Universidade Federal do Pará
Instituto de Ciências Biológicas
Programa de Pós-graduação em Biotecnologia

AutoAssemblyD

Software para submissão e gerenciamento de montagem de genomas a partir de modelos XML

Adonney Allan de Oliveira Veras

Universidade Federal do Pará

Instituto de Ciências Biológicas

Programa de Pós-graduação em Biotecnologia

AutoAssemblyD

Software para submissão e gerenciamento de montagem de genomas a partir de modelos XML

Adonney Allan de Oliveira Veras

Dissertação submetida ao Programa de Pós-graduação em Biotecnologia UFPA como requisito para a obtenção do grau de Mestre em Biotecnologia

Orientador: Dr. Artur Luiz da Costa da Silva

AGRADECIMENTOS

À Universidade Federal do Pará e ao Programa de Pós-graduação em Biotecnologia pela oportunidade de aprendizado;

A CAPES pela bolsa de pesquisa;

Ao professor Dr. Artur Luiz da Costa da Silva, por ter sido meu orientador, pela confiança e ajuda que me deu durante o mestrado visando sempre meu crescimento pessoal e profissional, e por não ter medido esforços para conclusão deste trabalho;

À Rommel Ramos e Pablo de Sá pela ajuda que me deram para o desenvolvimento e finalização do meu mestrado, especialmente ao professor Rommel por ter sido não somente meu amigo, mas também meu mentor durante essa fase e por ter auxiliado de forma inigualável no meu desenvolvimento pessoal;

À professora Dra. Maria Paula Schneider por ser um exemplo a ser seguido profissionalmente e ser uma ótima companhia dentro e fora do trabalho;

À minha família e amigos que sempre se fizeram presentes e apoiaram o meu desenvolvimento. E aos novos amigos que fiz no LPDNA e que com toda certeza ajudaram de alguma forma neste trabalho.

À minha filha Allana, pela compreensão nos momentos de ausência, amo você filha.

Aos meus pais, obrigado por tudo.

SUMÁRIO

V

LISTA DE ABREVIATURAS

SGS	Sequenciadores de Segunda Geração
XML	Linguagem de marcação (“Extensible Markup Language”)
DNA	Ácido desoxirribonucléico
NGS	Sequenciadores de próxima geração (“Next-Generation Sequencer”)
PGM	Sequenciador produzido pela Ion Torrent
SMRT	Sequenciamento de molécula única de DNA
INDEL	Erro de seqüenciamento onde pode ocorrer uma inserção ou deleção de uma base.
OLC	Metodologia de montagem (“Overlap-layout-consensus”)
DBG	Metodologia de montagem (“Grafo de Bruijn”)
SBH	Sequenciamento por Hibridização
RPC	Chamada de procedimento remoto
RMI	Invocação de método remoto
IP	Endereçamento de rede
TCP	Protocolo de controle de transporte (“Transport Control Protocol”)
UDP	Protocolo de transporte não orientado a conexão (“User Datagram Protocol”)
W3C	Organização para padronização da Internet
ZIP	Programa utilizado para compactação de dados

VI

ÍNDICE DE FIGURAS

VII

ÍNDICE DE TABELAS

VIII

ÍNDICE DE QUADROS

IX

ÍNDICE DE GRÁFICOS

X

RESUMO

As tecnologias de sequenciamento de segunda geração proporcionaram um grande avanço dos estudos genômicos, tornando sua utilização um marco que revolucionou a biologia. Estas plataformas são caracterizadas pela redução no tempo de sequenciamento, alta produção de dados e baixo custo por base sequenciada, contudo, estes equipamentos em sua maioria produzem dados compostos por leituras curtas o que representa um grande desafio para reconstrução do genoma, devido a essa nova característica das leituras ferramentas computacionais tiveram que ser desenvolvidas para realizar a tarefa de montagem exemplo delas temos Velvet, Allpaths, ABySS, SOAP*denovo*2, Edena.

No entanto, a maioria destes aplicativos são executados através de linhas de comandos extensas compostas por vários parâmetros e devem obedecer a uma sintaxe adequada a sua utilização, pois em caso de erros existe a possibilidade de não obtenção do melhor resultado, com o intuito de resolver este problema apresentamos o AutoAssemblyD, que além de proporcionar a utilização destes montadores através de uma interface gráfica também possibilita a gerência destas execuções de forma remota.

XI

ABSTRACT

Technologies for second-generation sequencing provided a major breakthrough of the genome, making its use a landmark that has revolutionized biology. These platforms are characterized by a reduction in sequencing time, high data production and low cost per base sequenced, however, these devices produce data mostly consist of short readings which represents a major challenge for reconstruction of the genome due to this new feature readings of computational tools had to be developed to accomplish the task of assembling their example we Velvet, AllPaths, Abyss, SOAPdenovo2, Edena.

However, most of these applications are executed through command lines extended and composed of several parameters must follow the standard syntax to use, because in case of errors in the syntax is the possibility of not obtaining the best result, with the aim of solve this problem we present the AutoAssemblyD that besides providing the use of these assemblers through a graphical interface also enables the management of these executions remotely.

1 INTRODUÇÃO

As tecnologias de sequenciamento de segunda geração (SGS) revolucionaram a biologia. Dentre as características relacionadas a estas, pode-se citar: a redução do custo e tempo de sequenciamento de genomas e geração de grande quantidade de dados quando comparado com o método de Sanger (El-Metwally *et al.*, 2013; Henson *et al.*, 2012).

Contudo, surgiram desafios como: processamento da grande quantidade de dados produzidos por estas plataformas, principalmente quando não se dispõe de um parque computacional robusto, e a montagem de genomas a partir de leituras curtas, onde há a necessidade de realizar várias montagens com diferentes parâmetros para se aperfeiçoar os resultados (Koren et al., 2014; Schuster, 2008)

Esses desafios impulsionaram o desenvolvimento de ferramentas computacionais para a montagem de genomas a partir de leituras curtas, como Velvet (Zerbino & Birney, 2008), ABySS (Simpson *et al.*, 2009), ALLPATHS (Butler *et al.*, 2008), SOAPdenovo (Li *et al.*, 2010) e MAQ (Li *et al.*, 2008).

Grande parte dos programas de montagem é executada através de extensas linhas de comandos, compostas por um ou vários parâmetros que devem ser definidos, o que dificulta a utilização por usuários com pouco conhecimento de computação (Ramos *et al.*, 2012; Earl *et al.*, 2011).

Desta forma, o uso de uma interface gráfica facilita a utilização dos algoritmos de montagem, como o software VAGUE (Powell & Seemann, 2013), que propõem uma interface gráfica para execução do montador Velvet, onde todos os parâmetros necessários para execução são definidos pelo usuário. No entanto, a disponibilidade apenas para o montador Velvet é uma limitação, considerando a grande variedade de montadores disponíveis (Earl *et al.*, 2011). Assim, este trabalho apresenta o AutoAssemblyD, uma ferramenta gráfica para o gerenciamento de montagem de genomas a partir de qualquer montador, através da criação de templates XML, além de possibilitar a montagem em máquinas remotas por processamento distribuído.

1.1 Sequenciadores de primeira geração.

O método de sequenciamento por terminação de cadeia desenvolvido por Frederick Sanger revolucionou a maneira de se realizar o sequenciamento de DNA, pois apresentava uma alta eficiência, fator que tornou o método desenvolvido por Sanger a principal tecnologia ("primeira geração") de aplicações de sequenciamento de DNA (Liu *et al.*, 2012).

Alguns anos mais tarde, a Applied Biosystems fez diversas melhorias na plataforma disponibilizando para o mercado o primeiro sequenciador automático, conhecido como AB 370. Essa plataforma iniciava suas atividades com o "throughput" de 500 Kilobases sequenciadas por dia, e leituras que poderiam alcançar o comprimento de 600 bases (Liu *et al.*, 2012).

A plataforma sofreu atualizações e passou a ser chamada de AB 3730xl, nesta versão o "throughput" era de 2.88 MB sequenciadas por dia, com comprimento de leitura de 900 bases. A principal desvantagem dessa tecnologia estava relacionada ao alto custo e o baixo "throughput". Contudo, esta plataforma foi amplamente utilizada para completar o Projeto Genoma Humano (Liu *et al.*, 2012).

Com o intuito se sanar os problemas relacionados às plataformas de primeira geração, em 2005 surge uma das primeiras plataformas de segunda geração, com a ampla utilização destas novas tecnologias podemos notar um salto nos estudos genômicos como demonstrado na Figura 1.

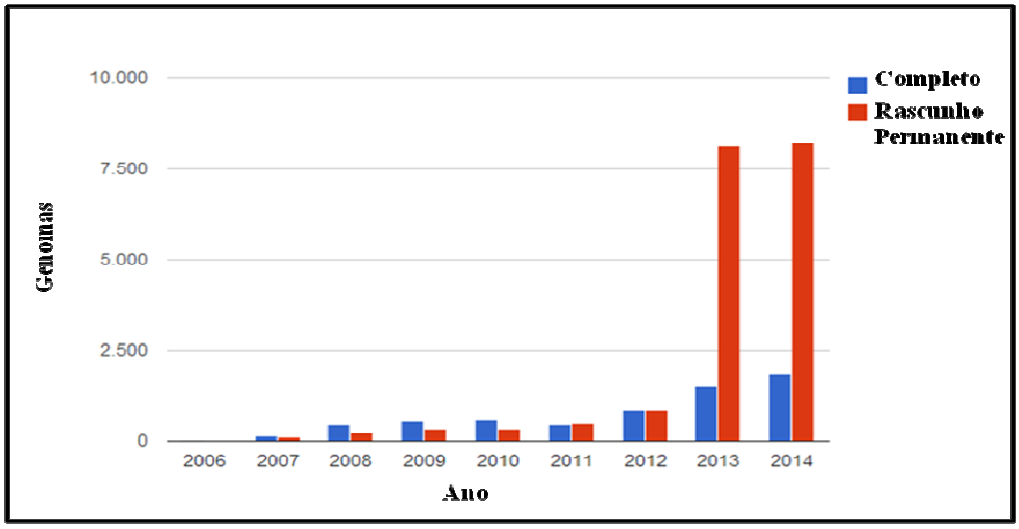

Figura 1. Avanços nos estudos genômicos proporcionados pelas plataformas de segunda geração**.** http://www.genomesonline.org.

1.2 Sequenciadores de segunda geração

As plataformas NGS surgiram em meados de 2005, quando a empresa Roche disponibilizou para o mercado o sequenciador 454, mais tarde outras tecnologias foram lançadas comercialmente como a SOLiD da Life Tecnologies, e GA Illumina (El-Metwally et al., 2013; Henson et al., 2012).

As plataformas de segunda geração também conhecidas como NGS (*Next generation sequencing*) se diferenciam das tecnologias que utilizam o método de Sanger, principalmente pela capacidade de realizar análises massivamente paralelas. Além de reduzir o custo do sequenciamento e apresentar elevado "throughput". Desta maneira, o desenvolvimento desses equipamentos proporcionou uma revolução na biologia (Faino and Thomma, 2014).

No entanto, apesar da alta produção de dados e alta cobertura, estas plataformas produzem leituras curtas, o que resulta no desafio de realizar montagem dos genomas, tanto por questões algorítmicas, como quanto a necessidade de infra-estruturas computacionais para o processamento do volume de dados gerados por estes equipamentos, que cresce de forma exponencial (Zhang *et al.*, 2011).

1.2.1 Roche 454

O sequenciador 454 produzido pela ROCHE (www.454.com) foi a primeira plataforma de segunda geração disponibilizada comercialmente em 2005, e utiliza o método de sequenciamento denominado pirosequenciamento. Na primeira versão, o seu "throughput" era capaz de gerar 20 MB por sequenciamento, as leituras produzidas apresentavam comprimento de aproximadamente 150 pb. Na versão atual conhecida como GS FLX, as leituras produzidas podem chegar a 700 pb, em alguns casos podem até superar esse valor chegando a 1000 pb. Com o "throughput" de 7 GB por rodada (Tabela 01), a média da taxa de erro fica em torno de 0.5% (Liu *et al.*, 2012).

Com o avanço dessas tecnologias houve um notável aumento do comprimento da leitura, possibilitando a utilização da plataforma para outras aplicações, como sequenciamento *de novo* de genomas completos e resequenciamento de genomas. No entanto, a plataforma apresenta problemas quanto à indels, principalmente no reconhecimento de homopolímeros que se trata da repetição de uma base consecutiva

como: AAAAAA ou GGGGGG, esta plataforma é capaz de trabalhar as bibliotecas genômicas fragments e paired-end (El-Metwally et al., 2013; Shendure and Ji, 2008).

1.2.2 Plataforma 454 GS Junior

A plataforma 454 GS Junior (Roche) faz uso da mesma abordagem utilizada na versão (454 GS FLX), utilizando as técnicas de PCR por emulsão e o pirosequenciamento. Este sequenciador "*benchtop*" tem "throughput" em torno de 70 MB por rodada, com leituras produzidas que podem alcançar o comprimento de mais de 600 bases (Loman *et al.*, 2012).

O 454 GS Junior é apto ao sequenciamento *de novo*, e resequenciamento de regiões alvo (Loman *et al.*, 2012).

1.2.3 Illumina GA/HiSeq

A plataforma GA (*Genome Analyzer*) foi disponibilizada em 2006 e as leituras produzidas inicialmente eram de 35 pb, com um "throughput" de 1GB e a taxa de erro de 1%. O tipo de erro mais comum inerente ao equipamento é a substituição de bases (Henson *et al.*, 2012).

Atualmente, a plataforma HiSeq 2000 (http://www.illumina.com) gera leituras de 100 pb com "throughput" de 600 GB (Tabela 01). A estratégia de sequenciamento utilizada pelo HiSeq 2000 é a mesma do Genome Analyzer, que tem como característica o sequenciamento por síntese – SBS, a taxa de erro deste equipamento fica em torno de menos de 2%, após o processo de filtragem dos dados e as bibliotecas genômicas que esta plataforma trabalha são fragments e paired-end. As aplicações mais comuns dessa plataforma são resequenciamento, montagem *de novo*, e análises de RNA-seq (Henson *et al.*, 2012).

1.2.4 Miseq

O Miseq realiza sequenciamento por síntese, assim como nas versões anteriores dos sequenciadores Illumina. No entanto, sua tecnologia reduziu drasticamente o tempo necessário por rodada e o tamanho do equipamento quando comparado ao Illumina HiSeq, devido as suas características ela é considera "*benchtop*". O "throughput" oriundo deste

equipamento é maior que 1 GB e o comprimento das leituras podem chegar a 150 pb (Henson *et al.*, 2012).

1.2.5 AB SOLiD

A plataforma SOLiD disponibilizada no final de 2006 (*Sequencing by Oligo Ligation Detection*) realiza o sequenciamento baseado na ligase, na primeira versão suas leituras alcançavam 35 pb de comprimento, com "throughput" de 3 GB por sequenciamento. Atualmente, a versão comercial é conhecida por SOLiD 5500xl (anteriormente conhecida como SOLiD4hq), sendo capaz de gerar até 180 GB (Tabela 1), produzindo leituras de comprimento de 75 pb apresentando uma taxa de erro de 0.01% (Henson *et al.*, 2012; Schlebusch & Illing, 2012).

Esta plataforma é a única a produzir dados no formato de *color space,* com 16 possíveis combinações de dois nucleotídeos em quatro possibilidades de cores (Figura 2). Esse sistema de codificação reduz a taxa de erros, possibilita diferenciar polimorfismos verdadeiros de erros de sequenciamento, e aumenta a confiabilidade na identificação de variações genômicas como SNP's adjacentes, inserções, deleções e rearranjos estruturais, quanto às bibliotecas genômicas que a plataforma trabalha temos fragments, paired-end e Mate-paired (Lee et al., 2013; Zhang et al., 2011).

Tabela 1. Tabela com as características dos Sequenciadores de 2ª geração adaptado de Liu et al., (2012) e Scholz et al., (2011).

Sequenciador	454 GS FLX	HiSeq 2000	SOLiDv4
Comprimento da Leitura	700 pb; ~1000 pb	50SE, 50PE, 101PE	50 + 35 pb ou 50 + 50 pb
Dados de Saída por execução	0.7GB	600 GB	180 GB
Tipo de erro	Indel	Substituição	A-T Bias

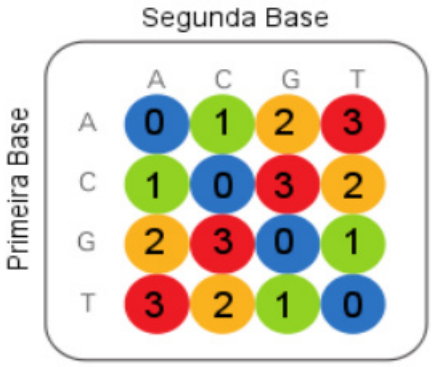

Figura 2. Código de Cores SOLiD

Como resultado do sequenciamento, são produzidos dois arquivos: um com a extensão csfasta e outro qual, nestes arquivos estão presentes as sequências de DNA no formato do código de cores e os valores de qualidade Phred, respectivamente. Abaixo segue um exemplo desses arquivos (Figura 3).

```
>427_21_241_R3      01
G0132103112
>427_22_754_R3
G1301220201
-------------------------------
>427_21_241_R3      02
25 30 21 23 27 29 19 18 22
>427_22_754_R3
26 29 22 24 25 26 20 21 19
```

Figura 3. Exemplo de arquivo csfasta (01) com as bases em código de cores e qual (02) com as qualidades Phred de cada base do arquivo csfasta.

1.3 Sequenciadores de terceira geração

A alta acurácia, grande geração de dados e redução no tempo de preparação das bibliotecas antes do sequenciamento são características das plataformas de terceira geração (Loman *et al.*, 2012).

Uma inovação das plataformas de terceira geração, esta relacionada à forma de detecção do sinal, que pode ser feito usando três novos métodos: detecção da variação de pH em função da liberação do íon de H+ durante a incorporação da base no sequenciamento, SMRT (*Single-molecule real-time*), e o Nanopore que consiste em um

orifício de diâmetro nanométrico, que devido a sua alta sensibilidade, quando as bases de DNA passam através dele causam oscilações de corrente elétrica o que possibilita a identificação das mesmas (Hayden *et al.*, 2012).

Atualmente, como plataformas de terceira geração comercializadas têm-se: HeliScope (Helicos), Ion Torrent PGM (Life Technologies), PacBio (Pacific Biosciences) (Loman *et al.*, 2012; Quail *et al.*, 2012; Zhang *et al.*, 2011), GridIon e MinIon System (https://www.nanoporetech.com).

1.3.1 HeliScope (Helicos)

O HeliScope foi o primeiro sequenciador a utilizar a tecnologia de SMRT (*Single-molecule real-time*), que possibilita o sequenciamento de uma única molécula de DNA em tempo real. Esse equipamento é dotado de um sistema sensível capaz de captar os nucleotídeos assim que estão sendo sintetizados. Atualmente, o seu "throughput" varia de 20 Gb até 28 Gb por rodada, gerando leituras de comprimento de 30pb a 35 pb (Tabela 2). Sua acurácia está em torno de 99 %, a partir do dado bruto (Loman *et al.*, 2012).

1.3.2 Ion Torrent Personal Genome Machine (PGM)

A plataforma Ion Torrent PGM faz parte da terceira geração de sequenciadores, além de ter inaugurado a era dos sequenciadores "pós-luz". Ao realizar a detecção dos nucleotídeos através da tecnologia dos semicondutores, esse sistema realiza a detecção dos íons de hidrogênio liberados durante a polimerização do DNA (Henson *et al.*, 2012).

O "throughput" é personalizável com três capacidades distintas: 10 MB (chip314), 100 MB (chip316) e 1GB (chip 318). O sequenciamento ocorre aproximadamente em duas horas, gerando leituras com comprimento de 400 pb. Sendo que o erro característico desta plataforma é de INDEL (inserção/deleção), em torno de 1% (El-Metwally et al., 2013; Henson et al., 2012).

1.3.3 PacBio RS system (Pacific Biosciences)

Fabricado pela Pacific BioSciences, o PacBio é uma plataforma de sequenciamento que tem dentre suas características a redução do tempo necessário para o sequenciamento, pois a preparação do modelo utilizado no sequenciamento toma de 4 a 6 h. O método de

sequenciamento é o SMRT (*Single-molecule real-time*) que possibilita o sequenciamento sem a necessidade do processo de amplificação. Atualmente seu "throughput" é maior que 35 MB (Tabela 2), com leituras comprimento que pode variar de 4.200 a 8.500 bp (http://www.pacificbiosciences.com) (Henson *et al.*, 2012).

Tabela 2. Tabela com as características dos Sequenciadores 3ª geração
Adaptado de Salmela et al., (2010) e Henson et al., (2012).

Sequenciador	454 GS Junior	Miseq	PGM	PacBio	HeliScope
Comprimento da Leitura	600 pb	150 pb	200 pb	~5000 pb	30-35 pb
Dados de Saída por execução	70 MB	1.6 GB	1 GB	35-45 MB	20-28 GB
Tipo de erro	Indel	Substituição	Indel	CG Deleção	Deleção

1.3.4 GridIon & MinIon System (Oxford Nanopore)

O GridIon é uma plataforma de sequenciamento que utiliza a tecnologia de nanopore para decodificação das bases de DNA e RNA, esta tecnologia opera com um cartucho no qual estão presentes todos os reagente necessários para realização de um experimento, cada nó do GridIon é reconhecido como um dispositivo de rede não havendo, portanto, a necessidade de um servidor dedicado, os dados podem ser disponibilizados em um HD externo conectado a porta USB ou em um equipamento na rede local. Outra inovação da plataforma é capacidade de trabalhar agregando múltiplos nós como um grande cluster computacional (https://www.nanoporetech.com).

A versão miniaturizada do GridIon é conhecida como MinIon, esta plataforma trabalhar seguindo o mesmo workflow de trabalho do GridIon, contudo é um equipamento que se assemelha em muito com um pendrive, devido as suas dimensões, ele é portável e pode ser utilizado diretamente na porta USB de um computador (https://www.nanoporetech.com).

1.4 Qualidade das sequências

O primeiro programa a utilizar valores de qualidades para bases de DNA foi o Phred desenvolvido por Ewing e colaboradores (1998). Esta ferramenta é capaz de calcular índices de qualidade relacionados à probabilidade de erro. Assim, a utilização de filtros de qualidade com base nessa métrica aumentam a acurácia do processo de montagem dos genomas (Ewing *et al.*, 1998; Liu *et al.*, 2012).

O *basecalling,* que consiste no processo de captura dos valores de comprimento de onda gerados durante o sequenciamento. O cálculo dos valores de qualidade são executados obedecendo a fórmula Q = -10 x log10(p). Assim um valor de qualidade Phred 30 implica em um erro a cada 1000 nucleotídeos, conforme a Tabela 3 (Ewing *et al.*, 1998).

Tabela 3. Qualidade Phred relacionada com a probabilidade de erro e precisão da base

Qualidade Phred	Probabilidade de Erro	Precisão
10	1 em 10	90 %
20	1 em 100	99 %
30	1 em 1000	99,9 %
40	1 em 10000	99,99 %
50	1 em 100000	99,999 %

Quanto às plataformas NGS (Next generation sequencing), apesar de produzirem dados com uma elevada cobertura, a avaliação da qualidade de dados se torna importante, pois possibilita a eliminação de bases das extremidades cujos valores de qualidades estejam abaixo do valor de corte, o FASTQC (http://www.bioinformatics.babraham.ac.uk/) é uma aplicação que possibilita a avaliação dos dados permitindo ao usuário visualizar em um gráfico de diagramas de caixas ou boxplot, informações sobre valores de qualidade, mediana dos dados e distribuição das leituras em relação a qualidade que elas apresentam (Figura 4) . Contudo, para realizar o filtro de qualidade, pode-se utilizar ferramentas computacionais como: Galaxy (Blankenberg *et al.*, 2010), ShortRead (Morgan *et al.*, 2009), FASTX-toolkit (http://hannonlab.cshl.edu/fastx_toolkit) e o QualityAssessment (Ramos *et al.*, 2011), o que aumenta a acurácia nos dados e reduz a possibilidade de erros durante a montagem.

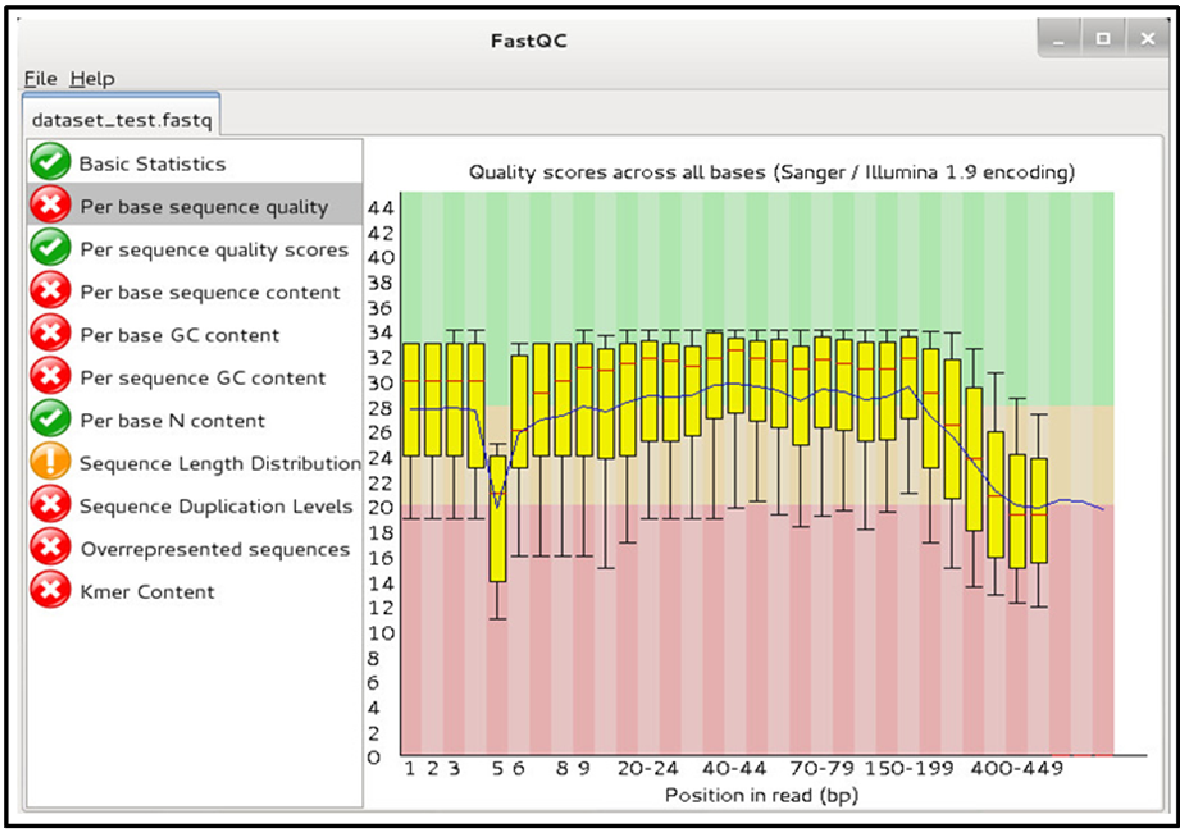

Figura 4. Tela do FastQC, a linha azul que corta o boxplot é a qualidade dos dados e a vermelha é a mediana dos dados.

1.5 Correção de Erros

Outra abordagem que objetiva o aumento da confiabilidade dos dados é a correção de erros. Ela pode ser dividida em três categorias, espectro de kmer (*k-spectrum based*), árvore de sufixos (*suffix tree/array-based*) e alinhamento de sequências múltiplas (MSA-based) (El-Metwally et al., 2013; Yang et al., 2013).

Nos últimos anos várias ferramentas computacionais para correção de erros vêm sendo desenvolvidas, implementando um alto grau de sofisticação para este processo. Dentre elas podemos citar como exemplos Quake (Kelley *et al.*, 2010), EULER-SR (Chaisson *et al.*, 2009) que utiliza o método de alinhamento espectral, SHREC (Schröder *et al.*, 2009) e a versão híbrida do SHREC (Schröder *et al.*, 2009; Salmela, 2010) que implementa árvore de sufixos e pode aceitar como entrada (Tabela 4), dados de várias plataformas de sequenciamento (Salmela, 2010).

Em programas de montagens de genomas, o processo de correção de erros é

executado em uma fase chamada de pré-processamento, no entanto, alguns realizam essa operação de forma automática, dentre estes pode-se citar ALLPATHS (Butler *et al.*, 2008) e SOAPdenovo2 (Luo *et al.*, 2012).

Tabela 4. Algoritmos de correção de erro.
Adaptado de (El-Metwally et al., 2013)

Algoritmos	Erros alvo	Input	Output
Quake	Substituição	fastq	fastq
Reptile	Substituição	fastq	fa,erros
Hammer	Substituição	fastq	Raw kmers
Shrec	Substituição	fastq	fastq
Hybrid-Shrec	Substituição Deleção Inserção	fasta	fasta
HiTEC	Substituição	Fasta,fastq	fasta

1.6 Metodologias para Montagem de Genomas

A montagem consiste no agrupamento por identidade das leituras geradas pelo sequenciamento, através da sobreposição das sequências, a fim de reconstruir o genoma original (Miller *et al.*, 2010).

Os agrupamentos das leituras geram uma leitura maior chamada de *contig* (sequência contígua), a ordenação e orientação dos contigs são realizadas na etapa de finalização, onde são construídos os *scaffolds* (também chamados de super contigs ou metacontigs) (Miller *et al.*, 2010).

Várias ferramentas computacionais foram desenvolvidas para realizar a tarefa de montagem dos dados, as primeiras a serem utilizadas eram baseadas na abordagem de algoritmos gulosos como SSAKE (Warren *et al.*, 2007), SHARCGS (Dohm *et al.*, 2007), VCAKE (Jeck *et al.*, 2007).

Como arquivo de entrada para o processo de montagem, os formatos mais utilizados são fasta, fastq e sff (Miller *et al.*, 2010).

As duas abordagens mais utilizadas atualmente para montagem de genomas são: “Reference Assembly” onde um genoma montado e disponível é utilizado como referência no processo de mapeamento das leituras. A outra abordagem é conhecida por *de novo*, que não faz uso de um genoma de referência para realizar a montagem, e exige um alto grau de processamento computacional (Pop, 2009).

1.6.1 Montagem por Referência (Reference Assembly)

Na montagem por referência também conhecida como montagem comparativa (Pop & Phillippy, 2004) é utilizado o alinhamento das sequências contra um genoma de referência (Figura 4), o qual geralmente trata-se de um organismo com características filogenéticas próximas ao organismo ao sequenciado (Pop, 2009).

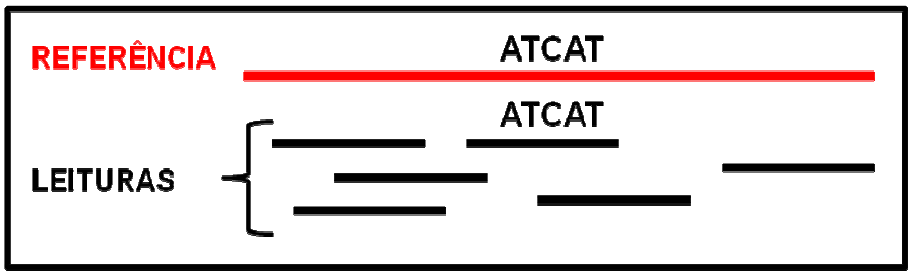

Figura 5. Figura ilustrando a montagem por referência

Esta abordagem é amplamente utilizada para a avaliação de variantes e polimorfismos de nucleotídeos únicos.

Dentre as ferramentas computacionais que utilizam essa abordagem podemos citar, SHRiMP (Rumble *et al.*, 2009), MAQ (Li *et al.*, 2008), BWA (Li *et al.*, 2008) e Bowtie (Langmead *et al.*, 2009).

1.6.2 Montagem *de novo*

A abordagem de montagem *de novo* realiza a reconstrução do genoma a partir de milhares de leituras curtas, sem nenhum genoma guia (Pop, 2009).

Alguns fatores podem dificultar o processo de montagem como: a ausência de bibliotecas pareadas, que auxiliam na resolução de regiões repetitivas, e a grande

quantidade de leituras geradas pelas plataformas NGS. A montagem *de novo* é um problema da classe "NP-hard" onde uma solução computacional eficiente não é conhecida (Pop, 2009).

A maioria das ferramentas computacionais utilizadas para montagem de dados gerados pelas plataformas de sequenciamento NGS utiliza a teoria dos grafos, a qual é amplamente utilizada em ciência da computação para solução de problemas complexos (Miller *et al.*, 2010).

Assim esses algoritmos podem ser divididos em três abordagens: algoritmos gulosos (*greedy*), overlap-layout-consensus (OLC) e Grafo *de Bruijn* (DBG) (Miller *et al.*, 2010).

1.6.2.1 Algoritmos Gulosos

Os algoritmos gulosos (*Greedy Algorithms*) são considerados os mais intuitivos na montagem de dados oriundos dos sequenciadores. O processo é iniciado com a busca por similaridade entre as leituras (Figura 5), realizando a comparação de cada sequência com todas as outras, e atribuindo valores para essas potenciais sobreposições. Após a finalização desta etapa o algoritmo realiza a fusão das leituras que possuem os maiores valores de sobreposição. Sendo assim podemos caracterizar essa estratégia pela execução uma operação básica, onde dada qualquer leitura é realizado a adição de uma nova leitura enquanto essa operação for possível (Pop *et al.*, 2002).

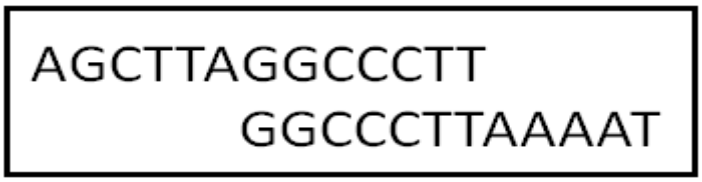

Figura 6. Similaridade entre leituras. Adaptado Pop 2009.

Dentre os softwares que utilizam essa estratégia estão o SSAKE (Warren *et al.*, 2007), VCAKE (Jeck *et al.*, 2007) e o SHARCGS (Dohm *et al.*, 2007). No entanto, essas ferramentas utilizam uma abordagem "*greedy*" melhorada, com uma nova forma de extensão de leituras. Assim uma leitura é escolhida para formação de uma "*seed"* e a partir dela os contigs são estendidos pelas sobreposições encontradas nas leituras, resultando na

diminuição de *mis-assemblies* causados pelas regiões repetitivas (Pop & Salzberg, 2008).

Outra estratégia que vem sendo utilizada para realização de montagens *de novo* é a utilização de dados de diferentes plataformas de sequenciamento. Um exemplo desse tipo de procedimento é o pipeline proposto por Goldberg *et al.*, 2006 que fez a união do montador VCAKE (Jeck *et al.*, 2007) com o Newbler (Reinhardt *et al.*, 2009), utilizando dados gerados pelas plataformas ABI 3730xl e o 454 Roche (Miller *et al.*, 2010).

1.6.2.2 Overlap-layout-consensus

Na estratégia Overlap-layout-consensus (OLC) o processo de montagem é dividida em três fases. Na primeira, é realizada a comparação de todas as leituras entre si, para auxiliar a execução desta tarefa a OLC faz uso de um algoritmo heurístico que realiza a quebra de leituras dentro de todos os k-mers existentes, o resultado desse processamento é uma lista de leituras candidatas que compartilham k-mers. O processo de descoberta das sobreposições é sensível aos valores de alguns parâmetros utilizados no algoritmo heurístico, como o valor de k-mer, tamanho mínimo de sobreposição e a porcentagem de identidade mínima exigida para uma sobreposição. A definição de valores estringentes implica em uma maior acurácia dos contigs, contudo ocorre uma influência na quantidade de contigs produzidos (Miller *et al.*, 2010).

As informações geradas na primeira fase são utilizadas na fase seguinte para gerar o grafo de sobreposições. Este grafo contém todas as informações referentes às leituras, representadas por nós (*nodes*) e a sobreposição entre duas leituras é representada como uma aresta (Pop, 2009).

Na última fase a sequência consenso é formulada a partir do alinhamento de múltiplas sequências (MSA). No passo seguinte as bases são carregadas em memória para que a consenso seja então construída após o grafo ser percorrido por um método conhecido por caminho Hamiltoniano (Miller *et al.*, 2010).

Dentre as ferramentas computacionais que utilizam essa estratégia para realizar a montagem de genomas pode-se citar: Celera (Myers *et al.*, 2000), Arachne (Batzoglou *et al.*, 2002) esses montadores são da era dos sequenciadores baseados no método de Sanger, desenvolvidos para trabalhar com leituras longas; CABOG (Miller *et al.*, 2010), que surgiu a partir do montador Celera, após uma revisão para que fosse possível trabalhar com dados gerados pela plataforma 454 Roche; Edena (Hernandez *et al.*, 2008), montador trabalha

com leituras curtas e de tamanho uniforme; Newbler (Reinhardt *et al.*, 2009), amplamente utilizado e distribuído pela Roche Life Sciences e na sua implementação ele faz uso da estratégia OLC duas vezes, na primeira vez ele gera os *unitigs* (contigs exclusivamente montáveis) que são construídos seguindo a sobreposição entre as pontes do grafo, enquanto não for encontrada uma "*fork*" (quando uma leitura A se sobrepõem a outras duas leituras B e C, no entanto as leituras B e C não se sobrepõem umas com as outras. Isso pode ocorrer devido a regiões repetitivas ou erros de sequenciamento Figura 6) (Pop, 2009).

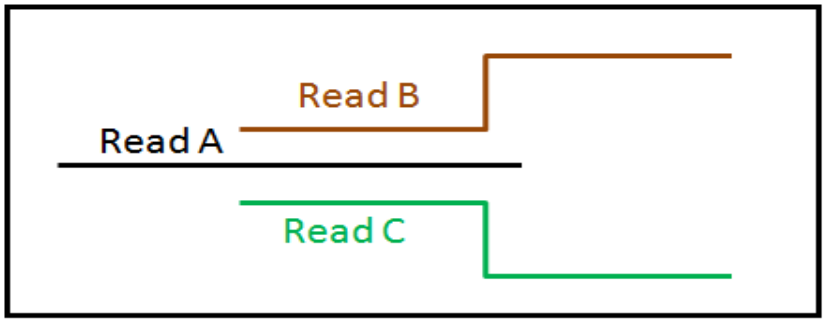

Figura 7. Surgimento do "fork" onde uma leitura A alinha com as leituras B e C, contudo B e C não alinham entre si, resultando em uma bifurcação.

1.6.2.3 Grafo *de Bruijn*

O Grafo *de Bruijn* tem sua origem nas análises dos resultados dos primeiros trabalhos com dados produzidos utilizando sequenciamento por hibridização (SBH), conceitualmente SBH é semelhante ao conjunto de fragmentos (Pevzner *et al.*, 2001). Essa abordagem inicia com a quebra de um conjunto de leituras dentro de seus k-mer spectrum (Figura 7), em seguida ocorre à construção do grafo contendo os segmentos das leituras como nós e a ligação desses nós ocorre caso eles estejam de forma adjacentes à leitura original (Pop & Salzberg, 2008).

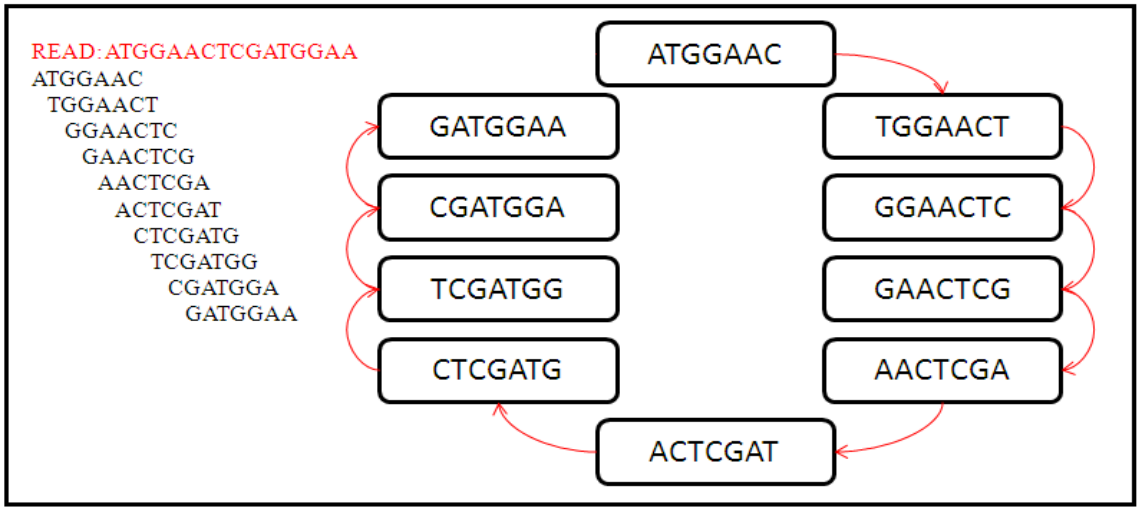

Figura 8. Demonstração do Layout das leituras e sobreposição usando kmer, ao lado a construção do grafo de Bruijn. Adaptado de Schatz *et al.*, 2010.

A abordagem do Grafo *de Bruijn* apresenta várias vantagens sobre a abordagem de OLC, dentre elas pode-se citar: há não necessidade de processamento de sobreposições entre as leituras, por isso a etapa de expansão das sobreposições não é mais necessária, em vez disso as sobreposições são implicitamente representas no grafo *de Bruijn,* existência de uma grande quantidade de algoritmos eficientes para encontrar um caminho através do grafo (Pop, 2009).

Apesar da série de vantagens citadas anteriormente, a utilização dessa abordagem em algoritmos de montagem não é trivial, em função do grande número de caminhos Eulerianos que podem ser encontrados no grafo, que correspondem a muitas maneiras diferentes que um genoma pode ser reconstruído em torno de suas regiões repetitivas. Assim a tarefa do algoritmo de montagem é encontrar apenas um caminho que corresponda à forma mais acurada de reconstrução do genoma (Miller *et al.*, 2010).

Atualmente essa estratégia é amplamente utilizada em algoritmos para montagem de dados de plataformas de sequenciamento NGS como, por exemplo, Velvet (Zerbino & Birney, 2008) software de montagem que utiliza esta abordagem; ABySS (Simpson *et al.*, 2009) tem como característica a utilização de recursos para o processamento distribuído, o que possibilita a montagem de genomas de mamíferos; ALLPATHS (Butler *et al.*, 2008) faz a montagem de genomas grandes usando bibliotecas paired-end, além disso, faz uso de procedimentos para correção de erros de substituição baseado em valores de qualidade; SOAPdenovo2 (Luo *et al.*, 2012) faz uso de várias técnicas implementadas em outros

montadores como o Velvet, CABOG e ALLPATHS (Quadro 1), normalmente utilizado em montagem de genomas grandes (Miller *et al.*, 2010).

Quadro 1. Lista de Montadores

Montadores	algoritmo	Tecnologia
SSAKE, SHARGS E VCAKE	Greedy	Illumina
Edena	OLC	Illumina
CABOG	OLC	Multi-Plataformas
SGA	OLC	Illumina
SOAPdenovo e Allpaths	De Bruijn	Illumina
Velvet	De Bruijn	Illumina
ABySS	De Bruijn	Illumina

1.7 Sistemas Distribuídos

Os sistemas distribuídos são um conjunto de componentes interligados utilizando os recursos de uma rede de computadores para realização de suas tarefas por meio de troca de mensagens (COULOURIS *et al.*, 2007).

A utilização das redes de computadores proporcionou uma melhor utilização dos recursos computacionais (compartilhamento de impressoras, arquivos e utilização de softwares), no entanto, com o avanço das tecnologias podemos contar atualmente com computadores dotados de processadores com vários núcleos de processamento com um custo reduzido, porém na maioria do tempo estes núcleos ficam ociosos ou são subutilizados com tarefas que não requerem nem um terço do que o equipamento realmente pode oferecer (COULOURIS *et al.*, 2007).

Com utilização de técnicas de sistemas distribuídos estes recursos podem ser melhor utilizados um exemplo disto, é a possibilidade do processamento de tarefas em computadores ociosos na rede, sem que o usuário necessite fazer intervenções tornando esse processo transparente (COULOURIS *et al.*, 2007).

O surgimento dessas tecnologias impulsionou a criação de várias outras como, a programação distribuída que faz uso dos recursos de redes para implementação de softwares capazes de atuar em diferentes tipos de arquitetura computacionais (Sistemas

operacionais, plataformas de Hardware). No entanto, os desafios enfrentados pelos projetistas dessas aplicações são diversos dentre eles, heterogeneidade, segurança e confiabilidade (COULOURIS *et al.*, 2007).

As relações entre os processos em um sistema distribuído usualmente ocorrem utilizando o modelo cliente-servidor, neste modelo os processos assumem papeis que podem variar de acordo com a tarefa que está sendo executada. Assim quando um processo oferece serviços, recebe a definição de servidor, contudo, em determinada situação ele pode assumir o papel de processo cliente (COULOURIS *et al.*, 2007).

Na Bioinformática existem vários algoritmos para montagem de genomas grandes que foram desenvolvidos utilizando técnicas de sistemas distribuídos dentre eles, ABySS, Allpaths, Velvet.

1.8 RMI JAVA

O RMI (*Remote Method Invocation*) Java foi incorporado a API (*Application Programming Interface*) do Java na versão 1.1 do JDK. Ele é uma extensão do modelo de objetos Java no estilo RPC (*Remote Procedure Call* – onde uma chamada de um método remoto é realizada utilizando a troca de mensagens), proporcionando suporte a objetos distribuídos utilizados na linguagem. A ideia original no desenvolvimento do RMI é proporcionar ao desenvolvedor a possibilidade de escrever códigos de objetos remotos de forma similar a escrita de um código local, tornando transparente os detalhes relacionados à comunicação em um ambiente distribuído, esta ideia está presente também no RPC, no entanto, todas as máquinas virtuais Java representam os dados da mesma maneira, não havendo a necessidade de conversão de dados. Além do mecanismo de serialização de objetos em Java que permite que objetos completos sejam passados como parâmetros de invocações remotas. A forma de trabalho do RMI é diferente do socket uma vez que no socket o endereço IP e a porta precisam ser conhecidos pelo cliente (Costa, 2008).

Esta tecnologia foi divida em camadas (Figura 8) de abstração para simplificar a implementação e a manutenção, pois, assim qualquer camada poderia ser melhorada ou trocada sem afetar o sistema como um todo. O RMI é divido em três camadas que são Stub e Skeleton, Remote Reference Layer, Camada de Transporte. Contudo, atualmente o RMI Java não faz uso do Skeleton em sua plataforma (Costa, 2008).

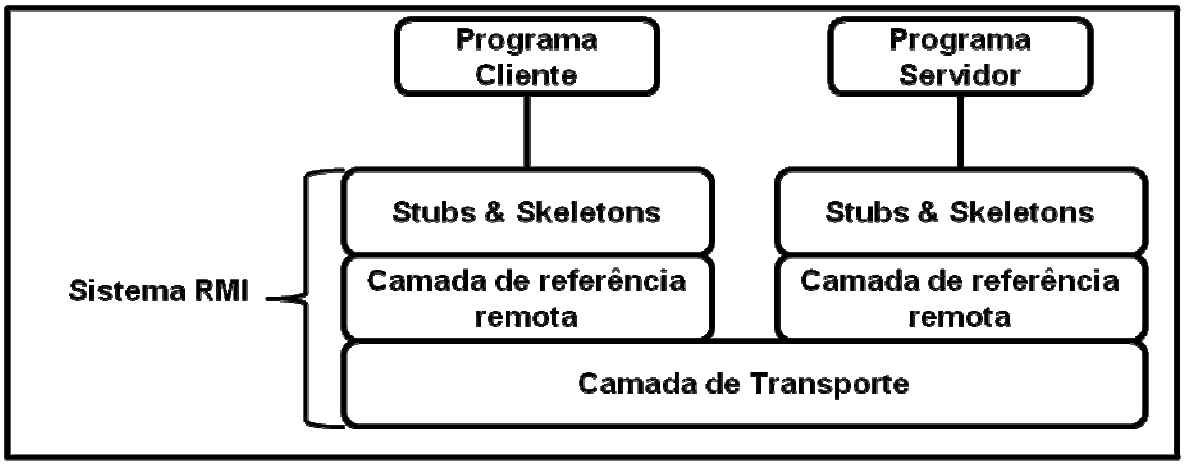

Figura 9. Arquitetura RMI Java, demonstração das camadas.

Stub e Skeleton são as camadas responsáveis por interceptar as chamadas de métodos feitas pelo cliente, fazendo com que a interface através de sua variável de referência redirecione as chamadas para o RMI remoto. O Stub é executado no lado cliente, sua função consiste na transformação dos argumentos do método de invocação em um formato que possa trafegar pela rede, este processo é chamado de serialização. No lado servidor o Skeleton realiza a montagem desses dados em um processo chamado de desserialização (Costa, 2008).

Nesta arquitetura, existe a camada de Referência Remota (*Remote Reference Layer*) é que possibilita o acesso entre a camada Stub e Skeleton e a camada de transporte adjacente, ela é responsável criação e gestão de referências a objetos remotos (Grosso, 2001).

Camada de Transporte (*Transporte Layer*) é baseada nas conexões TCP/IP entre máquinas na rede, ela é responsável por fornecer a conectividade básica assim, as solicitações dos objetos são encaminhados através desta camada pelos meios de comunicação disponíveis (rede cabeada, rede sem fio, etc.) (Grosso, 2001).

A implementação da arquitetura do RMI baseia-se na definição do comportamento e na implementação do comportamento. Assim, o RMI permite que os códigos que definem o comportamento esteja separados dos que o implementam. No RMI Java o comportamento é definido utilizando uma interface Java e a sua implementação é feita utilizando uma classe (Grosso, 2001).

A tarefa de realizar a localização do RMI remoto é feita através de um serviço de nomeação ou diretório (*naming or directory*). Este sistema funciona como uma forma de endereçamento bem definido formado composto pelo IP do host e a porta utilizada no serviço (*host:port*) (Costa, 2008).

Dentre os serviços de nomeação e diretório mais utilizados pelo RMI temos JNDI (*Java Naming and Directory Interface*) e o *RMI Registry* que é fornecido pelo próprio RMI que ocupa por padrão a porta 1099. Assim, que um serviço remoto é criado utilizando um programa servidor, um objeto que usa este serviço é criado, em seguida ele é exportado para o RMI, após isto o RMI cria o serviço que ficará aguardando as conexões do cliente (Costa, 2008).

O cliente acessa o *RMI Registry* através da classe estática *Naming* que fornece o método *lookup*( ), o qual é utilizado pelo cliente para se registrar no *RMI Registry*. A entrada deste método é uma URL que especifica o nome do servidor e o nome do serviço desejado (Grosso, 2001).

rmi://<host_name>[:port_number]/<service_name>.

1.9 Socket Java

O Socket é uma interface que foi introduzida pela Berkeley Unix, essa tecnologia foi amplamente adotada como forma de comunicação de processos com conexões a Internet. O mesmo pode ser definido como um ponto final na comunicação de aplicações que utilizam recursos de rede (Costa, 2008).

Dentre os exemplos da utilização socket podemos citar, estabelecimento de conexão entre equipamentos e o envio e recebimento de dados. A utilização desta estrutura torna transparente aos desenvolvedores detalhes de como, tipo de transmissão, tamanho de pacote e retransmissão (Costa, 2008).

Os sistemas operacionais modernos têm uma implementação da interface socket desenvolvida pela Berkeley Unix, um exemplo disto é o Windows que faz uso do socket pelo Winsock. A implementação do socket no Java é feita através das classes Socket e ServerSocket, presentes no pacote Java.net. O socket faz uso de dois modos de operação

que são, orientado a conexão que se baseia no protocolo TCP (*Transport Control Protocol*) e o não orientado conexão, empregando o protocolo UDP (*User Datagram Protocol*), a forma de operação é similar as operações executadas pelos protocolos, pois quando se utiliza a operação orientada a conexão tem se uma maior confiabilidade na transmissão dos dados (Figura 9). Assim como o protocolo TCP/IP, antes de qualquer transmissão de informação uma conexão deve ser estabelecida, só podendo ser encerrada assim que os dados forem recebidos na ordem em que forem transmitidos, aumentando o nível de segurança e a confiabilidade (Costa, 2008).

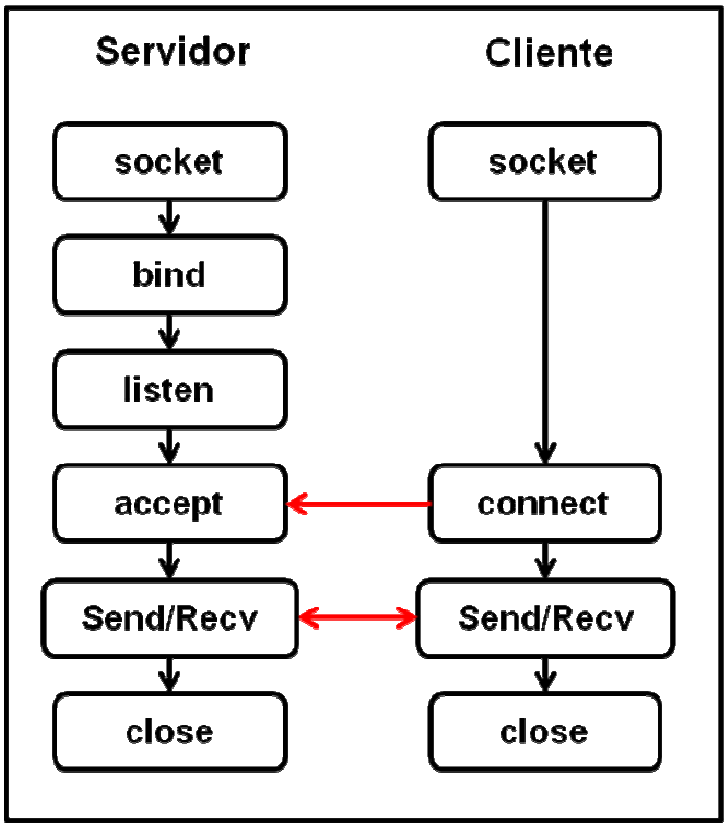

Figura 10. Utilização de sockets orientados à conexão.

- Socket ( ) - Estabelece um socket, usando uma chamada do sistema;
- Bind ( ) - Cria uma identificação constituída por endereço IP da máquina onde reside o servidor e número de porta associado à aplicação;
- Listen ( ) - Processo que fica aguardando o pedido de conexão;
- Accept ( ) - O pedido de conexão é aceito através desta chamada;
- Send/Recv ( ) - canal onde é feito o tráfego das informações;

- Close ( ) - fecha o socket criado liberando os recursos associados ao mesmo.

As operações não orientadas a conexão (Figura 10) têm como finalidade a transmissão de informação da forma mais rápida possível, não garantindo a confiabilidade na entrega dos dados, além destes poderem chegar em ordem diferente da que foram transmitidos (Costa, 2008).

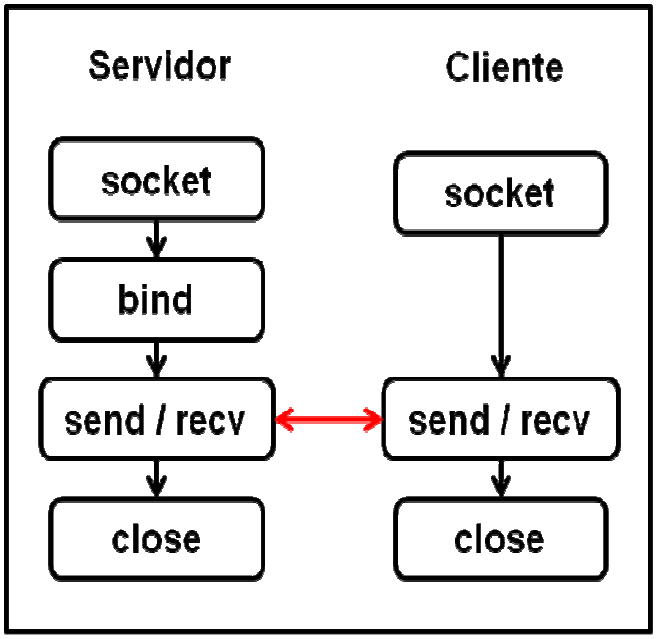

Figura 11. Utilização de sockets não orientados à conexão.

- Socket ( ) - Estabelece um socket, usando uma chamada do sistema;
- Bind ( ) - Cria uma identificação constituída por endereço IP da máquina onde reside o servidor e número de porta associado à aplicação;
- Sendto / Recvfrom ( ) - Utilizado para enviar e receber informações respectivamente;
- Close ( ) - fecha o socket criado liberando os recursos associados ao mesmo.

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver uma ferramenta gráfica para submissão e gerenciamento de montagem de genomas a partir de qualquer montador através da criação de modelo XML.

2.2 Objetivos Específicos

- Desenvolver um módulo do aplicativo para criação do arquivo modelo com as informações referentes aos montadores;
- Desenvolver um módulo do aplicativo para atualização, possibilitando a reutilização de arquivos modelo com modificações nos valores dos parâmetros;
- Desenvolver um módulo do aplicativo para leitura do arquivo modelo;
- Desenvolver um módulo do aplicativo para realizar a execução dos montadores;
- Desenvolver um módulo para gerencia de montagens remotas.

3 MATERIAIS E MÉTODOS

3.1 Implementação

Para o desenvolvimento do AutoAssemblyD foi utilizado a linguagem de programação JAVA (http://www.oracle.com), fazendo uso dos recursos de orientação objeto, o ambiente de desenvolvimento integrado (IDE) utilizado foi o NetBeans versão 7.2.1.

A interface gráfica do AutoAssemblyD foi desenvolvida utilizando a biblioteca gráfica Swing (Biblioteca que fornece um conjunto de classes destinadas a construção de interfaces gráficas), Xstream para criação do arquivo modelo e no gerenciamento das tarefas remotas foi utilizado as bibliotecas RMI e Socket disponíveis no pacote Java.net e Java.io.

3.2 Organismo modelo

Corynebacterium pseudotuberculosis linhagem 31 (Cp31) foi utilizado como modelo para a validação do processo de montagem pelo aplicativo AutoAssemblyD. Este

genoma foi seqüenciado utilizando a plataforma Ion Torrent PGM com chip 318, o tratamento de qualidade foi realizado com software Quality Assessment, o tamanho do arquivo de seqüências obtido no seqüenciamento foi de 3.9 GB (Ramos *et al.*, 2011).

3.3 Montadores

Alguns montadores foram utilizados na validação do aplicativo: Velvet (Zerbino & Birney, 2008), trata-se de um montador *de novo* que faz uso do paradigma do grafo *de Bruijn* para realizar a montagem de leituras curtas no formato *base space* ou *color space*; SOAP*denovo* (Li *et al.*, 2010) é um algoritmo adequado a montagem de genomas eucariotos, este aplicativo que utiliza a biblioteca MPI (Message Passing Interface) para realizar a distribuição da construção do grafo *de Bruijn* entre equipamentos na rede; Mira (http://www.chevreux.org/projects_main.html) utiliza overlap-layout-consensus, e as leituras podem ser curtas ou longas, além de possuir um pipeline para dados produzidos pela plataforma Ion Torrent PGM; Edena (Hernandez et al., 2008), também utilizada overlap-layout-consensus, e pode ser utilizada para dados produzidos pela plataforma Illumina, as leituras que são utilizadas como entrada deve ter o mesmo comprimento.

4 RESULTADOS E DISCUSSÃO

4.1 AutoAssemblyD

4.1.1 Dados de entrada

O AutoAssemblyD utiliza como entrada um arquivo modelo (Figura 11) gerado pela própria aplicação. A estrutura deste modelo é definida pelo padrão XML (*Extensible Markup Language*), que é uma recomendação da W3C (*World Wide Web Consortium*), a principal organização de padronização da World Wide Web**.** Ela foi inicialmente desenvolvida como uma especificação técnica para criação e troca de dados através da Internet (Cerami, 2005). Sua característica principal é criação de uma estrutura única para diversas linguagens, independente da linguagem de programação.

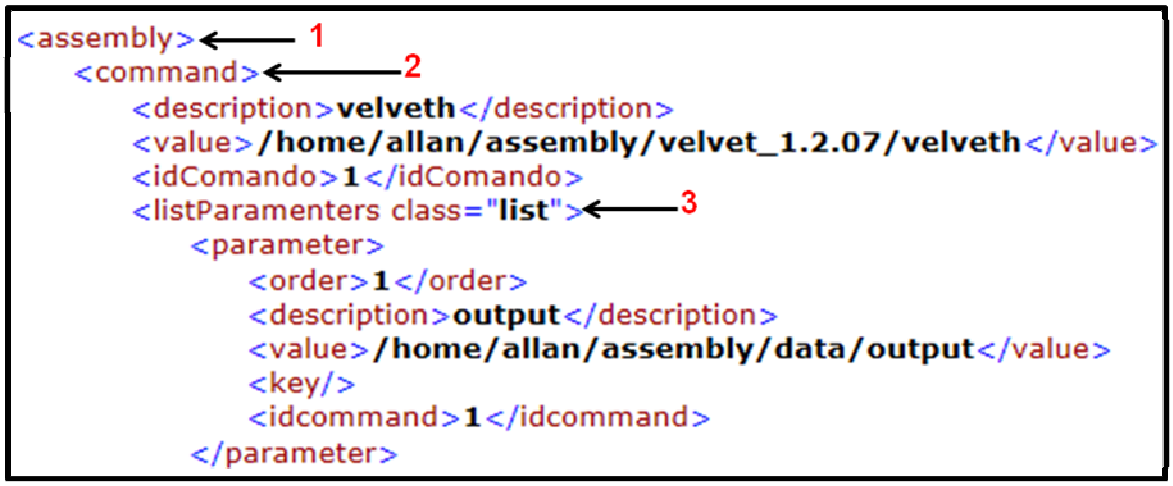

Figura 12. Trecho do template no padrão XML. Item 1 – TAG raiz do template nomeada com o mesmo nome informado para o montador, item 2 – TAG que identifica o início de um comando, item 3 – TAG que identifica a lista de parâmetros de um determinado comando.

4.1.2 Criação do arquivo modelo

Para iniciar a utilização do AutoAssemblyD é necessário a criação de um arquivo modelo, o qual será utilizado pela ferramenta na execução do montador desejado. Ao iniciar a aplicação a tela padrão que é executada após a inicialização é a Create File (Figura 12), na qual o usuário fará a inserção das informações referente ao nome do montador, comando(s) e parâmetro(s) existentes.

No intuito de auxiliar o usuário nesta tarefa a aplicação dispõe de dois quadros que possibilitam o acompanhamento dos dados à medida que são inseridos.

O primeiro quadro exibe informações sobre os comandos já inseridos, e no quadro seguinte é exibido às informações referentes aos parâmetros associados a determinado comando. Assim, quando o usuário clicar em qualquer comando será exibido todos os parâmetros já inseridos associados ao comando selecionado no quadro de parâmetros.

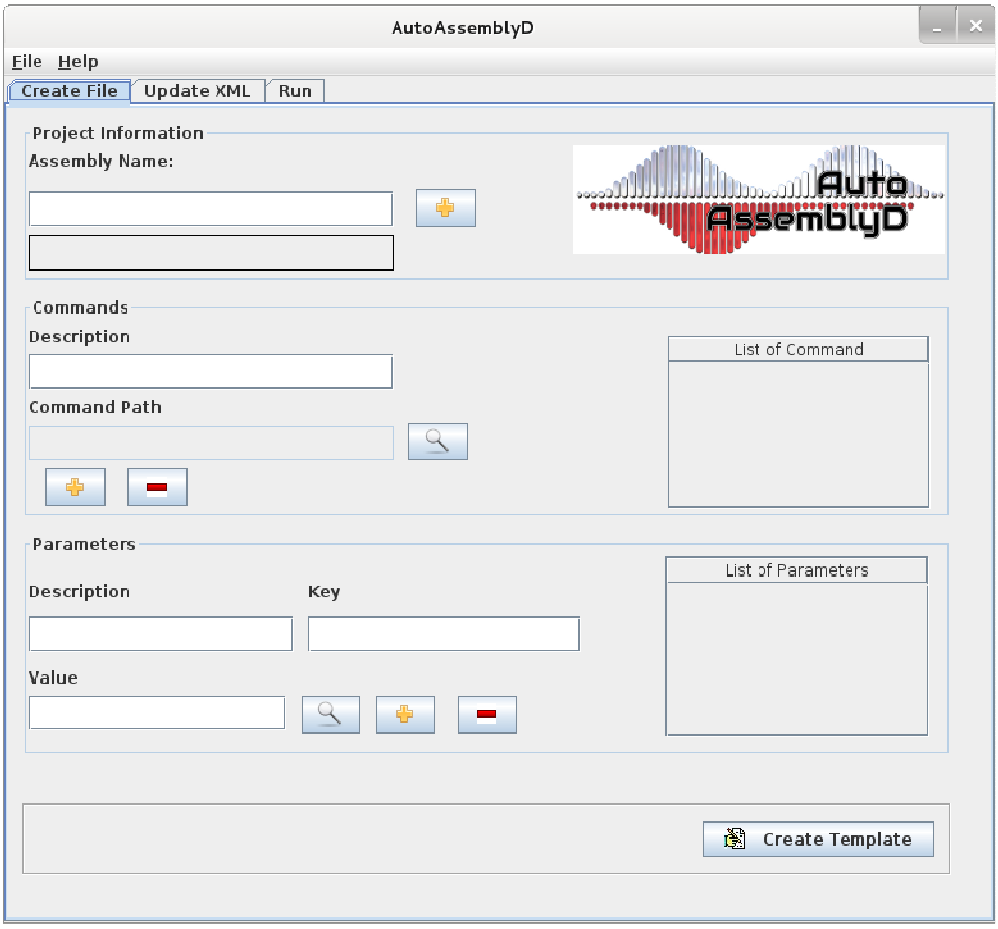

Figura 13. Tela Create File, tela responsável pela inserção dos dados dos montadores.

4.1.3 Update XML

Nesta tela (Figura 13) aos usuários tem a possibilidade realizar modificações nos valores de parâmetros de arquivos modelo existentes.

Após selecionar um comando no quadro de acompanhamento (item List of Commands) seus parâmetros serão exibidos no quadro ao lado, em seguida selecione o parâmetro desejado, o mesmo será disponibilizado no campo de edição abaixo (Value of field selected), após as devidas alterações o usuário deverá selecionar *update field.* Assim que todas as alterações tiverem sido concluídas, o usuário deverá pressionar botão *Update File*.

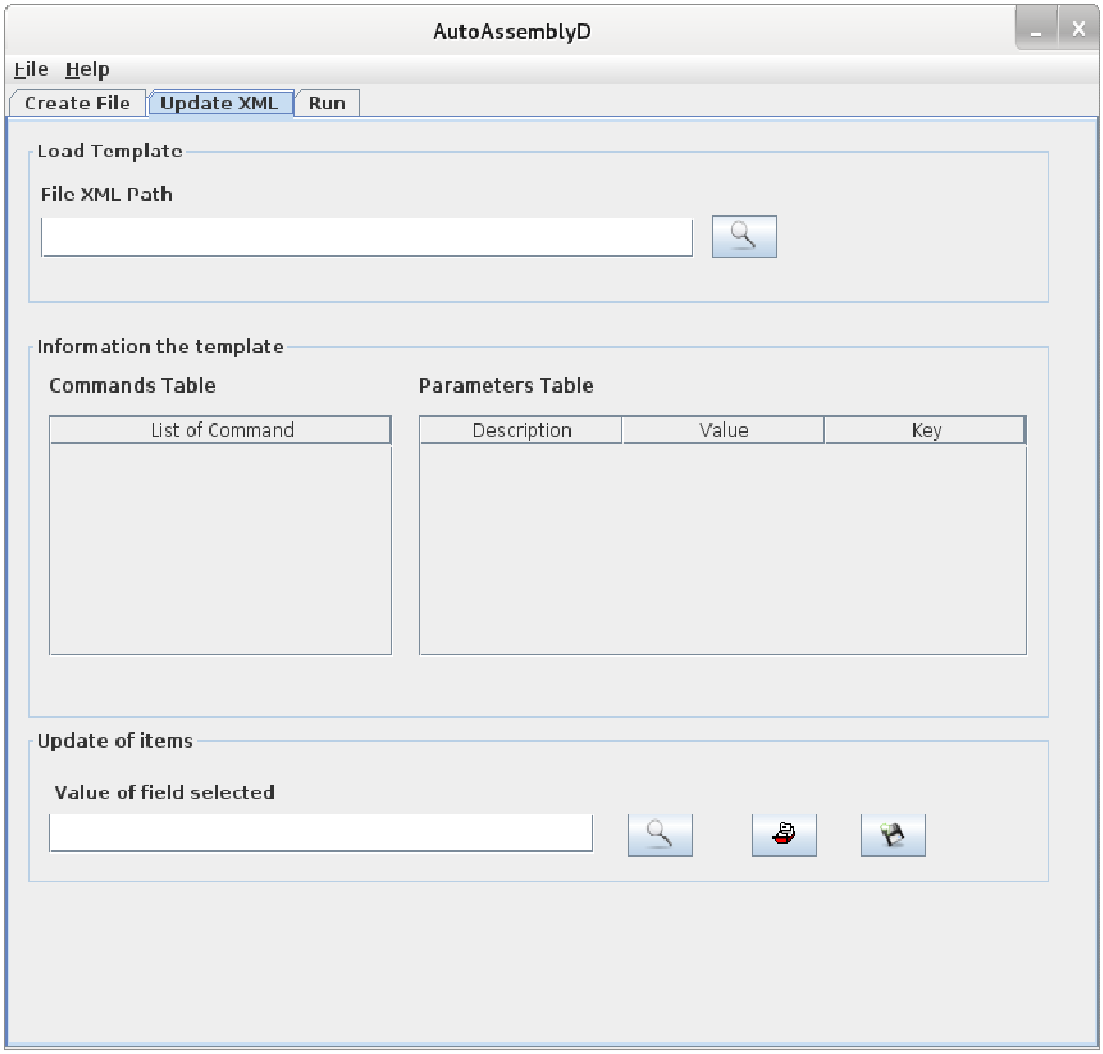

Figura 14. Tela de atualização de templates Update XML.

4.1.4 Execução do Montador

Após a etapa de preparação do modelo, será possível realizar a execução do mesmo. Na tela de execução *Run* (Figura 14) o usuário fará a carga do arquivo modelo selecionando o botão *Load*, após a carga do arquivo modelo seus comandos serão exibidos no quadro de acompanhamento (List of Commands), a visualização dos parâmetros poderá ser feita após a seleção prévia de um comando (Os valores serão exibidos no quadro ao lado – Parameters Table).

Após a conferência do arquivo modelo o mesmo poderá ser executado, contudo, o procedimento de execução pode ser realizado de duas maneiras, local ou remota.

Vale salientar que ao optar por uma execução local ou remota o equipamento o qual se deseja realizar a montagem deve obedecer aos pré-requisitos exigidos pelo montador.

No que diz respeito à execução do arquivo modelo utilizado o AutoAssemblyD local, após a conferência do arquivo modelo basta que seja pressionado o botão de Run para que o processo seja iniciado.

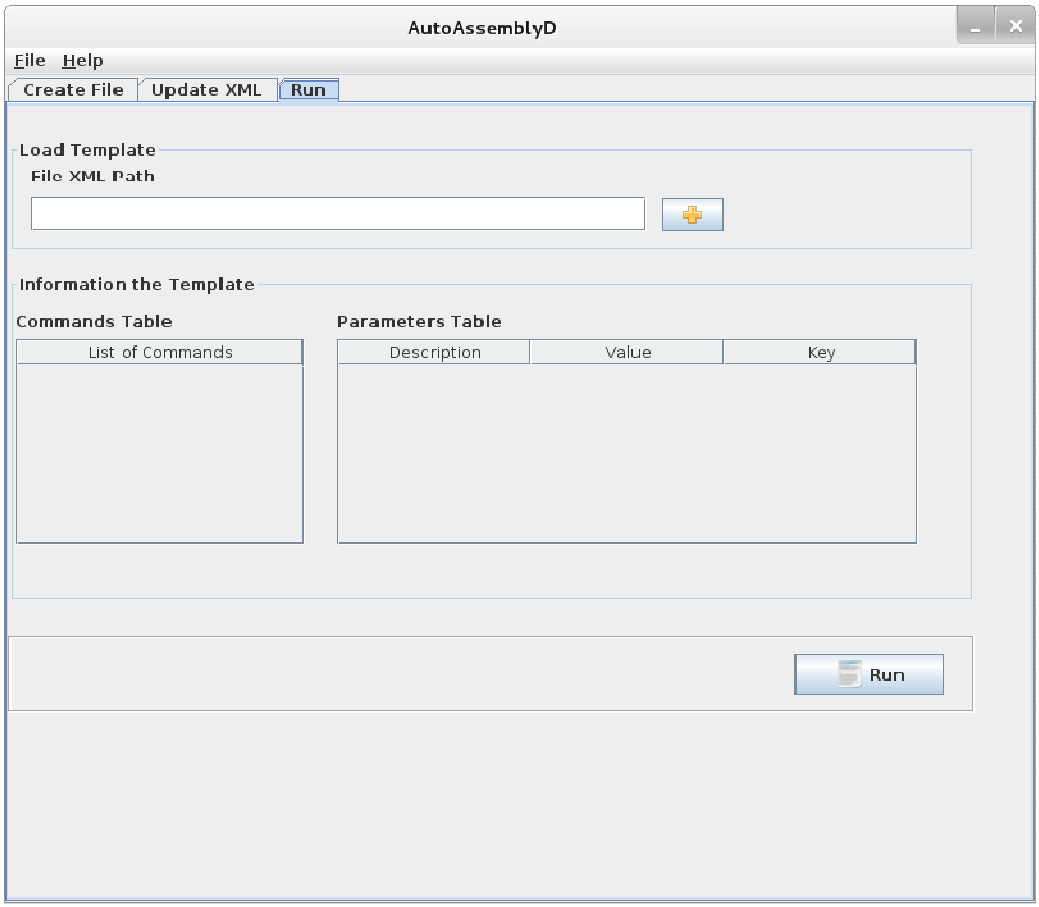

Figura 15 - Tela Run (AutoAssemblyD local).

A execução do AutoAssemblyD remoto (Figura 15) consiste no gerenciamento das montagens executadas utilizando os recursos ociosos na rede, como no item anterior neste módulo é necessário realizar a carga do arquivo modelo, no entanto, a janela apresenta uma sessão onde são exibidos todos os clientes disponíveis na rede (com o AutoAssemblyD previamente instalado). Após a seleção do cliente é necessário que seja enviado ao cliente o arquivo no padrão de compressão ZIP contendo os arquivos necessários para montagem.

Ao finalizar os requisitos o botão Remote Run deve ser pressionado para que o processo de montagem remota seja executado.

Na lista de clientes é disponibilizada uma coluna que informa ao usuário o estado

atual do cliente assim, quando o mesmo está disponível o estado demonstrará “Avaliable”, quando alguma montagem for enviada a este cliente o status é modificado para “Busy” indicando que o mesmo esta ocupado executando alguma montagem.

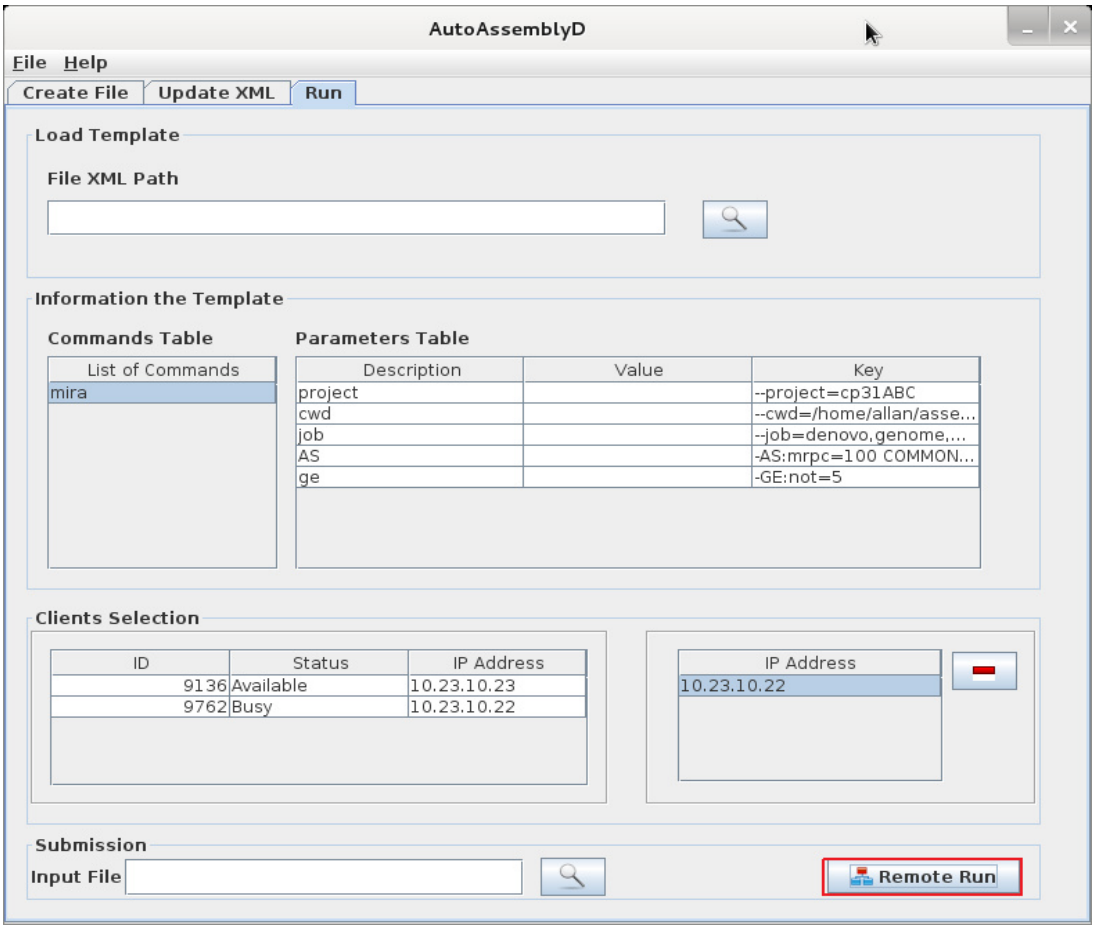

Figura 16. Execução remota do AutoAssemblyD.

5 MONTAGEM DA *CORYNEBACTERIUM PSEUDOTUBERCULOSIS*

Para validação da ferramenta foi realizado a montagem do organismo modelo através do arquivo template XML, gerado com informações do montador Mira previamente criado no AutoAssemblyD, resultando em um genoma draft da *Corynebacterium pseudotuberculosis* linhagem 31.

Os valores obtidos de N50, maior contig, menor contig, quantidade total de contigs e total de bases foram 116555 pb, 298150 bp, 45 pb, 40 , 2392285 pb respectivamente e a análise do conteúdo GC resultou em uma porcentagem de 52.07 % do genoma.

6 ANÁLISE DE TRANSFERÊNCIA DE DADOS

Uma análise do tempo de transferência de arquivos de dados com diferentes tamanhos através da rede foi realizada resultando nos dados demonstrados na Tabela 4 e no gráfico 1.

Tabela 5. Dados da Análise de transferência de dados.

Tempo gasto da transferência	Tamanho do arquivo transferido
11	0,823 GB
62	4,3 GB
183	12 GB
357	24 GB

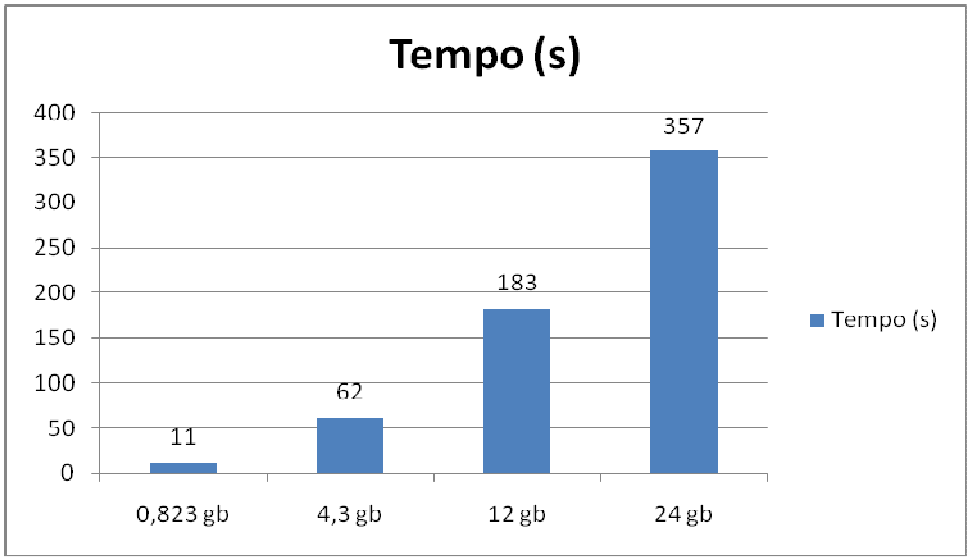

Gráfico 1. Tempo gasto versos tamanho do arquivo.

7 CONCLUSÃO

O desenvolvimento do AutoAssemblyD proporcionou uma maneira mais simples na utilização de ferramentas para montagem de genomas, tendo em vista que até o presente momento a grande maioria destes algoritmos são executados em extensas linhas de comandos no terminal do sistema operacional, dificultando ainda mais o uso destas ferramentas por usuários iniciantes.

Com o AutoAssemblyD é possível realizar a execução da tarefa de montagem utilizando uma interface gráfica intuitiva que auxilia no processo de criação do template XML com as informações sobre o montador que posteriormente é utilizado na própria aplicação. Além, da possibilidade da execução e gerência remota de processos de montagem utilizando equipamentos ociosos na rede.

Com o uso das técnicas de sistemas distribuídos o AutoAssemblyD deixa o processo de controle e gerência remota completamente transparente ao usuário, implementando maior flexibilidade e eficiência na utilização dos algoritmos de montagem de genomas.

8 REFERÊNCIAS

Batzoglou S, Jaffe D, and Stanley K (2002). ARACHNE: a whole-genome shotgun assembler. Genome …, 177–189.

Blankenberg D, Gordon A, Von Kuster G, Coraor N, Taylor J, and Nekrutenko A (2010). Manipulation of FASTQ data with Galaxy. Bioinformatics (Oxford, England), 26(14), 1783–5.

Butler J, MacCallum I, Kleber M, Shlyakhter I a, Belmonte M K, Lander E S, Nusbaum C, and Jaffe D B (2008). ALLPATHS: de novo assembly of whole-genome shotgun microreads. Genome research, 18(5), 810–20.

Cerami E (2005). XML for Bioinformatics. (I Springer Science Business Media Ed) (1ª ed., p. 304). New York: Springer-Verlag.

Chaisson M J, Brinza D, and Pevzner P A (2009). De novo fragment assembly with short mate-paired reads: Does the read length matter?, (858), 336–346.

Costa D G (2008). Java em Rede - Recursos Avançados de Programação (1ª ed., p. 324). Rio de Janeiro.

COULOURIS G, DOLLIMORE J, and KINDBERG T (2007). Sistemas Distribuídos: Conceitos e Projeto (4ª Edition., p. 780).

Dohm J C, Lottaz C, Borodina T, and Himmelbauer H (2007). SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome research, 17(11), 1697–706.

Earl D, Bradnam K, St John J, Darling A, Lin D, Fass J, Yu H O K, Buffalo V, Zerbino D R, Diekhans M, Nguyen N, Ariyaratne P N, Sung W-K, Ning Z, Haimel M, Simpson J T, Fonseca N a, Birol İ, Docking T R, Ho I Y, et al. (2011). Assemblathon 1: a competitive assessment of de novo short read assembly methods. Genome research, 21(12), 2224–41.

El-Metwally S, Hamza T, Zakaria M, and Helmy M (2013). Next-Generation Sequence Assembly: Four Stages of Data Processing and Computational Challenges. (S Markel Ed)PLoS Computational Biology, 9(12), e1003345.

Ewing B, Green P, and Spring C (1998). Base-Calling of Automated Sequencer Traces Using Phred. ?II. Error?Probabilities, 186–194.

Faino L, and Thomma B P H J (2014). Get your high-quality low-cost genome sequence. Trends in plant science, 19(5), 288–91. Elsevier Ltd.

Goldberg S M D, Johnson J, Busam D, Feldblyum T, Ferriera S, Friedman R, Halpern A, Khouri H, Kravitz S A, Lauro F M, Li K, Rogers Y, Strausberg R, Sutton G, Tallon L, Thomas T, Venter E, Frazier M, and Venter J C (2006). A Sangerpyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes.

Grosso W (2001). Java RMI (1ª ed., p. 572). O'Reilly.

Hayden E C, Technologies O N, Biology G, Island M, Macarthur D, Diego S, Systems I T, Biosciences P, Park M, and Nanopore O (2012). Nanopore genome sequencer makes its debut, 1–4.

Henson J, Tischler G, and Ning Z (2012). Next-generation sequencing and large genome assemblies. Pharmacogenomics, 13(8), 901–15.

Hernandez D, François P, Farinelli L, Osterås M, and Schrenzel J (2008). De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome research, 18(5), 802–9.

Jeck W R, Reinhardt J a, Baltrus D a, Hickenbotham M T, Magrini V, Mardis E R, Dangl J L, and Jones C D (2007). Extending assembly of short DNA sequences to handle error. Bioinformatics (Oxford, England), 23(21), 2942–4.

Kelley D R, Schatz M C, and Salzberg S L (2010). Quake: quality-aware detection and correction of sequencing errors. Genome Biology, 11(11), R116. BioMed Central Ltd.

Koren S, Treangen T J, Hill C M, Pop M, and Phillippy A M (2014). Automated ensemble assembly and validation of microbial genomes. BMC Bioinformatics, 15(1), 126.

Langmead B, Trapnell C, Pop M, and Salzberg S L (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome biology, 10(3), R25.

Lee C, Chiu Y, Wang L, Kuo Y, Chuang E Y, and Tsai M (2013). Common applications of next-generation sequencing technologies in genomic research, 2(1), 33–45.

Li H, Ruan J, and Durbin R (2008). Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome research, 18(11), 1851–8.

Li R, Zhu H, Ruan J, Qian W, and Fang X (2010). De novo assembly of human genomes with massively parallel short read sequencing. Genome …, 20(2), 265–72.

Liu L, Li Y, Li S, Hu N, He Y, Pong R, Lin D, Lu L, and Law M (2012). Comparison of next-generation sequencing systems. Journal of biomedicine & biotechnology, 2012, 251364.

Loman N J, Misra R V, Dallman T J, Constantinidou C, Gharbia S E, Wain J, and Pallen M J (2012). Performance comparison of benchtop high-throughput sequencing platforms. Nature biotechnology, 30(5), 434–9.

Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, Tang J, Wu G, Zhang H, Shi Y, Liu Y, Yu C, Wang B, Lu Y, Han C, Cheung D W, Yiu S-M, Peng S, Xiaoqian Z, Liu G, Liao X, Li Y, Yang H, Wang J, Lam T-W, and Wang J (2012). SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience, 1(1), 18.

Miller J R, Koren S, and Sutton G (2010). Assembly algorithms for next-generation sequencing data. Genomics, 95(6), 315–27. Elsevier Inc.

Morgan M, Anders S, Lawrence M, Aboyoun P, Pagès H, and Gentleman R (2009). ShortRead: a bioconductor package for input, quality assessment and exploration of high-throughput sequence data. Bioinformatics (Oxford, England), 25(19), 2607–8.

Myers E W, Sutton G G, Delcher A L, Dew I M, Fasulo D P, Flanigan M J, Kravitz S A, Mobarry C M, Reinert K H, Remington K A, Anson E L, Bolanos R A, Chou H H, Jordan C M, Halpern A L, Lonardi S, Beasley E M, Brandon R C, Chen L, Dunn P J, Lai Z, Liang Y, Nusskern D R, Zhan M, Zhang Q, Zheng X, Rubin G M, Adams M D, and Venter J C (2000). A whole-genome assembly of Drosophila. Science (New York, N.Y.), 287(5461), 2196–204.

Pevzner P a, Tang H, and Waterman M S (2001). An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences of the United States of America, 98(17), 9748–53.

Pop M (2009). Genome assembly reborn: recent computational challenges. Briefings in bioinformatics, 10(4), 354–66.

Pop M, and Phillippy A (2004). Comparative genome assembly. Briefings in …, 5(3), 237–248.

Pop M, and Salzberg S L (2008). Bioinformatics challenges of new sequencing technology. Trends in genetics : TIG, 24(3), 142–9.

Pop M, Salzberg S, Shumway M, and Feature C (2002). Genome sequence assembly: Algorithms and issues. Computer, 35(7), 47–54.

Powell D R, and Seemann T (2013). VAGUE: a graphical user interface for the Velvet assembler. Bioinformatics (Oxford, England), 29(2), 264–5.

Quail M a, Smith M, Coupland P, Otto T D, Harris S R, Connor T R, Bertoni A, Swerdlow H P, and Gu Y (2012). A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC genomics, 13(1), 341. BMC Genomics.

Ramos R T, Carneiro A R, Baumbach J, Azevedo V, Schneider M P, and Silva A (2011). Analysis of quality raw data of second generation sequencers with Quality Assessment Software. BMC research notes, 4(1), 130. BioMed Central Ltd.

Ramos R T J, Carneiro A R, Azevedo V, Schneider M P, Barh D, and Silva A (2012). Simplifier: a web tool to eliminate redundant NGS contigs. Bioinformation, 8(20), 996–9.

Reinhardt J A, Baltrus D A, Nishimura M T, Jeck W R, Jones C D, and Dangl J L (2009). De novo assembly using low-coverage short read sequence data from the rice pathogen Pseudomonas syringae pv. oryzae. Genome research, 19(2), 294–305.

Rumble S M, Lacroute P, Dalca A V, Fiume M, Sidow A, and Brudno M (2009). SHRiMP: accurate mapping of short color-space reads. PLoS computational biology, 5(5), e1000386.

Salmela L (2010). Hitseq paper, 26(10), 1284–1290.

Schatz M C, Delcher A L, and Salzberg S L (2010). Assembly of large genomes using second-generation sequencing. Genome research, 20(9), 1165–73.

Schlebusch S, and Illing N (2012). Next generation shotgun sequencing and the challenges of de novo genome assembly. South African Journal of Science, 108, 1–8.

Schröder J, Schröder H, Puglisi S J, Sinha R, and Schmidt B (2009). SHREC: a short-read error correction method. Bioinformatics (Oxford, England), 25(17), 2157–63.

Schuster (2008). Next-generation sequencing transforms today's biology, 5(1), 16–18.

Shendure J, and Ji H (2008). Next-generation DNA sequencing. Nature biotechnology, 26(10), 1135–45.

Simpson J T, Wong K, Jackman S D, Schein J E, Jones S J M, and Birol I (2009). ABySS: a parallel assembler for short read sequence data. Genome research, 19(6), 1117–23.

Warren R L, Sutton G G, Jones S J M, and Holt R a (2007). Assembling millions of short DNA sequences using SSAKE. Bioinformatics (Oxford, England), 23(4), 500–1.

Yang X, Chockalingam S P, and Aluru S (2013). A survey of error-correction methods for next-generation sequencing. Briefings in bioinformatics, 14(1), 56–66.

Zerbino D R, and Birney E (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome research, 18(5), 821–9.

Zhang J, Chiodini R, Badr A, and Zhang G (2011). The impact of next-generation sequencing on genomics. Journal of genetics and genomics = Yi chuan xue bao, 38(3), 95–109. Elsevier Limited and Science Press.

9 ANEXO I - ARTIGO CIENTÍFICO

O artigo "AutoAssemblyD: a graphical user interface system for several genome assemblers", foi publicado no dia 23 de setembro no periódico "BIOINFORMATION" (ISSN 0973-2063). BIOINFORMATION é uma revista que publica pesquisas voltadas as descoberta de conhecimento por meio de análise matemática e computacional de dados biológicos.

open access www.bioinformation.net *Software*

Volume 9(16)

AutoAssemblyD: a graphical user interface system for several genome assemblers

Adonney Allan de Oliveira Veras[1], Pablo Henrique Caracciolo Gomes de Sá[1], Vasco Azevedo[2], Artur Silva[1] & Rommel Thiago Jucá Ramos[1]*

[1]Institute of Biological Sciences, Federal University Pará, Belém, Pará, Brazil; [2]Institute of Biological Sciences, Federal University Minas Gerais, Belo Horizonte, Minas Gerais, Brazil; Rommel Thiago Jucá Ramos - Email: rommelthiago@gmail.com *Corresponding author

Received August 27, 2013; Accepted August 28, 2013; Published September 23, 2013