Storing and querying a genome in a blockchain

Sarah Wagner
Mentor: Gamze Gursoy
Advisor: Mark Gerstein

Abstract

The genomic characterization of millions of individuals promises to be immensely useful for medical research. Having a larger number of studied individuals is almost assured to boost statistical power for future discoveries; therefore, genomes from increasingly more individuals likely will be sequenced going forward. One of the main barriers to expanding sequencing is the ability to store and compute on this large-scale data. While cloud storage and computing offer solutions to maintain and compute on ever increasing amount of data, the security, data integrity and robustness vulnerabilities such as single-point-of-failure have not yet been addressed. This project developed novel tools for decentralized storage and computing of genome sequencing data on private Blockchain networks.

These tools provide open-source Blockchain-based storage and computation tools for advanced genomic analyses such as variant calling. This provides two tools: the first is called SAMChain, and allows users efficiently store sequence alignment maps in a blockchain. The second is called SCtools and provides analysis functions such as (a) querying, (b) read streaming, (c) depth analysis, (d) pile-ups for variant calling, and (e) re-creating BAM files. This was achieved by using the MultiChain platform that provides support for creating data streams in blockchain. Multiple data streams were created, and these streams were divided intelligently and for allow for hashing of the inserted data, allowing for fast retrieval of the genomic data. Although inserting BAM files into a blockchain for efficient query typically requires an overwhelming amount of data and time, by simple format manipulations, we also reduced the amount of data to be stored.
Introduction

The understanding of genomic landscape from millions of different individuals is essential to characterize and investigate rare diseases and genotype-phenotype associations. Studying more individuals boost the statistical power for better association discoveries and discovery of rare genetic diseases. Therefore, it is unavoidable to envision more and more individuals will be sequenced going forward. In addition to increasing statistical power, modern advances in personalized medicine result in increased number of individuals willing to sequence their own genome for disease-risk predictions as well as ancestry analysis, which also brings the genomic data and predictive tools to health care. It is almost inevitable to think that genome sequencing will become part of routine clinical care in the future.

The growth in amount of genomic data faces bottlenecks from computational requirements and capacity of servers. NIH and many other institutions are moving towards cloud computing based services. However, security is one of the concerns when dealing with cloud-based storage and data analysis tools as cloud storage and computing are based on centralized architecture that are vulnerable to single point of failure. Since genomic data is in the core of understanding human health, the integrity and security should be a priority when providing solutions to storage and analysis. Securely maintaining the genomics data is imperative, because if genomic sequences are corrupted, changed, or lost, then the biomedical discoveries and patient care would not be possible.

An ideal implementation of genomic data dissemination would protect from both loss and manipulation. To protect against manipulation and single point of failure, researchers are increasingly moving away from relying on trusted parties or authority figures and instead placing that trust in an algorithm such as Blockchain technology (Figure 1). Blockchain employs an ideal implementation due to three key properties: decentralization, immutability, and security. Decentralization prevents a single entity from controlling the data; immutability guarantees that data cannot be altered; and security is ensured by protecting accounts with enhanced cryptographic methods. There are already personalized medicine start-ups with the promise of keeping individuals' genomes in blockchains.
A recent review by Ozercan et al outlines the current status of the commercial and academic proposals to share genomics data using blockchain platforms. Among these newly emerged platforms, gene-chain and Zenome provide solutions for genomic data distributions for commercial use of genomic data. Nebula genomics create tokens as an incentive for individuals to share their genomes with entities using the blockchain technology. CrypDist enables synchronized data sharing among researchers with protection against data removal. This technology allows for sharing of large genomic data files such as reference aligned genomes (BAM files). However, due to the inefficiencies in inserting large datasets to a blockchain, they instead share the links to the data. Although, all of these platforms allow secure storage and sharing of the genomic data, none of them offer solutions to perform computation on the data stored in a blockchain. One of the biggest caveats of blockchain technology is the inefficiency of storing and querying data due to the potential for chains to reach large sizes. The required storage space and computational power of blockchains is greater than a centralized database application due to the blocks needing to contain information of the rest of the chain before them. The decentralized system also create a higher latency during storage and retrieval of data.

Here we introduce a first open-source, proof-of-concept application of storing and analysis of BAM files in a Blockchain. We provide two tools: First one is called SAMchain that allows users efficiently store sequence alignment maps in a blockchain and second one is called SCtools that provides analysis functions such as (a) querying, (b) read streaming, (c) depth analysis, (d) pile-ups for variant calling, and (e) re-creating BAM files. We achieve this by
using MultiChain platform that provides data streams with key:value pairs property. We created multiple data streams to use them as hash tables for fast retrieval of the genomic data. Although inserting BAM files into a blockchain for efficient query typically requires an overwhelming amount of data and time, by simple format manipulations, we also reduced the amount of data to be stored.

Project Description

MultiChain data streams allow a blockchain to be used as a general purpose database. The data published in every stream is stored by every node in the network. Each data stream on a MultiChain blockchain consists of a list of items. Each item in the stream contains the following information, as a JSON object:\cite{multichain}: A publisher (string), key:value pairs (between 1-256 ASCII character, excluding whitespace and single/double quotes) (string), data (hex string), a transaction ID (string), blocktime (integer) and Confirmations (integer). The idea is that when data needs to be queried or streamed, it can be retrieved by searches using the key:value pairs.

We took an approach that will potentially maximize the efficiency of storing and querying data. The goal is to store minimum amount of data while reducing the time and memory cost of the query and increasing the utility of the stored data. For that, we first separated mapped and unmapped reads by using the FLAG attribute of SAM files. We generated a stream for unmapped reads to store directly every line in the unmapped set as a stream item. For mapped reads, to achieve scalable storage and computation, we first categorized the information in a BAM read based what a user might want to query for. We decided that the most common user queries are the query by read name, query by genomic location and query by mapping quality. These three features are used to index the data streams in the blockchain. We then binned the BAM file in three ways: binned by the genomic location, binned by hex converted read name and binned by mapping quality. Binning is done for the purpose of avoiding large data streams for efficient querying. After binning, the information in each BAM read is inserted in four different kind of streams: main streams that are composed of location stream, readname stream, mapping quality stream and a shadow stream (Figure 2a). The fixed streams indexed with these features provide rapid retrieval and query of the data. The features that can be searched by a user are stored as key:value pairs such as read name, genomic location, mapping quality, and alignment score. Data section of streams are composed of CIGAR string of the BAM file, MD:Z tags and a feature called MODCIGAR, where we find out the difference between the reference sequence and the read sequence with respect to the CIGAR string. This is a much smaller string than the sequence of the reads. This allows us to reduce the amount of data to be stored in a blockchain. Other features of a BAM read are stored as is in the shadow streams as they are accessed by users rarely. Keeping the rare access features in separate streams
allow us to retrieve the necessary data quickly (Figure 2b). The information in these streams are complementary to the main streams and can be pulled for reconstruction of the BAM files. BAM attributes FLAG, RNEXT, PNEXT, TLEN and QUAL are stored in the data section of the shadow streams. The key section of the shadow streams have the same information as the main streams for indexing purposes. We compared the per node storage requirement of a blockchain with a BAM file.

Figure 2

We developed SCtools to extract information from SAMchain for downstream analysis. We provide a code base that has the ability to query on a blockchain. The key:value property of the data streams in MultiChain2.0 with the ability to query on multiple keys provide an opportunity to extract data from a blockchain without the need for costly calculations. Our query module can retrieve data from a chain based on the queried read name, mapping quality, alignment score, and the position on the reference genome. For example, if a user query on a chain for the reads that are mapped to a genomic region, our query module first finds the data streams that are binned using genomic location and then the exact data stream bin that contains the queried region and pulls the data from that particular stream.
This reduces the query time significantly. Similar approach is taken when queried by read name and mapping quality. Below, we described the modules of our query system.

**Read streaming:** This module allows users to retrieve the entire read based on the queries mentioned above. For example, if a user has a list of read names to stream, the module will reconstruct the reads and stream them as following. It will first locally download the stream with the unmapped reads, convert it to a dataframe and query and return the reads in the list. For the mapped reads, it will query the streams and shadow streams with the read names on the chain without locally downloading it, transform the query results to SAM format and retrieve them. Transformation is based on converting hex into strings, parsing them, querying the sequences from the reference genome and modifying them based on the MOD string.

**Depth analysis:** This module is to query sequencing depth for a given position or a range of positions in the genome. For example, when a user prompts a location on the genome to query the sequencing depth, this module first find the stream that has the reads from the genomic location. It will then check whether the location will be in the flanking region of any other stream. After finding the relevant streams, it queries them to list the stream items with all possible start locations that a read covering the queried location can have. After checking the CIGAR of these reads to make sure they do not have clipping or deletion in the queried location, it returns the resulting number of reads as the depth of the queried location.
**Pile-ups for variant calling:** A great function that samtools provide is the pile-ups for a queried location or all of the locations in the genome. Pile-up files contain the number of reads that mapped to a location, the reference allele for that location, and the sequenced nucleotide in each read for that location. It allows users to visualize the genetic variation and calculate allele frequencies for the variants. We developed this functionality in an effort to create pile-up files from the blockchain. This functionality combines the depth analysis, read streaming as well as querying from the reference genome and return results in the pile-up format.

**Re-construction of alignment maps:** This functionality is for the users to have a copy an alignment map. Based on the desired alignment format (SAM / BAM /CRAM), this module combines the read streaming with unaligned reads and header and re-constructs the alignment files.

We calculated the time requirement of read streaming, depth analysis and pile-ups empirically. We also compared the md5 of our re-constructed BAM files with the original ones and ensured that we generate the exact same alignment files.

We envision a real-world scenario in which individuals create private blockchains to store their personal genomes to share with their healthcare providers. By simply giving ssh access, the healthcare providers and associated genetics researchers can stream or query patients’ genomes without the need of downloading or transferring data. This reduces not only the risk of data corruption but also access to the private data by adversaries. Blockchain provides immutability such that the data cannot be altered or changed either intentionally or by accident.

Our framework is the first open-source application to allow query and streaming of data from blockchain to the best of our knowledge. This is a substantial improvement over the current biomedical applications of blockchains. Currently, the security of the data is provided by blockchain but the computations on the genomes are based on plain text. However, in the case of privacy concerns, our framework can also be extended to store the data in a homomorphically encrypted fashion in the data streams. However, this will add storage and computation overhead to the solution.

Our blockchain solutions can be generalized to other large-scale data storage and querying problems beyond BAM files. Data including but not limited to electronic health records, vcf files from multiple or single individuals, and somatic mutation datasets from cancer patients can be indexed with our hashing schemes and stored in blockchain while rapid and partial retrieval of the data can be achieved.