velvetg. O velvetg usa os arquivos de saída gerados pelo velveth e constrói o gráfico de Bruijn. Também faz simplificação e correção de erros sobre o gráfico. Como último passo, ele extrai os contigs da sequência. O valor usado para o parâmetro k foi de 31.

6 RESULTADOS

Este capítulo tem como objetivos apresentar e analisar os resultados encontrados após o processo de montagem dos genomas do cromossomo humano 14 e o genoma de novo do primata Muriqui nos montadores avaliados.

6.1 Métricas

Para os resultados apresentados neste estudo, foram executadas três séries de testes, variando a quantidade de threads usadas. Uma thread é uma unidade de controle dentro de um processo (CARVER ; TAI (2006)). Quando se executa um programa em multitarefa, é criada uma thread principal, que é responsável por criar as demais threads necessárias para a execução do programa. A quantidade de threads que serão utilizadas para a execução do programa é especificada no início da execução ou na programação.

Após a execução de cada montador, foi calculado o tempo médio das séries para definir o speedup. Em computação paralela, o speedup é utilizado para calcular o quanto um algoritmo paralelo é mais rápido que um algoritmo sequencial correspondente (BERTSEKAS et al. (1997)). Para o cálculo do speedup, são necessários os parâmetros p, que determinam o número de processadores, T_1 é o tempo de execução do algoritmo sequencial e T_p é o tempo de execução do algoritmo paralelo com p processadores. (Equação 6.1)

$$S_p = \frac{T_1}{T_p} \tag{6.1}$$

O servidor utilizado possui dois processadores com oito núcleos cada, em um total de 16 núcleos. O número de threads utilizadas variou de 1 a 24.

Outra métrica importante é a eficiência. Eficiência é uma métrica de desempenho utilizada em conjunto com o speedup. A eficiência especifica o percentual de uso dos processadores durante o processo de execução. Esses valores variam entre 0 e 1. Quanto mais próximo de 1, melhor é o processo de paralelização do programa.

Para o cálculo da eficiência são necessários os parâmetros S_p, valor de speedup encontrado e p é o número de processadores utilizados. (Equação 6.2)

$$E_p = \frac{S_p}{p} \tag{6.2}$$

6.2 Cromossomo Humano 14

Neste trabalho, o genoma do cromossomo humano 14 foi usado como referência para validar os resultados obtidos na montagem do genoma de novo do Muriqui. Na Figura 6.1, podemos observar o desempenho dos montadores avaliados utilizando o cromossoma humano como entrada de dados.

Os montadores Allpaths-LG e SOAPdenovo2 obtiveram ganhos próximos entre 1 e 16 cores. Quando foram utilizadas entre 16 e 24 cores, a curva se tornou plana, pois atingiu o número máximo de cores do servidor.

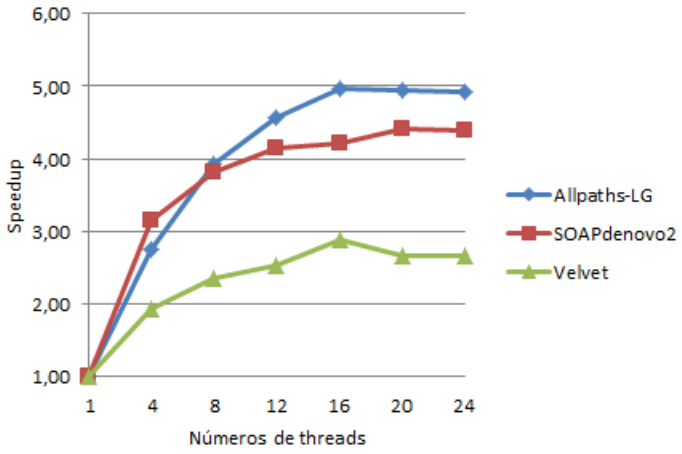


Figura 6.1: Conjunto de dados cromossomo humano 14.

A Tabela 6.1 mostra os tempos obtidos durante o processo de montagem usando 1 e 16 threads.

Tabela 6.1: Montagem do cromossomo humano 14.

Montadores	Tempo		Speedup	Eficiência
	1 Thread	16 Threads		
Allpaths-LG	102600s	20640s	4,97	0,31
SOAPdenovo2	3314s	787s	4,21	0,26
Velvet	14880s	5160s	2,88	0,18

O genoma do cromossomo humano 14 foi utilizado para validar os testes com o genoma de novo do primata Muriqui. A qualidade dos resultados obtidos neste estudo foi próxima aos resultados encontrados em outros estudos já realizados. Grande parte dos estudos já realizados se preocupa somente com a qualidade da montagem, como o GAGE (SALZBERG et al. (2012)), que avaliou a qualidade da montagem e não o desempenho obtido pelos montadores. Neste estudo foi utilizado o genoma do cromossomo humano 14.

6.3 Genoma de novo Muriqui

Os resultados obtidos durante o processo de montagem do genoma de novo do primata Muriqui, utilizando os montadores avaliados no estudo, tiveram os resultados de desempenho diferentes aos encontrados durante o processo de montagem do genoma do cromossoma humano 14. Na Figura 6.2 são observados os resultados dos montadores utilizando o genoma de novo do primata Muriqui.

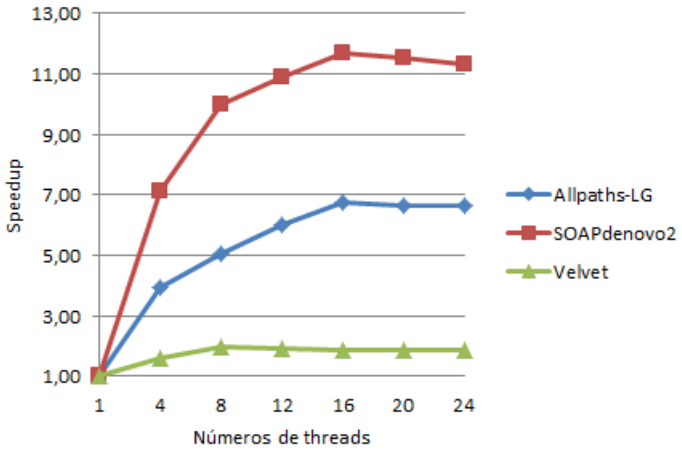


Figura 6.2: Conjunto de dados do primata Muriqui.

O montador Allpaths-LG teve um ganho de aproximadamente 35% em relação ao obtido na montagem do cromossomo humano 14. Já o montador SOAPdenovo2 obteve o melhor ganho entre os montadores e teve um speedup superlinear, quando foram utilizados entre 4 e 12 cores.

A Tabela 6.2 mostra os tempos obtidos durante o processo de montagem usando 1 e 16 threads

Os dados do genoma de novo do primata Muriqui se encontram divididos em quatro conjuntos de dados, os quais foram executados separadamente em cada

Tabela 6.2: Montagem do genoma de novo Muriqui.

Montadores	Tempo		Speedup	Eficiência
	1 Thread	16 Threads		
Allpaths-LG	270660s	40200s	6,73	0,42
SOAPdenovo2	30742s	2631s	11,68	0,73
Velvet	42250s	22411s	1,88	0,11

montador. Cada conjunto de dados possui tamanho diferente, o que influenciou nos resultados obtidos pelos montadores.

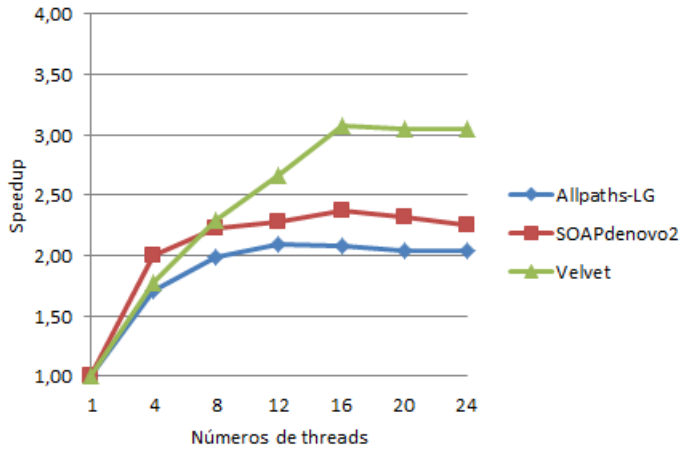


Figura 6.3: Conjunto de dados Muriqui8.

A Figura 6.3 mostra o desempenho dos montadores utilizando o conjunto de dados Muriqui8, que tem a menor quantidade de dados entre os conjuntos avaliados do genoma do primata Muriqui. O montador Velvet teve um melhor desempenho em relação aos outros montadores.

Utilizando o conjunto de dados Muriqui9, o montador SOAPdenovo2 mostrou uma melhor eficiência em relação aos montadores Allpaths-LG e Velvet (Figura 6.4). Observamos aqui também a ocorrência de speedup superlinear na execução do montador SOAPdenovo2 com 4 e 8 threads.

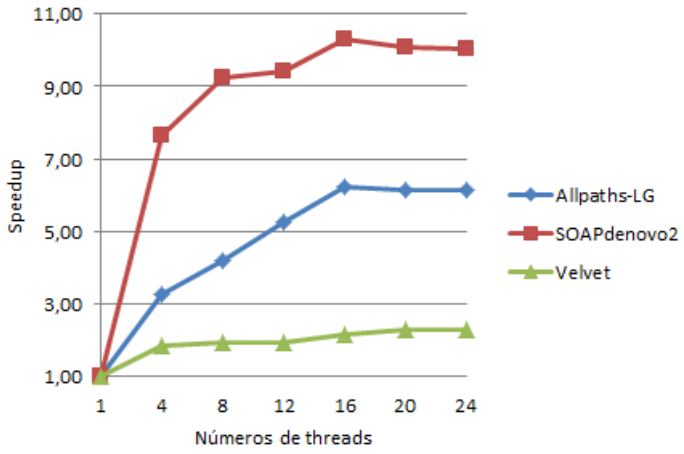


Figura 6.4: Conjunto de dados Muriqui9.

Nas Figuras 6.5 e 6.6 são vistos os resultados utilizando-se os conjuntos de dados Muriqui10 e Muriqui12. Mais uma vez, o montador SOAPdenovo2 teve um ganho superior em relação aos demais montadores.

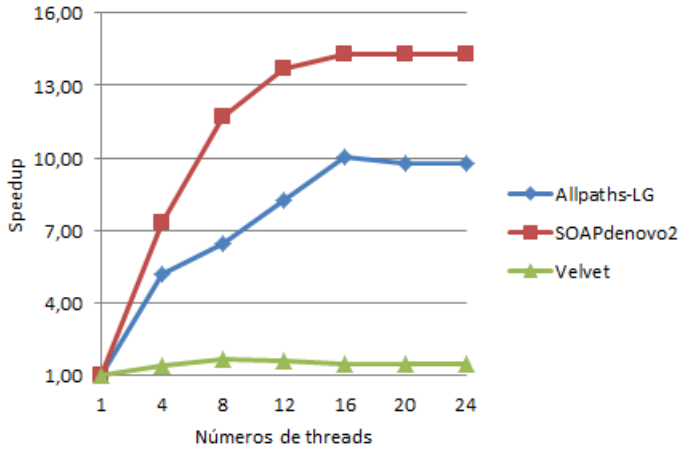


Figura 6.5: Conjunto de dados Muriqui10.

O desempenho dos montadores utilizando-se os conjuntos de dados do genoma do primata Muriqui será detalhado nas seções a seguir.

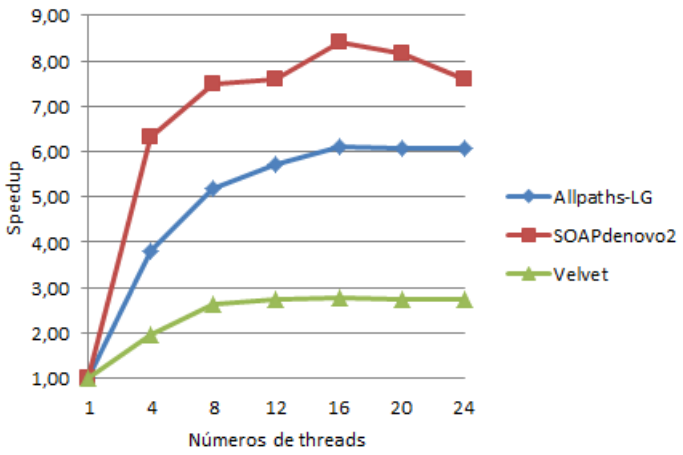


Figura 6.6: Conjunto de dados Muriqui12.

6.3.1 Allpaths-LG

O montador teve um bom desempenho utilizando tanto os dados do cromossomo humano 14 como do Muriqui, porém o desempenho do segundo foi melhor do que o do primeiro. Com os conjuntos de dados Muriqui9 e Muriqui12, os desempenhos obtidos foram muito próximos. Quando o conjunto de dados Muriqui8 foi utilizado, o desempenho do montador piorou. Isso é uma característica desses montadores, já que trabalham melhor quando processando grandes volumes de dados. A Figura 6.7 mostra os resultados que foram obtidos usando o conjunto de dados do genoma Muriqui.

Por esse motivo, o Muriqui10 obteve o melhor desempenho entre os conjuntos de dados. Na Tabela 6.3 é visto o desempenho do genoma Muriqui usando separadamente cada um dos conjuntos de dados. Observamos o desempenho do montador utilizando 1 ou 16 cores

Outro fato importante é visto na Figura 6.8: como o montador se comporta

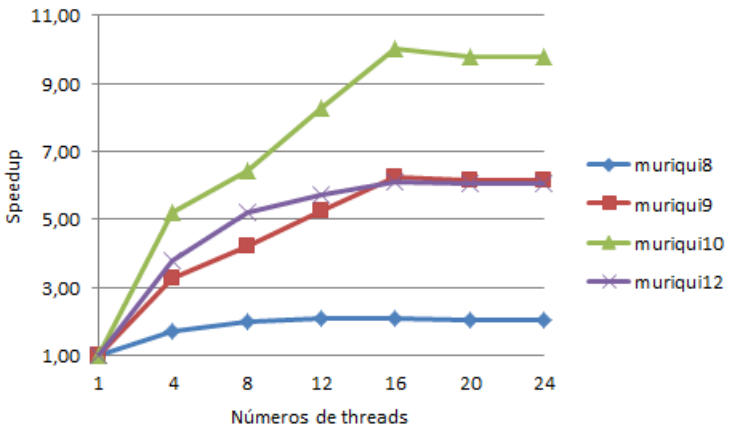


Figura 6.7: Desempenho do conjunto de dados Muriqui.

Tabela 6.3: Desempenho do montador utilizando o genoma Muriqui.

Conjuntos de dados	Tempo		Speedup	Eficiência
	1 Thread	16 Threads		
Muriqui8	12960s	6240s	2,08	0,13
Muriqui9	48120s	7740s	6,22	0,38
Muriqui10	126300s	12600s	10,02	0,63
Muriqui12	83280s	13620s	6,11	0,38

com a variação de threads durante o processo de montagem do genoma Muriqui. À medida que a quantidade de threads aumenta, o valor percentual do processamento paralelo aumenta.

6.3.2 SOAPdenovo2

O montador SOAPdenovo2, como descrito anteriormente, teve melhorias em seu algoritmo em relação às versões anteriores. Pode ser visto na Figura 6.9 o resultado de todos os conjuntos de dados executados separadamente. Como observado, o melhor ganho de speedup foi obtido no processo de montagem do conjunto de dados

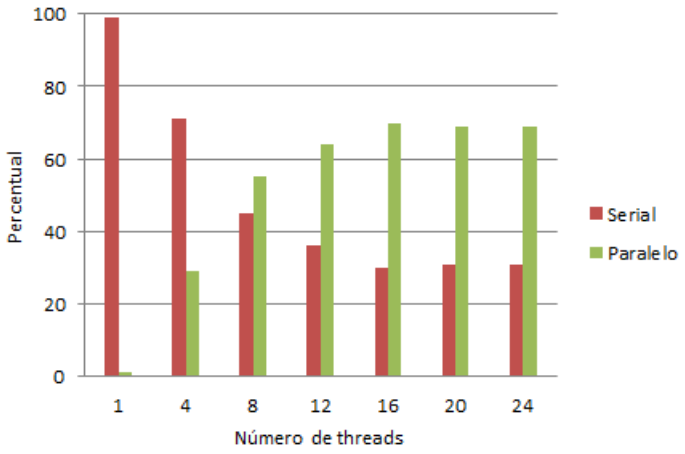


Figura 6.8: Percentual de uso do montador Serial vs Paralelo.

Muriqui10.

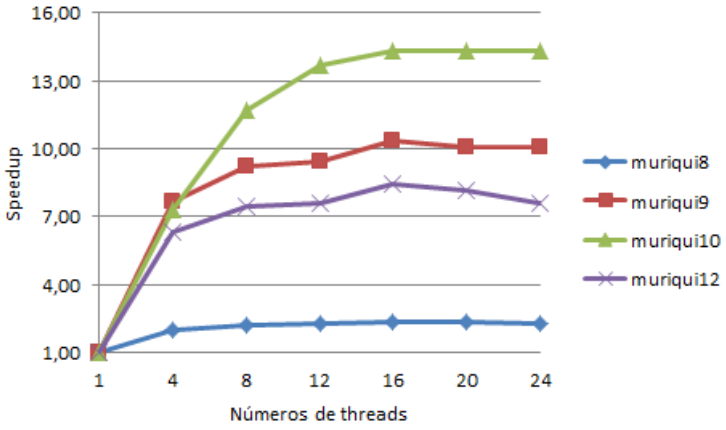


Figura 6.9: Desempenho dos conjunto de dados Muriqui.

Também verificamos a ocorrência de speedup superlinear, ao executar o conjunto de dados Muriqui10. Isso ocorreu, principalmente, porque o processo de leitura dos arquivos é executado em paralelo, usando todas as threads definidas, como visto na Tabela 6.4.

Tabela 6.4: Desempenho dos conjuntos de dados Muriqui.

Conjuntos de dados	Tempo		Speedup	Eficiência
	1 Thread	16 Threads		
Muriqui8	181s	76s	2,37	0,15
Muriqui9	6913s	669s	10,3	0,64
Muriqui10	18960s	1327s	14,3	0,89
Muriqui12	4687s	558s	8,4	0,53

O processo de montagem do programa SOAPdenovo2 tem três passos importantes. O primeiro passo lê os arquivos e cria o pré-grafo. O segundo passo remove as pontas, e o último passo alinha as leituras feitas. Na Tabela 6.5, é descrito o tempo utilizado em cada passo. O tempo total para executar o conjunto de dados Muriqui8 com 1 thread foi de aproximadamente 10 minutos; quando executado o mesmo conjunto de dados utilizando 16 threads, o tempo total foi de aproximadamente 4 minutos. As diferenças entre os tempos de execução utilizando 1 ou 16 threads é pequeno, isto ocorre porque o conjunto de dados tem um tamanho pequeno.

Tabela 6.5: Percentual de tempo utilizado na montagem do conjunto de dados Muriqui8.

Passo	1 Thread		16 Thread	
	Tempo	Percentual	Tempo	Percentual
1	74s	41%	20s	26%
2	54s	30%	45s	59%
3	53s	29%	11s	15%
Total	181s	100%	76s	100%

Na Tabela 6.6 são apresentados os resultados utilizando-se o conjunto de dados Muriqui10. Quando há apenas uma thread, a maior parte do tempo de execução é gasto no primeiro passo. Quando o programa é executado com 16 threads, o tempo de execução no primeiro passo cai significativamente. O tempo total para executar o conjunto de dados Muriqui10 com 1 thread foi de aproximadamente 5 horas, e para executar o mesmo conjunto de dados com 16 threads o tempo total foi

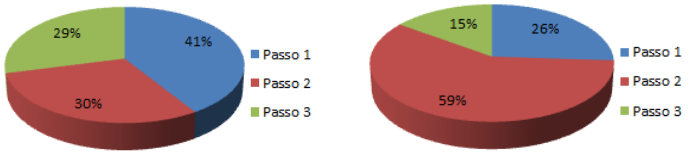
de aproximadamente 25 minutos.

Tabela 6.6: Percentual de tempo utilizado na montagem do conjunto de dados Muriqui10.

Passo	1 Thread		16 Thread	
	Tempo	Percentual	Tempo	Percentual
1	14789s	78%	372s	28%
2	1896s	10%	703s	53%
3	2275s	12%	252s	19%
Total	18960s	100%	1327s	100%

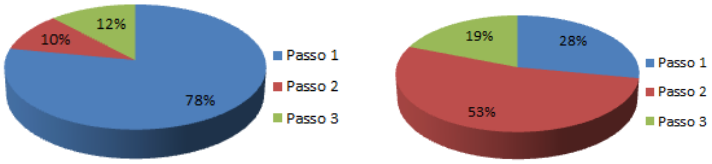
Como o conjunto de dados Muriqui10 é maior que o Muriqui8, a quantidade de threads utilizadas durante o processo de leitura influencia no tempo total de montagem do genoma. A nova versão do algoritmo do SOAPdenovo2 torna o montador muito mais eficiente em relação às versões anteriores.

Nas Figuras 6.10 e 6.11, é visto como a nova versão do algoritmo do SOAPdenovo2 divide o percentual de uso em cada passo da montagem.



(a) Muriqui8 com 1 thread (b) Muriqui8 com 16 threads

Figura 6.10: Percentual de uso durante o processo de montagem do Muriqui8.



(a) Muriqui10 com 1 thread (b) Muriqui10 com 16 threads

Figura 6.11: Percentual de uso durante o processo de montagem do Muriqui10.

Para auxiliar na identificação das causas do speedup superlinear durante o

processo de montagem do conjunto de dados do Muriqui10, foi utilizado o programa Intel VTune Amplifier XE 2013 Update 12.

Na Figura 6.12, é vista a utilização de recursos do montador SOAPdenovo2 pelo Intel VTune. Durante o processo de montagem, os arquivos sequenciados são divididos em partes especificadas pelo parâmetro -k e são processados pela quantidade de threads especificadas usando o parâmetro -p.

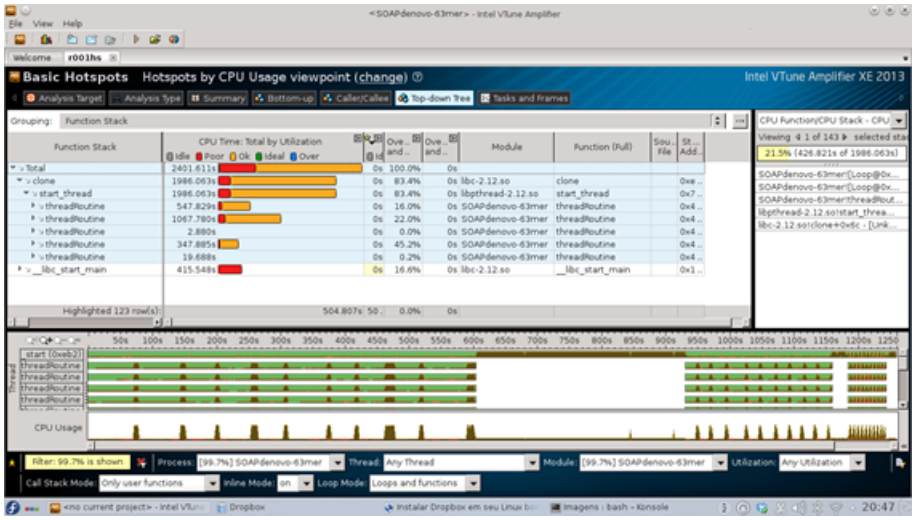
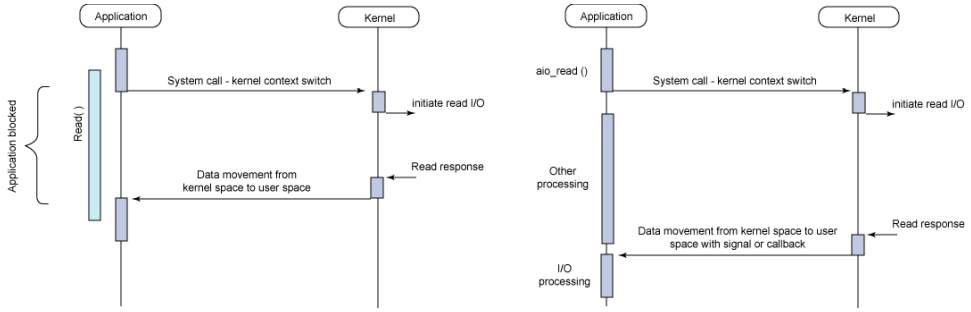


Figura 6.12: Utilização de recursos pelo montador SOAPdenovo2.

O programa SOAPdenovo2 possui um alto grau de paralelismo, pois na maior parte do tempo de montagem são utilizadas todas as threads que foram especificadas na linha de comando.

Durante a leitura dos arquivos de entrada, com o conjunto de leituras curtas geradas pelo sequenciador Illumina, é utilizada a rotina de leitura assíncrona aio_read (Figura 6.13).

O uso dessa leitura permite que a montagem do grafo de Bruijn e as operações de entrada e saída sejam realizadas em paralelo. Em particular, as operações de



(a) I/O Síncrona (b) I/O Assíncrona

Figura 6.13: Modelos de I/O.

entrada e saída são beneficiadas pela buerização do sistema operacional, que faz a leitura em clusters de 4 Kbytes, que contêm vários setores de 512 bytes.

Essa é, no nosso ponto de vista, a principal explicação para a ocorrência do speedup superlinear no montador SOAPdenovo2. Esse tipo de implementação, com rotinas de entrada e saída assíncronas, não é encontrado em nenhum dos outros montadores analisados em nosso trabalho.

Com a utilização dessas funções no código do montador, o SOAPdenovo2 obtém um ganho maior na medida em que os conjuntos de dados também aumentam de tamanho.

6.3.3 Velvet

Na Figura 6.14, o desempenho do montador utilizando os conjuntos de dados Muriqui8, Muriqui9 e Muriqui12 tem o mesmo desempenho com 1 ou 4 threads. Quando são utilizadas 8 threads, o desempenho do conjunto de dados Muriqui8 é melhor que os demais. Por sua vez, o conjunto de dados Muriqui10 se manteve sempre com desempenho abaixo dos demais, isto ocorre porque o processo de leitura

das sequências, em grande parte do tempo, é feito utilizando somente 1 thread.

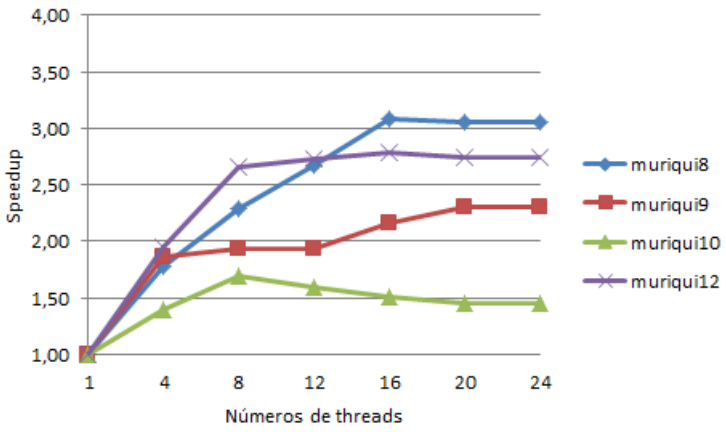


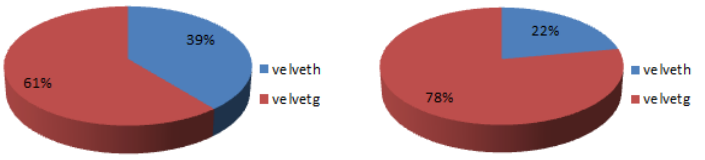
Figura 6.14: Desempenho dos conjunto de dados Muriqui.

Na Tabela 6.7 é visto o speedup obtido após a execução da montagem dos dados do genoma do primata Muriqui.

Tabela 6.7: Desempenho dos conjuntos de dados do genoma do primata Muriqui.

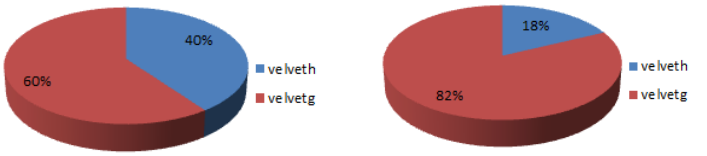
Conjuntos de dados	Tempo		Speedup	Eficiência
	1 Thread	16 Threads		
Muriqui8	960s	312s	3,08	0,19
Muriqui9	8820s	4080s	2,16	0,13
Muriqui10	20770s	13816s	1,50	0,09
Muriqui12	11700s	4203s	2,78	0,17

Para o processo de montagem, o programa Velvet utiliza os comandos velveth e velvetg. Quando o comando velvetg é utilizado, o tempo de processamento é muito alto, independentemente da quantidade de threads especificadas para o processo de montagem. Nas Figuras 6.15 e 6.16 é visto o percentual de execução de cada comando durante o processo de montagem.



(a) Muriqui8 com 1 thread (b) Muriqui8 com 16 threads

Figura 6.15: Percentual de uso durante o processo de montagem do Muriqui8.



(a) Muriqui10 com 1 thread (b) Muriqui10 com 16 threads

Figura 6.16: Percentual de uso durante o processo de montagem do Muriqui10.

6.4 Intel Xeon Phi

Escolhemos o montador SOAPdenovo2 para a execução dos testes no Intel Xeon Phi por ter apresentado os melhores resultados entre os montadores, utilizando-se uma arquitetura com processadores multicore. Os dados usados para a execução dos testes foram o genoma do cromossomo humano 14 e o conjunto de dados Muriqui10. Para a execução do montador no coprocessador foram feitas somente as alterações necessárias para a compilação do código no Intel Xeon Phi.

Nas Figuras 6.17 e 6.18 são vistos os resultados do processo de montagem variando a quantidade de threads entre 1 e 60 no coprocessador e entre 1 e 24 threads no processamento em CPU.

Os resultados mostram que o processo de montagem usando o coprocessador obteve ganhos superiores aos obtidos em CPU, quando o coprocessador usou entre 1 e 40 threads. Quando o coprocessador utilizou a partir de 50 threads o processamento em CPU, obteve um melhor desempenho, no entanto os tempos de execução

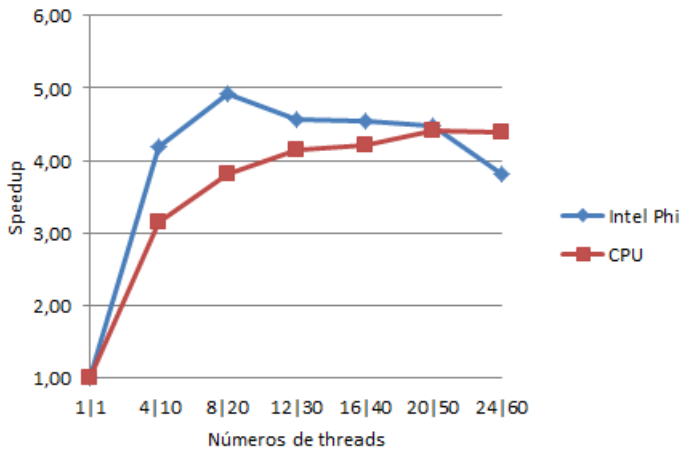


Figura 6.17: Desempenho do genoma cromossomo humano 14 usando Intel Xeon Phi vs CPU.

tiveram uma diferença de até 25% no processo de montagem usando o genoma do cromossomo humano 14 entre os melhores tempos de cada montador.

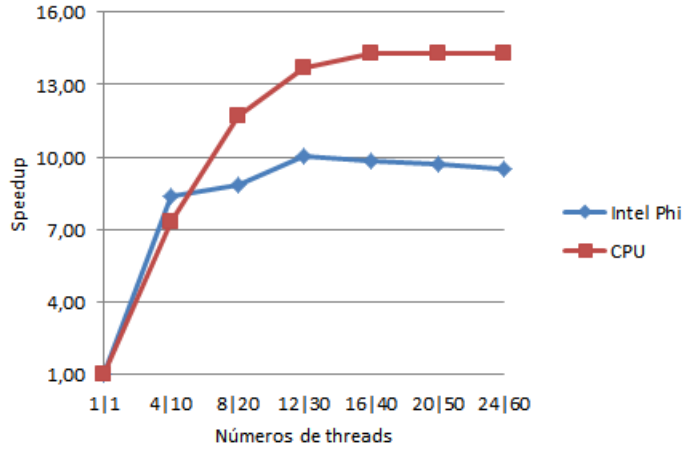


Figura 6.18: Desempenho do conjunto de dados Muriqui10 usando Intel Xeon Phi vs CPU.

As Tabelas 6.8 e 6.9 mostram os resultados encontrados pelo montador utilizando o coprocessador e CPU.

Tabela 6.8: Desempenho do genoma cromossomo humano 14.

Intel Xeon Phi			CPU		
Tempo	Threads	Speedup	Tempo	Threads	Speedup
2781s	1	1,00	3314s	1	1,00
662s	10	4,20	1054s	4	3,14
565s	20	4,92	867s	8	3,82
608s	30	4,57	798s	12	4,15
612s	40	4,54	787s	16	4,21
622s	50	4,47	751s	20	4,41
728s	60	3,82	756s	24	4,38

Ao utilizar o conjunto de dados Muriqui10, os tempos de execução tiveram uma variação de até 10% no processo de montagem. Utilizando o conjunto de dados Muriqui10, o montador SOAPdenovo2 obteve um desempenho 30% superior, executando o processo de montagem em CPU.

Tabela 6.9: Desempenho do conjunto de dados muriqui10.

Intel Xeon Phi			CPU		
Tempo	Threads	Speedup	Tempo	Threads	Speedup
14706s	1	1,00	18960s	1	1,00
1758s	10	8,37	2585s	4	7,33
1664s	20	8,84	1620s	8	11,70
1464s	30	10,05	1387s	12	13,67
1490s	40	9,87	1327s	16	14,29
1516s	50	9,70	1327s	20	14,29
1543s	60	9,53	1327s	24	14,29

Esse baixo desempenho do coprocessador durante o processo de montagem está associado ao fato de que não houve nenhuma alteração no código do programa SOAPdenovo2. Somente foram feitas as alterações necessárias para a execução do montador no coprocessador. Outro fator que pode ter causado esse baixo desempenho é que o conjunto de dados Muriqui10 é aproximadamente 20% maior do que o genoma do cromossomo humano 14.

6.5 Tempo de montagem

Outro fator importante é o tempo diferente de montagem dos genomas em cada montador, pois cada um processa os dados de forma variada. Entre os montadores avaliados, o SOAPdenovo2 obteve os melhores tempos de montagem, tanto usando o genoma do cromossomo humano 14 como o genoma do primata Muriqui.

Nas Tabelas 6.10 e 6.11 são vistas as diferenças de tempos de montagem dos genomas usando o SOAPdenovo2 entre os montadores Allpaths-LG e Velvet. As maiores diferenças de tempos encontradas entre os genomas usados foram em relação ao montador Allpaths-LG

Tabela 6.10: Tempo de montagem do cromossomo humano 14.

Montadores	Tempo de montagem	
	1 Thread	16 Threads
Allpaths-LG	102600s	20640s
SOAPdenovo2	3314s	787s
Velvet	14880s	5160s
Montadores	Diferença de tempos	
	1 Thread	16 Threads
Allpaths-LG	31,0	26,2
Velvet	4,5	6,6

Embora o montador Allpaths-LG tenha obtido um speedup superior ao montador Velvet, os tempos de montagem dos genomas usando o Allpaths-LG foram maiores em relação aos tempos encontrados no processo de montagem usando o Velvet.

Tabela 6.11: Tempo de montagem do genoma do primata Muriqui.

Montadores	Tempo de montagem	
	1 Thread	16 Threads
Allpaths-LG	270660s	40200s
SOAPdenovo2	30742s	2631s
Velvet	42250s	22411s
Montadores	Diferença de tempos	
	1 Thread	16 Threads
Allpaths-LG	8,8	15,3
Velvet	1,4	8,5

6.6 Uso de memória

Outro fator importante monitorado foi a quantidade de memória consumida pelos montadores. Para analisar a utilização de memória em cada montador durante o processo de montagem, foi usado o genoma do cromossomo humano 14 e o conjunto de dados Muriqui10; ambos os genomas em seu processo de montagem utilizaram 16 threads. Para avaliar o uso de memória durante o processo de montagem, foi usado o comando smem, que exibe o uso da memória física por cada processo em execução.

Durante o processo de montagem, o montador Allpaths-LG utiliza os recursos de memória com maior eficiência tanto ao usar o genoma do cromossomo humano 14 quanto o conjunto de dados do Muriqui10. Nas Figuras 6.19 e 6.20 vemos a utilização de memória do montador Allpaths-LG.

Assim como o montador Allpaths-LG, o SOAPdenovo2 também utiliza a memória sem a necessidade do uso de swap durante o processo de montagem. A utilização de memória pelo montador aumenta durante o processo de montagem, como pode ser observado nas Figuras 6.21 e 6.22.

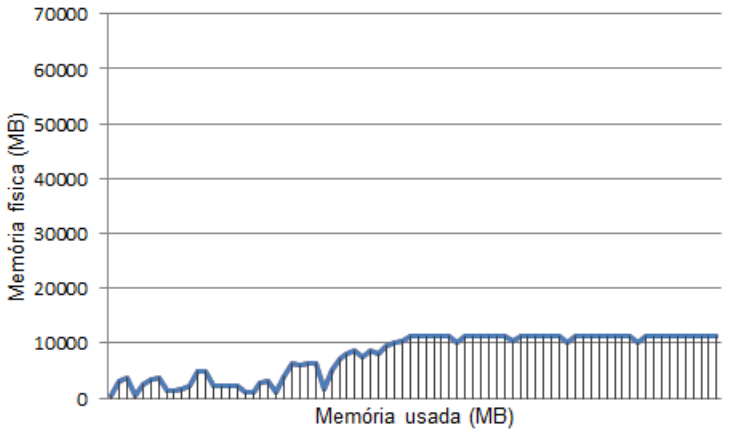


Figura 6.19: Memória utilizada pelo Allpaths-LG processando cromossomo humano 14.

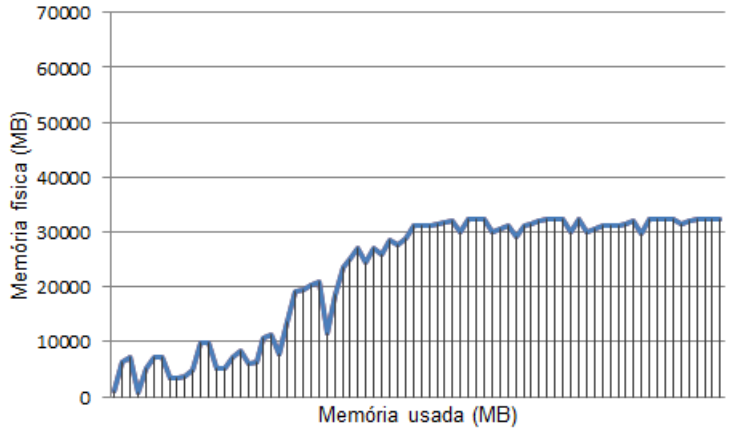


Figura 6.20: Memória utilizada pelo Allpaths-LG processando Muriqui10.

Entre os montadores, o Velvet utiliza toda a memória disponível durante o processo de montagem de ambos os conjuntos de dados utilizados. Para continuar o processo de montagem dos dados, foi necessário o uso de área swap pelo montador. O fato de o Velvet precisar utilizar área swap durante o processo de montagem causou impacto no seu desempenho. Nas Figuras 6.23 e 6.24 é vista a utilização de memória do Velvet durante o processo de montagem.

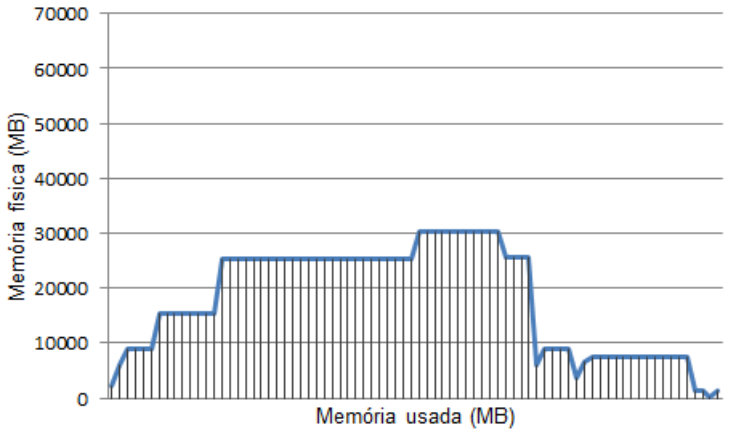


Figura 6.21: Memória utilizada pelo SOAPdenovo2 processando cromossomo humano 14.

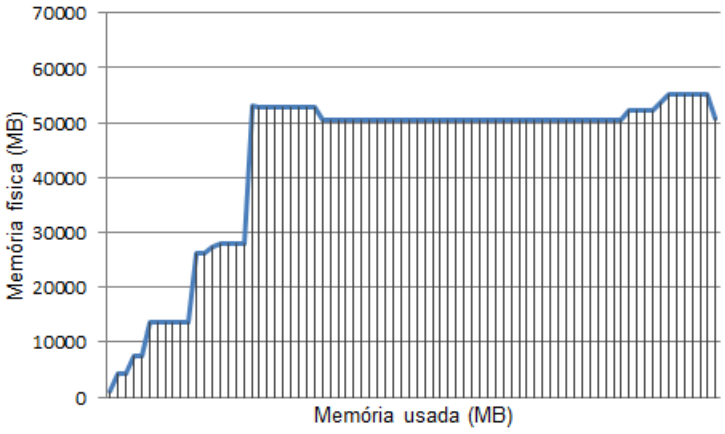


Figura 6.22: Memória utilizada pelo SOAPdenovo2 processando Muriqui10.

O uso da memória *swap* em disco por um processo em execução tem um grande impacto no seu desempenho, devido ao tempo perdido enviando e trazendo páginas da memória para o disco (e vice-versa). Isso explica o baixo desempenho do programa Velvet, que faz uso pesado de swap quando o volume de dados processados é alto, como no caso do processamento do genoma do primata Muriqui.

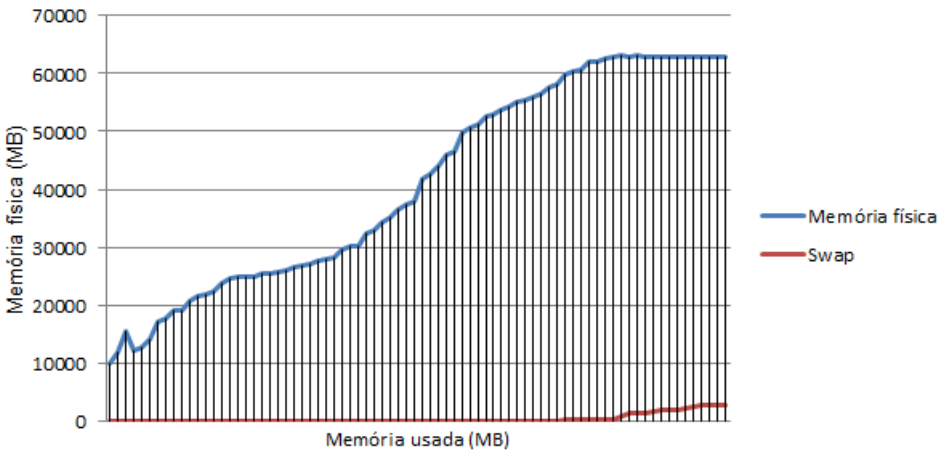


Figura 6.23: Memória utilizada pelo Velvet processando cromossomo humano 14.

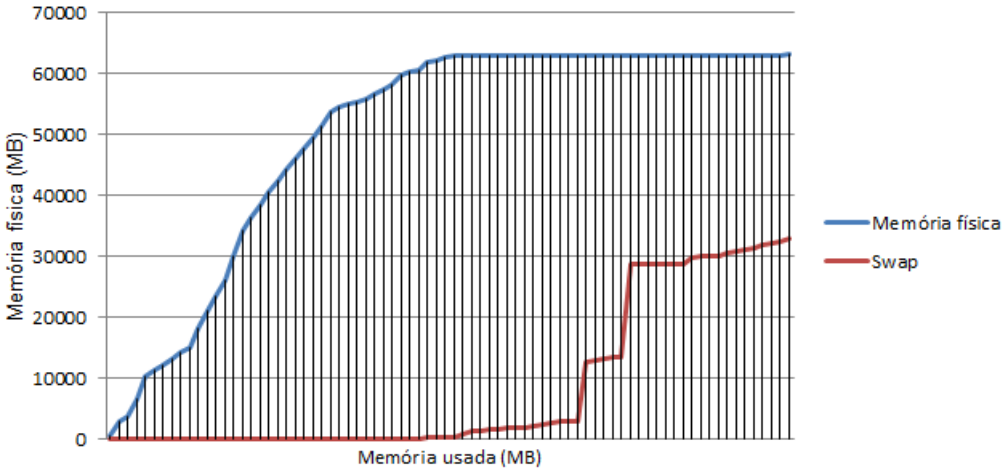


Figura 6.24: Memória utilizada pelo Velvet processando Muriqui10.

6.7 Qualidade da montagem

Como dito anteriormente, a qualidade da montagem não foi usada como um parâmetro para avaliar os montadores utilizados neste estudo. Ambos os genomas tiveram uma qualidade de montagem dentro dos padrões desejados. Os dados utilizados para a verificação de qualidade de montagem foram o cromossomo humano

14 e o conjunto de dados Muriqui.

A qualidade da montagem dos genomas também é um fator de importância para a avaliação dos montadores. Para avaliar a qualidade da montagem dos genomas, são utilizados alguns parâmetros, como o N50, N90 e o número de scaffolds.

N50 é um parâmetro que define o tamanho mínimo dos contigs que contêm 50% das bases utilizadas durante o processo de montagem. O N50 é amplamente usado em Bioinformática, especialmente no que se refere aos comprimentos dos contigs.

Na Tabela 6.12 são apresentados os resultados da montagem usando o genoma do cromossomo humano 14

Tabela 6.12: Resultado da montagem do genoma do cromossomo humano 14.

Montadores	Num. Contigs	Num. Scaffolds	N50
Allpaths-LG	3741	104	91
SOAPdenovo2	18832	455	458
Velvet	38729	1664	941

Para o cálculo do tamanho do parâmetro N50, todos os contigs são organizados do maior valor para o menor valor e determinado o conjunto mínimo de contigs cujo tamanho total é 50% do genoma montado. Já o parâmetro N90 representa 10% da soma de todos os comprimentos dos contigs.

Os resultados obtidos após o processo de montagem usando o conjunto de dados Muriqui são apresentados na Tabela 6.13.

A baixa qualidade no resultado obtido durante o processo de montagem do genoma do Muriqui ocorreu principalmente pela própria qualidade dos arquivos se-

Tabela 6.13: Resultado da montagem do conjunto de dados Muriqui.

Montadores	Num. Contigs	Num. Scaffolds	N50
Allpaths-LG	1047	29	25
SOAPdenovo2	5273	126	128
Velvet	10844	466	263

quenciados. Durante o processo de sequenciamento dos arquivos, foram encontradas várias áreas que não foram identificadas pelo sequenciador, o que gera uma quantidade muito grande de áreas com informações perdidas de nucleotídeos.

7 CONCLUSÃO

Neste estudo foram avaliados alguns montadores paralelos de genomas, nominadamente, Allpaths-LG, SOAPdenovo2 e Velvet. O desempenho desses montadores foi avaliado com o uso de conjuntos de dados de diversos tamanhos e ainda executados sob um variado número de processadores, além do acelerador Xeon Phi. Os resultados obtidos indicam que esses montadores têm grande variação de desempenho em função do tamanho do conjunto de dados e dos recursos computacionais disponíveis para o processo de montagem. O desempenho dos montadores avaliados é menor quando são utilizados conjuntos de dados de menor tamanho, e melhora na medida em que o tamanho desses conjuntos de dados aumenta.

Dos montadores avaliados neste estudo, os que obtiveram melhor desempenho, independentemente do tamanho dos dados processados, foram Allpaths-LG e SOAPdenovo2. Observou-se, contudo, que o montador Velvet também apresenta bom desempenho no processamento de alguns dos conjuntos de dados utilizados. O montador SOAPdenovo2 mostrou-se mais eficiente que os demais montadores, principalmente processando os dados do genoma do primata Muriqui.

Mais especificamente, encontramos um speedup de 11,68 com o montador SOAPdenovo2 usando o genoma do Muriqui com uma eficiência de 0,73. Esse resultado é 1,7 vez maior que o Allpaths-LG e 6,5 vezes maior que o montador Velvet.

No estudo realizado por Zhang (ZHANG et al. (2011)), também se verificou que alguns montadores, como os que foram utilizados em nosso estudo, não têm um bom desempenho quando utilizam conjuntos de dados com um tamanho muito reduzido. Também no estudo realizado por Zhang, os montadores SOAPdenovo e Velnet

obtiveram o melhor desempenho entre os montadores avaliados. De modo geral, os montadores apresentam um desempenho melhor quando utilizam uma quantidade de dados com um maior tamanho.

Nossos resultados apresentam uma variação de speedup entre os montadores utilizando tamanhos de dados diferentes. Se forem comparados os conjuntos de dados Muriqui8 com um tamanho de 1,7 GB e o Muriqui10 com tamanho de 16,6 GB, o montador Allpaths-LG obtém uma diferença de speedup 2,9. Já o SOAPdenovo2 obtém uma diferença de 6,0 de speedup entre os conjuntos de dados. Entretanto, o montador Velvet teve um speedup baixo, quando utilizou o conjunto de dados Muriqui10. Isso ocorre porque o montador utiliza área swap durante o processo de montagem dos dados.

O speedup de cada montador varia de acordo com a forma como ele trata os dados durante o seu processamento. O SOAPdenovo2 é muito eficiente durante o passo de leitura dos arquivos. Os montadores usados neste estudo tiveram comportamentos diferentes em relação ao uso de recursos computacionais como, por exemplo, na utilização de processamento e memória. Na utilização de recursos de processamento, os montadores Allpaths-LG e SOAPdenovo2 obtiveram um melhor desempenho, utilizando esses recursos de forma mais eficiente.

Em uma análise de tempo de processamento de montagem, o SOAPdenovo2 foi mais rápido em relação aos demais montadores, e foi também em grande parte dos dados usados para a montagem o que obteve o melhor speedup. Entre os montadores, o Velvet foi o que teve o pior speedup. Em relação ao tempo de montagem, usando o Velvet, ele obteve melhor resultado em comparação com o montador Allpaths-LG. O Allpaths-LG tem um speedup em alguns casos de até 3,5 vezes maior que o Velvet, porém o seu tempo de montagem é o mais alto entre os montadores avaliados.

Para a execução do processo de montagem dos dados usados neste estudo, foram necessárias aproximadamente 747 horas (31 dias), e foi gerada uma quantidade de dados próxima de 1 TB de resultados após o processo de montagem. O tempo e a quantidade de dados gerados após o processo de montagem são referentes a todas as séries que foram executadas em cada montador para gerar as informações utilizadas neste trabalho.

Os resultados apresentados neste estudo utilizando os montadores Allpaths-LG, SOAPdenovo2 e Velvet podem ser usados como referência para a montagem de genomas que possuam as mesmas características dos utilizados nos testes. Embora não comparemos a qualidade da montagem obtida pelos montadores avaliados, os resultados das montagens realizadas foram de qualidade aceitável.

REFERÊNCIAS

BERTSEKAS, D. P. et al. **Parallel and Distributed Computation**: numerical methods (optimization and neural computation). Nashua: Athena Scientific, 1997.

BUTLER, J. et al. ALLPATHS: de novo assembly of whole-genome shotgun microreads. **Genome Research**, Bethesda, v. 18, n.5, p.810–820, May 2008.

CARVER ; TAI. **Modern Multithreading**: implementing, tesding, and deabbuging multithreaded java and c++/pthreads/win32programs. Hoboken: John Wiley & Sons, 2006.

CHAVES, P. B. et al. Genetic diversity and population history of a critically endangered primate, the northern muriqui (Brachyteles hypoxanthus). **PloS one**, San Francisco, v. 6, n.6, 2011.

DAHM, R. Friedrich Miescher and the discovery of DNA. **Developmental Biology**, San Diego, v. 278, n.2, p.274–288, 2005.

FAKRUDDIN, M. et al. Pyrosequencing - Principles and Applications. **International Journal of Life Science @ Pharma Research**, [S.l.], v. 2, n.2, Apr./Jun 2012.

GRIFFITHS, A. J. F. et al. **Introdução à Genética - 9ª Ed.** São Paulo: Guanabara Koogan, 2008.

HEBENSTREIT, M. **Conguring Intel Xeon Phi coprocessors inside a cluster**. Denver: INTEL Corporations, 2013.

HEILIG, R. et al. The DNA sequence and analysis of human chromosome 14. **Nature**, London, v. 421, n.6923, p.601–607, Feb 2003.

HOGEWEG, P. The Roots of Bioinformatics in Theoretical Biology. **PLoS Computational Biology**, San Francisco, v. 7, n.3, Mar 2011.

IHGSC. Finishing the euchromatic sequence of the human genome. **Nature**, London, v. 431, n.7011, p.931–945, Oct 2004.

INTEL. **Intel Xeon Phi Coprocessor System Software Developers Guide**. Denver: INTEL Corporations, 2013.

LANDER, E. S. ; WATERMAN, M. S. Genomic mapping by fingerprinting random clones: a mathematical analysis. **Genomics**, San Diego, v. 2, n.3, p.231–239, Apr 1988.

LANDER, E. S. et al. Initial sequencing and analysis of the human genome. **Nature**, London, v. 409, n.6822, p.860–921, 2001.

LUO, R. et al. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. **GigaScience**, London, v. 1, n.1, p.18, Dec 2012.

MILLER, J. R. ; KOREN, S. ; SUTTON, G. Assembly algorithms for next-generation sequencing data. **Genomics**, San Diego, v. 95, n.6, p.315–327, Jun 2010.

PEVZNER, P. A. ; TANG, H. ; WATERMAN, M. S. An Eulerian path approach to DNA fragment assembly. **PNAS - Proceedings of the National Academy of Sciences of the United States of America**, Washington, DC, v. 98, n.17, p.9748–9753, Aug 2001.

RONAGHI, M. Pyrosequencing Sheds Light on DNA Sequencing. **Genome Research.**, Bethesda, v. 18, n.1, p.3–11, Jan 2001.

SALZBERG, S. L. et al. GAGE: a critical evaluation of genome assemblies and assembly algorithms. **Genome Research**, Bethesda, v. 22, n.3, p.557–567, Mar 2012.

SANGER, F. ; COULSON. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. **Journal of molecular biology**, London, v. 94, n.3, p.441–448, May 1975.

VENTER, J. C. et al. The Sequence of the Human Genome. **Science**, Washington, DC, v. 291, n.5507, p.1304–1351, Feb 2001.

WATSON, J. D. ; CRICK, F. H. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. **Nature**, London, v. 171, p.737–738, Apr 1953.

ZERBINO, D. R. ; BIRNEY, E. Velvet: algorithms for de novo short read assembly using de bruijn graphs. **Genome Research**, Bethesda, v. 18, n.5, p.821–829, May 2008.

ZHANG, W. et al. A Practical Comparison of de Novo Genome Assembly Software Tools for Next-Generation Sequencing Technologies. **PLoS ONE**, San Francisco, v. 6, n.3, Mar 2011.