2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Representing Variant Calling Format as Directed Acyclic Graphs to enable the use of cloud computing for efficient and cost effective genome analysis

Sanna Aizad, Ashiq Anjum
Department of Computer Science
University of Derby
Derby, UK
s.aizad@derby.ac.uk
a.anjum@derby.ac.uk

Rizos Sakellariou
School of Computer Science
University of Manchester
Manchester, UK
rizos@manchester.ac.uk

Abstract— **Ever since the completion of the Human Genome Project in 2003, the human genome has been represented as a linear sequence of 3.2 billion base pairs and is referred to as the "Reference Genome". Since then it has become easier to sequence genomes of individuals due to rapid advancements in technology, which in turn has created a need to represent the new information using a different representation. Several attempts have been made to represent the genome sequence as a graph albeit for different purposes. Here we take a look at the Variant Calling Format (VCF) file which carries information about variations within genomes and is the primary format of choice for genome analysis tools. This short paper aims to motivate work in representing the VCF file as Directed Acyclic Graphs (DAGs) to run on a cloud in order to exploit the high performance capabilities provided by cloud computing.**

Keywords- DAGs; VCF; genome analysis; cloud computing

I. INTRODUCTION

The Human Genome Project, started in 1990 and concluded in 2003, aimed to sequence the entire human genome. It resulted in an official gene map, also known as the "reference genome" which consists of 3.2 billion base pairs present in a human genome [1]. Although this boosted genomics in unimaginable ways, the existing linear gene map looks at only one way the human genome could be viewed. The current research in genomics demands a "map" or arrangement of the genome such that thousands of genomes may be combined to effectively solve multiple types of research problems. The scientific community has come to an understanding that the best way round it is to map a "pan genome" [18] which represents all the variations in genomes with respect to the reference genome. This would help give the larger picture of genomes with respect to each other as compared to the traditional view of looking at genome variations with respect to their nucleotide positions mapped to a reference genome.

The stage has been set by initiatives such as 1000 Genomes Project [2] and The 100,000 Genomes Project [3] along with an advancement of sequencing technologies. Companies like Illumina [4] use Next-Generation Sequencing (NGS) to sequence genomes in a cost- and time-effective manner. The NGS information is distributed in Variant Calling Format (VCF) [5], [25]. The VCF files are used for diagnosing genetic disorders by running the variations through a series of analysis in addition to clinical annotations to bring out meaningful insights [6]. There are different platforms available such as the Omacia Platform [6] and CAPER 3.0 [7], which address data-intensive analysis on a cloud-based environment.

As a standard introduced by 1000 Genomes Project [2], the Variant Calling Format (VCF) [5], [25] stores small-scale variant information such as that about SNPs, insertions and deletions. In other words, the VCF file contains genetic variation data/DNA polymorphism data. The VCF file has many advantages, the most important of which is that it is standardized [5]. The VCF stores only the variations along with the reference genome, eliminating redundancy of data by not storing the portions of the genome which are the same as the reference genome. At the same time, it is flexible enough to contain structural variants. The variations are listed along with the reference haplotype allowing VCF to express any type of variation. It explicitly states the type of variation along with the sequence of variation, as well as the genotypes of multiple samples, if they are available, for the particular variation. Since VCF is usually associated with Next-Generation Sequencing data (such as that generated from the 1000 Genomes Project [2]), it is, therefore, the primary format choice for genome analysis tools.

However, the VCF is a plain text file containing highly detailed information. It is somewhere between human readable and machine readable [25] and, therefore, understanding the data that a VCF file contains is in itself a challenge. Usually, genome analysis requires looking at more than one VCF file whereby a personal computer may struggle to load and process the data in its memory which is why, analysis tools like Omicia [6] and CAPER 3.0 [7] turn to cloud computing.

Here, we explore the possibility of representing the VCF files as a graph model, thus making it easy to move the data in memory and, thus, eliminating the need to load and read VCF files every time analysis is to be done. It will also make possible to represent every possible path within a genome in memory, making analysis faster, less expensive, and more accurate.

978-1-5090-6611-7/17 $31.00 © 2017 IEEE
DOI 10.1109/CCGRID.2017.116

II. TYPES OF GRAPHS

Different types of graph representations of the genome can be found in literature. Listed below are some of these genome graphs and the data they represent.

A. Sequence Graphs

A Sequence Graph allows representation of different orientations of homology of a genome [26]. It lets a multiple sequence alignment capture structural variations which may go undetected in a matrix representation. Currently, the main application of sequence graphs is in multiple sequence alignment where each genome is represented as acyclic breakpoint graph [8].

B. Population Graphs

A Population graph captures variations between many individuals. Multiple sequences are represented as a graph [27] which is aligned to new reads containing variations. Since coordinates have to be extracted from a reference genome to get an alignment, the reference coordinates are carried to the graph. This type of graph may miss structural variations such as novel insertions in a reference genome.

C. Assembly Graphs

A set of unaligned sequences can be represented as an assembly graph. [28]. Assembly graphs are popularly used to assemble fragments of a single genome to get the original sequence in sequence technology. de Bruijn graphs have also been used for sequence assembly.

D. Variation Graphs

A Variation Graph (VG) is able to represent many genomes in the same context by aligning a sequence graph of a genome to the variations. The principle of Variation Graphs is same as Sequence Graphs, as they also look to link successive sequences through directed edges while representing the sequence as nodes.

E. Compressed de Bruijn Graphs

The relationship between genomes is graphically represented using maximal exact matches (MEMs) using compressed de Bruijn Graphs [9]. It helps reveal highly conserved or segregated sequences across a population which play an important role in determining phenotypical roles.

III. REPRESENTING THE VARIANT CALLING FORMAT AS A DIRECTED ACYCLIC GRAPH (DAG)

Cloud computing is being used to solve large-scale problems in science [16]. With exabytes of data being generated by genomics, researchers are turning towards cloud computing to answer important biological questions. Applications which are run on clouds can be modeled by Directed Acyclic Graphs (DAGs) representing workflows [19] of tasks to be run on the cloud [16]. A DAG uses its vertices to represent the number of jobs that need to be processed in order to complete a task, while its edges are used to define the precedence constraints. This model of workflow works well for High Performance Computing [17]. There are a number of studies available [9], [10], [11] which show that high performance can be achieved when executing DAG-based workflows on parallel clusters [12], [19].

The challenge here is to represent the VCF files as a graph model which can be converted to DAG so that high performance may be achieved for genome analysis using cloud computing. De Bruijn graphs have been used in genome assembly as well as in population studies to represent overlapping information. The de Bruijn graph is a directed graph which is able to capture variations as disjoint cycles in the graph. Assembly graphs can be represented as variation graphs, by converting a de Bruijn graph to a variation graph. Since, one way or another, it is possible to represent the different types of genome graphs as variation graphs, the question arises whether it is possible to represent a VCF as a variation graph, and then convert the variation graph to a directed acyclic graph?

Given the information present in a VCF, it cannot be directly converted to variation graph. However, it is possible to map a VCF file to a reference genome, which, together will make a variation graph. The reference genome is available as a FAST-All (FASTA) sequence format (which is a text file representing the nucleotide or protein sequence). The VCF file does not contain sequences which are same as the reference genome, but only those sequences which are different along with the position information of this occurrence. This means that the variations must be aligned to the genome.

The reference genome is first converted to a sequence graph. To convert a sequence graph to a variation graph, the variations need to be incorporated. To do this, the sequence graph is cut where a variation occurs (this information is picked from the VCF file) generating an alternative graph with the variant sequence known as the variation graph. To know where to cut the sequence graph, the VCF is aligned to the sequence graph through partial order alignment [14]. Figure 1 below shows a partial order alignment of two sequences.

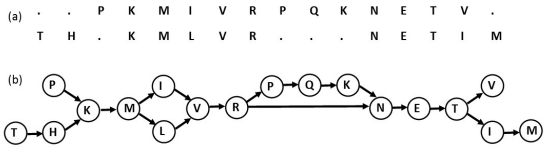

Figure 1. Sequence Alignment in the Partial Order Alignment (POA) representation (a) Row-Column Alignment representation of a pairwise protein sequence. (b) Partial Order alignment of a pairwise protein sequence alignment [14].

The basic data structure of a variation graph will contain nodes, edges and paths (Figure 2). Each variation will follow a different path within the graph. Once the variation graph is constructed, each variation path can be treated as a sub-graph. Since the path will traverse each node once, the subgraph can be now treated as a Directed Acyclic Graph or DAG. The DAGs consisting of nodes, edges and one path

can now be moved in memory. These subgraphs can be reassembled when required because the path is now known.

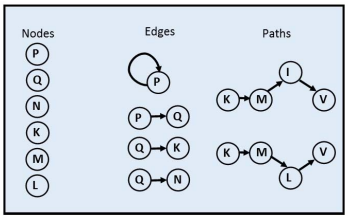

Figure 2. The data structure of a Variation Graph showing nodes, edges and paths.

The subgraphs are basically representing the variations that occur in different genomes with respect to the reference genome. Figure 3 shows the different paths a variation graph can take (which are also the subsequent subgraphs (or DAGs) of the variation graph).

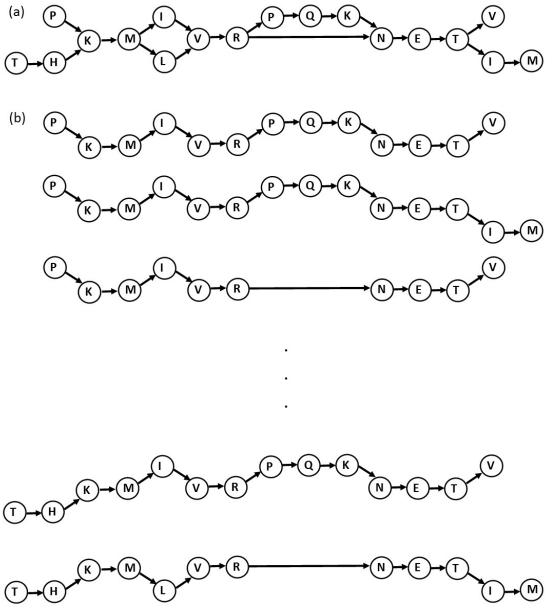

Figure 3. (a) Variation Graph of two sequences [14]. (b) Variation graph broken down to subgraphs. Each graph contains nodes, edges and paths. The subgraphs are constructed as DAGs to be run in memory within a cloud environment.

Using the VG tool by VGteam [15], the reference sequence hs37d.fa and VCF file for phase 3 of the 1000 Genomes Project [5] generates VG by cutting the reference genome at the variation and adding the alternate sequence to it. It indexes the sequence as a key-value store. For this step, 64GB of RAM or more are required. It then constructs the VG by aligning one chromosome at a time and takes half an hour when using 24 cores. It is at this stage that the VG is producing many partially ordered subgraphs. These can be interpreted as DAGs. Intoducing a workflow at this stage will allow the DAGs to move in memory [13]. The workflow will be able to coordinate the DAGs and reconstruct the variation graph, if required.

The DAGs can be processed in memory to exploit the high performance of cloud computing. DAGs have been used in the past to harness the power of multicore and hybrid platforms [21], [22], [23], [24]. DAGs allow processing to be broken down to parallel tasks which can then be assigned to different kernels. Communication between tasks is implicit, allowing the use of data dependecies of DAG. This means that global synchronization of tasks can be avoided, and in turn, this increases scalability [20].

IV. CONCLUSION

The VCF file is the standard starting point of genome analysis. Since it represents a huge amount of data, and more than one VCF file is required during the analysis, it makes sense to shift the analysis process to a cloud. This enables the genome analysis to make use of high performance computing to generate analysis efficiently and in almost real time. But in order to further facilitate this, taking a step back and representing the VCF files as DAGs will enhance the analysis by shifting the data on to a cloud from almost the beginning of the process.

REFERENCES

[1] International Human Genome Sequencing Consortium, “Finishing the euchromatic sequence of the human genome,” Nature, vol. 431, pp. 931-945, 2004.

[2] “1000 Genomes Project,” [Online]. Available: http://www.internationalgenome.org/.

[3] Genomics England, “The 100000 Genomes Project,” [Online]. Available: https://www.genomicsengland.co.uk/the-100000-genomes-project/. [Accessed 8 1 2017].

[4] Illumina, “Illumina,” [Online]. Available: http://www.illumina.com/technology/next-generation-sequencing.html. [Accessed 8 1 2017].

[5] “IGSR: The International Genome Sample Resource,” [Online]. Available: http://samtools.github.io/hts-specs/VCFv4.2.pdf. [Accessed 6 1 2017].

[6] E. M. Coonrod, R. L. Margraf, A. Russel, K. V. Voelkerding and M. G. Reese, “Clinical analysis of genome next-generation sequencing data using the Omicia platform,” Expert Rev Mol Diagn., 2013.

[7] S. Yang, X. Zhang, L. Diao, F. Guo, D. Wang, Z. Liu, H. Li, J. Zheng, P. Jingshan , E. C. Nice, D. Li and F. He, “CAPER 3.0: A Scalable Cloud-Based System for Data-Intensive Analysis of Chromosome-Centric Human Proteome Project Data Sets,” J. Proteome Res, vol. 14, no. 9, pp. 3720-3728, 2015.

[8] M. A. Alekseyev and P. A. Pevzner, “Breakpoint graphs and ancestral genome reconstructions,” Genome Res. , vol. 19, no. 5, p. 943–957, 2009.

[9] S. Marcus, H. Lee and M. Schatz, “SplitMEM: Graphical pan-genome analysis with suffix skips,” Bioinformatics, vol. 30, no. 24, pp. 3476-3483, 2014.

[10] G. Cordascoa, R. D. Chiarab and A. L. Rosenberg, “On scheduling DAGs for volatile computing platforms: Area-maximizing schedules,” Journal of Parallel and Distributed Computing, vol. 72, no. 10, pp. 1347-1360, 2012.

[11] G. Malewicz, A. L. Rosenberg and M. Yurkewych, “Toward a theory for scheduling dags in Internet-based computing,” IEEE Transactions on Computers, vol. 55, no. 6, pp. 757 - 768, 2006.

[12] A. L. Rosenberg, “On scheduling mesh-structured computations for Internet-based computing,” IEEE Transactions on Computers, vol. 53, no. 9, pp. 1176 - 1186, 2004.

[13] M. Taufer and A. L. Rosenberg, “Scheduling DAG-based workflows on single cloud instances: High-performance and cost effectiveness with a static scheduler,” The International Journal of High Performance Computing Applications, vol. 31, no. 1, pp. 19-31, 2017.

[14] C. Lee, C. Grasso and M. F. Sharlow, “Multiple sequence alignment using partial order graphs,” Bioinformatics, vol. 18, no. 3, pp. 452-464, 2002.

[15] E. Garrison, “VGteam,” [Online]. Available: https://github.com/vgteam/vg. [Accessed 8 1 2017].

[16] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan and A. Y. Zomaya, “CA-DAG: Communication-Aware Directed Acyclic Graphs for Modeling Cloud Computing Applications,” in 2013 IEEE Sixth International Conference on Cloud Computing, 2013.

[17] M. F. Wheeler, G. Pencheva, R. Tavakoli, Z.-Y. Shae, H. Jamjoom, J. Sexton, V. Sachdeva, K. E. Jordan, H. Kim, M. Parashar and M. AbdelBaky, “Enabling High-Performance Computing as a Service,” Computer, vol. 45, pp. 72-80, 2012.

[18] The Computational Pan-Genomics Consortium, “Computational pan-genomics: status, promises and challenges,” Briefings in Bioinformatics, pp. 1-18, 2016.

[19] E. Deelman, D. Gannon, M. Shields and I. Taylor, “Workflows and e-Science: An overview of workflow system features and capabilities,” Future Generation Computer Systems, vol. 25, no. 5, pp. 528-540, 2009.

[20] G. Bosilca, A. Bouteiller, A. Danalis, T. Herault, P. Lemarinier and J. Dongarra, “DAGuE: A generic distributed DAG engine for high performance computing,” in 2011 IEEE International Parallel & Distributed Processing Symposium, 2011.

[21] A. Buttari, J. Dongarra, J. Kurzak, J. Langou, P. Luszczek, and S. Tomov, “The impact of multicore on math software,” in Applied Parallel Computing. State of the Art in Scientific Computing, 8th InternationalWorkshop, PARA, ser. Lecture Notes in Computer Science, vol. 4699.Springer, 2006, pp. 1–10.

[22] E. Chan, F. G. Van Zee, P. Bientinesi, E. S. Quintana-Ort´ı, G. QuintanaOrt´ı, and R. van de Geijn, “Supermatrix: a multithreaded runtime scheduling system for algorithms-by-blocks,” in PPoPP ’08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming. ACM, 2008, pp. 123–132.

[23] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, “StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures,” in Euro-Par 2009 Euro-par’09 Proceedings, ser. LNCS, Delft Pays-Bas, 2009. [Online]. Available: http://hal.inria.fr/inria-00384363/en/

[24] R. Dolbeau, S. Bihan, and F. Bodin, “HMPP: A hybrid multi-core parallel programming environment,” in Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007), 2007.

[25] P. Danecek, A. Auton, G. Abecasis, C. A.Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T.Sherry, G. McVean, R. Durbin and 1000 Genomes Project Analysis Group, “The variant call format and VCFtools,” Bioinformatics, vol 27, no. 15, pp. 2156-2158, 2011.

[26] B. Paten, A. Novak and D. Haussler, “Mapping to a Reference Genome Structure,” arXiv:1404.5010, 2014.

[27] K. Schneeberger, J. Hagmann, S. Ossowski, N. Warthmann, O. Kohlbacher and D. Weigel, “Simultaneous alignment of short reads against multiple genomes,” Genome Biology, vol 10, no. 9, pp. R98.1-12, 2009.

[28] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, A. V. Pyshkin, A. V. Sirotkin, N. Vyahhi, G. Tesler, M. A. Alekseyev and P. A. Pevzner, “SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing,” Journal of Computational Biology, vol 19, no. 5, pp. 455-477, 2012.