Next Generation Sequencing Technologies

Julian Pierre, Jordan Taylor, Amit Upadhyay, Bhanu Rekepalli

Abstract:

The process of generating genome sequence data is constantly getting faster, cheaper, and more accurate. Unfortunately, assembling the data into a finished genome sequence is still a challenge despite our technological advances. While we have a variety of assembly tools, many of these tools differ in performance and final composition of an assembled sequence. Sequencing results in bioinformatics have shown the need for a benchmark in sequence assemblers.

Introduction:

One fundamental principle of biology is that within each cell of DNA there are genes that encode RNA which is used to produce proteins that regulate all of the biological processes within an organism. Bioinformatics is the application of computer technology to aid in the management of biological information, and is a field that encompasses different tools and techniques from three separate disciplines; molecular biology, computer science, and data analysis algorithms. Through technological advances in bioinformatics we have a better understanding of how gene sequences code specific proteins from various types of data.

Background:

To understand the process of generating genome sequence data, there first needed to be an understanding related to the field of bioinformatics and what genome sequences encompasses. Being familiar with the National Center for Biotechnology Information (NCBI) [2], Multiple Sequence Alignment (MSA) [3], and Sequence Analysis methods [4] proved helpful in the study on gene sequencing. However, a web journal titled “What is bioinformatics? An introduction and overview” [5] provided a broad array of topics related to bioinformatics, such as they types of formats used, Different databases, ways data gets organized, and different ideas for the future of bioinformatics.

To prepare to work with large amounts of data, there was a need to attend training on parallel computing. The training attended addressed some of the new features of the Beacon Many Integrated Core (MIC) architecture, as it combines many Central processing Units (CPU) into one chip providing better performance with faster computations. Although Beacon supports FORTRAN, C, and C++; some things to note is that even though you can use C++ code to call functions, you aren’t able to transfer classes, and you don’t send a function to a Message passing Interface (MPI), but have the function go to an Open Multi Processing Interface (OpenMPI) to run the code in parallel. This would allow the use of more than one core when running a program, thus taking full advantage of the parallel processing performed on Beacon.

Project:

The process of generating genome sequence data is constantly getting faster, cheaper, and more accurate thanks to the high throughput Next Generation Sequencing (NGS) machines. Sequence assembly is the merging and ordering of shorter fragments called “reads”, sampled from the larger sequence. Sequence assemblers generally take a file of short sequence reads and a quality-value file as the input [6]. The quality value reflects the how accurate an alignment is of a particular sequence. Due to the high memory requirements of high-throughput short reads from NGS machines, sequence data is always initially formatted to a specific data structure to reduce the total amount of memory used. To understand NGS assemblers we first conducted an experiment on Short Sequence Alignment (SSA) Tools.

Aligners:

Methods:

The aligners chosen for comparison were BLAST (Basic Local Alignment Search Tool) and MUMmer (Maximal Unique Matches or MUM). To get an understanding on how aligners worked, two FASTA [7] formatted files related to the same nucleotide were used. One file contained the database and the other the query sequence. Before these files can be used they needed to reformat the database to read nucleotide sequences using BLAST. BLAST achieves its speed is by using sequences in a binary format thus avoiding the overhead of parsing sequences stored in ASCII format. Once formatted the new output files to represent the database had the extension .nhr, .nin, and .nsq [8], which represented the header file, the index file, and the sequence file.

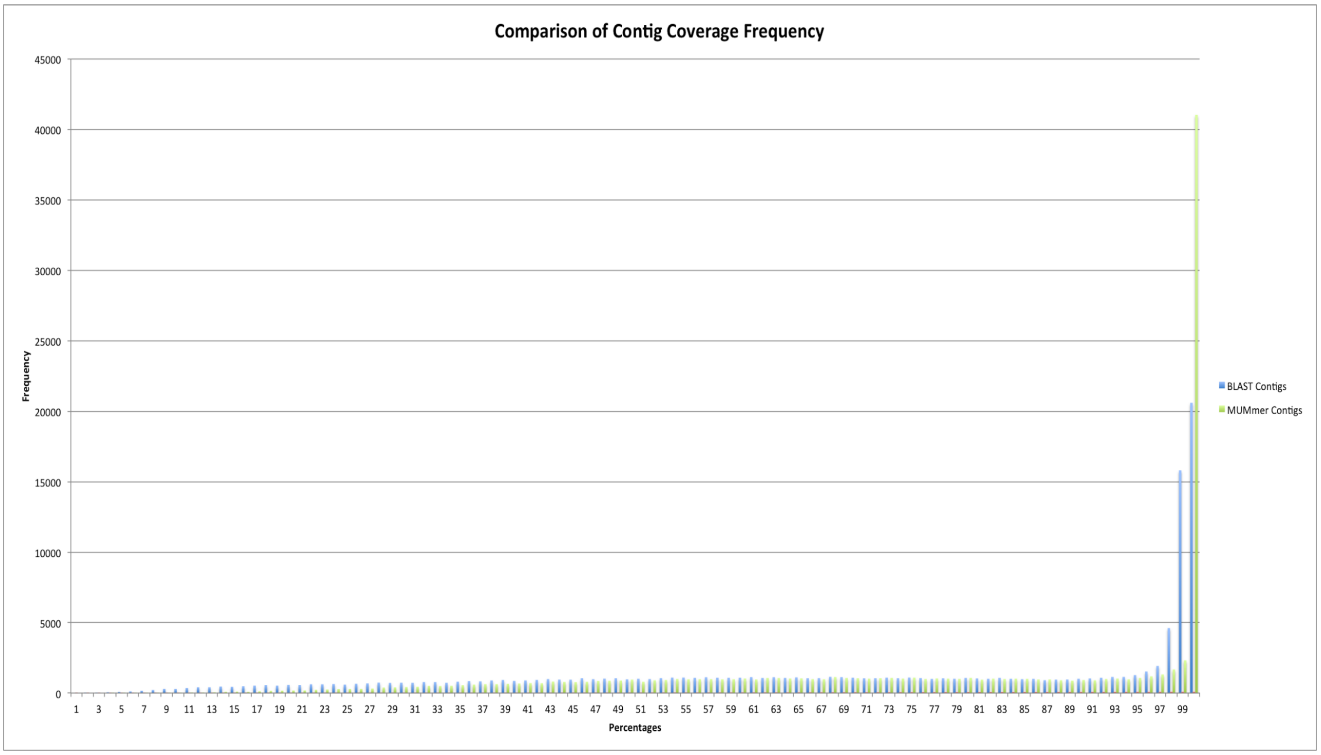
Once the file was formatted we could run the BLAST program against the database in relation to the sequence file. This returned a tabular output file that contained Node surface area of DNA, the contig [9], and their respective lengths, mismatches, gaps, and other related fields. To begin comparisons, a Python script was written to read the data and to find out how many values were returned in this file. The programming language chosen was Python as it is more of a script base language than an actual programming language. This would make it easier for a non-programmer to understand or modify to achieve the same or different results. When the file was parsed, the results showed that the particular nucleotide sequence had 552,305 reads, and produced 160,749 contigs. To see how accurate the sequence was aligned, the script was modified the script to show that the reads returned a value of 198,259 hits [10], while the contigs returned 123,070 hits.

The process for MUMmer was similar as it used the same dataset used in the BLAST testing. However, instead of using ASCII format like BLAST, MUMmer uses a suffix-tree [11] architecture which makes it an important data structure for large-scale genome analysis. The use of suffix-trees provides a faster alignment time than BLAST while the memory usage is decreased. To read the MUMmer output file, the program was modified, which returned the results that were 121,829 contig that were hit compared to blast 123,070. As these values on their own don’t signify anything, another test was initiated to compare how accurately they aligned their contig values. To accomplish this we had to compare the contig values based on its coverage. This was accomplished by first getting the lengths of each node related to a specific contig and get their lengths while taking only the difference between overlapping lengths.

Results:

The results in Fig. 1 shows the number of contigs based on that percent of coverage.

Fig. 1:



Shows frequency of contig coverage for both BLAST and MUMmer.

What this graphs shows is that the contig coverage using MUMmer is greater than that covered by BLAST on the higher percentages. As a result, even though MUMmer produced fewer contigs than BLAST, it is a more accurate Alignment tool than BLAST.

Assemblers:

Research:

For the past 29 years Sanger Sequencing [12] produced longer reads with a low error rate, but it was relatively more expensive to produce the reads. Sanger sequencing contigs, overlapping segments of DNA, were initially built using string graphs. However, due to the rapid rate of increase in competing technologies, there has been a change in the field has moved towards shorter reads at a much lower cost for a given volume of reads. Genome assemblers

generally take a file of short sequence reads and a file of quality-value as the input. Tools such as PHRED reflect the actual error rates in the aligned sequences in data providing a quality value [13]. The higher this quality value, the better the alignment. Because the quality-value file for the high throughput short reads is usually memory-intensive, first generation tools such as PHRAP (PHRagment Assembly Program), Celera, and ARCHNE, were used for numerous high quality assemblies. These tools used an overlap layout consensus approach (Zerbino and Birney 2008) and didn't follow the paradigm (Kunin *et al.* 2008) [14]. This was obscuring the intra-strain variation results found and assumes that deviations were errors than real genetic variations.

Algorithm types:

Currently there are two main formats styles: string-based implemented with the Greedy extension algorithm, called "Greedy Algorithm" [15], which are mainly reported for the assembly of small genomes [16-18],and graph-based model formatting\which are designed at handling complex genomes [19-21]. The near-identity of sequences of the Greedy Algorithm is characterized by a small positive number instead of a large one. In other words an alignment is assessed by counting the number of its differences (i.e. the columns that do not align identical nucleotides). The distance between strings and is then defined as the minimum number of differences in any alignment of those strings. Greedy alignment algorithms work directly with a measurement which is the difference between two sequences, rather than their similarity.

While the graph-based approaches are generally superior in terms of assembly quality, the computer resources required for building and storing a large graph is very high. One primary area of concern is how to process repetitive fragments from complicated genome through the assembly of next-generation short reads. Due to the length of some sequences, Paired-end (PE) sequencing can compensate for the read lengths. PE sequencing reads both the forward and reverse template strands of each cluster during one Paired end read, and both reads contain long range positional information allowing for highly precise alignment of reads. One possible solution to this problem can be to use longer reads, but currently that isn't possible with our current technology [22].

Assemblers, such as SSAKE [23], SOAPdenovo [20], AbySS [21], and Velvet [24] exploit PE sequencing information to reduce gaps, an insertion or deletion within the input sequence alignment as missing data [25], from assembled contigs. For string-based assemblers the time and memory cost is proportionate to the dataset size. SSAKE, a string based assembler, runs in less time than other peer assemblers, but the RAM (or memory) usage increases dramatically as the dataset size increases. In comparison graph based Short-read Sequencing tools such as Velvet, SOAPdenovo, ALLPATHS [26], and ABySS implement assembly task with fairly little computational power, and are more suited for large datasets which use the De Bruijn graph method [27]. In this method a certain proportion of base errors are incorporated into contigs during construction of the graph with k-mers(DNA 'words' of length *k*) generated from the input, thus creating a series of overlapping reads and represents a candidate Hamiltonian cycle assembly reading each series of k-mers only once.

NGS Related Issues:

Because adjacent reads usually overlap, data loss is a primary concern when using Short-read Sequencing tools. When the data is read by the assembler some base pairs are lost either by being discarded as mistakes or repeat sequences, or by being joined in the wrong place or orientation [28]. In fact it is now also recognized that short reads have made the assembly problem significantly harder due to the complexity involved in resolving long repeats. To solve this problem, the assembler tools used are based on the assumptions that if two reads share a sufficiently long subsequence then they belong to the same location in the genome.

Currently, there are more than 20 different assemblers, and these assemblers have been designed to mitigate the complexity of assembling Next Generation Sequence (NGS) reads. The issue with that many different assemblers is that there is no single computational method that is accepted as the best way to find similarities between genomes of different species. This raised the concern for a computational benchmark for assemblers. To achieve this benchmark, competitions such as the Assemblathon and Assemblathon 2 were conducted to compare methods of assembling full genomes from the short segments of genetic information produced by genetic sequencing technologies. Unfortunately the results from both these competitions showed that large differences exist between the assemblies, and that there are inconsistencies when using the same assembler (i.e. two groups could run the same program and get different results).

Conclusion:

In summary, MUMmer is a more accurate alignment tool than BLAST and there are numerous sequencing problems caused by the new sequencing methods of Next Generation Sequencing. These sequencing issues may be fixed by taking longer sequences instead of a large amount of short sequences. Due to the different parameters related to different assemblers, you should not depend on a single metric, and choose assemblers that excel in a specific area of interest.

References:

1. Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter *Molecular Biology of the Cell*, 2007, ISBN 978-0-8153-4105-5
2. NCBI - develops new information technologies to aid in the understanding of fundamental molecular and genetic process that control health and disease
3. MSA -the Alignment of three of more biological sequences usually DNA, proteins, or RNA

peptide sequence - the order of which amino acids connect to peptide bonds

4. N.M.Luscombe, D. Greenbaum, and M. Gerstein(2001) Yearbook of Medical Informatics *What is bioinformatics An introduction and overview*
5. Sensen, C. W. "Sequenceing Terminology." *Essentials of Genomics and Bioinformatics*. Weinheim: Wiley-VCH, 2002. N. pag. Print
6. FASTA - is a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes zhanglab.ccmb.med.umich.edu/FASTA/
7. Farrar, Michael S. "NCBI BLAST Database Format." HHMI, Janelia Farm Research Campus, Mar.-Apr. 2010. Web.
8. Contig - is a set of overlapping DNA segments that together represent a region of DNA.
9. Staden,R (1980) A new computer method for the storage and manipulation of DNA gel reading data", Nucleic Acids Res. 8, 3673-3694
10. hits - has an area of coverage
11. Suffix-tree - is a data structure for representing all the substrings of a string, whether that string is a DNA sequence, a protein sequence, or plain text.

Kurtz, Stefan, Adam Phillippy, Arthur L. Delcher, and And Others. "Versatile and Open Software for Comparing Large Genomes." *Genome Biology*. N.p., 30 Jan. 2004. Web.

12. Earl D, Bradnam K, John JS, Darling A, Lin D, et al. (2011) Assemblathon 1: A competitive assessment of de novo short read assembly methods.Genome Res 21: 2224–2241.

13. Sensen, C. W. "Sequenceing Terminology." *Essentials of Genomics and Bioinformatics*. Weinheim: Wiley-VCH, 2002. N. pag. Print

14. Timothy Graham Amos "Metagenomics" From Organism Diversity to Micro-heterogeneity: Confident Assessment of Fine-scale Variation within Metagenomic Data: Thesis 2011

15. Zheng Z. Scott S. Lukas W. and Webb M. (2000)"A Greedy Algorithm for Aligning DNA Sequences" JOURNAL OF COMPUTATIONAL BIOLOGY Volume 7, Numbers 1/2, 2000 Mary Ann Liebert, Inc. Pp. 203–214

16. Dohm JC, Lottaz C, Borodina T, Himmelbauer H (2007) SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res 17: 1697–1706. doi: 10.1101/gr.6435207.

17. Bryant DW Jr, Wong WK, Mockler TC (2009) QSRA: a quality-value guided de novo short read assembler. BMC Bioinformatics 10: 69. doi: 10.1186/1471-2105-10-69.

18. Jeck WR, Reinhardt JA, Baltrus DA, Hickenbotham MT, Magrini V, et al. (2007) Extending assembly of short DNA sequences to handle error. Bioinformatics 23: 2942–2944. doi: 10.1093/bioinformatics/btm451.

19. Li R, Fan W, Tian G, Zhu H, He L, et al. (2010) The sequence and de novo assembly of the giant panda genome. Nature 463: 311–317.

20. Li R, Zhu H, Ruan J, Qian W, Fang X, et al. (2010) De novo assembly of human genomes with massively parallel short read sequencing. Genome Res 20: 265–272. doi: 10.1101/gr.097261.109.

21. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, et al. (2009) ABySS: a parallel assembler for short read sequence data. Genome Res 19: 1117–1123. doi: 10.1101/gr.089532.108.

22. Zhang W, Chen J, Yang Y, Tang Y, Shang J, et al. (2011) A Practical Comparison of De Novo Genome Assembly Software Tools for Next-Generation Sequencing Technologies. PLoS ONE 6(3): e17915. doi:10.1371/journal.pone.0017915

23. Warren RL, Sutton GG, Jones SJ, Holt RA (2007) Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23: 500–501. doi: 10.1093/bioinformatics/btl629.

24. Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 18: 821–829. doi: 10.1101/gr.074492.107.

25. Steve.E, Tandy W. (2011) Phylogenetic analyses of alignments with gaps. Tech Report 807: statistics.berkeley.edu

26. Butler, J. et al. Genome Res. 18, 810–820 (2008)

27. Phillip C, Pavel P, and Glenn T (2011) Nature Biotechnology 29, 987–991 doi:10.1038/nbt.2023 Published online 08 November 2011

28. Monya Baker (2012) Nature Methods "De novo denome assembly: what every biologist should know" doi: 10.1038/nmeth.1935