TECHNISCHE UNIVERSITÄT DRESDEN

CENTER FOR INFORMATION SERVICES
& HIGH PERFORMANCE COMPUTING
PROF. DR. WOLFGANG E. NAGEL

Performance Tuning & Parallelisation of Inchworm

Ankur Sharma

Dresden, April 3, 2014

Contents

List of Figures

1 Introduction

The high demand of digging deeper through the structural characteristic and behaviour of Deoxyribonucleic Acid (DNA) and Ribonucleic Acid (RNA) has driven the development of highly efficient and high-throughput sequencing, which is better known as Next Generation Sequencing (NGS). The sequencing of cDNA or the RNA sequencing is a technology that exploits the properties of next-generation sequencing to demonstrate the snapshot of RNA presence and quality from a genome at given moment of time. A variety of parallel assembly programs have been developed as an open source project as well as commercial distribution in the past few decades. Although most of these programs have been successful in assembling genomes, transcriptome assembly depicted some unique challenges in almost every phase of the program starting from the performance to the efficiency of results generated by them. *Trinity* is one of those assemblers that is highly efficient and presents a novel method for efficient and robust de novo reconstruction of transcriptomes from RNA-seq data. Careful analysis of *Trinity* revealed a lot of performance bottlenecks in the *Inchworm* phase that could be resolved in order to generate better results in a more efficient way. This document thus discusses the performance optimisations and parallel master slave approach of computing sequence assemblies deployed in *Inchworm* that boosted the performance to a great extent compared to the original stable implementation.

1.1 Biological Background

The detailed analysis of an assembly tool like *Inchworm* involves familiarity with a lot of biological terms and techniques. This section discusses some of those key biological elements that play crucial role in understanding the implementation and a deeper insight of how the changes in the assembly algorithm is effecting the results. In further sections, some of the important terms frequently used in this documentation are discussed:

1.1.1 Key Terms

Deoxyribonucleic Acid (DNA) : It is a molecule that encodes the genetic information and instructions used in the development and functioning of all living organisms and even many viruses. Hence it is a well defined and suited biological information storage system. It consists of a double stranded structure coiled around each other in an anti parallel fashion. Both these strands store the same information which is replicated when the two strands get separated.

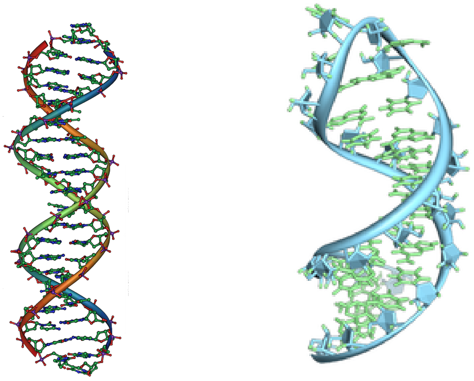

Figure 1: Double stranded DNA and single stranded RNA molecule

Ribonucleic Acid (RNA) : It is a large biological molecule that performs some essential roles in the coding, decoding, regulation and expression of genes. Unlike the DNA molecule, RNA has a single stranded structure and has a much shorter chain of nucleotides. Some RNA molecules also play an active role within cells by catalysing most of the biological reactions, controlling gene expressions, or sensing and communicating responses to cellular signals.

Transcriptomes : The transcriptomes refer to a set consisting of all the RNA molecules including mRNA, rRNA, tRNA and other non coding RNA produced in one or group of cells.

Reads : These are a short sequences that represents the DNA sequencing data and is generated by very expensive high-throughput sequencers and are typically less than 1000 base pairs in length.

k-mer : It basically refers to n-tuple of nucleic acid and amino acid sequences that can be used to identify certain regions within biomolecules like RNA. Either they as such or their statistics giving discrete probability distribution of a number of possible k-mer combinations can be used for finding regions of interest within a molecule. These sequences (string) consist of a tuple of length k with 4 different type of elements which are A (Adenine), T (Thymine), C (Cytosine) and G (Guanine).

Contig : A contig is a set of overlapping DNA segments that together represent a point of interest in a DNA molecule. In bottom-up sequencing projects, a contig refers to overlapping sequence data (reads) and in a top-down sequencing projects, they refers to the overlapping clones that form a physical map of the genome that is used to guide sequencing and assembly. Contigs can thus refer both to overlapping DNA sequence and to overlapping fragments contained in clones depending on the context.

Sequence Assembly : Sequence assembly refers to aligning and merging fragments of a much longer DNA sequence in order to reconstruct the original sequence. This is needed as DNA sequencing technology cannot read whole genomes in one go, but rather reads small pieces of between 20 and 30000 bases, depending on the technology used. Typically the short fragments, called reads, result from shotgun sequencing genomic DNA, or gene transcript.

1.2 Trinity

Trinity assembler, developed at the Broad Institute and the Hebrew University of Jerusalem, represents a method for *de novo* assembly of full-length transcripts that uses construction and analysis of sets of *de Bruijn* graph. It fully reconstructs large fraction of transcripts, including spliced isoforms and transcripts from recently duplicated genes. *Trinity* partitions the sequence data into many individual *de Bruijn* graphs, each representing the transcriptional complexity at at a given gene or locus, and then processes each graph independently to extract full-length splicing isoforms and to tease apart transcripts derived from paralogous genes.

Figure 2: Structure of trinity

1.2.1 Structure

Trinity has an excellent modular structure which distributes the entire assembly into three different pipelined phases. This specific design of *Trinity* allows us to analyse it in an efficient manner by decomposing the intermediate results of different phases and running only specific module to be tested.

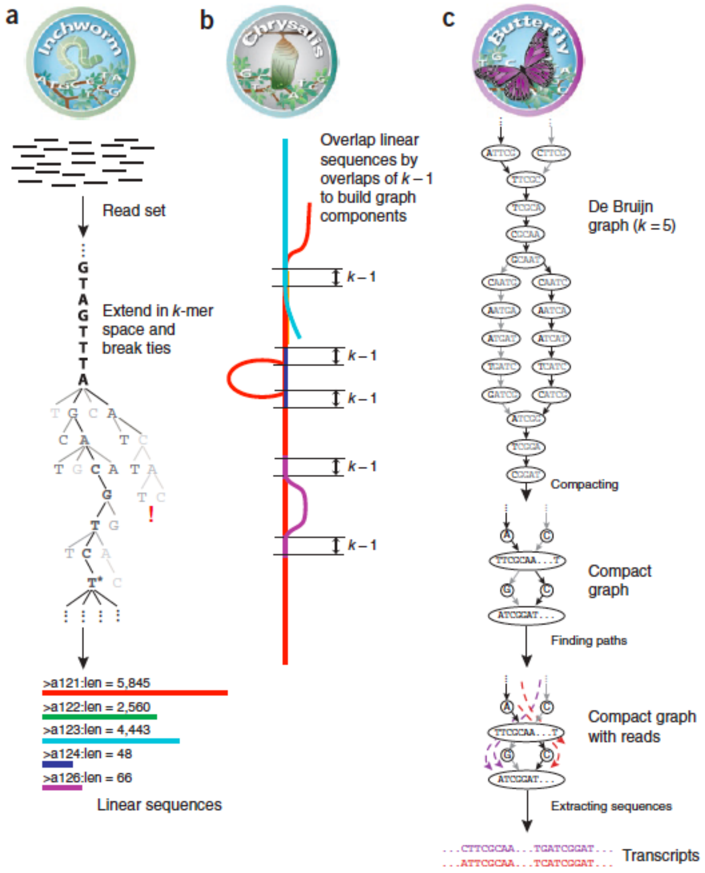

Figure 3: *Trinity* - Workflow

1.2.2 Phases

Trinity consists of three modules namely *Inchworm*, *Chrysalis* and *Butterfly*

1.3 Applications

2 Inchworm

2.1 Phases

2.1.1 Parsing

2.1.2 Sorting

2.1.3 Pruning

2.1.4 Assembling

2.2 Implementation

3 Performance Analysis

3.1 Datasets

3.2 Hardware & Software Environment

3.3 Phase Analysis of Inchworm

4 Performance Optimisation

4.1 Parsing

4.1.1 Memory Mapping

4.1.2 Observed Improvements

4.2 Pruning

4.2.1 Parallel Pruning

4.2.2 Performance Gain

4.3 Assembly

4.3.1 Master-Slave Approach

4.3.2 Benefits

4.4 Overall Improvements

5 Conclusion

5.1 Future work