Himmelfarb Health Sciences Library, The George Washington University
Health Sciences Research Commons

Biochemistry and Molecular Medicine Faculty Publications

Biochemistry and Molecular Medicine

2015

Advantages of distributed and parallel algorithms that leverage Cloud Computing platforms for large-scale genome assembly

Phil Kumari

Raja Mazumder
George Washington University

Vahan Simonyan

Konstantinos Krampis

Recommended Citation

Kumari, P., Mazumder, R., Simonyan, V., Krampis, K. (2015). Advantages of distributed and parallel algorithms that leverage Cloud Computing platforms for large-scale genome assembly. F1000 Research, 4:20.

RESEARCH NOTE

Advantages of distributed and parallel algorithms that leverage Cloud Computing platforms for large-scale genome assembly. [v1; ref status: not approved 2, http://f1000r.es/4yw]

Priti Kumari[1], Raja Mazumder[2], Vahan Simonyan[3], Konstantinos Krampis[4]

[1]Institute for Genome Sciences, University of Maryland School of Medicine, Baltimore, MD 21201, USA
[2]Department of Biochemistry and Molecular Biology, George Washington University Medical Center, Washington, DC 20037, USA
[3]Center for Biologics Evaluation and Research, US Food and Drug Administration, Rockville, MD 20852, USA
[4]Biological Sciences Department, Hunter College, City University of New York, New York, NY 10023, USA

v1 **First published:** 22 Jan 2015, **4**:20 (doi: 10.12688/f1000research.6016.1)
Latest published: 22 Jan 2015, **4**:20 (doi: 10.12688/f1000research.6016.1)

Abstract

Background: The transition to Next Generation sequencing (NGS) sequencing technologies has had numerous applications in Plant, Microbial and Human genomics during the past decade. However, NGS sequencing trades high read throughput for shorter read length, increasing the difficulty for genome assembly. This research presents a comparison of traditional versus Cloud computing-based genome assembly software, using as examples the Velvet and Contrail assemblers and reads from the genome sequence of the zebrafish (*Danio rerio*) model organism.
Results: The first phase of the analysis involved a subset of the zebrafish data set (2X coverage) and best results were obtained using K-mer size of 65, while it was observed that Velvet takes less time than Contrail to complete the assembly. In the next phase, genome assembly was attempted using the full dataset of read coverage 192x and while Velvet failed to complete on a 256GB memory compute server, Contrail completed but required 240hours of computation.
Conclusion: This research concludes that for deciding on which assembler software to use, the size of the dataset and available computing hardware should be taken into consideration. For a relatively small sequencing dataset, such as microbial or small eukaryotic genome, the Velvet assembler is a good option. However, for larger datasets Velvet requires large-memory compute servers in the order of 1000GB or more. On the other hand, Contrail is implemented using Hadoop, which performs the assembly in parallel across nodes of a compute cluster. Furthermore, Hadoop clusters can be rented on-demand from Cloud computing providers, and therefore Contrail can provide a simple and cost effective way for genome assembly of data generated at laboratories that lack the infrastructure or funds to build their own clusters.

Open Peer Review

Referee Status: ☒ ☒

	Invited Referees 1	 2
version 1 published 22 Jan 2015	☒ report	☒ report

1 **Keith E Robison**, Warp Drive Bio USA

2 **Surya Saha**, Boyce Thompson Institute for Plant Research USA

Discuss this article

Comments (0)

Corresponding author: Konstantinos Krampis (agbiotec@gmail.com)

How to cite this article: Kumari P, Mazumder R, Simonyan V and Krampis K. **Advantages of distributed and parallel algorithms that leverage Cloud Computing platforms for large-scale genome assembly. [v1; ref status: not approved 2, http://f1000r.es/4yw]** *F1000Research* 2015, **4**:20 (doi: 10.12688/f1000research.6016.1)

Grant information: This project has been funded in whole or part with federal funds from the National Institute of Allergy and Infectious Dis-eases, National Institutes of Health, Department of Health and Human Services under contract numbers N01-AI30071 and/or HHSN272200900007C.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: No competing interests were disclosed.

First published: 22 Jan 2015, **4**:20 (doi: 10.12688/f1000research.6016.1)

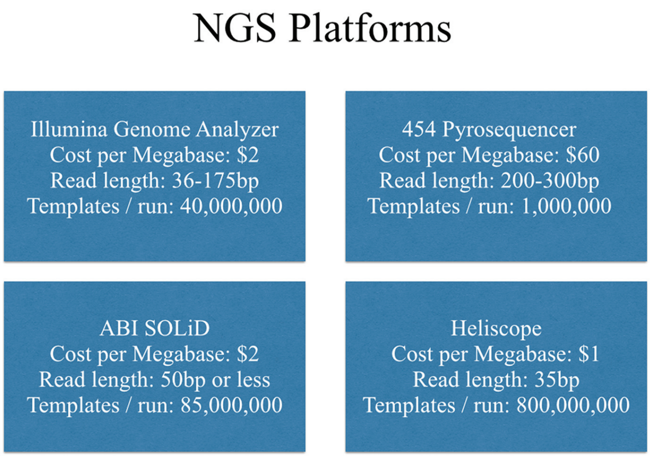
Background

Genome sequencing technologies and platforms

The earliest landmark in genome sequencing is that of the Bacteriophage MS2[1] between 1972 and 1976. Following that, two DNA sequencing techniques for longer DNA molecules were invented, first the Maxam-Gilbert[2] (chemical cleavage) method and then the Sanger[3] (or dideoxy) method. The Maxam-Gilbert method was based on nucleotide-specific cleavage by chemicals, and is best applied to oligonucleotides that and short sequences usually smaller than 50 base-pairs in length. On the other hand, the Sanger method was more widely used because it leveraged the Polymerase Chain Reaction (PCR) for automation of the technique, which also allows to sequence long strands of DNA including entire genes. In more detail, the Sanger technique is based on chain termination by dideoxynucleotides triphosphate (ddNTPs) during PCR elongation reactions (review in[4]). Although the automated Sanger method had dominated the industry for almost two decades, with sequencing applications and broad demand for the technology in genome variation studies, comparative genomics, evolution, forensics, diagnostic and applied therapeutics, it was still limiting due to its high cost and labor intensive process[5].

The high demand for low-cost sequencing led to the development of high-throughput technologies also known as Next Generation Sequencing (NGS), that parallelize the sequencing process and lower the cost per sequenced DNA base-pair. NGS techniques achieve this by automating template preparation and using high-speed, precision fluorescence imaging, for highly parallel identification of the nucleotides. NGS technologies involve sequencing of a dense array of DNA fragments by iterative cycles of enzymatic manipulation and imaging-based data collection. The major NGS platforms are Roche/454FLX (http://454.com/), Illumina/Solexa Genome Analyzer (http://www.illumina.com/systems/genome_analyzer_iix.ilmn), Applied Biosystems SOLiD system (http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing.html), Helicos Heliscope (http://www.helicosbio.com/Products/HelicosregGeneticAnalysisSystem/HeliScopetradeSequencer/tabid/87/Default.aspx) and Pacific Biosciences (http://www.pacificbiosciences.com/). Summary statistics for the throughput and characteristics of these NGS platforms are shown Figure 1, while additional review studies are available in the literature[6–8].

Although these platforms are quite diverse in sequencing biochemistry as well as in how the array is generated, their workflows are conceptually similar. First, a library is prepared by fragmenting PCR-amplified DNA randomly, followed by *in vitro* ligation to a common adaptor, that is small DNA oligonucleotide with known sequence. In the next step, clustered amplicons that are multiple copies of a single fragment are generated and serve as the sequencing fragments. This can be achieved by several approaches, including *in situ* polonies[9], emulsion[10] or bridge PCR[11,12]. Common to these methods is that PCR amplicons derived from any given single library molecule end up spatially clustered, either to a single location on a planar substrate (*in situ* polonies, bridge PCR), or to the surface of micron-scale beads, which can be recovered and arrayed (emulsion PCR).



NGS Platforms

Illumina Genome Analyzer
Cost per Megabase: $2
Read length: 36-175bp
Templates / run: 40,000,000

454 Pyrosequencer
Cost per Megabase: $60
Read length: 200-300bp
Templates / run: 1,000,000

ABI SOLiD
Cost per Megabase: $2
Read length: 50bp or less
Templates / run: 85,000,000

Heliscope
Cost per Megabase: $1
Read length: 35bp
Templates / run: 800,000,000

Figure 1. Comparison of various NGS Platforms.

The advantage of the NGS platforms is sequencing using single, amplified DNA fragments, avoiding the need for cloning required by the Sanger method. This also makes the technology applicable to genomes of un-cultivated microorganism populations, such as for example in metagenomics. A disadvantage of the new technology is that sequence data is in the form of short reads, presenting a challenge to developers of software and genome assembly algorithms. More specifically, it can be difficult to correctly assign reads and separate between genomic regions that contain sequence repeats even if using high-stringency alignments, especially if the lengths of the repeats are longer than the reads. In addition, repeat resolution and read alignments are further complicated by sequencing error, and therefore genome assembly software must tolerate imperfect sequence alignments. A reduced alignment stringency can return many false positives that results in chimeric assemblies where distant regions of the genome are mistakenly assembled together. The limitation of short reads is compensated by multiple overlaps of the sequenced DNA fragments, resulting in many times coverage of the genome sequence. Finally, genome assembly is hindered by regions that have nucleotide composition for which PCR does not achieve its optimum yield, resulting in non-uniform genome coverage that in turn leads to gaps in the assembly.

To alleviate some of these problems in genome assemblies with short reads, "paired-end" reads are used that are generated by a simple modification to the standard sequencing template preparation, in order to get the forward and reverse strands at the two ends of a DNA fragment. The unique paired-end sequencing protocol allows the user to choose the length of the insert (200–500 bp) and use the positional and distance information of the reads for validating the genome assembly, allowing for resolution of alignment ambiguities and chimeric assemblies.

Genome assembly algorithms and software

Two approaches widely used in NGS genome assemblers are the Overlap/Layout/Consensus (OLC)[13] and the *de Bruijn* Graph (DBG)[14] method, both using K-mers as the basis for read alignment. A graph is an abstraction used in computer science, and is composed of "nodes" connected by "edges". If the edges connecting the nodes can be traversed in one direction, the graph is a directed graph, whereas if the edges can be traversed in both the directions the graph is bi-directed[15].

The OLC approach for genome assembly is the typical method for Sanger datasets, and is implemented in software such as Celera Assembler, PCAP and ARACHNE[16–18]. With this approach reads are represented as nodes in the graph, and nodes for overlapping reads are connected by an edge. In more detail, OLC assembles proceed in three phases: in the first phase overlap discovery involves all-against-all comparison, pair-wise read alignment. The algorithm used for that purpose is a seed and extend heuristic algorithm that pre-computes K-mers from all reads, then selects overlap candidate reads that share K-mers and calculates alignments using the K-mers as alignment seeds. This step of the algorithm is sensitive to settings of K-mer size, minimum overlap length and percent identity required to retain an overlap as true positive, in addition to base calling errors and low-coverage sequencing. Using the results from the read alignment, an overlap graph is constructed that provides an approximate read layout. The overlap graph does not include the sequence base calls rather than just the overlapping read identifiers, so large-genome graphs may fit into practical amounts of computer memory. The graph also has metadata on the nodes and edges, in order to distinguish the lengths, 5′ and 3′ ends, forward and reverse complements, and the type of overlap between the reads. In the second phase, the precise layout and the consensus sequence of the graph are determined by performing Multiple Sequence Alignment (MSA)[19]. For that purpose, the algorithm must load all the sequences into memory, and this becomes one of the most computationally demanding stages of the assembly. Finally, in the third phase the assembly algorithm follows a Hamiltonian path[20] to "walk" through the graph visiting every node only once, and the contigs formed by merging the overlapping reads are determined.

The second approach in genome assembly is based on the *de Brujn* Graph (DBG) algorithms, and example software using this approach includes Velvet, Contrail, ALLPATHS and SOAPdenovo[21–24]. In the DBG approach each node in the graph corresponds to a unique K-mer present in the set of reads, and a directed edge connects two nodes labeled 'a' and 'b', if the k – 1 length suffix of 'a' is the same as the k – 1 length prefix of 'b'. For example, a graph node representing the K-mer (K=3) ATG will have an edge with the node representing TGG. The DBG algorithms are more efficient for large datasets, as they doe not require all-against-all read alignment and overlap discovery, while it also doe not store individual reads or their overlaps in the computer memory. A *de Brujn* graph can either be uni-directed or bi- directed, with a single edge connecting two nodes, or edges with two directions for the 5' or 3' genome strands. A bidirected graph has the advantage of allowing simultaneous assembly of sequence reads from both strands of the genome[15]. The DBG assembly algorithms traverse the graph using an Eularian path[14], where each edge is visited exactly once.

Methods

Assembly dataset: The sequence reads dataset were Illumina reads from the fish species *M.* zebrafish downloaded from Assemblathon database (http://assemblathon.org). The genome was approximately 1GB in size, the total library coverage estimate for this dataset was 197X. The read length for the given dataset was 101 and in the first phase 38,364,464 reads with coverage of only 2X, both paired and unpaired, totaling 2,762,241,408 base pairs were used (filtered for bad quality reads). Following that, different variations in the size of dataset used for running the assembly involving 1/4, 1/2 and the entire dataset of fish genome. The dataset consisted of pairs of files with each corresponding to the paired-end reads. *De novo* assembly was performed using two different assemblers, Velvet and Contrail, both using the graph-theoretic framework of De-Brujn Graph. Velvet requires high performance computer servers with large RAM memory, whereas Contrail is a distributed memory assembler that performs the computation in parallel over several servers on a computer cluster.

Assembly statistics: The output of the assembly was a file with a list of contigs of various lengths. The file contains contigs along with various details like length and coverage. For the Velvet assembler, the output file was parsed using a Perl script called velvet_stats.pl to calculate various assembly statistics. Similarly, for the Contrail assembler a Perl script called contrail_stats.pl was used. These scripts are available upon request from the authors. Using the Perl scripts the following statistics were calculated for comparing the quality of the two assemblies:

1) N50 score: The length of the largest contig for which the following is true: the sum of its length and the lengths of all larger contigs equals to 50% of the total contig length. The N50 size is computed by sorting all contigs from largest to smallest and by determining the minimum set of contigs whose sizes total 50% of the entire genome.

2) Maximum contig length: Contig with largest number basepairs.

3) Minimum contig length: Contig with smallest number of basepairs.

4) Mean contig length: Average of all the contigs length.

To the run the assembly on the full dataset using contrail, all the read files were merged into a single file, and copied in a Hadoop compute cluster available at JCVI that consisted of 8 servers with 32 GB RAM and 16cores each.

Assembly algorithms and software

Both the Velvet and Contrail assemblers are designed for short reads. Velvet requires a compute server with large RAM memory, whereas Contrail relies on Hadoop to distribute the assembly graphs across multiple servers. Velvet is a *de novo* genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454. It currently takes in short read sequences, removes errors then produces high quality, unique contigs using paired-end read and long read information when available, for resolving genomic repeats that complicate contig calculation. Velvet resolves errors which can arise due to both the sequencing process or to the polymorphisms by removing the "tips" in the *de Bruijn* graph that are chain of nodes that is disconnected on one end, or "bubbles" using the Tour Bus algorithm, both originating from nucleotide differences due to sequencing error or polymorphisms in diploid genomes.

Contrail enables *de novo* assembly of large genomes from short reads by leveraging Hadoop, a software library and framework that allows distributed processing of large data sets across clusters of computers using a simple programming model. The Hadoop Distributed File System (HDFS) is the primary storage system in this framework, and creates multiple data blocks distributed across compute server throughout a cluster to enable reliable, extremely rapid parallel computations. HDFS is highly fault-tolerant and is designed to be deployed on low-cost, commodity hardware. Build on top of HDFS is the Hadoop-MapReduce Framework that uses a "Map" step in which individual data records tagged with a "key" are processed in parallel. In a second step, a "Reduce" function performs aggregation and summarization, in which all associated records based on the key are brought together from across the nodes of the cluster.

More specifically, the Hadoop implementation of the Contrail algorithm in the Map phase scans each read and emits the key-value pairs (u, v) corresponding to overlapping k-mer pairs that form an edge. After the map function completes, a key sorting phase that is executed in parallel groups together edges for the same K-mer based on the key value. The initial graph construction creates a graph with nodes for every K-mer in the read set. This is followed by aggregation of identical K-mers in the Reduce phase, where also linear paths of the de *Bruijn* graph are calculated and continuously overlapping K-mers are simplified into single graph nodes representing longer stretches of sequence.

Contrail Assembly Steps: The Contrail source code was downloaded from sourceforge.net repository (http://sourceforge.net/apps/mediawiki/contrail-bio/index.php?title=Contrail). Since Contrail uses Hadoop MapReduce programming framework, the reads file from the fish dataset were stored in the Hadoop File System (HDFS), on a local Hadoop cluster with 5TB storage capacity distributed across 8 nodes, with each node having 32GB RAM and 16 cores memory. In order to run the assembly the following steps were followed:

1) Reads files were downloaded from Assemblathon database using wget http://bioshare.bioinformatics.ucdavis.edu/Data/hcbxz0i7kg/Fish/62F6HAAXX.1.1.fastq.gz

2) Copy the unzipped fastq files into the Hadoop File System (HDFS) using 'put' command: /opt/hadoop/bin/hadoop/fs –put/source folder/destination folder in HDFS. 'Put' command copies files from the local file system to the destination HDFS, also splitting and distributing the file equally across several compute nodes on the cluster.

3) From within the Hadoop Server enter the directory storing the source code for contrail assembler.

4) Run the assembly using the command: contrail.pl –reads [readfile_path] –hadoop [destination folder] –start [stage_of_assembly] [K-mer length].

5) From the output directory obtain the statistic folder, final graph folder and the final contig folder (fasta format) using the Hadoop 'get' command. 'Get' command copy files to the local file system from HDFS.

6) Extract the files in the final graph folder using cat and gunzip command and store all the graph in a single file: cat part* | gunzip >final_graph.

7) Parse the final graph with the perl script contrail_stats.pl and save the output in a file.

Velvet Assembly Steps: The Velvet assembler was downloaded from the EBI website (http://www.ebi.ac.uk/~zerbino/velvet/). The first component of the assembler is Velveth takes in a number of sequence files, produces a hash table, and then outputs two files in an output directory, "Sequences" and "Roadmaps", which are necessary for the next step of the assembler called Velvetg. Users need to provide the output directory folder, file format of the sequence stored in the read file, hash length or K-mer length, read type, and read file name. The second Velvetg component of the assembler is where de Brujn graph is built. It has various options like specifying coverage cutoff or minimum length of the contigs to output. Velvet has optional parameters that allow to assemble paired reads data as well. To activate the use of read pairs, two parameters must be specified: the expected (i.e. average) insert length (or at least a rough estimate), and the expected short-read coverage. The insert length is understood to be the length of the sequenced fragment, i.e. it includes the length of the reads themselves.

1) Run Velveth on one fourth of the entire dataset using unpaired reads using the command velveth/outputdir 65 –fastq –short read_file_1 read_file_2.

2) Run Velvetg on the output obtained from Step 1. using velvetg/outputdir.

3) Run the Velvet on paired read files using velveth/outputdir 65 –fastq –shortPaired read_file_1 read_file_2.

4) Run velvetg on the output obtained from Step 3 using velvetg output_directory/-ins_length 101 -exp_cov 20.

5) After the assembly is complete, parse the stats.txt file from the output directory using the Perl program velvet_stats.pl.

Results and discussion

The comparison of the *de novo* assembly quality for the Velvet and Contrail algorithms presented in the current study, is based on the size of the dataset and assembly statistics using both paired and un-paired sequence data. In the first phase of the research, we compared assembly results for a range of K-mers, using a minimal portion of *zebrafish* data set (single lane un-paired, approximately 2X coverage). For this dataset, Figure 2 shows the graph plot with the relationship between the K-mer value and the maximum contig size in the assembly output for both Velvet and Contrail assemblers. The graph indicates that the maximum contig length increases with increase in the K-mer size, but above K-mer value of 65 that is a little more than half of the actual read length (101 for our dataset), the contig size decreases indicating lower assembly quality. The reason is that at larger K-mer lengths, the DBG algorithm connects two K-mers with an edge on the assembly graph only if K-1 length suffix of the first K-mer is same as K-1 length prefix of the second K-mer. As the K-mer approaches the actual read size, the probability that any pair of K-mers will have a K-1 length of suffix/prefix identity, decreases significantly given also the presence of sequencing errors. From Figure 2 it is also apparent that for both assemblers the maximum contig size was achieved at K-mer size 65, and therefore the remaining experiments in this study were carried out using that specific value.

In the next step, assembly was performed with Velvet for the same 2X coverage dataset using paired reads, but no significant difference was observed in the maximum contig size (Table 1, **row 1–2**). Paired read assembly was not performed for Contrail, as the algorithm implementation used in the current study did not provide this feature. The un-paired datasets with Contrail returned half the maximum contig size of Velvet (Table 1, **row 3**), and took significantly more computing time to complete due to the overhead required for initiating the parallel computation with the Hadoop cluster (see Methods section for details).

In the second phase of the research we attempted to perform assembly using the entire dataset of the zebrafish genome that has approximate coverage of 192X. For running the assembly using Velvet, first the sub-command Velveth was used that creates a "roadmap" of read overlaps, followed subsequently by Velvetg to create and traverse the *de Bruijn* graph. While we used a 1 Terabyte RAM memory server to run the assembler and the Velveth step of the assembly completed, Velvetg failed to complete, and similar situation was observed when using paired reads (Table 1, **row 4–5**). The error reported by the Velvet software in both cases indicated that there was not sufficient memory for the completion of the assembly computation. On the other hand, the Contrail assembler successfully completed the run and returned the best N50 score and largest contig size in the current study, albeit requiring 240 hours running time (Table 1, **row 6**).

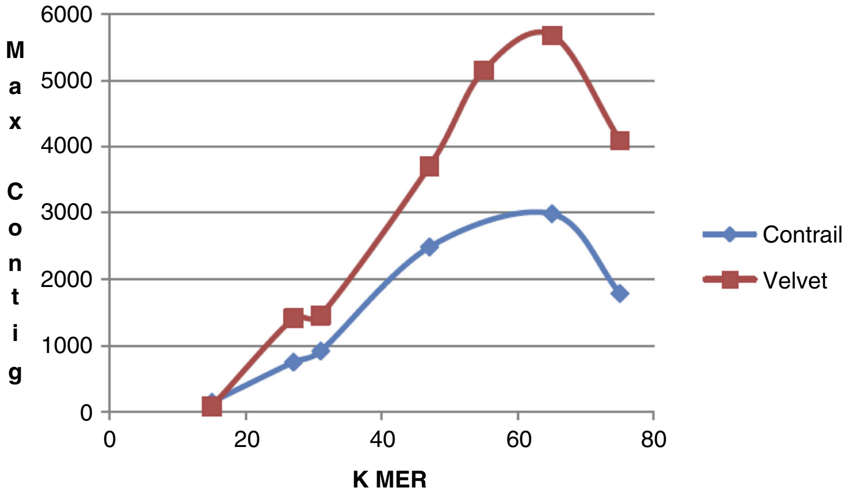


Figure 2. Maximum Contig VS K-mer length.

Table 1. Assembly statistics for the Velvet and Contrail assemblers at K-mer value 65.

#	Assembler	Dataset size	N50 score (bp)	Max Contig size (bp)	Approx. Time (hrs)	Memory RAM (GB)
1	Velvet	minimal (2x) unpaired	1693	5682	6	256
2	Velvet	minimal (2x) paired	1500	6238	6	256
3	Contrail	minimal (2x) unpaired	1380	2987	24hrs	32GB * 8
4	Velvet	full dataset unpaired	-	-	System Crash	1000
5	Velvet	full dataset paired	-	-	System Crash	1000
6	Contrail	full unpaired	4458	46958	240hrs	32GB * 8
7	Velvet	¼ dataset unpaired	1279	16548	9	512
8	Velvet	¼ dataset paired	2540	23780	12	512
9	Velvet	½ dataset unpaired	2477	20658	24	1000
10	Velvet	½ dataset paired	-	-	System Crash	1000

Since Velvet failed to complete the assembly for the entire genome, a partial set of the reads from the zebrafish dataset was used as input. First, by using one-fourth of the total dataset the assembly completed in nine and twelve hours for un-paired and paired reads respectively (Table 1, **row 7–8**). In this case, there was significant difference in the assembly quality between paired and unpaired reads, with the maximum contig size for unpaired reads approximately 7,000bp shorter when compared to paired reads. Next, one-half of the zebrafish dataset was used as input to Velvet, and with unpaired reads the total assembly time completed successfully in 24 hours (Table 1, **row 9**). In detail, the Velveth step consumed approximately 8hours, while Velvetg required an additional 14 hours. The maximum contig size obtained was about half of that with the Contrail assembler when using the full dataset of unpaired reads (Table 1, **row 6**). In addition the assembly for Velvet with one-half of the unpaired reads dataset, was found to be comparable to the Velvet results using one-fourth of the paired reads (Table 1, **row 8–9**). This demonstrates that is feasible to achieve better assembly with less data, when paired end reads are available. Finally, using half of dataset with paired reads, the Velveth step completed in 10hrs while Velvetg failed on our 1TB compute server due to insufficient memory space (Table 1, **row 10**).

Conclusions

The present study presents an example comparison of performance characteristics for distributed genome assemblers that leverage parallel computing and commodity, cloud computing clusters, versus assemblers that require large memory, specialized and expensive compute servers. The overall conclusion is that for assembling small datasets, reads from bacterial genomes or small eukaryotic genomes, it is better to use assembly software such as Velvet. Nonetheless, for larger genomes and datasets a single server cannot scale, and the assembler fails to load the entire graph into the memory and the assembly does not complete. Genome assemblers such as Contrail present an alternative option, as it performs assembly in parallel using cloud computing and commodity server clusters. Contrail is based on the Hadoop programming framework that is open source, freely available and also can be installed on local cluster. If a local clusters are not available, Hadoop compute servers can be rented from cloud providers. Depending on the size of the assembly to be carried additional nodes can be attached or rented as needed, and expand the capacity of the Hadoop cluster on the cloud or locally. Therefore, additional research by the bioinformatics community on cloud-based, scalable assemblers can result in cost effective solutions for assembly of genomes of any size for both well established institutions or smaller, academic research groups.

Author contributions

P.K. performed the research and wrote the manuscript, R.M. and V.S. contributed to the research and writing the manuscript, K.K. reviewed and edited the manuscript.

Competing interests

No competing interests were disclosed.

Grant information

This project has been funded in whole or part with federal funds from the National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services under contract numbers N01-AI30071 and/or HHSN272200900007C.

I confirm that the funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Min Jou W, Haegeman G, Ysebaert M, *et al.*: **Nucleotide sequence of the gene coding for the bacteriophage MS2 coat protein.** *Nature.* 1972; **237**(5350): 82–8.
   PubMed Abstract | Publisher Full Text
2. Maxam AM, Gilbert W: **A new method for sequencing DNA.** *Proc Natl Acad Sci U S A.* 1977; **74**(2): 560–4.
   PubMed Abstract | Publisher Full Text | Free Full Text
3. Sanger F, Coulson AR: **A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase.** *J Mol Biol.* 1975; **94**(3): 441–8.
   PubMed Abstract | Publisher Full Text
4. Metzker ML: **Emerging technologies in DNA sequencing.** *Genome Res.* 2005; **15**(12): 1767–76.
   PubMed Abstract | Publisher Full Text
5. Saiki RK, Gelfand DH, Stoffel S, *et al.*: **Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase.** *Science.* 1988; **239**(4839): 487–91.
   PubMed Abstract | Publisher Full Text
6. Metzker ML: **Sequencing technologies - the next generation.** *Nat Rev Genet.* 2010; **11**(1): 31–46.
   PubMed Abstract | Publisher Full Text
7. Bentley DR: **Whole-genome re-sequencing.** *Curr Opin Genet Dev.* 2006; **16**(6): 545–52.
   PubMed Abstract | Publisher Full Text
8. Ansorge WJ: **Next-generation DNA sequencing techniques.** *N Biotechnol.* 2009; **25**(4): 195–203.
   PubMed Abstract | Publisher Full Text
9. Mitra RD, Church GM: ***In situ* localized amplification and contact replication of many individual DNA molecules.** *Nucleic Acids Res.* 1999; **27**(24): e34–e39.
   PubMed Abstract | Publisher Full Text | Free Full Text
10. Dressman D, Yan H, Traverso G, *et al.*: **Transforming single DNA molecules into fluorescent magnetic particles for detection and enumeration of genetic variations.** *Proc Natl Acad Sci U S A.* 2003; **100**(15): 8817–22.
   PubMed Abstract | Publisher Full Text | Free Full Text
11. Adessi C, Matton G, Ayala G, *et al.*: **Solid phase DNA amplification: characterisation of primer attachment and amplification mechanisms.** *Nucleic Acids Res.* 2000; **28**(20): E87.
   PubMed Abstract | Publisher Full Text | Free Full Text
12. Fedurco M, Romieu A, Williams S, *et al.*: **BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies.** *Nucleic Acids Res.* 2006; **34**(3): e22.
   PubMed Abstract | Publisher Full Text | Free Full Text
13. Myers EW, Sutton GG, Delcher AL, *et al.*: **A whole-genome assembly of Drosophila.** *Science.* 2000; **287**(5461): 2196–2204.
   PubMed Abstract | Publisher Full Text
14. Pevzner PA, Tang H, Waterman MS: **An Eulerian path approach to DNA fragment assembly.** *Proc Natl Acad Sci U S A* 2001; **98**(17): 9748–9753.
   PubMed Abstract | Publisher Full Text | Free Full Text
15. Kundeti VK, Rajasekaran S, Dinh H, *et al.*: **Efficient parallel and out of core algorithms for constructing large bi-directed de Bruijn graphs.** *BMC Bioinformatics.* 2010; **11**(1): 560.
   PubMed Abstract | Publisher Full Text | Free Full Text
16. Denisov G, Walenz B, Halpern AL, *et al.*: **Consensus generation and variant detection by Celera Assembler.** *Bioinformatics.* 2008; **24**(8): 1035–1040.
   PubMed Abstract | Publisher Full Text

17. Cancel-Tassin G, Latil A, Valeri A, *et al.*: **PCAP is the major known prostate cancer predisposing locus in families from south and west Europe.** *Eur J Hum Genet.: EJHG.* 2001; **9**(2): 135–42.
PubMed Abstract | **Publisher Full Text**

18. Batzoglou S, Jaffe DB, Stanley K, *et al.*: **ARACHNE: a whole-genome shotgun assembler.** *Genome Res.* 2002; **12**(1): 177–189.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

19. Lipman DJ, Altschul SF, Kececioglu JD: **A tool for multiple sequence alignment.** *Proc Natl Acad Sci U S A* 1989; **86**(12): 4412–4415.
Publisher Full Text | **Free Full Text**

20. Chvátal V, Erdos P: **A note on Hamiltonian circuits.** *Discrete Math.* 1972; **2**(2): 111–113.
Publisher Full Text

21. Zerbino DR, Birney E: **Velvet: algorithms for de novo short read assembly using de Bruijn graphs.** *Genome Res.* 2008; **18**(5): 821–829.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

22. Schatz MC, Langmead B, Salzberg SL: **Cloud computing and the DNA data race.** *Nat Biotechnol.* 2010; **28**(7): 691–3.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

23. Butler J, MacCallum I, Kleber M, *et al.*: **ALLPATHS: de novo assembly of whole-genome shotgun microreads.** *Genome Res* 2008; **18**(5): 810–820.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

24. Li Y, Hu Y, Bolund L, *et al.*: **State of the art *de novo* assembly of human genomes from massively parallel sequencing data.** *Hum Genomics.* 2010; **4**(4): 271–7.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Open Peer Review

Current Referee Status: ☒ ☒

Version 1

Referee Report 27 February 2015

doi:10.5256/f1000research.6440.r7606

☒ **Surya Saha**
Boyce Thompson Institute for Plant Research, Ithaca, NY, USA

De novo assembly of large eukaryotic genomes remains a challenging task even with the increasing availability of high quality long reads and short reads from paired-end and mate-pair libraries. An in-depth comparison of the performance of assemblers using multiple data sets and operating on widely used high memory multi-core systems versus distributed platforms can be a valuable contribution to the field.

However, the authors fail to deliver on the promise of the paper on several counts. They present a limited and unsound comparison of only two assemblers (Velvet and Contrail) using two poorly selected datasets from a single species. The experiment design and analysis is replete with numerous errors, large and small.

The Background section is superfluous and contains uncited historical details that are not applicable to the context of the paper. The authors chose to highlight sequencing technology (Figure 1) that is outdated by at least two years (Illumina Genome Analyzer) and not available any more (Helicos, Roche 454) while neglecting to mention the workhorses of sequencing cores today. There was no discussion of the widely used platforms like Illumina Hiseq or desktop sequencers like Illumina Miseq or Ion Torrent PGM. There are also many factual errors such as the maximum length of 50bp for sequences generated using Maxam-Gilbert method and costs/ read lengths of different sequencing platforms. Maxam-Gilbert sequences range from 250bp to 500bp or longer [1]. The sequencing costs and read lengths are outdated for Illumina Genome Analyzer and incorrect for SOLiD. The authors describe the concepts of overlap-layout-consensus (OLC) and *de Bruijn* graph assembly but neglect to mention string graph [2] theory.

The methods used for evaluation are reasonably well documented. But a majority of the description could have been moved to supplementary data leaving room for a more qualitative discussion of the motivation for the methods used in the paper. The authors chose to evaluate only two assemblers without giving an explanation of why these two were selected. Velvet was selected as an example of serial de Bruijn graph assembler and Contrail as a compute-distributed assembler. The dataset selected by the authors is from a Lake Malawi cichlid (*Maylandia zebra* or *Metriaclima zebra*) incorrectly named as *M.* zebrafish in the paper. The reason for selecting this particular dataset and not a range of genomes with varying complexity is not explained despite the impact on the evaluation. The two cichlid data sets have either very low coverage or very high coverage, both of which are detrimental for *de novo* assembly. The authors refer to Perl scripts that are not included in the paper. This is not acceptable given the availability and ease of use of code sharing platforms like Git. Such scripts can also be included in supplementary data as text files. The only summary statistics used for evaluating assemblies are N50 and maximum contig size. These are not informative with regards to the quality of the assembly and will have ramifications for analysis. Error correction is mentioned but no further explanation is given. None of the many other well-known preprocessing practices such as k-mer based normalization and merging of paired-end reads were used before *de novo* assembly.

The authors attempt to assemble the full dataset as well as various sub-sampled versions using the two assemblers. The sub-sampling procedure is not described in the methods. The analysis in the Discussion section, like previous sections, has several shortcomings. The authors refer to the Assemblathon2 [3] comparison of assembly algorithms which set the standard for metrics to use for evaluation of assemblers. The evaluation of assemblies from Velvet and Contrail is quite inadequate as they authors did not check the assemblies for errors. They did not validate the assemblies by checking for the presence of core genes as described in Assemblathon2 [3] or by comparing to the published genome. The assemblies were simply evaluated for contiguity using N50 and maximum contig size, both of which can be improved with parameters that can potentially increase the number of misassemblies.

The authors conclude that assemblers based on serial or non-distributed algorithms cannot be used for large scale *denovo* assembly due to out of memory errors in Velvet. Velvetg fails to complete due to lack of memory but they do not explore the issue further. Large *de Bruijn* graphs are often caused by presence of sequencing errors in the reads. Velvet may be able to assemble the given dataset once poor quality reads are filtered out. Merging of paired-end reads and k-mer-based normalization are also effective strategies to reduce memory requirements. It is also not clear why the authors did not perform assemblies with 50% and 25% unpaired dataset with Contrail. They would provide additional data points for comparison with Velvet even for the underpowered experimental design used in this paper.

The information presented in this paper is outdated and the experiments and analysis are woefully inadequate to judge the effectiveness of serial versus distributed genome assemblers. Moreover, the paper does not utilize or even address the commonly used approach of combining long reads with higher coverage paired-end and mate-pair short reads to generate assemblies for large eukaryotic genomes.

References

1. Franca LT, Carrilho E, Kist TB: A review of DNA sequencing techniques.*Q Rev Biophys*. 2002; **32** (2): 169-200 PubMed Abstract
2. Myers EW: The fragment assembly string graph. *Bioinformatics*. 2005; **21** (suppl 2): ii79-85 PubMed Abstract | Reference Source
3. Bradnam KR, Fass JN, Alexandrov A, Baranay P, Bechner M, Birol I, Boisvert S, Chapman JA, *et al.*: Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *Gigascience*. 2013; **2** (1). PubMed Abstract | Free Full Text

I have read this submission. I believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Competing Interests: No competing interests were disclosed.

Referee Report 10 February 2015

doi:10.5256/f1000research.6440.r7605

Keith E Robison
Warp Drive Bio, Cambridge, MA, USA

This title of this manuscript would lead a reader to believe that a careful comparison of two broad strategies for *de novo* sequence assembly. Unfortunately, what the manuscript delivers is an error-rich and outdated introduction, incompletely defined methods and an extremely limited comparison on a single dataset of two assembly programs.

In their introduction the authors dig far into the history of DNA sequencing nearly to the very beginning. However, this summary is filled with dubious assertions that lack citations. For example, they credit PCR with boosting Sanger sequencing over Maxam-Gilbert sequencing, but Sanger sequencing had already all-but-extinguished Maxam-Gilbert before PCR had become commonly used in any facet of sequencing, and even today Sanger sequencing does not have a reliance on PCR (the authors may be confusing cycle sequencing, which relies on a linear amplfication using thermostable polymerases, with PCR).

The authors present in figure form a comparison of four "next generation sequencing" systems (a term that really should be retired, given the fact that these systems are over a decade old now). The figure is exquisitely badly formatted and nearly unreadable when printed due to using a thin white font on dark backgrounds; if the information was of any value it should have been formatted as a table. Alas, the information in the table is worse than its formatting, being completely out-of-date.

The statistics given for Illumina sequencing, which name an instrument (the GA) discontinued several years ago, give a cost per base that is roughly right for the MiSeq platform, but the read lengths on that platform are far longer (now 2x300). Several of the other Illumina platforms offer longer read lengths than given with a cost per basepair which are several orders of magnitude lower than given in the figure.

Another quadrant of the figure describes the SOLiD system, which has rarely been used for *de novo* assembly. In any case, the number of reads per run and read length are both wrong, which leads to the cost per basepair being off by over an order of magnitude.

A third quadrant gives obsolete statistics for the 454 platform, which hit read lengths of over 800 bases (or >2X that given in the figure). However, that really doesn't matter except historically since Roche discontinued the 454 platform in 2014. Even worse is the 4th quadrant, which describes the Helicos sequencer, a comparny that went bankrupt in 2011.

The Ion Torrent systems, despite being used frequently for small genome *de novo* assembly, are not mentioned anywhere. Missing from the table, but briefly mentioned in the text, is the Pacific Biosciences platform. Given that PacBio has been used extensively for *de novo* assembly, this is unexcusable. Furthermore, the paper fails to mention that PacBio is very different in its read characteristics, particularly read length.

A section on the experimental workflows for these systems attempts to summarize all of them in one paragraph. There is one serious error here; in library preparation PCR is performed after ligation of adaptors and not before. More seriously, the described workflow does not apply to either of the single molecule systems which they have mentioned; neither uses PCR and Helicos didn't even have a ligation step.

A brief summary of *de novo* sequence assembly algorithms has a few small errors (for example, while many implementation of overlap-layout-consensus (OLC) use k-mers to speed execution, k-mer analysis is not an inherent facet of the algorithm as the authors suggest). More serious is that only de Bruijn graph and OLC are discussed; string graphs are omitted and would be very relevant to the purported scope of this paper.

For the paper, the authors downloaded a single dataset from the Assemblathon dataset, for Zebrafish (oddly described as "fish species M.zebrafish"). No explanation is given why this dataset was chosen. Some assembly runs involved removing low quality data from the dataset, but no explanation is given as to what criteria were used to define low quality or tools used to remove them. This relates to another gaping hole in the manuscript: numerous approaches for preprocessing data have been described in the literature, including read filtering, read trimming, error correction, paired end merging and k-mer based normalization; none of these topics are broached. This will become clearly unfortunate later in their manuscript.

While the title promises a significant comparison of methods, the manuscript describes using only two programs: Velvet standing in for single compute node de Bruijn graph algorithms and Contrail for distributed computing de Bruijn graph assemblers. While a few other DBG assemblers are mentioned, the existence of other DBG assemblers which can run across multiple compute nodes are not (e.g. Ray, ABySS). Since the manuscript focuses on the Hadoop aspect of Contrail (which is the framework it uses to distribute the computing across multiple nodes), the paper could leave the unfortunate impression that this is the only attempt in the field, rather than one of many mechanisms (e.g. MPI)

The authors begin by trying a sampling of k-mer values for both Velvet and Contrail using a 2X dataset. The method used for downsampling the dataset is not given (while the text promises that the Perl programs used are available on request, this should be seen as an unacceptably inadequate mechanism; at a minimum they must be supplementary materials but better would be deposition in a public code repository). They measure two figures-of-merit (N50 and maximum contig size), but plot only one of them (though this plot is the best single element in the paper). A justification for using a 2X sample, rather than a larger one, is not given. This opens the question whether a larger k-mer length might have worked better on a larger dataset.

The authors proceed to try both programs on the entire dataset; Contrail succeeds but Velvet fails. Velvet fails again on 50% of the data if in paired end mode (though again, the method of downsampling is not given) but runs on that dataset in unpaired read mode. Velvet is tried also, in both modes, on a 25% dataset and succeeds. The authors present the figures-of-merit as a table, with no apparent order. Since Contrail (at least the version used) had only an unpaired mode, it is run only once. This data would be far more useful plotted as a graph as well, with the table sorted in some order relevant to the user, such as the 2X, 25%, 50% and 100% of dataset.

A serious issue at this point is the author's choice of N50 and maximum contig length as their sole figures-of-merit, which they mistakenly label as measures of assembly quality. At no point do the authors attempt to assess the correctness of their assemblies, despite this being a standard method in assembler comparisons (such as the Assemblathon from which the authors obtained the data used). Both N50 and maximum contig length can be inflated by overly aggressive assembly that yields misassembly artifacts, and N50 can be inflated by the choice of a minimum contig length cutoff. Indeed, the authors fail to report a genome size their assemblies, and so these assemblies could represent only a fraction of the target genome.

The authors observe that the 25% and 50% datasets gave similar results for their figures-of-merit, and observe that less data can give equal or better results. They appear to have not asked if this has been observed before (it has). Nor do they run Contrail on the subsampled datasets to see if the trend holds there as well.

The issue of Velvet crashing on the larger datasets is presented as highly significant; indeed the conclusion is drawn that multi-machine programs such as Contrail are required for this data. This is highly unfortunate on two grounds.

First, as noted before, the authors performed nearly no preprocessing of the data (other than the ill-documented poor quality read removal). Sequencing errors will enlarge the de Bruijn graph, so error correction or read trimming can reduce the memory requirements of an assembler. Paired end merging can similarly reduce memory requirements, albeit at some risk of telescoping small repeats. Merging is particularly relevant for Contrail, since it does not explicitly handle paired ends. K-mer based read normalization can greatly reduce memory requirements for assembly.

Second, a number of programs have demonstrated assembly of vertebrate-scale short read datasets on single machines, indeed single machines with far less memory than the 1Tbyte found compute node used for Velvet in the paper. Examples include Minia, with a de Bruijn graph structure designed to be extremely memory efficient, and Readjoiner, which uses a string graph paradigm (which, as noted above, is a strategy ignored by the paper in the introduction).

Finally, the authors fail to make any attempt to place this in a relevant modern context. Given that short reads from short inserts alone are mathematically incapable of assembling anything but the simplest plasmid or viral genomes, the current thrust in *de novo* assembly is assembling either entirely from long reads or integrating short reads with long reads or mate pairs to accurately yield long (increasingly, chromosome-scale) scaffolds. Failing to place the very limited findings of this manuscript in such a context could be characterized as a final failing.

I have read this submission. I believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Competing Interests: No competing interests were disclosed.