A PARALLEL SCALABLE MULTI-CORE SHORT READ ASSEMBLER

Veeram Venkata Siva Prasad[1] and Gunisetti Loshma[2]

[1]M.Tech(CSE), Sri Vasavi Engineering college, Tadepalligudem, West Godavari District, Andhra Pradesh, India

vvsprasad1242@gmail.com

[2]Associate Professor, CSE, Sri Vasavi Engineering College, Tadepalligudem, West Godavari District, Andhra Pradesh, India

loshma@gmail.com

ABSTRACT

Bioinformatics and computational biology are rooted in life sciences as well as computer and information sciences and technologies. Bioinformatics applies principles of information sciences and technologies to make the vast, diverse, and complex life sciences data more understandable and useful. Computational biology uses mathematical and computational approaches to address theoretical and experimental questions in biology. Short read sequence assembly is one of the most important steps in the analysis of biological data. There are many open source software's available for short read sequence assembly where MAQ is one such popularly used software by the research community.

In general, biological data sets generated by next generation sequencers are very huge and massive which requires tremendous amount of computational resources. The algorithm used for the short read sequence assembly is NP Hard which is computationally expensive and time consuming. Also MAQ is single threaded software which doesn't use the power of multi core and distributed computing and it doesn't scale. In this paper we report HPC-MAQ which addresses the NP-Hard related challenges of genome reference assembly and enables MAQ parallel and scalable through Hadoop which is a software framework for distributed computing.

In this paper we try to perform thread level parallelism using openMP .and it can reduce computational time.

KEYWORDS

High Performance computing, Hadoop, Whole Genome Reference Assembly, Computational Quantitative Biology, HPC-MAQ, openMP, thread level parallelism.

1. INTRODUCTION

In the recent past there are attempts on understanding life through the eyes of physics, chemistry, and mathematics. The objective is to quantify biology so that we can understand biological objects as a measurable system. As our understanding of biology is increasing the data, database, and complexity of biology problems are increasing. We call this the field as computational quantitative biology, where we use computers to quantify biology. In computational quantitative biology, for a bioinformatician, assembly challenges are immense. In assembly we try to construct (assemble) the target string or a sequence from billions of tiny substrings or subsequence's. The target sequence can be a whole genome, a set of transcriptases, a set of Chip sequences in epigenomics, or even multiple microbial genomes in meta genomics. Assembly challenge is a complex computational algorithmic challenge rather

DOI : 10.5121/ijitcs.2012.2207

than a biology challenge. To appreciate the magnitude of the challenge we can take a simple unicellular micro-organism amoeba dubia where the target genome or the DNA of this species is about 670 Billion nucleotides .When we sequence this species it will generate trillions of small fragments of the DNA.

A DNA is made of small molecules like Adenine, Thymine, Guanine, and Cytosine. In computational quantitative biology, these molecules are represented as symbols A, T, G, and C respectively. In simple terms, a target DNA sequence is a large string made of millions of A, T, G, or C. These symbols are called nucleotides or bases. In computational terms, the genome of amoeba dubia will be represented as a large string of size 670 billion combinations of these A, T, C, and G characters like "ACGTGACTGGTCATGAC…".

During the sequencing process a large bio-molecule like DNA, mRNA or ncRNA, is broken into multiple tiny fragments using next generation sequence (NGS) techniques (Fig. 1). In the next step, the nucleotides (A, T, C, G) within these tiny fragments are read by the NGS machine to generate small sequences of A/T/C/G character string or reads. The sequencing process generates billions and trillions of small set of data sequences as the result of any experiment. These reads of tiny fragments are 25 charters to 400 characters long. For Illumina machines this length is between 25 to 75 based on the version of the sequencing machine ; and, between 30 to 400 for Roche 454 machine depending on the quality of the read. Before a biologist can make any meaning of the original biological sample, these small sequences need to be quantified through the assembly process. During the assembly phase, these tiny reads (fragments) are assembled to reconstruct the target sequence or the original large string (Fig 1). It is like taking a large book of thousands of pages and shred it through a shredding machine. Now the task is to reconstruct the original book from the pieces. In their words, the target sequence (DNA/RNA) of millions of bases needs to be reconstructed from billions of tiny sequences or reads that range from 25 to 76 bases for Illumina or 30 to 400 bases for Roche 454. This is like solving a billion piece jigsaw puzzle. In general, these tiny reads are compared for overlap or similarity between them and connected one after the other to construct the original target string. Many authors have attempted to address this challenge through various bioinformatics systems and pipelines.

In bioinformatics, 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 small pieces between 20 and 1000 bases, depending on the technology used. Typically the short fragments, called reads, result from shotgun sequencing genomic DNA, or gene transcript (ESTs).

The problem of sequence assembly can be compared to taking many copies of a book, passing them all through a shredder, and piecing a copy of the book back together from only shredded pieces. The book may have many repeated paragraphs, and some shreds may be modified to have typos. Excerpts from another book may be added in, and some shreds may be completely unrecognizable.

The assembly process can be of two types, viz., reference assembly and de-novo assembly. In reference assembly we have a similar target sequence that is used as a reference; whereas, in de-novo no target sequence is available either to compare or use as a reference.

In terms of complexity and time requirements, de-novo assemblies are orders of magnitude slower and more memory intensive than mapping assemblies. This is mostly due to the fact that the assembly algorithm need to compare every read with every other read (an operation that is

has a complexity of $O(n^2)$ but can be reduced to $O(n \log(n))$. Referring to the comparison drawn to shredded books in the introduction: while for mapping assemblies one would have a very similar book as template (perhaps with the names of the main characters and a few locations changed), the de-novo assemblies are more hardcore in a sense as one would not know beforehand whether this would become a science book, or a novel, or a catalogue etc.

Reference assembly plays a very significant role in computational quantitative biology. In case of species where the genome is already available, it will be used almost end-to-end to determine the variation of an individual against the reference – example will be clinical genomics. It is also often used as a part of the pipeline in denovo assembly to learn about the orientation of contigs and as a guide. One reliable and popular system for reference assembly is MAQ (Mapping and Assembly with Quality) authored by Heng Li, et al. that uses a known superstring as a reference and assembles the tiny reads from NGS by aligning with the reference sequence.

Biological problems are in general NP-hard (Non deterministic Polynomial) this means that if the problem complexity increases, the resource required to arrive a solution is not linear. Moreover biological problems are hard one-way functions – it is easy to break a large string into tiny pieces but very hard to do the inverse – combine the small pieces together into the original. To process the human genome that is about 3.14 billion bases (Bb), it needs super computers. Supercomputers are expensive to acquire and maintain and is not affordable by small & medium enterprises (SME) like, clinics, or small biotech companies. High performance computing is a paradigm that allows supercomputers to be available on rent. Also, algorithms like Map Reduce and tools like Hadoop allows many of these one-way NP-hard problems to be solved within a desired time using High Performance computing infrastructure.

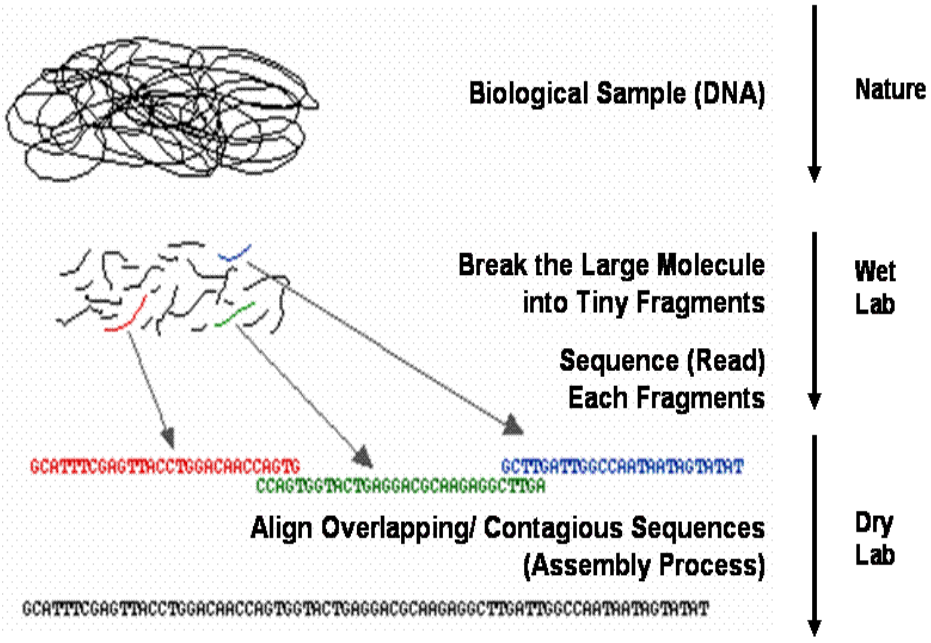


Fig. 1: The Sequencing and Assembly process

2. HIGH PERFORMANCE COMPUTING - THE NEW PARADIGM

Supercomputing is the biggest, fastest computing right this minute and latter "Likewise, a supercomputer is one of the biggest, fastest computers right this minute". So, the definition of supercomputing is constantly changing. Rule of thumb: A supercomputer is typically at least 100 times as powerful as a PC. According Jargon: Supercomputing is also known as High Performance Computing (HPC) or High End Computing (HEC).

High performance computing is effective because of It's about size and speed.

Size: Many problems that are interesting to scientists and engineers can't fit on a PC - usually because they need more than few GB of RAM, or more than a few 100 GB of disk. Speed: Many problems that are interesting to scientists and engineers would take a very long time to run on a PC: months or even years. But a problem that would take a month on a PC might only take a few hours on a supercomputer.

In general High performance computing is useful for performing following

• Simulation of physical phenomena, such as Weather forecasting,, Black holes colliding , Oil reservoir management.

• Data mining: finding needles of information in a haystack of data, such as Gene sequencing, Signal processing, Detecting storms that could produce tornado.

• Visualization: turning lots of data into pictures that a scientist can understand Several scientific applications require computing and/or storage resources that go beyond the processing power of a single multi-core machine. High performance computing (HPC) clusters provide the necessary hardware and software infrastructure to efficiently run computing intensive applications.

The power to model and manipulate our world in silicon has enabled vast changes in how we conduct science, business, and even our everyday lives. From the next scientific breakthrough to new and better products to a greener world, High Performance Computing (HPC) is playing a central role in all these efforts. In simple terms, HPC enables us to first model then manipulate those things that are important to us. HPC changes everything. It is too important to ignore or push aside. Indeed, HPC has moved from a selective and expensive endeavor to a cost-effective technology within reach of virtually every budget.

3. MAQ - MAPPING AND ASSEMBLING WITH QUALITY

MAQ is software that builds mapping assemblies from short reads generated by the next-generation sequencing machines. It is particularly designed for Illumina-Solexa 1G Genetic Analyzer, and has preliminary functions to handle ABI SOLiD data. Maq first aligns reads to reference sequences and then calls the consensus. At the mapping stage, maq performs ungapped alignment. For single-end reads, maq is able to find all hits with up to 2 or 3 mismatches, depending on a command-line option; for paired-end reads, it always finds all paired hits with one of the two reads containing up to 1 mismatch. At the assembling stage, maq calls the consensus based on a statistical model. It calls the base which maximizes the posterior probability and calculates a phred quality at each position along the consensus. Heterozygotes are also called in this process.

By using MAQ we can do 1. Fast align Illumina/SOLiD reads to the reference genome. With the default options, one million pairs of reads can be mapped to the human genome in about 10 CPU hours with less than 1G memory.2.Accurately measure the error probability of the alignment of each individual read. 3.Call the consensus genotypes, including homozygous and heterozygous polymorphisms, with a Phred probabilistic quality assigned to each base. 4.Find short indels with paired end reads. 5.Accurately find large scale genomic deletions and translocations with paired end reads. 6.Discover potential CNVs by checking read depth. 7.Evaluate the accuracy of raw base qualities from sequencers and help to check the systematic errors.

By using MAQ we cannot perform 1.Do de novo assembly. (Maq can only call the consensus by mapping reads to a known reference.) 2.Map shorts reads against themselves. (Maq can only find complete overlap between reads.) 3.Align capillary reads or 454 reads to the reference. (Maq cannot align reads longer than 63bp.)

4. HIGH PERFORMANCE COMPUTING TO MAPREDUCE

HPC-MAQ uses *map-reduce* technique for scalability. MapReduce is a functional programming technique used in parallel programming; it has been implemented by Google for their Internet search engine. Since its introduction in 2004 by Google, MapReduce has become the programming model of choice for processing large data sets. MapReduce borrows ideas from functional programming, where a programmer can define both a *map* task that maps a data set into another data set, and a *reduce* task that combines intermediate outputs into a final result. Although MapReduce was originally developed for use by web enterprises in large data-centers, this technique has gained a lot of attention from the scientific community for its applicability in large parallel data analysis including geographic, high energy physics, genomics, etc.

HPC-MAQ uses the MapReduce algorithm where the problem is split into smaller tasks at the *map* phase and collects the results through the *reduce* phase. Looking into its potential, Apache open source community implemented MapReduce and made it available as Hadoop. It comprises of independent instances of tasks that are executed simultaneously.. Map Reduce is a software framework that supports distributed computing on large data sets on clusters of computers. It is the combination of two processes named *map* and *reduce*. HDFS is a file system designed for storing very large files with streaming data access patterns, running on clusters on commodity hardware. Hadoop streaming comes with the Hadoop distribution that allows the user to create and run Map Reduce jobs with any executable or script as the mapper and/or the reducer. In HPC-MAQ the input reads of a genome are split into multiple chunks and each of these chucks are processed in parallel through Hadoop streaming.

5. MAPPING AND ASSEMBLING WITH QUALITY TO HIGH PERFORMANCE COMPUTING- MAPPING AND ASSEMBLING WITH QUALITY

MAQ version 0.6.8, software comprises of mainly two applications. These are maq and maqview. Maq is used for assembly and maqview is used for visual mapping of the reads into the reference sequence. Maq commands are divided into three categories that are key functions, information extraction,and format conversion. Within each category there are various sub-functions that are invoked through the commands parameters.

Maq contains two important parameters read and reference in which read is of the form fastq and reference is of the form fasta.

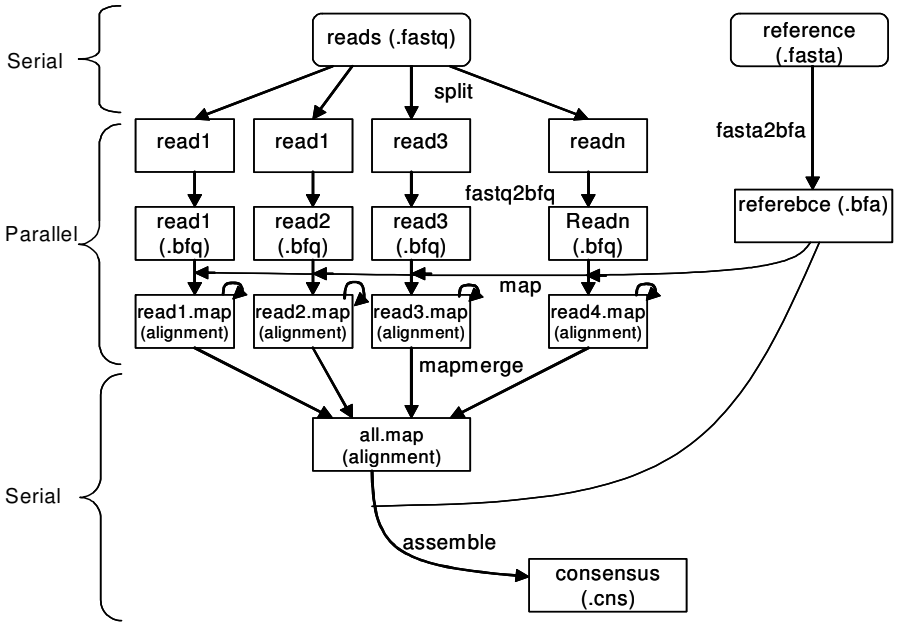


Fig. 2: High performance computing –Mapping and Assembling with Quality Pipeline

5.1 FASTQ FORMAT

FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are encoded with a single ASCII character for brevity. It was originally developed at the Wellcome Trust Sanger Institute to bundle a FASTA sequence and its quality data, but has recently become the de facto standard for storing the output of high throughput sequencing instruments such as the Illumina Genome Analyzer

A FASTQ file normally uses four lines per sequence. Line 1 begins with a '@' character and is followed by a sequence identifier and an optional description (like a FASTA title line). Line 2 is the raw sequence letters. Line 3 begins with a '+' character and is optionally followed by the same sequence identifier (and any description) again. Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.

A minimal FASTQ file might look like this:

@SEQ_ID

GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGT
TT

+

!"*((((***+))%%%++)(%%%%).1***-+*"))**55CCF>>>>>>CCCCCCC65

The original Sanger FASTQ files also allowed the sequence and quality strings to be wrapped (split over multiple lines), but this is generally discouraged as it can make parsing complicated due to the unfortunate choice of "@" and "+" as markers (these characters can also occur in the quality string).

5.2 FASTA FORMAT

In bioinformatics, FASTA format is a text-based format for representing either nucleotide sequences or peptide sequences, in which base pairs or amino acids are represented using single-letter codes. The format also allows for sequence names and comments to precede the sequences. The format originates from the FASTA software package, but has now become a standard in the field of bioinformatics.

A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. The word following the ">" symbol is the identifier of the sequence, and the rest of the line is the description (both are optional). There should be no space between the ">" and the first letter of the identifier. It is recommended that all lines of text be shorter than 80 characters. The sequence ends if another line starting with a ">" appears; this indicates the start of another sequence. A simple example of one sequence in FASTA format:

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]

LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLF
SAIPYIGTNLVEWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGL
TSDSDKIPFHPYYTIKDFLGLLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEW
YFLFAYAILRSVPNKLGGVLALFLSIVILGLMPFLHTSKHRSMMLRPLSQALFWTLTMDL
LTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGXIENY

5.3 SEQUENCE REPRESENTATION

After the header line and comments, one or more lines may follow describing the sequence: each line of a sequence should have fewer than 80 characters. Sequences may be protein sequences or nucleic acid sequences, and they can contain gaps or alignment characters.

The nucleic acid codes supported are:

A adenosine	C cytidine	G guanine
T thymidine	N A/G/C/T (any)	U uridine
K G/T (keto)	S G/C (strong)	Y T/C (pyrimidine)
M A/C (amino)	W A/T (weak)	R G/A (purine)
B G/T/C	D G/A/T	H A/C/T
V G/C/A	- gap of indeterminate length	

the accepted amino acid codes are:

A alanine	P proline

B aspartate/asparagine	Q glutamine
C cystine	R arginine
D aspartate	S serine
E glutamate	T threonine
F phenylalanine	U selenocysteine
G glycine	V valine
H histidine	W tryptophan
I isoleucine	Y tyrosine
K lysine	Z glutamate/glutamine
L leucine	X any
M methionine	* translation stop
N asparagine	- gap of indeterminate length

MAQ also offers many handy Perl scripts and abstracts many MAQ functions. It may be noted that the map function in MAQ and map function in MapReduce are different – map function used in Hadoop MapReduce is to split a large task into smaller tasks; whereas, map function used by MAQ is to map the reads into a layout. The HPC-MAQ pipeline is depicted in Fig. 2 where MAQ commands are run in parallel, and realized through following steps.

1) Split the Read Files: In this step, the sequence file comprising of tiny reads in FASTQ (with standard quality format) format is split into multiple files – each with 2,000,000 sequences. These 2,000,000 small tiny sequences are in fact 8,000,000 UNIX records with 8,000,000 new-line characters. This is done using the split command in UNIX; however, other tools can also be used for the split function.

2) Convert Reference File into binary: In this step the reference sequence FASTA file is converted into binary .bfa format using MAQ fasta2bfa tool.

3) Convert Read files into binary: These chunks of 2,000,000 reads are converted into MAQ .bfq files in parallel using MAQ fastq2bfq tool. This step is performed in parallel with Step 2 above.

4) Hadoop Streaming: Each chunk of 2 millions reads aretaken and given to map phase of MapReduce using Hadoop streaming. Hadoop takes these independent reads and spawns instances.

5) Continue till finish: The Hadoop task of Step 4 above continues with map phase of MAQ using MAQ map function.Here each MAQ map runs on individual read file in parallel.The map command of MAQ aligns reads in the bfq format to the reference in the bfa format. The output out.map is a compressed binary file. It stores the sequences, qualities, positions of the mapped reads and related information such as the number of mismatches and the mapping quality.

6) Reduce phase: Reduce the Hadoop outputs using MAQ mapmerge command to merge a batch of read alignments together and get a single map (default all.map) for the entire NGS read.

7) Final assembly: In the final phase use the MAQ assembler by using MAQ assemble command.

6. IMPLEMENTATION OF HPC-MAQ THROUGH HADOOP MAPREDUCE

HPC-MAQ uses Open-source MAQ software (http://maq.sourceforge.net) and Open-source Hadoop software (http://hadoop.apache.org). Large read files are split using UNIX split command as already mentioned in this article. The UNIX shell script hpcmaq.sh is used to parallelize the MAQ tasks. Compiled versions of Python scripts mapper1.pyc, mapper2.pyc, mapper3.pyc, reducer1.pyc and reducer2.pyc are used to parallelize MAQ using Hadoop streaming.

1) core-site.xml: This configuration file is used to specify the name node..

2) hdfs-site.xml: This configuration file is used to specify the path for name node and data node.

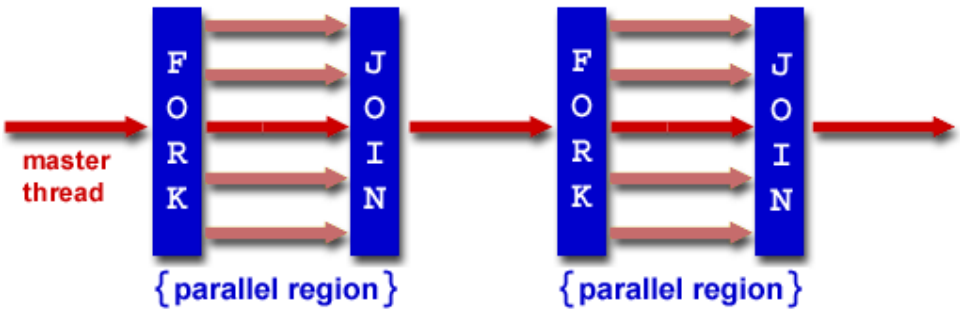
3) mapred-site.xml: This configuration file is used to specify the name of the job tracker, number of map and reduce tasks.

4) HPCmaq.sh: This is the main shell script which is used to run Hadoop map/reduce tasks by using the concept of Hadoop Streaming.

7.OPENMP:

An Application Program Interface (API) that may be used to explicitly direct multi-threaded, shared memory parallelism.In short **Open Multi-Processing** that is Open specifications for Multi-Processing via collaborative work between interested parties from the hardware and software industry, government and academia.

OpenMp uses different programming models one of this is



8. PARALYZED SCALABLE MULTI CORE SHORT READ ASSEMBLER:

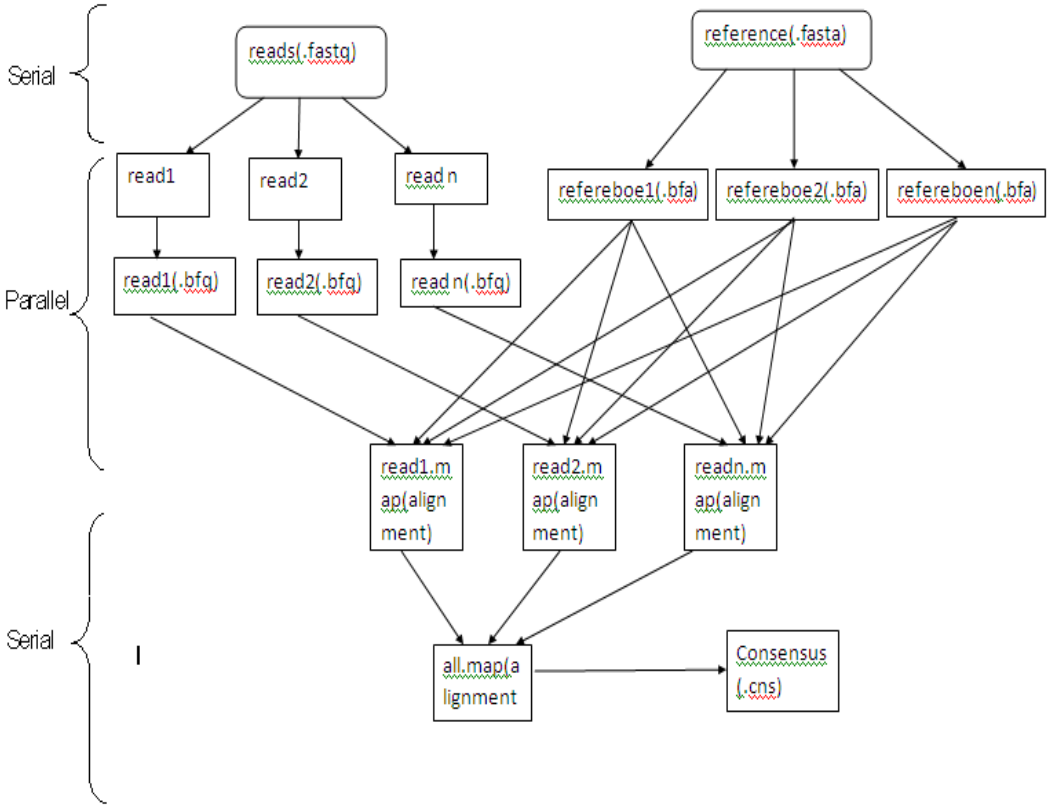


FIGURE: PARALLELIZED MULTI CORE SHORT READ ASSEMBLER PIPELINE

In the above diagram we stated that the how we can do further parallelization that is by doing thread level penalization using openMp or anyone which is used for multiple thread processing .
In this we have divided reference into number of parts as reference1,reference2…reference n
And they parallel compute read1.map,read2.map,….read n.map using openMp

9.RESULTS

TABLE 1: BENCHMARK RESULTS OF HPC-MAQ AGAINST MAQ

Software	Read Data	Reference	Time Taken
MAQ	43 million reads	hg19	34h 15m 16sec
HPC-MAQ	43 million reads	hg19	10 h 20m 20sec

Time taken to run maq with 43 million reads and human genome (hg19) as reference is 34h 15m 16sec. Time taken to run hpc-maq with the same 43 million reads and human genome (hg19) as reference on a 4 node cluster is 10h 20m 20sec.

10. CONCLUSION

In omic sciences assembly step is a mandatory requirement for any experiment using next generation sequencer. However, like many other biology problems that are NP-hard, reference assembly for human genome requires a supercomputer to run for days. The assembly of a whole genome from small fragments of sequences has always been a challenge in computational biology. Assembly step is a mandatory requirement for any experiment using next generation sequencer. However, like many other biology problems that are NP-hard, reference assembly for human genome requires a supercomputer to run for days. In this paper we have presented HPC-MAQ that made popular MAQ software HPC enabled that can run on commodity machines by which time and cost can reduce to a larger extent. This can also be parallelized again at the fine grain level which will reduce lot of computational time that is paralyzed scalable multi core short read assembler. In this paper we also implemented paralyzed scalable multi core short read assembler which is advanced version of HPC-MAQ. Paralyzed scalable multi core short read assembler can reduce computational time to a fine grain level.

11. REFERENCES

[1] Sizing up genomes: Amoeba is king,
http://www.genomenewsnetwork.org/articles/02_01/Sizing_genomes.shtml

[2] Joao Setubal, Joao Meidanis, Introduction to Computational Molecular Biology, Thomson,2005

[3] Pavel A Pevzner, Computational Molecular Biology – An Algorithmic Approach, MIT Press, 2000

[4] Heng Li, Jue Ruan, and Richard Durbin, Mapping short DNA sequencing reads and calling variants using mapping quality scores, doi:10.1101/gr.078212.108,2008

[5] Ramana M Idury, Michael S Waterman, ANew Algorithm for DNA Sequence Assembly,Journal of Computational Biology, Vol 2, No 2,1995

[6] Stefan Canzar, Lagrangian Relaxation Solving NP-hard Problems in Computational Biology via Combinatorial Optimization, Doctoral thesis, Max-Planck-Institut for Informatics

[7] Humanme Assembly Information, Genome Reference Consortium,
http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human/data/index.shtml

[8] Dean J, Ghemawat S., (2004) MapReduce:Simplified Data Processing on Large Clusters, OSDI'04, San Francisco, CA .

[9] Hadoop, open-source software for reliable, scalable,distributed computing, http://hadoop.apache.org/

[10] Hadoop streaming http://hadoop.apache.org/common/docs/r0.15.2/streaming.html

[11] MAQ (Mapping and Assembly using Quality) version 0.6.8 (2008), http://maq.sourceforge.net/
[12]MAQ (2008a), Manual http://maq.sourceforge.net/maq-man.shtml

[13] The first International Workshop on Mapreduce and its Applications (MAPREDUCE'10) http://graal.enslyon.fr/mapreduce/

[14] Human Genome v19 (hg19), http://hgdownload.cse.ucsc.edu/downloads.html#

[15]computational biology challenges https://idal-siege.inria.fr/dri/bis2011/Sjolander.pdf

[16] computational biology challenges http://maranas.che.psu.edu/pub/maranas-etal03.pdf

[17] SOLiD fasta format http://marketing.appliedbiosystems.com/images/Product_Microsites/Solid_Knowledge/Download/SOLiD_Data_Format_and_File_Definitions_Guide.pdf

[18] fastq fomat www.ncbi.nlm.nih.gov/pubmed/21252073

[19] a parallel short read reference assembler http://genome.cshlp.org/content/19/6/1117?cited-by=yes&legid=genome;19/6/1117

[20]. Short sequence reads and reduced representation libraries

http://genome.cshlp.org/content/20/2/249.short?cited-by=yes&legid=genome;20/2/249

[21]denovo reference assembly www.illumina.com/Documents/products/.../technote_denovo_assembly.pdf

[22]assembling software's www.clcbio.com/index.php?id=1323

[23]high performance computing www.ukhec.ac.uk/publications/reports/beowulf_paper.pd

[24] challenges in high performance computing http://www.tessera.com/technologies/microelectronics/Documents/Packaging_Challenges_in_High_Performance_Computing.pdf

[25] performance analysis and applications of high performance computing www.lbl.gov/cs/CSnews/cloudcomBP.pdf

[26]hadoop streaming hadoop.apache.org/common/docs/r0.15.2/streaming.pdf

[27]next generation sequencing www.hhmi.org/bulletin/aug2008/pdf/Sequencing.pdf

[28] dna sequencing www.genetics.wustl.edu/rmlab/pdf/Next_gen_DNA.pdf

[29]openMP: http://openmp.org/wp/

12.AUTHORS PROFILE

Veeram venkata siva Prasad did his BTech (Information Technology) in Gudlavalleru Engineering College ,Gudlavalleru , krishna district. Now he perusing his M.Tech. (Computer Science and Engineering) in Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem, West Godavari District, and Andhra Pradesh, India .His area of interest is cloud computing and parallel processing and image processing.

Gunisetti Loshma did her B.E (Computer Technology) and M.Tech (Computer Science and Engineering) and is currently working as an Associate Professor in CSE Department at Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem, West Godavari District, and Andhra Pradesh, India. She has 10years of teaching experience in teaching. She has published number of papers in various journals and conferences.