Distributed Pipeline for Genomic Variant Calling

Richard Xia, Sara Sheehan, Yuchen Zhang, Ameet Talwalkar, Matei Zaharia
Jonathan Terhorst, Michael Jordan, Yun S. Song, Armando Fox, David Patterson
Division of Computer Science
University of California, Berkeley
Berkeley, CA 94720
{rxia, ssheehan, yuczhang, ameet, matei}@eecs.berkeley.edu
{jordan, yss, fox, pattrsn}@eecs.berkeley.edu
terhorst@stat.berkeley.edu

Abstract

Due to recent advances in nucleotide sequencing technology, the cost of genomic sequencing is decreasing at a pace that vastly exceeds Moore's law. The computational methods needed to process short read data are struggling to keep pace; indeed, current sequencing pipelines take days to execute for even a single human genome. In this work, we describe the Big Genomics Inference Engine (BIGGIE), a faster variant calling pipeline designed for modern distributed clusters that supports the efficient processing of thousands to millions of genomes. Moreover, the modular design of our system should foster the rapid development and painless integration of new algorithms. BIGGIE hinges on the observation that we do not need to perform the same computation on all regions of the genome, but rather can first classify the genome into high- and low-complexity regions and subsequently allocate resources accordingly.

1 Introduction

Next-generation sequencing is revolutionizing biological and clinical research [10]. Indeed, sequencing advances will soon allow sequencing a human genome for under $1000. Long hampered by the difficulty and expense of obtaining genomic data, life scientists now face the opposite problem: faster, cheaper technologies are beginning to generate such massive amounts of new sequencing data (Figure 1) that our technological capacity to conduct genomic analyses struggles to keep pace.

Much of genomic data processing is a statistical inference problem, involving inferring the DNA sequence of an individual from short "reads" of DNA produced by sequencing machines. Unfortunately, current pipelines take days to execute for even a single human genome. To support the processing of thousands to millions of genomes, as well as to aid in rapid development of new algorithms, we propose to build the **Big G**enomics **I**nference **E**ngine (BIGGIE), a faster, more scalable, and customizable pipeline for genomic data processing designed to run on clusters.

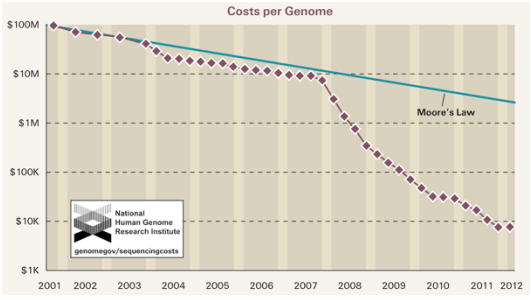


Figure 1: The cost of genome sequencing has been falling faster than Moore's Law.

The key insight we propose to exploit is that variation between genomes of different individuals is non-uniformly distributed, and by parti-

tioning the genome into regions of high and low complexity, we can apply different algorithms based on the complexity of each region. Abstractly speaking, we *classify* the complexity of a region of the genome by the quality of read alignments in that region. Areas with high complexity may have many poor alignments or alignments with many variations relative to the reference genome, while areas with low complexity have mostly good and concordant alignments. Simpler, faster algorithms may be sufficient for processing areas of low complexity, and therefore we can focus our resources and employ more sophisticated algorithms on a much smaller subset of the data. In addition, low-complexity regions give us a natural location to split the processing of the genome across machines. This paper explores both of these benefits by showing that simple, fast algorithms can be effective on a substantial percentage of the genome, and describes how we plan to incorporate them in BIGGIE.

2 Background

DNA sequencing machines analyze cell samples and output data in the form of *reads*, which are short sequences of DNA. Variant calling pipelines aim to convert these raw, noisy short reads into a set of variants that describe differences between a particular sample and the human reference genome. The first step in most pipelines involves aligning the reads to an existing human reference genome.[1] The alignment step is both algorithmically challenging and computationally intensive, though recent advances have been made, e.g., [12], and in this work we consider the alignment step as a black-box.

Next, in part based on the alignment from the previous step, variant calling algorithms are used to identify various types of variants. Variants can be categorized as either single-nucleotide polymorphisms (SNPs, pronounced "snips") or structural variants. A SNP is genomic variation in which a single nucleotide differs from the reference genome. In contrast, structural variants are more complex variations relative to the reference genome such as deletions, duplications, insertions, translocations, etc., and can be as large as thousands to millions of bases in size.[2] Once several genomes have been processed with a variant calling pipeline, subsequent analysis involves population-level inference, including disease gene mapping, modeling demographic history, and genetic ancestry estimation. Indeed, this downstream research crucially relies on accurate variant calls from many individuals.

However, current variant calling algorithms suffer from both low accuracy and high computational costs. Variant callers disagree as much as 20% of the time [4], and it quite difficult, and sometimes not possible, to run many of the leading structural variant algorithms on full genomes due to scalability issues [9]. In this work, we address both the accuracy and scalability issues associated with leading variant calling tools and pipelines, and focus on building a general framework for analyzing genome variation in a distributed fashion.

All types of human genome variation are of great interest to the bioinformatics and genetics communities. Both rare and common SNPs, copy-number differences, and structural variation have been linked to common diseases, and there is much ongoing work to entangle the various contributions. Since SNPs are the most accessible and easily understood measures of genetic variation, they are also used to detect admixture, demography, and population structure, in humans and many other organisms. Easily obtaining fast and accurate variant calls is thus very important for downstream biological analysis. Crucially, we observe that we do not need to perform the same computation on every base. Instead, we first learn high and low complexity regions within the alignment data and subsequently allocate resources accordingly.

3 Overview

The following subsections describe the various steps in BIGGIE as well as how we plan to build them for an efficient distributed system. Several steps rely on domain knowledge for improved work distribution and communication. Moreover, the modularity and efficiency of our system will provide

[1] An alternative data processing path, which we do not directly consider in this work, involves *de novo* assembly of the reads into new genome.

[2] 'Indels' are a third type of variant which can roughly be described as a short structural variant, i.e., less than 50 nucleotides.

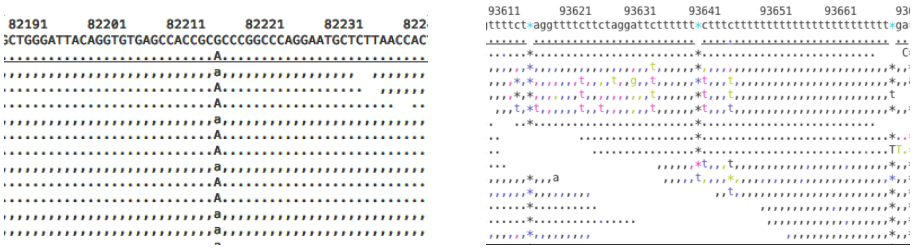
Name	Weight	Description
Substitution	3	# aligned reads showing a substitution with respect to the reference
Insertion	10	# aligned reads showing an insertion with respect to the reference
Deletion	10	# aligned reads showing a deletion with respect to the reference
Low Quality	3	# reads aligned with low map quality (a common indicator of repetitive region)

Table 1: Relative weight of features for computing complexity.

a short turnaround time between algorithm development and benchmarking, allowing researchers to quickly explore the design space of different algorithmic choices and ease of developing new algorithms. We assume that we start with an aligned set of reads for each genome to be processed, in the form of a BAM file [5].

3.1 Alignment Classification

We classify the regions of the alignment as either "high complexity" or "low complexity," allowing us to both operate on different regions in parallel as well as perform different actions based on the quality of the alignments. Regions of low complexity are characterized by few alignment problems, and regions of high complexity are characterized by either few alignments or many poor alignments. We can call SNPs relatively easily in areas of low complexity (see Figure 2(a)), while areas of high complexity might suggest a structural variant (see Figure 2(b)). The intuition is that load-balancing can be improved if we better differentiate between areas of high and low complexity. Moreover, simpler and faster algorithms can be used to process areas of low complexity, and we can focus our resources and employ more sophisticated algorithms on the high complexity regions.



(a) (b)

Figure 2: (a) **Low-complexity region** - Samtools alignment viewer on chromosome 20 of NA12878. The reference sequence is displayed on top, and base mismatches are indicated with an actual letter in the read (upper and lower case letters refer to the strand direction). In this case, there is an obvious SNP with the value 'A'. (b) **High-complexity region** - There are many poor quality alignments with several base mismatches in a relatively small region. Bases written in displayed in different colors represent varying degrees of poor quality bases. Stars (*) represent a gap in the alignment resulting from an insertion called by at least one read. Many of these poor alignments are due to the highly repetitive sequence in the area, and more sophisticated techniques, e.g., targeted assembly, are typically needed to call variants in these regions.

As part of our initial prototype, we categorized complexity using a sliding window of 65 bases around the base of interest. The features we used to calculate our complexity score, along with the relative weights associated with each feature, are described in Table 1. Scoring was normalized by the number of aligned reads, and we chose a threshold of 0.1 for high complexity regions. To test the efficacy of our classifier, we used both real data (Yoruban trio data from the 1000 genomes project [11]) and simulated data from Chromosome 20. The simulated data consisted of SNPs taken from dbSNP randomly inserted into the reference for Chromosome 20, which is 63M bases in length, of which 94.4% are known (not Ns). Artificial reads were then generated from this simulated genome using simNGS [13]. For the simulated data, 2.2% of the non-N bases had a high complexity score, and for the real data 11.5% had a high complexity score. The increase for real data is partly due to indels in the reads, but probably also due to lower and more variable coverage. We hope to further refine this complexity metric and provide the possibility for different types of classification. However, this metric already shows that we can drastically reduce the set of regions we need to analyze in detail.

Algorithm	# false positives	# false negatives
GATK	66	1151
Our results	4	3

Table 2: Results of SNP calling on simulated reads from Chromosome 20.

Algorithm	# SNPs	# Mendelian conflicts
CASAVA	128K	162
Our results	120K	84

Table 3: Results of SNP calling on Yoruban trio data from Chromosome 20.

3.2 SNP Calling

We plan to employ SNP calling algorithms on the low-complexity regions, and in our prototype we implemented a simple consensus-based approach, which roughly builds on the intuition that if a SNP exists, then the majority of the reads which cover that position should agree on a common nucleotide which is different from the reference.[3] This algorithm is embarrassingly data-parallel and requires no interprocess communication, and may be trivially parallelized over a cluster. To verify the effectiveness of performing SNP calling based on separate high and low complexity regions, we ran our simple SNP caller only on low-complexity regions using the experimental data described in Section 3.1. Our SNP caller reports fewer false positives and runs much faster than the GATK, an existing tool which can call SNPs (2 minutes for our SNP caller versus roughly 2 hours for GATK). In Table 2, we compare the false positive and false negative rates of GATK with our SNP caller on simulated data. In Table 3, we compare the number of SNPs and Mendelian conflicts of CASAVA [14] with our SNP caller on real data, using the Yoruban trio to determine Mendelian conflicts.

There are several techniques used for SNP calling, and the modularity of our system allows us to easily switch between these different options. More advanced variant detection algorithms exploit common ancestry among the individuals being studied to pool statistical strength among samples. These estimators have been shown to perform better than the naïve approach described above, particularly in the case of missing data and/or low coverage sequencing experiments [7]. Accordingly, we plan to design our system to be flexible enough to support communication between nodes during this step. After analyzing several existing SNP callers, such as ones which create a hidden error model or ones which reuse information across multiple genomes, we hope to better understand the data dependencies and communication patterns of common SNP callers.

3.3 Structural Variant Calling

After processing the low-complexity regions, we are subsequently left with a set of disjoint high-complexity regions, and we can process these regions in an embarrassingly parallel fashion, using more statistically rigorous and computationally intensive techniques such as targeted assembly [1]. Accurate calling of structural variants is a challenging area of bioinformatics that is not as well-studied as SNP calling, and is hampered by significant workflow overhead. By targeting areas in the genome with potential structural variants and providing a modular platform to develop and test new and existing algorithms, we hope that our system can foster algorithmic development in this area.

4 Related Work

There are several existing SNP (and other variant) callers such at SAMtools mpileup [5], GATK [3], and Platypus [8]. SAMtools has largely been replaced by GATK as the industry standard. Although GATK does have the ability to perform in a distributed environment, this functionality is not straightforward for the typical user, and usually GATK ends up being very slow. Although GATK provides more functionality than simple SNP calling, we hope that in this respect our system is faster and more lightweight than GATK. Most variant callers consider each candidate SNP independently, but the caller Platypus aggregates information within a window of the genome to take into account local information In addition, there are many programs to call more complex structural variation, although the field is less developed than SNP calling. BIGGIE will ideally set us up to efficiently tackle this problem later on.

[3] More specifically, we use the standard 80-20 approach [3] to deal with diploid genomes and differentiate between homozygous and heterozygous SNPs.

References

[1] Ma'ayan Bresler M., Sheehan S., Chan A., and Song, Y., "Telescoper: *de novo* assembly of highly repetitive regions." *Bioinformatics* (2012), 28: 311-317.

[2] Barnett D., Garrison E., Quinlan A., Strömber M., and Marth G., "BamTools: a C++ API and toolkit for analyzing and managing BAM files." *Bioinformatics* (2011), 27: 1691-1692.

[3] DePristo M., Banks E., Poplin R., Garimella K., Maguire J., Hartl C., Philippakis A., del Angel G., Rivas M., Hanna M., McKenna A., Fennell T., Kernytsky A., Sivachenko A., Cibulskis K., Gabriel S., Altshuler D., and Daly M., "A framework for variation discovery and genotyping using next-generation DNA sequencing data." *Nature Genetics* (2011), 43:491-498.

[4] Haussler D. Keynote Address: The UCSC Cancer Genomics Hub. (2012) https://www.usenix.org/conference/osdi12/keynote-address.

[5] Li H., Handsaker B., Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. and 1000 Genome Project Data Processing Subgroup, "The Sequence alignment/map (SAM) format and SAMtools." *Bioinformatics* (2009), 25: 2078-9.

[6] National Center for Biotechnology Information (US), (2005-), http://www.ncbi.nlm.nih.gov/projects/SNP/.

[7] Nielsen, R., Paul, J.S., Albrechtsen, A., and Song, Y.S. "Genotype and SNP calling from next-generation sequencing data." *Nature Reviews Genetics* (2011), 12:443-451.

[8] Rimmer A., Mathieson I., Lunter G., and McVean G., "Platypus: An Integrated Variant Caller" (2012), www.well.ox.ac.uk/platypus.

[9] Curtis, K., personal communication, (2012).

[10] Schuster, S. "Next-generation sequencing transforms today's biology." *Nature Methods* (2008), 5: 16-18.

[11] The 1000 Genomes Project Consortium. "A map of human genome variation from population-scale sequencing." *Nature* (2010) 467:10611073.

[12] Zaharia M., Bolosky W., Curtis K., Fox A., Patterson P., Shenker S., Stoica I., Karp R., Sittler T., "Faster and More Accurate Sequence Alignment with SNAP." (2011) http://arxiv.org/abs/1111.5572.

[13] Massingham T. simNGS and simLibrary – Software for Simulating Next-Gen Sequencing Data. (2010) http://www.ebi.ac.uk/goldman-srv/simNGS/.

[14] CASAVA. (2012) http://support.illumina.com/sequencing/sequencing_software/casava.ilmn.