A Data Management Solution for the DNA App Store

Konstantinos Xirogiannopoulos
University of Maryland
Department of Computer Science
Email: kostasx@cs.umd.edu

Virinchi Srinivas
University of Maryland
Department of Computer Science
Email:

***Abstract*—There is starting to emerge widespread interest in fleshing out the vision of a DNA App Store; a platform for developing applications that aim at providing users with specific tidbits of insight about themselves based on their genotypes, in the context of the genotypes of multiple other individuals. In order to provide these insights efficiently and accurately, applications would need to leverage the rapidly growing data-pool of sequenced human genomes. Currently however, when it comes to genotype datasets in particular, data is stored in *flat files* and has to be manually loaded in memory every time so it can be queried in an ad-hoc and sometimes suboptimal fashion. In this paper we propose a data management pipeline towards making this vision of a DNA app store not only possible but also highly versatile and scalable.**

I. Introduction

In recent years, the fields of Computational Biology and Genomics have come a long way in developing technologies towards being able to sequence and analyze the human genome. As these technologies progress, so does our understanding of human genetics as well as the insight that can be gained from genomic analysis. More specifically, there has been significant interest in performing *variant calling* towards identifying and later attempting through a series of analyses, to explain variations in the genomes of different individuals. A wide array of companies have also started to emerge, whose goal is to find common as well as differentiating traits between individuals based on their genotypes. This has sparked the vision for creating the DNA AppStore; an infrastructure that leverages genomic sequencing datasets through a series of applications that would allow for specific types of queries against one's genome, that are usually answered using the pool of genomes collected from as many individuals as possible. These applications could range from simple retrieval and similarity queries, to complex analytics for one or more individuals at a time. Any application in such an environment would require efficient "*sequence once, query often*" data access to a large database filled with a variety of different data around the genomes of many individuals. For the purposes of this study, we assume that these queries are typically going to be queries on top of genotype data resulting from variant calling, and we focus on the efficient structured storage of this data.

II. Current state of the DNA App Store

The high-level philosophy behind the value of the DNA App Store is that "*The genome is an asset that you have for life, and youll keep going back to it.*" [1]. Companies that seem to be interested in the idea of the DNA App Store include Helix, 23andMe, Human Longevity (HLI), and Ancestry. Ancestry and 23andMe are currently offering genome sequencing services for their users while HLI is mostly focused on analytics of already existent genome data. *Helix* has plans to offer more competitive sequencing prices than either of the two aforementioned companies but is still at very early stages in development. Human Longevity is reportedly building the largest genotype and phenotype database in the world and is hoping to enable personalized disease treatment through leveraging these databases. *23andMe* provides carrier status reports, ancestry, wellness and traits reports based on the sequenced genomes stored in their database. They are also the only company that currently exposes a REST API[1] for developers to access a small portion of their datasets. Through their endpoints, developers are able to retrieve all SNP-chips for a specific user's genome or query SNPs in a particular location in the genome. Given a profile_id, users can also query phenotype information, as well as information about the different haplogroups the individual belongs to. We unfortunately have no further insight on the back-end storage infrastructure any of these companies are utilizing in order to enable their services.

Some of the prevalent types of queries that seems to be of interest for the DNA App Store include queries regarding genomic similarity. One example would be "How similar am I genetically with X" type queries. Albeit less optimal than ad-hoc solutions like GQT, SQL does allow for queries like this while providing a rich platform for defining user-defined aggregates and many different variations of the above query. One variation would be "Show me the individuals that I have similarity $> 60\%$ with". Another type of query would be specific string querying on top of the genome; queries of the type "Do I have the YY gene?" or if a specific character trait is correlated with a particular gene variant, users will want to ask "Do I have this variant/mutation?" These types of queries are very natural for classic databases and are some of the queries we expect to be more commonplace in a system like the DNA App Store.

[1] https://api.23andme.com/

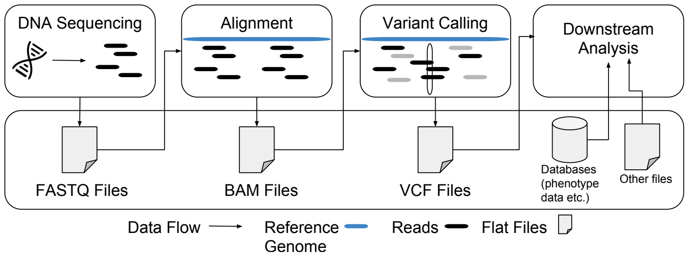

Fig. 1. Current workflow using flat files (figure based on Figure 1 in [2]) **We need something here to show what happens to the VCF file data currently in analysis (GQT, samtools etc.)**

III. Related Work

There has been a body of recent work that explores using *database systems* in order to properly organize and efficiently query genomics data.

Dorok et al. [2], [3] proposes a *base-centric* relational schema over the standard read-centric notion of storing raw sequence data, which avoids the on-the-fly cost of splitting reads when trying to search for specific bases. Their approach also leverages the inherent compression capabilities of column stores that do not require decompression at processing time. The fact that they are however using in-memory databases for this, although will increase query performance, will not scale to the volume of these datasets as cost of main memory is still quite far from reaching that of disk storage. Another work by Dorok et. al. [4] utilizes user-defined aggregation functions in order to compute variant calls within the database on top of this schema using a combination of SQL and user-defined aggregation functions for implementing heuristics towards determining a genotype for each site in the requested range. This approach however significantly limits variant calling capabilities in the general case in that it renders a wide diversity of variant calling algorithms unusable, unless they are entirely re-implemented as user-defined functions on top of this schema.

The approach by Röhm et. al. [5] also towards storing raw sequence data into an RDBMS integrates genotyping by utilizing user-defined functions into a DBMS but instead chooses to store genome sequences as strings which require on-the-fly splitting via other user-defined functions. This work is mainly focused on trading off support for external tools for strict data modeling. The main purpose of this system is to store data from every stage of analysis including reads, alignments, meta-data about each of the reads, as well as the output of tertiary analysis (here they're storing consensus calling data). They are however using very small datasets to prove their points (in the order of a few GBs) and are not focusing on how this schema would scale. This line of work is in some ways however related to our goal of properly structuring genomics data towards higher availability and easier analysis however we focus specifically on genotype data that has resulted from variant calling.

There have also been a variety of systems [6], [7] that propose storing SNPs in structured relational databases which allow for rich and efficient querying and analysis using SQL. The challenge with this architecture is the fact that it logically must include a single table containing every single variant that is observed for every individual as well as their homogeneity. This the space complexity of this single relation is of the order $v * i$ where v is the number of unique variants and i is the number of individuals in the dataset. Each work chooses to deal with this challenge differently; [6] partitions this table into a large set of smaller identical tables, without however specifically explaining whether or how these tables would be distributed across a cluster. SNPpy [7] on the other hand, endorses relational databases for their data validation mechanisms inherent in their data model as well as their ability to store multiple types of meta-data that can naturally be merged (joined) with the snp data towards more complex analyses. Their results were however computed on very small simulated datasets (around 1.5M SNPs), so they did not encounter the aforementioned issue.

Moreover, there have been a few data warehouse solutions such as Atlas [8] and BioWarehouse [9]; these solutions use disk-based relational databases but focus their efforts towards the issue of data integration across various different datasets relevant with bioinformatics and their solutions are not optimized for genomic data.

More recently, the Genotype Query Tools (GQT) [10] system has been proposed which enables fast queries over highly-compressed genotype data from VCF files. GQT however focuses specifically and is built in a way to efficiently support a specific subset of *individual centric queries*. When it comes to variant-centric, or more complex queries such as searching for variants in specific chromosomal regions of the individuals, GQT breaks down. The reason is because GQT does not include any metadata associations with each variant they are storing in their index, making their approach very limited for more complex queries on top of genotype data.

There are also other open-source projects trying to deal with genomics data management, one of which is GenomicsDB [**?**], which is being developed by Intel. This project uses the TileDB database (built for efficient sparse matrix/array storage) to store variant data in a 2D array, queried using a command line tool (similar to GQT) and returning results in a JSON format.

IV. Proposed Architecture

A high level representation of the current pipeline architecture is depicted in Figure 1. consists of Currently, genotype data is stored in flat files called VCF (Variant Call Format) files. These VCF files are then loaded into memory and analyzed using systems like *samtools/bcftools*[2] for analysis. When it comes to variant-centric analysis, these tools start to break down due to the ever-expanding volume of data flowing into these flat files. This has caused ad-hoc solutions that consequently limit the scope of analyses possible on top of these solutions by using e.g. compression schemes in order

[2]https://samtools.github.io/bcftools/bcftools.html

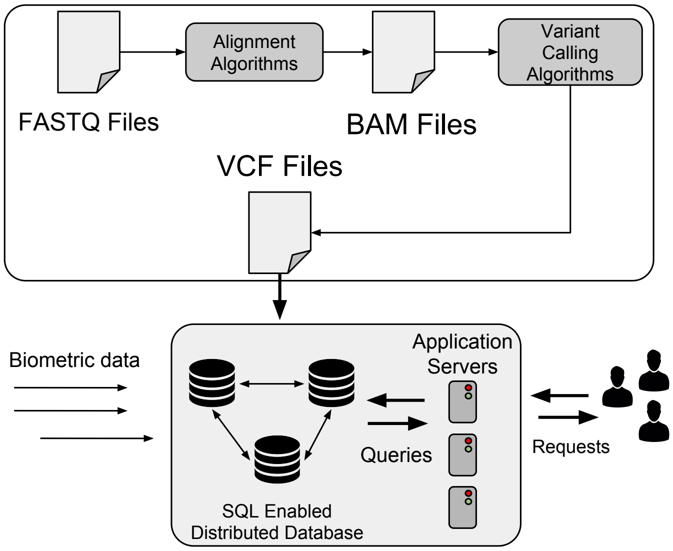

Fig. 2. Our proposed architecture for the DNA App store

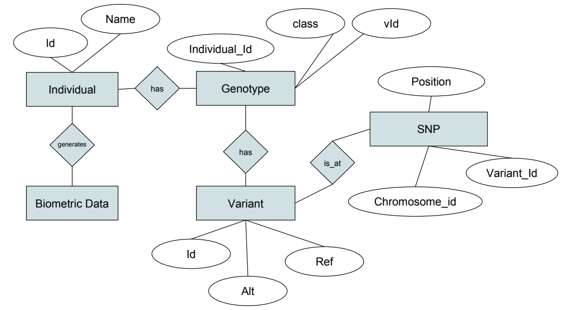

Fig. 3. Our proposed schema for a distributed database for the DNA App Store

to be able to deal with the sheer volume of these datasets. In the context of a DNA App Store however, various applications will require have access to a wide diversity of datasets, that they will undoubtably want to *combine*, and apply extensive analysis on top of. The first step in that direction is bringing a higher degree of *structure* to the genotype data that currently exists which will facilitate in analyzing it in a *scalability* and *extensible* manner.

Our vision for how a data management system that would enable such rich analytics on top of genotype data is one that is optimized for the write-once, query-often demands of the DNA App Store. The general idea is simple; apply ETL (Extract Transform and Load) over standard VCF files, insert them into a specific relational format dictated by a schema and then load it into a *distributed database management system*.

Distributed databases are storage systems in which the data does not reside in a single machine. These systems provide replication across multiple nodes in a network which in turn offers both *high availability* and *fault-tolerance*. This means that the data store can handle a large amount of requests, even in situations when requests are hitting the same data at the same time, since most of the data will be replicated and will be able to be accessed from more than one machine at a time. The proposed schema is seen in Figure 3. Genotype data has the unique characteristic that in order to enable complex analytics on top of it, one needs to store information about every single unique *variant* for every *individual* in one enormous logical relation; in our case that's the `Genotype` table. Using a distributed database, we can choose how to *partition* this gigantic table in various ways across multiple machines.

We propose partitioning this `Genotype` table by `individual_id` in a way such that all variants for a specific individual are stored *in the same machine*. Advanced compression techniques already existent in most database systems will allow for further compressing the massive genotype table at every machine.

As mentioned, common queries would include finding the number of common genotypes between two individuals by using SQL; the following query will return how many variants Konstantinos Xirogiannopoulos has in common with Michael Jordan:

```
With MJ as (Select G.class, G.vId
From Genotype G
Where G.Individual_id = (Select I.id from
Individuals I where name='Michael Jordan'))
Select Count(G.Vid) as common-variants
From Genotype G, MJ
Where G.Individual_Id = (Select I.id from
Individuals I where name='Konstantinos
Xirogiannopoulos') and
G.vId = MJ.vId and G.class = MJ.class
```

Note that the similarity check does not necessarily have to be done in the database. Applications could simply query for the genotypes of these individuals and conduct their own analysis tailored to their purpose.

Because this is a write-once, read-many architecture, there are no issues of maintaining consistency in this context, since the genotype data is *static*. Consistency would have been an issue if there were also a large number of incoming updates to the genome data, however in this case we have no updates to the genotypes that may interfere with the queries made by the applications since an individuals variant data does not change once identified and inserted into the database.

Considering that there could potentially be as many as $100,000,000$ variants in the entire human genome, if we were to maintain the genotypes for as many as $1,000,000$ individuals, and store say 8 bits per individual-variant pair (to account for multiple alleles, or complex structural variants), that would require $8 * 10^14$ bits, or 100 Terabytes, before applying any standard compression techniques native in many database systems. We obviously don't have this volume of data available yet, but is steadily flowing in as more genomes are sequenced. Disk storage is also increasingly becoming more inexpensive, and it is not unusual for machines to have a few Terabytes of storage. By using a distributed architecture like the one proposed it is therefore apparent that a system like this is *horizontally scalable*. Increase in data volume to extents much higher than this can be dealt with by simply throwing more machines at the problem.

V. Conclusion and Future Directions

There are of course many more things to take into account like the actual rate at which queries are expected to hit this database, how we could incorporate biometric data into the equation, what interesting queries can be made by doing so and what potential problems arise in these cases. Moreover there could also be security as well as data ethics issues associated with the DNA App store as data anonymization is definitelly non trivial and there may be interest in users genome data being completely anonymous but also that some of their phenotypes are concealed. It is difficult to pinpoint whether these will be real issues or to estimate their magnitude in regards to the users as well as the infrastructure. Another future direction is incorporating more complex structural variants inside this database in ways that would enable combining these datasets with the rest of the data in the database (probably using SQL), towards potentially more insightful analytics.

We have proposed a database schema and architecture that would allow for structured queries of a large and extensible unified database system that we believe is a practical first step towards organizing the back-end for the vision of the DNA App store. We believe that allowing for SQL support over structured datasets would allow for the realization of this vision as it would allow a standard, structured way of accessing and merging genomics data for complex analysis.

Acknowledgment

References

[1] "DNA App Store: An online store for information about your genes will make it cheap and easy to learn more about your health risks and predispositions." https://www.technologyreview.com/s/600769/10-breakthrough-technologies-2016-dna-app-store/, accessed: 2016-04-29.

[2] S. Dorok, "The relational way to dam the flood of genome data." in *SIGMOD PhD Symposium*, 2015, pp. 9–13.

[3] S. Dorok, S. Breß, J. Teubner, and G. Saake, "Flexible analysis of plant genomes in a database management system." in *EDBT*, 2015, pp. 509–512.

[4] S. Dorok, S. Breß, and G. Saake, "Toward efficient variant calling inside main-memory database systems," in *Database and Expert Systems Applications (DEXA), 2014 25th International Workshop on*. IEEE, 2014, pp. 41–45.

[5] U. Röhm and J. Blakeley, "Data management for high-throughput genomics," *arXiv preprint arXiv:0909.1764*, 2009.

[6] M. Bouffard, M. S. Phillips, A. M. Brown, S. Marsh, J.-C. Tardif, and T. van Rooij, "Damming the genomic data flood using a comprehensive analysis and storage data structure," *Database*, vol. 2010, p. baq029, 2010.

[7] F. Mitha, H. Herodotou, N. Borisov, C. Jiang, J. Yoder, and K. Owzar, "Snppy-database management for snp data from genome wide association studies," *PloS one*, vol. 6, no. 10, p. e24982, 2011.

[8] S. P. Shah, Y. Huang, T. Xu, M. M. Yuen, J. Ling, and B. F. Ouellette, "Atlas–a data warehouse for integrative bioinformatics," *BMC bioinformatics*, vol. 6, no. 1, p. 1, 2005.

[9] T. J. Lee, Y. Pouliot, V. Wagner, P. Gupta, D. W. Stringer-Calvert, J. D. Tenenbaum, and P. D. Karp, "Biowarehouse: a bioinformatics database warehouse toolkit," *BMC bioinformatics*, vol. 7, no. 1, p. 1, 2006.

[10] R. M. Layer, N. Kindlon, K. J. Karczewski, A. R. Quinlan, E. A. Consortium *et al.*, "Efficient genotype compression and analysis of large genetic-variation data sets," *Nature methods*, vol. 13, no. 1, pp. 63–65, 2016.