SparkBeagle: Scalable Genotype Imputation from Distributed Whole-Genome Reference Panels in the Cloud

Altti Ilari Maarala
Department of Computer Science, University of Helsinki
Helsinki, Finland
ilari.maarala@helsinki.fi

Kalle Pärn
Finnish Institute for Molecular Medicine
Helsinki, Finland
kalle.parn@helsinki.fi

Javier Nuñez-Fontarnau
Finnish Institute for Molecular Medicine
Helsinki, Finland
javier.nunez-fontarnau@helsinki.fi

Keijo Heljanko
Department of Computer Science, University of Helsinki
Helsinki Institute for Information Technology, HIIT
Helsinki, Finland
keijo.heljanko@helsinki.fi

ABSTRACT

Massive whole-genome genotype reference panels now provide accurate and fast genotyping by imputation for high-resolution genome-wide association (GWA) studies. Imputation-assisted genotyping can increase the genomic coverage of genotypes and thus satisfy the resolution required in comprehensive GWA studies in a cost-effective manner. However, the imputation of missing genotypes from large reference panels is a compute-intensive process that requires high-performance computing (HPC). Although HPC uses extremely distributed and parallel computing, current imputation tools, and existing algorithms have not been developed to fully exploit the power of distributed computing. To this end, we have developed SparkBeagle, a scalable, fast, and accurate distributed genotype imputation tool based on popular Beagle software. SparkBeagle is designed for HPC and cloud computing environments and it is implemented on top of the Apache Spark distributed computing framework. We have carried out scalability experiments by imputing 64,976,316 variants of 2504 samples from the 1000 Genomes reference panel in the cloud. SparkBeagle shows near-linear scalability while increasing the number of computing nodes. A speedup of 30x was achieved with 40 nodes. The imputation time of the whole data set decreased from 565 minutes to 18 minutes compared to a single node parallel execution. Near identical imputation accuracy was measured in the concordance analysis between the original Beagle and the distributed SparkBeagle tool.

KEYWORDS

bioinformatics, computational genomics, genotyping, parallel computing, distributed systems, big data

ACM Reference Format:
Altti Ilari Maarala, Kalle Pärn, Javier Nuñez-Fontarnau, and Keijo Heljanko. 2020. SparkBeagle: Scalable Genotype Imputation from Distributed Whole-Genome Reference Panels in the Cloud. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '20), September 21–24, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3388440.3414860

BCB '20, September 21–24, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7964-9/20/09.
https://doi.org/10.1145/3388440.3414860

INTRODUCTION

Determining genotype-phenotype associations can contribute to our understanding of complex diseases and foster the development of precision medicine for personalized treatments, patient risk stratification, and targeted drugs [32]. Genotyping is a routine method for detecting genomic variation and is a fundamental process for studying the association of genetic variation with traits or diseases. Advances in high-throughput sequencing technology have enabled large-scale genome-wide association (GWA) studies that require comprehensive detection of single-nucleotide polymorphisms (SNPs) in a sample population, which can consist of more than 100,000 samples where each sample may contain more than 100,000 SNPs [26]. In addition, whole-genome sequencing (WGS) based high-coverage reference panels now allow accurate large-scale genotyping by imputation [22, 24].

Imputation is performed with computational methods by estimating missing SNPs from a reference panel and already genotyped SNPs. Genotype imputation from a subset of genotyped SNPs is an efficient method for discovering the rest of the variants in a genome, in addition to direct genotyping. Imputation assisted genotyping can significantly reduce the time and cost used for direct genotyping with DNA microarrays [24]. Currently, reference panel sizes are growing while more genomes are sequenced, which in turn, enables a more accurate imputation of low-frequency and rare variants [16, 23]. However, imputing missing genotypes of multiple study samples from thousands of reference genotypes is computationally demanding and can take days to weeks even with the most powerful workstation.

While Moore's law is repealing, the sequential CPU performance gain has slowed down dramatically [18] and therefore the high-performance computing needs to scale to a high number of cores distributed across a large computing cluster in order to cope with ever-growing data volumes. This trend is realized in increasing the concurrency at multiple levels: the number of computing cores in a single processor, the number of processors in a computer, and the number of computers in a computing cluster. In contrast, traditional bioinformatics algorithms and pipelines are typically developed on

demand by the researchers largely relying on existing sequential algorithms or algorithms that use shared-memory to parallelize computing inside a single computer. This has led to pipelines that utilize a mixture of command-line tools making them poorly scalable, inflexible, and not easily able to exploit the computing capacity available in large computing clusters. Thus, bioinformatics computing frameworks and algorithms need to change to efficiently exploit all the parallelism available in a distributed computing cluster relying on parallel programming models.

Inspired by our previous results on large-scale genomics data processing using big data platforms with Hadoop-BAM [25], ViraPipe [21], and SeqPig [29], we anticipate that similar distributed and parallel computing methods are ideal for solving scalability problems in large-scale genotype imputation as well. We develop and implement an Apache Spark [36] accelerated distributed parallelization of the widely adopted Beagle imputation tool, SparkBeagle, to boost and to scale genotype imputation from deep coverage reference panels in the cloud. We conduct experiments in a high-performance cloud computing cluster to evaluate the scalability and accuracy of SparkBeagle from 1000 Genomes and HapMap reference panels. Finally, we discuss the results and conclude the paper with a brief summary.

MATERIALS AND METHODS

Related work

Current imputation tools are based on machine learning and related statistical methods such as Hidden Markov Models (HMM) but are not designed for distributed computing [35]. However, genotype data can be partitioned at the chromosomal level and for specific regions (i.e. loci), enabling decomposition of the data for distributed imputation. That is, SNP data for imputation is parallelizable at chromosome locus as nearby variants are known to inherit together considering linkage disequilibrium [31]. Arguably, one reason behind the absence of distributed imputation tools is the data structures of genomics file formats that are not designed for distributed file systems. Especially, compressed variant data formats such as BCF (bgzip compressed VCF[1]), and BED are not distributable without additional external libraries. The Hadoop-BAM [25] was the first library to support the parallel processing of distributed SAM, BAM, VCF, and BCF files in the Hadoop Distributed File System (HDFS) [30]. More recently, Hadoop-BAM has been developed under Disq[2] and Spark-BAM[3] projects for better Spark integration.

Apache Spark [36] and Hadoop-BAM is leveraged in many bioinformatics tools and pipelines such as widely used Genome Analysis Toolkit (GATK)[4] and ADAM[5] for more scalable genomic data analysis in the cloud. Zhou et al. [38] propose MetaSpark for distributed read mapping on reference genomes. Ferraro et al. [11] demonstrate Spark enhanced FastKmer for analyzing k-mer statistics from a large collection of genomic sequences and FASTdoop [10] for managing FASTA and FASTQ files with Hadoop MapReduce. Huang et al. [17] propose the Sparkhit framework for scalable genome analytics by harnessing the Spark MLlib machine learning library and integrating existing genomic data processing tools in the cloud. O'Brien et al., demonstrate VariantSpark [27], a scalable Spark-based variant analysis tool for variant visualization and annotation. Linderman et al. [20] propose DECA for scalable exome copy-number variant calling based on ADAM and Spark. Halvade [8] and Crossbow [14] demonstrate scalable Hadoop MapReduce based sequence alignment methods. CloudBrush [4] is a distributed *De Novo* assembler developed on the Hadoop MapReduce framework while Reflexiv[6] exploits Spark for scalable genome assembly in the cloud.

Beagle utilizes Li and Stephens Hidden Markov Model (HMM) based algorithm for enabling accurate parallel imputation on a single node over adjacent SNP marker windows [3]. In addition to Beagle, other multithreaded single node shared memory imputation tools such as Minimac [12] and Impute [15] have been developed. Minimac3, Minimac4, and Impute4 have been compared with Beagle in [3] demonstrating that Beagle 5.0 overcomes the latest versions of Minimac and IMPUTE in terms of both imputation speed and accuracy. However, being based on single node shared memory computing none of them is capable of massive parallelization over multiple computing nodes with distributed memory in a distributed computing cluster. Imputation is typically distributed in batches of chromosomal chunks to multiple nodes with Slurm[7] or similar workload managers. Without a distributed computing framework and data storage, data chunks are typically distributed to targeted nodes and batch jobs are run in parallel on corresponding nodes [19]. Such systems may suffer from a lack of robustness, low level of parallelism, load imbalance, and lack of fault tolerance.

CloudAssoc [5] is the first massively distributed implementation of genotype imputation tools reported at the present utilizing a distributed filesystem. CloudAssoc is implemented in MapReduce on the Hadoop framework using Impute2 for the imputation algorithm and Hadoop Distributed File System for distributed data storage. However, the implemented CloudAssoc software is not currently publicly available. Minimac3 has been lately deployed on Michigan Imputation Server[8], which exploits Hadoop MapReduce for distributing the imputation jobs [6]. SparkBeagle exploits Apache Spark [36] which is a modern big data processing engine based on distributed in-memory computing for accelerating the execution of distributed computing tasks. SparkBeagle is open source software and freely available at https://github.com/NGSeq/SparkBeagle.

Implementation

Storing and processing of rapidly accumulating genomic data requires a large amount of high-performance storage space, working memory, computing power, and network capacity. Big data infrastructures, cloud computing frameworks, distributed filesystems, and databases have been evolving while the price of DNA sequencing, data storage, and computing memory has been decreasing. Moreover, distributed computing frameworks, such as Apache Spark [36], enable scalable, reliable, efficient, and relatively low-cost computing in the cloud. Parallel data analysis with multiple distributed computing nodes brings a huge performance advantage

[1] https://samtools.github.io/hts-specs/VCFv4.3.pdf
[2] https://github.com/disq-bio/disq
[3] http://www.hammerlab.org/spark-bam/
[4] https://github.com/broadinstitute/gatk
[5] https://github.com/bigdatagenomics/adam
[6] https://rhinempi.github.io/Reflexiv/index.html
[7] https://slurm.schedmd.com/
[8] https://imputationserver.sph.umich.edu/

compared to single workstations. Moreover, cloud services provide infrastructure for deploying computing clusters in a flexible and cost-effective manner.

Apache Spark [36] is a general framework for processing big data workloads in the cloud. Spark accelerates distributed data analysis with in-memory processing where working sets of data can be reused and pipelined in-memory from one pipeline stage to another for the analysis job. Computation in Spark is based on Resilient distributed data sets (RDD) [37], which are distributed and cached to the working memory of multiple computing nodes in a cluster to be processed as parallel tasks by Spark executors. Moreover, Hadoop Distributed File System (HDFS) [30] enables distributed data processing with data-parallel paradigms, such as MapReduce and Spark, by minimizing data transfer between the nodes [7]. This is achieved by dividing each computing task into partitions and each Spark executor process locally stored blocks of the data whenever available. HDFS replicates data blocks to three separate nodes as default for improving fault tolerance and data locality. If the needed data block is not available locally, its location is requested from the HDFS NameNode and the NameNode returns the closest location of the block replica.

SparkBeagle aims to provide easily scalable and robust genotype imputation in the cloud while minimizing user intervention such as transferring, compressing, splitting, and merging files manually. SparkBeagle distributes the imputation workload in Beagle to multiple nodes in the cloud and is designed to scale automatically while increasing the computing cluster size. Beagle I/O routines are rewritten to support distributed computing with Spark and Hadoop Distributed File System extensively. Figure 1 represents the abstraction of SparkBeagle architecture.

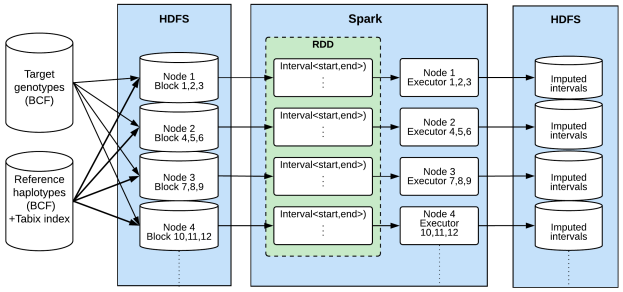

Figure 1: Architectural abstraction of the implementation.

Distributed imputation algorithm. In SparkBeagle, the genotypes are imputed over chromosomal regions in parallel on distributed computing nodes utilizing resources of a computing cluster efficiently, thus giving a great performance advantage compared to parallel imputation tools developed for single node multiprocessor execution. A parallel imputation example scenario is illustrated in the Figure 2. The reference panel of each chromosome is distributed automatically to Hadoop Distributed File System (HDFS) in block compressed (BGZF)[9] VCF format (BCF). Both the reference panel and the target data set are stored to HDFS for improving parallel

[9]http://www.htslib.org/doc/bgzip.html

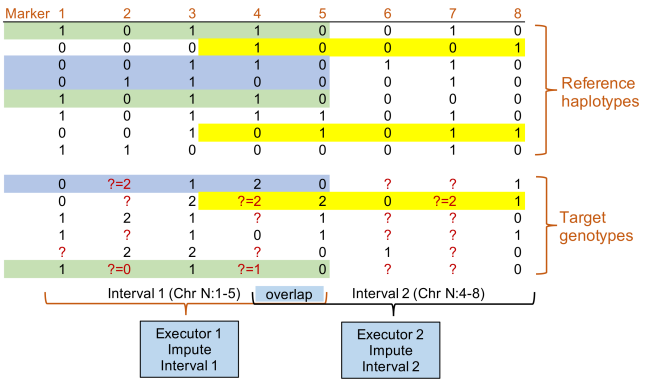

Figure 2: Parallel imputation scenario over overlapped marker intervals. Homozygous major allele = 0, homozygous minor allele = 1, heterozygous allele = 2, missing target genotype = ?. Marker numbering here is artificial (in a real scenario, the numbers would represent physical positions of reference alleles). The genotypes in a target sample are imputed from reference haplotypes highlighted in the same color. Allele encoding is following: 0|0 = 0, 1|1=1, 0|1=2, 1|0=2. In practice, genotype likelihoods of corresponding alleles are used for genotype prediction.

I/O performance. Also, an uncompressed VCF format is supported. Distributed reference panel is read from HDFS blocks on each node in parallel, thus reducing the time to read the panel into Spark working memory. The target data set is read once per chromosome into working memory and broadcasted to all the Spark executors. SparkBeagle exploits imputation in intervals of chromosomal regions, which enables highly parallel imputation on distributed panel data while still preserving accuracy.

Figure 3 presents the distributed imputation task at the software component level inside the SparkContext instance. SNP data is distributed at chromosomal regions, thus enabling parallel imputation in intervals without parallelizing the HMM algorithm itself. Each chromosome is processed individually and adjacent reference markers are partitioned to overlapping intervals while data is read in from the HDFS. The intervals are mapped to Spark executors for distributed imputation.

BGZF compressed input data is partitioned to overlapping intervals utilizing Tabix (http://www.htslib.org/doc/tabix.html) index (Figure 4). Intervals include markers within genomic regions where the size of the region is given in units of *centimorgans*. Given interval is mapped to genomic positions defined in PLINK genetic map format (http://zzz.bwh.harvard.edu/plink/data.shtml#map) and the markers within the interval range are read from the HDFS blocks of the distributed VCF file in parallel. If the VCF file is BGZF compressed (BCF), the Tabix index is used to find the corresponding byte positions in the compressed blocks. Tabix index maps the physical starting position of a marker interval to the corresponding block offset in a BCF file (Figure 4). The offset and the length of the marker interval are queried from the Tabix index and the actual data is accessed by seeking the BGZF block offset in the block compressed HDFS input data stream and reading the bytes

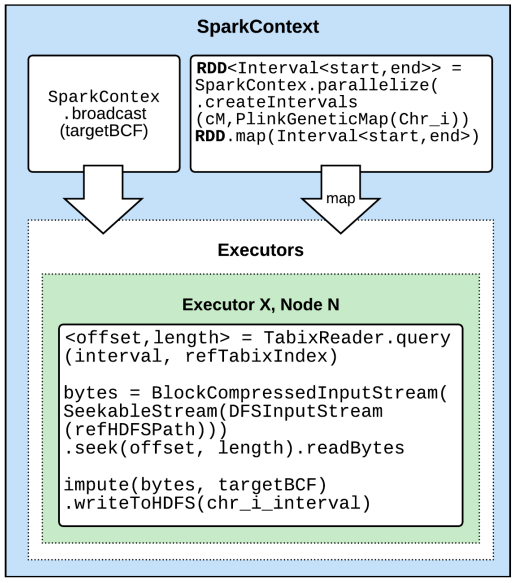

Figure 3: Distributed execution of an imputation task inside the SparkContext instance.

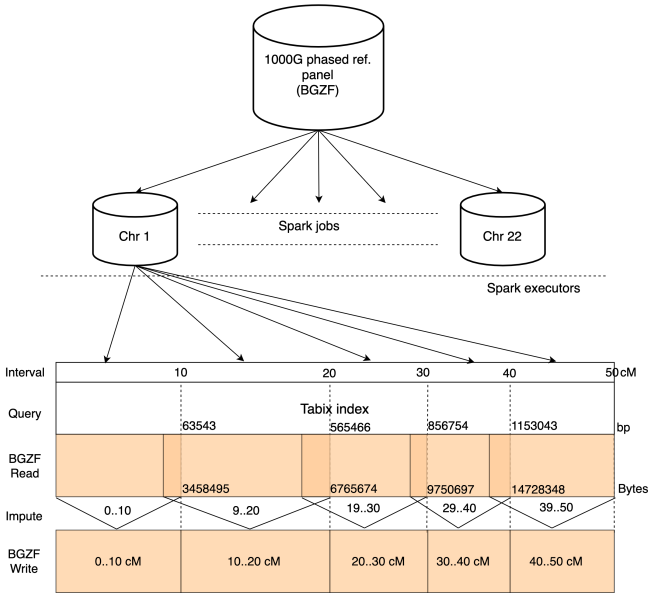

Figure 4: Overall schema of the data-parallel decomposition. Imputation is done in parallel in overlapping allele marker intervals. The physical size of imputation interval depends on the number of markers included in a genomic region defined in units of centimorgans (cM). The imputation interval size of 10 cM and an overlap size of 1 cM is used in the experiments.

by the length of the interval. DFSInputDataStream class takes care of reading the corresponding blocks from the HDFS inside a Spark executor (Figure 3). Eventually, the target SNPs are imputed within the reference panel interval executing the Beagle's original imputation method in parallel on multiple cores on distributed computing nodes. That is, Spark creates as many imputation tasks as there are the chromosomal intervals and processes each interval in parallel with Spark executors. The number of parallel imputation intervals is dependent on the size of the regions, the smaller the regions the more intervals there are. Overlapping markers are imputed twice, but only the genotypes at the end of the interval are persisted as imputation accuracy increases with distance from the beginning of the partition (Figure 4).

Data sets and test data preparation

1000 Genomes [33] phase 3 data set including 2504 haplotypes (81,214,785 variants, 769 GB) is used as a reference panel in both the scalability and the accuracy evaluations. 1000 Genomes data[10] is already phased and provided with Tabix indexes. Only autosomes (chromosomes 1-22) are included in the evaluations and GRCh37 based data is used in all the evaluations. The target data for the scalability test is derived from the 1000 Genomes reference panel by filtering every fifth marker by simply storing header lines and every fifth line after the header (16,238,469 variants, 154 GB). Thus, we uncompress, filter, and reindex the reference panel with Tabix to generate the target panel for the scalability test (shell script available in GitHub[11]). The genetic map files are downloaded from[12], uncompressed and stored to HDFS.

The accuracy test target data set is derived from the HapMap [13] phase 3 genotype data including 861 individuals (880,104 variants). The original HapMap phase 3 target data used in the concordance analysis is available at ftp://ftp.ncbi.nlm.nih.gov/hapmap/phase_3. The concordance analysis workflow is presented in the Figure 5. We use PLINK 1.9 to convert the HapMap genotype files to binary PLINK format e.g., with the following command (for each population separately):

$ plink –file hapmap3_r1_b36_fwd.ASW.qc.poly.recode –make-bed –out ASW

The binary output files of each population are merged with PLINK "–bmerge" option. Then, genotype call rate filter with 10% missingness is applied to exclude such positions which are only present in a few samples e.g., with the following command:

$ plink –bfile hapmap_merged –geno 0.1 –make-bed –out hapmap_gt09

Next, we convert data to VCF with PLINK "–recode vcf" option and recalculate allele count fields with the Bcftools plugin "fill-AN-AC". Finally, we apply quality control (QC) methods described by Anderson et al. [1] and presented in the section "1.3.2 Minimum quality control" of Pärn et al. [28]. One-fourth of the markers (220,017 variants) in the HapMap panel is filtered for the imputation target data. After pre-processing steps, the HapMap VCF data is compressed using bgzip and indexed with Tabix for imputation with SparkBeagle. The analysis-ready quality controlled and masked HapMap data set is available in GitHub[13].

[10] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/
[11] https://github.com/NGSeq/SparkBeagle/tree/master/test/preprocess.sh
[12] http://bochet.gcc.biostat.washington.edu/beagle/genetic_maps/
[13] https://github.com/NGSeq/SparkBeagle/tree/master/test/data/

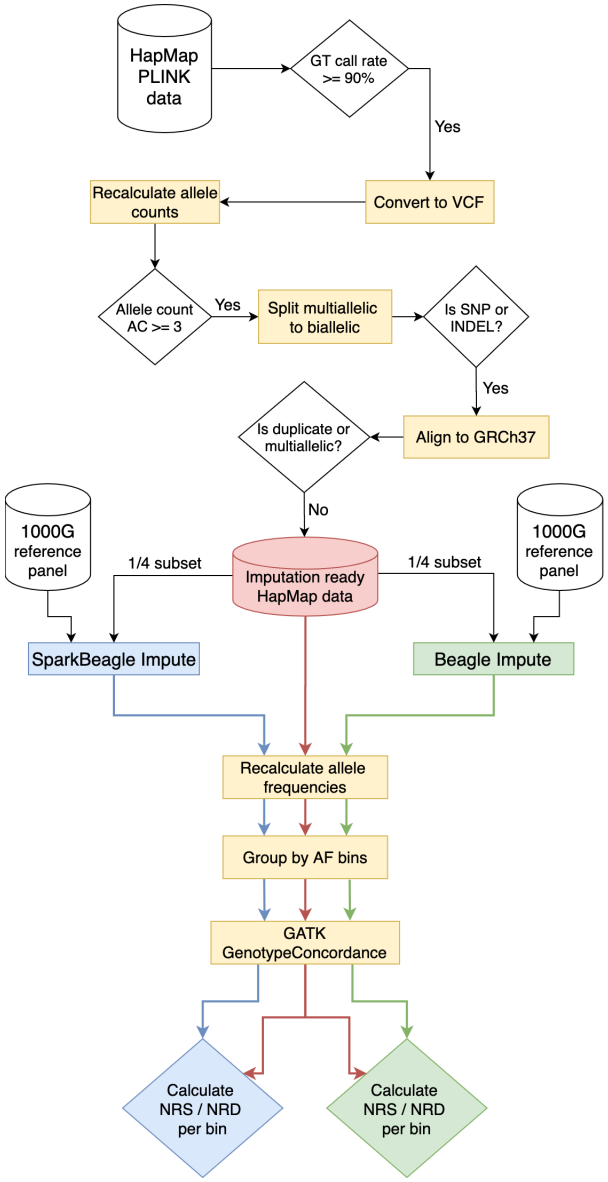

Figure 5: Concordance analysis workflow.

Computing environment

The experiments are run on the Apache Spark cluster in a cloud computing environment maintained by CSC Finland. The cluster consists of Spark worker nodes having 44 GB of RAM and 10 cores (Intel(R) Xeon(R) CPU Gold 6148, with hyper-threading) in each and two Spark master nodes having 86 GB of RAM and 20 cores in each. The whole cluster comprises 520 CPU cores, 1.15 TB of RAM, 25 GB/s network, 15 TB of HDD storage space in total. The Spark cluster is deployed with Hortonworks Data Platform (HDP) 3.1 distribution on virtual machines running CentOS 7 operating system. Spark 2.3.2 and Hadoop 3.1.0 versions are deployed for our experiments. HDP software stack provides YARN [34] resource manager which is used for scheduling Spark jobs and allocating CPU and memory resources dynamically amongst queued jobs. YARN uses containers as resource pools for sharing resources to Spark executors of the queued job. The number of physical CPU cores sets the limit for the maximum number of CPU cores per container and the amount of available memory limits the number of concurrently running executors in a node. The Yarn job queue is configured to use fair ordering policy and size based weighting for allocating resources fairly based on the computational burden of each job i.e. physically larger imputation intervals will have more resources. This configuration maximizes the uniform workload distribution in our experiments and performs the best.

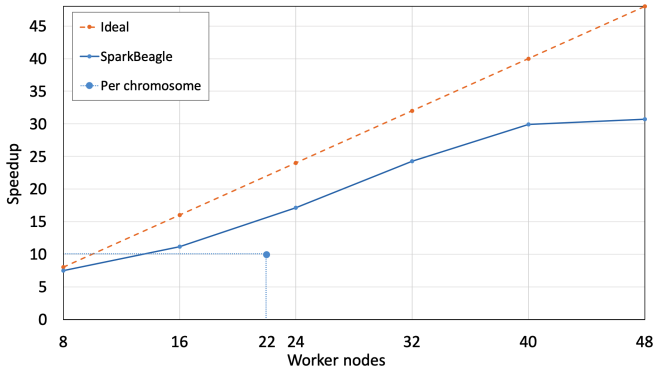

Figure 6: Speedup with increasing cluster size compared to Beagle running in parallel per chromosome on 22 nodes. Dashed line denotes an ideal speedup.

RESULTS

Performance and scalability

The scalability experiment imputes 64,976,316 masked variants of 2504 samples from phased 1000 Genomes reference panel while increasing the number of Spark worker nodes. As a baseline, we run an imputation test with Beagle version 5 on the same data set on a single node having 10 cores resulting in 565 minutes. In addition, we distribute the same data per chromosome on 22 nodes and run imputation with Beagle on each node resulting in 57 minutes and 10x speedup (Single point in Figure 6). Finally, we impute with SparkBeagle over all chromosomes with different amount of Spark worker nodes. All tests are run three times and the result is based on the average execution time. The results (Figure 6) show that total imputation speedup increases almost linearly up to 40 worker nodes. With 40 nodes and 400 cores, the speedup of 30x was achieved and the imputation wall-clock time decreased to 18 minutes. From 40 to 48 nodes, the speedup does not increase significantly anymore as the maximum parallel execution is limited by the amount of data partitions in practice (number of distributed intervals is 365 and each interval is imputed using one core out of the 480 total cores).

Accuracy

Near identical imputation accuracy with both implementations can be seen from non-reference sensitivity (NRS) and non-reference discrepancy (NRD) rates in Figure 7. Thus the parallelization does not

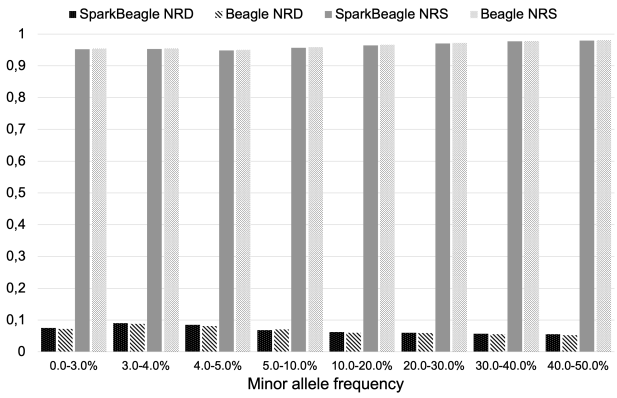

Figure 7: Concordance analysis compares non-reference sensitivity (NRS, the proportion of recovered variants) and non-reference discrepancy (NRD, the proportion of incorrectly imputed variants) across different minor allele frequency (MAF) bins.

affect the observed imputation accuracy. This result was obtained by comparing the possible changes in imputation accuracy between the original Beagle and SparkBeagle. We create a subset of 220,017 variants from the publicly available HapMap panel which we thoroughly QC-d (861 individuals) to mimic a genotyping chip data set. We then impute this data set with the publicly available 1000 Genomes reference panel using both of the imputation approaches. We filter the imputed data based on the HapMap position list and observed the results. The outcome is compared with the all 880,104 QC-d variants in the HapMap panel by running the concordance analysis (Figure 5) with the GATK GenotypeConcordance module. We perform the concordance analysis by applying the comparison methods from the genotype imputation protocol presented in Pärn et al. [28].

The results of the concordance analysis are summarized by two values: the proportion of variants that the imputation manages to recover compared to the reference data set, (measured by non-reference sensitivity, NRS), and the proportion of variants which are imputed incorrectly (measured by the non-reference discrepancy, NRD) [9]. We compare these values across multiple minor allele frequency (MAF) bins ranging from 3-50%. There are no lower MAF bins (MAF<3%) available for us due to the combined HapMap panel being a merge of multiple smaller populations, each with a separate allele count filter.

Discussion

SparkBeagle processes all chromosomal intervals massively parallel on multiple distributed computing nodes utilizing resources of each node more fairly, thus improving usability, load balancing, and CPU utilization. The best benefit is achieved when the reference panel or the target set contains at least thousands of samples and millions of markers, that is when the panel data is too large to be processed in memory on a single machine. SparkBeagle reads and writes VCF data both in compressed (BGZF) and uncompressed formats directly from/to HDFS.

Processing tends to be highly I/O bound and the performance is highly dependent on the underlying storage and network, thus the 30x speedup with 40 nodes can be considered very high with the underlying HDD storage. As the chromosomal imputation interval is the minimal unit of work that can be executed in parallel with our approach, the 18 minutes running time is close to the minimum as the largest chromosomal interval (629,203 markers in the region of 100-110 cM in chromosome 3) was imputed in 15 minutes. Three minute overhead is mostly due to reading, writing, decompressing, and compressing the data. The peak memory usage depends highly on the maximum number of markers within the imputation interval. 4 GB memory per interval of size 10 cM was found enough for running all 365 intervals in parallel. The peak memory usage per executor seems to grow near linearly in proportion to the number of reference markers within the interval which is in line with the memory usage of Beagle reported in [2]. However, the maximum number of the markers does not have a linear relation in the interval size, thus peak memory usage can not be estimated directly from the interval size. The maximum number of markers amongst the intervals can be calculated e.g., from the genetic map file for estimating the memory usage.

Imputation data produced 365 intervals in total, thus with 40 nodes (400 cores), every interval was imputed in parallel (one interval per core). That is, the speedup stops increasing after 40 nodes (Figure 6). A negligible increase in speedup is assumed to be produced by variation in cloud computing cluster performance between the runs. To achieve even better scalability with the test data set, shorter chromosomal intervals should be used, but this would potentially lead to decreased imputation accuracy. In contrast, chromosomally distributed Beagle can only speed up to the time taken to impute the chromosome containing the largest number of SNPs which is demonstrated in our experiments: on chromosome 2 the imputation takes 57 minutes, while with the SparkBeagle the imputation takes only 18 minutes with the same data set (Figure 6). Obviously, panel data could be split manually to overlapping intervals and distributed to several nodes for achieving better performance with the original Beagle. However, this would require writing additional tools for splitting, coordinating, and merging the results, effectively replicating large parts of SparkBeagle functionality while losing data locality and fault tolerance provided by HDFS, and advantage of Spark's cached in-memory data processing.

Near identical accuracy was obtained comparing SparkBeagle with Beagle and the negligible difference in the accuracy is due to heuristics of imputation algorithm, that is, the imputation result fluctuates slightly every time the sequential Beagle imputation is run. Panel data partition and imputation interval sizes must be considered when deciding the Spark runtime configuration: the more samples, the larger the physical size of the imputation interval grows, and thus, more memory per executor is allocated. In contrast, a smaller imputation interval can increase the speedup but reduce the imputation accuracy.

SparkBeagle can be run on public cloud infrastructures with minimal effort when using pre-installed platforms such as Amazon EMR, Google Dataproc, and Azure HDInsight for deploying and managing the Spark cluster. Spark provides connectors for supporting various cloud storage systems such as hdfs, s3a, wasb, abfs, and gs. However, the experiments have been run on HDFS, and using

different storage system may slow down the runtime performance and decrease the fault tolerance. To use a different storage system, the corresponding connector protocol is supplied via the SparkBeagle application parameter e.g., hdfs:// or s3a://. Authentication may be required with public cloud storages and more detailed information is given in Spark documentation[14]. SparkBeagle supports VCF formatted genotype data (reference panel is assumed to be already phased) at the moment, thus PLINK formatted HapMap data was pre-processed when preparing the experiments. The tests were based on human genomes, thus genotyping other species would require adjusting the number of chromosomes in the application parameters.

As a further study, we will investigate how the imputation interval size affects the imputation accuracy, speedup, and scalability with different data sets and computing cluster sizes. We are planning to integrate quality control methods to the SparkBeagle imputation workflow in the near future.

CONCLUSIONS

Genotype imputation with the whole genome-based reference panels offers a cost-efficient alternative for time-consuming and expensive microarray-based genotyping. Reference panel sizes are growing as whole-genome sequencing becomes more cheap and rapid enabling more accurate imputation for comprehensive GWA studies [16]. However, the current genotype imputation algorithms are computationally complex and widely used imputation tools scale only to multiple cores on a single machine. In this work, we have developed a scalable imputation tool, SparkBeagle, that distributes the imputation workload to multiple nodes in the cloud without user intervention and manual processing steps. SparkBeagle processes chromosomes in parallel chromosomal regions on multiple nodes utilizing resources of computing cluster efficiently, thus imputing multiple times faster than parallel imputation tools developed for single node multiprocessor execution.

Our experiments show that SparkBeagle scales near linearly with the increasing number of nodes in the cloud whilst preserving high accuracy and performance. 30x speedup was achieved on 40 Spark worker nodes (imputation time 18 minutes) compared with Beagle running on a single node (imputation time 565 minutes). Near identical imputation accuracy was observed in the results of both implementations. Thus, SparkBeagle can potentially respond to the computational requirements in the near future and foster genome-wide association studies by imputing ever-growing data sets more rapidly and with high precision. SparkBeagle can be deployed on most public cloud infrastructures and big data platforms with a little effort. Moreover, our Spark-based distributed imputation method can be applied to other imputation tools with a relatively small amount of effort, as far as they are able to impute over chromosomal regions. SparkBeagle and the test scripts are available online in GitHub[15].

ACKNOWLEDGMENTS

This work is supported by The Nordic Information for Action eScience Center (NIASC) [Grant No 62721] and Helsinki Institute for Information Technology program Foundations of Computational Health (FCHealth).

[14]https://spark.apache.org/docs/latest/cloud-integration.html
[15]https://github.com/NGSeq/SparkBeagle

REFERENCES

[1] Carl A Anderson, Fredrik H Pettersson, Geraldine M Clarke, Lon R Cardon, Andrew P Morris, and Krina T Zondervan. 2010. Data quality control in genetic case-control association studies. *Nature Protocols* 5 (08 2010), 1564–1573. https://doi.org/10.1038/nprot.2010.116

[2] Brian L Browning and Sharon R Browning. 2016. Genotype Imputation with Millions of Reference Samples. *The American Journal of Human Genetics* 98, 1 (2016), 116 – 126. https://doi.org/10.1016/j.ajhg.2015.11.020

[3] Brian L. Browning, Ying Zhou, and Sharon R. Browning. 2018. A One-Penny Imputed Genome from Next-Generation Reference Panels. *The American Journal of Human Genetics* 103, 3 (2018), 338 – 348. https://doi.org/10.1016/j.ajhg.2018.07.015

[4] Yu-Jung Chang, Chien-Chih Chen, Chuen-Liang Chen, and Jan-Ming Ho. 2012. A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework. *BMC Genomics* 13, 7 (2012), S28. https://doi.org/10.1186/1471-2164-13-S7-S28

[5] Weidi Dai, Qiuwen Wang, Meng Gao, and Lu Zhang. 2012. CloudAssoc: A pipeline for imputation based genome wide association study on cloud. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*. 1435–1438.

[6] Sayantan Das, Lukas Forer, Sebastian Schönherr, Carlo Sidore, Adam E Locke, Alan Kwong, Scott I Vrieze, Emily Y Chew, Shawn Levy, Matt McGue, David Schlessinger, Dwight Stambolian, Po-Ru Loh, William G Iacono, Anand Swaroop, Laura J Scott, Francesco Cucca, Florian Kronenberg, Michael Boehnke, Gonçalo R Abecasis, and Christian Fuchsberger. 2016. Next-generation genotype imputation service and methods. *Nature Genetics* 48, 10 (2016), 1284–1287. https://doi.org/10.1038/ng.3656

[7] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[8] Dries Decap, Joke Reumers, Charlotte Herzeel, Pascal Costanza, and Jan Fostier. 2015. Halvade: Scalable sequence analysis with MapReduce. *Bioinformatics* 31, 15 (2015), 2482–2488. https://doi.org/10.1093/bioinformatics/btv179

[9] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo del Angel, Manuel A Rivas, Matt Hanna, Aaron McKenna, Tim J Fennell, Andrew M Kernytsky, Andrey Y Sivachenko, Kristian Cibulskis, Stacey B Gabriel, David Altshuler, and Mark J Daly. 2011. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics* 43, 5 (2011), 491–498. https://doi.org/10.1038/ng.806

[10] Umberto Ferraro Petrillo, Gianluca Roscigno, Giuseppe Cattaneo, and Raffaele Giancarlo. 2017. FASTdoop: a versatile and efficient library for the input of FASTA and FASTQ files for MapReduce Hadoop bioinformatics applications. *Bioinformatics* 33, 10 (01 2017), 1575–1577. https://doi.org/10.1093/bioinformatics/btx010

[11] Umberto Ferraro Petrillo, Mara Sorella, Giuseppe Cattaneo, Raffaele Giancarlo, and Simona E. Rombo. 2019. Analyzing big datasets of genomic sequences: fast and scalable collection of k-mer statistics. *BMC Bioinformatics* 20, 4 (2019), 138. https://doi.org/10.1186/s12859-019-2694-8

[12] Christian Fuchsberger, Gonçalo R. Abecasis, and David A. Hinds. 2014. Minimac2: Faster genotype imputation. *Bioinformatics* 31, 5 (10 2014), 782–784. https://doi.org/10.1093/bioinformatics/btu704

[13] Richard A. Gibbs and John W. et al. Belmont. 2003. The International HapMap Project. *Nature* 426, 6968 (2003), 789–796. https://doi.org/10.1038/nature02168

[14] James Gurtowski, Michael C. Schatz, and Ben Langmead. 2012. Genotyping in the Cloud with Crossbow. *Current Protocols in Bioinformatics* 39, 1 (2012), 15.3.1–15.3.15. https://doi.org/10.1002/0471250953.bi1503s39

[15] Bryan N. Howie, Peter Donnelly, and Jonathan Marchini. 2009. A Flexible and Accurate Genotype Imputation Method for the Next Generation of Genome-Wide Association Studies. *PLOS Genetics* 5, 6 (06 2009), 1–15. https://doi.org/10.1371/journal.pgen.1000529

[16] Jie Huang and Bryan Howie. 2015. Improved imputation of low-frequency and rare variants using the UK10K haplotype reference panel. *Nature Communications* 6 (14 09 2015), 8111 EP –. https://doi.org/10.1038/ncomms9111

[17] Liren Huang, Jan Krüger, and Alexander Sczyrba. 2017. Analyzing large scale genomic data on the cloud with Sparkhit. *Bioinformatics* 34, 9 (12 2017), 1457–1465. https://doi.org/10.1093/bioinformatics/btx808

[18] Hennessy John L. and Patterson David A. 2019. A new golden age for computer architecture. *Commun. ACM* 62, 2 (2019), 48–60. https://doi.org/10.1145/3282307

[19] Max Lam, Swapnil Awasthi, Hunna J Watson, Jackie Goldstein, Georgia Panagiotaropoulou, Vassily Trubetskoy, Robert Karlsson, Oleksander Frei, Chun-Chieh Fan, Ward De Witte, Nina R Mota, Niamh Mullins, Kim BrÃijgger, S Hong Lee, Naomi R Wray, Nora Skarabis, Hailiang Huang, Benjamin Neale, Mark J Daly,

Manuel Mattheisen, Raymond Walters, and Stephan Ripke. 2019. RICOPILI: Rapid Imputation for COnsortias PIpeLIne. *Bioinformatics* 36, 3 (08 2019), 930–933. https://doi.org/10.1093/bioinformatics/btz633

[20] Michael D. Linderman, Davin Chia, Forrest Wallace, and Frank A. Nothaft. 2019. DECA: Scalable XHMM exome copy-number variant calling with ADAM and Apache Spark. *BMC Bioinformatics* 20, 1 (2019), 493. https://doi.org/10.1186/s12859-019-3108-7

[21] Altti I Maarala, Zurab Bzhalava, Joakim Dillner, Keijo Heljanko, and Davit Bzhalava. 2017. ViraPipe: Scalable parallel pipeline for viral metagenome analysis from next generation sequencing reads. *Bioinformatics* 34, 6 (11 2017), 928–935. https://doi.org/10.1093/bioinformatics/btx702

[22] Jonathan Marchini and Bryan Howie. 2010. Genotype imputation for genome-wide association studies. *Nature Reviews Genetics* 11 (02 06 2010), 499 EP –. https://doi.org/10.1038/nrg2796

[23] Mario Mitt, Mart Kals, Kalle Pärn, Stacey B Gabriel, Eric S Lander, Aarno Palotie, Samuli Ripatti, Andrew P Morris, Andres Metspalu, Tõnu Esko, Reedik Mägi, and Priit Palta. 2017. Improved imputation accuracy of rare and low-frequency variants using population-specific high-coverage WGS-based imputation reference panel. *European Journal of Human Genetics* 25, 7 (2017), 869–876. https://doi.org/10.1038/ejhg.2017.51

[24] Sarah C Nelson, Kimberly F Doheny, Elizabeth W Pugh, Jane M Romm, Hua Ling, Cecelia A Laurie, Sharon R Browning, Bruce S Weir, and Cathy C Laurie. 2013. Imputation-based genomic coverage assessments of current human genotyping arrays. *G3 (Bethesda, Md.)* 3, 10 (10 2013), 1795–1807. https://doi.org/10.1534/g3.113.007161

[25] Matti Niemenmaa, Aleksi Kallio, André Schumacher, Petri Klemelä, Eija Korpelainen, and Keijo Heljanko. 2012. Hadoop-BAM: Directly manipulating next generation sequencing data in the cloud. *Bioinformatics* 28, 6 (02 2012), 876–877. https://doi.org/10.1093/bioinformatics/bts054

[26] Jo Nishino, Hidenori Ochi, Yuta Kochi, Tatsuhiko Tsunoda, and Shigeyuki Matsui. 2018. Sample Size for Successful Genome-Wide Association Study of Major Depressive Disorder. *Frontiers in Genetics* 9 (2018), 227. https://doi.org/10.3389/fgene.2018.00227

[27] Aidan R. O'Brien, Neil F. W. Saunders, Yi Guo, Fabian A. Buske, Rodney J. Scott, and Denis C. Bauer. 2015. VariantSpark: Population scale clustering of genotype information. *BMC Genomics* 16, 1 (2015), 1052. https://doi.org/10.1186/s12864-015-2269-7

[28] Kalle Pärn, Marita A. Isokallio, Javier Nuñez-Fontarnau, Aarno Palotie, Samuli Ripatti, and Priit Palta. 2019. Genotype imputation workflow v3.0 V.2. https://doi.org/10.17504/protocols.io.xbgfijw

[29] André Schumacher, Luca Pireddu, Matti Niemenmaa, Aleksi Kallio, Eija Korpelainen, Gianluigi Zanetti, and Keijo Heljanko. 2013. SeqPig: Simple and scalable scripting for large sequencing data sets in Hadoop. *Bioinformatics* 30, 1 (10 2013), 119–120. https://doi.org/10.1093/bioinformatics/btt601

[30] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. 2010. The Hadoop Distributed File System. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. 1–10.

[31] Matthew Stephens and Paul Scheet. 2005. Accounting for Decay of Linkage Disequilibrium in Haplotype Inference and Missing-Data Imputation. *The American Journal of Human Genetics* 76, 3 (2005), 449 – 462. https://doi.org/10.1086/428594

[32] Vivian Tam, Nikunj Patel, Michelle Turcotte, Yohan Bossé, Guillaume Paré, and David Meyre. 2019. Benefits and limitations of genome-wide association studies. *Nature Reviews Genetics* (2019). https://doi.org/10.1038/s41576-019-0127-1

[33] The 1000 Genomes Project Consortium and Adam et al. Auton. [n.d.]. A global reference for human genetic variation. *Nature* 526 (30 09 [n. d.]), 68 EP –. https://doi.org/10.1038/nature15393

[34] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. 2013. Apache Hadoop YARN: Yet Another Resource Negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*. New York, NY, USA, Article 5, 16 pages. https://doi.org/10.1145/2523616.2523633

[35] Yining Wang, Guohui Lin, Changxi Li, and Paul Stothard. 2016. Genotype Imputation Methods and Their Effects on Genomic Predictions in Cattle. *Springer Science Reviews* 4, 2 (01 Dec 2016), 79–98. https://doi.org/10.1007/s40362-017-0041-x

[36] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (Boston, MA) *(HotCloud'10)*. Berkeley, CA, USA, 10–10.

[37] Matei Zaharia and Mosharaf et al. Chowdhury. 2012. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. 2–2.

[38] Wei Zhou, Ruilin Li, Shuo Yuan, ChangChun Liu, Shaowen Yao, Jing Luo, and Beifang Niu. 2017. MetaSpark: A spark-based distributed processing tool to recruit metagenomic reads to reference genomes. *Bioinformatics* 33, 7 (01 2017), 1090–1092. https://doi.org/10.1093/bioinformatics/btw750