A Highly Parallel Next-Generation DNA Sequencing Data Analysis Pipeline in Hadoop

Kareem S. Aggour*, Vijay S. Kumar*, Dipen P. Sangurdekar†, Lee A. Newberg†, Chinnappa D. Kodira†
*Knowledge Discovery Lab and †Computational Biology and Biostatistics Lab
GE Global Research
{aggour, v.kumar1, sangurde, newberg, kodira}@ge.com

***Abstract*—The era of precision medicine is best exemplified by the growing reliance on next-generation sequencing (NGS) technologies to provide improved disease diagnosis and targeted therapeutic selection. Well-established NGS data analysis software tools, in their unmodified form, can take days to identify and interpret single nucleotide and structural variations in DNA for a single patient. To improve sample analysis throughput, we developed a highly parallel end-to-end next-generation DNA sequencing data analysis pipeline in Hadoop. In our pipeline, each step is parallelized not only across samples but also within each individual sample, achieving a 30x speedup over a single server workflow execution. Furthermore, we extensively evaluate the viability of having our Hadoop-based pipeline as part of a larger commercial genomic services offering—we demonstrate how our pipeline scales sub-linearly both with the number of samples being analyzed and with the depth of coverage of those samples. In particular, on our commodity cluster, 10x as many samples resulted in only a 2.24x increase in the execution time, and a 4x increase in coverage depth resulted in only a 2.53x growth in execution time. We anticipate that such improvements will allow large cohort populations to be analyzed in parallel, and can fundamentally change the way DNA sequencing analyses are used by both researchers and clinicians.**

***Keywords*—NGS; Hadoop; bioinformatics infrastructure; high-performance computing; scalability; DNA; exome; genome**

I. INTRODUCTION

Next-generation sequencing (NGS) refers to the family of massively parallelized sequencing technologies that generate high-throughput DNA sequence reads at a relatively low cost per base of DNA. Existing commercial sequencing platforms offer variants on the sequencing-by-synthesis approach, such as dye incorporation followed by fluorescent detection (Illumina's Solexa, Pacific Biosciences' SMRT sequencing), emulsion PCR on template-specific glass beads (Roche 454, Life Technologies' SOLiD, Complete Genomics) and semi-conductor based detection of polymerization (Life Tech's Ion Torrent). These solutions offer a range of throughputs from small bacterial genomes in a few hours to fully sequenced human genomes in a few days. The rates of base pair read generation from sequencers has recently been growing by a factor of 3-5x per year [1]. As a result, the volume of raw NGS data generated from research activities is projected to increase exponentially in the coming years [2].

For large and complex genomes such as the human genome, researchers have the option of either sequencing the entire genome at a low depth of coverage (average number of quality sequencing reads covering any position along the genome) or targeting a clinically relevant section of the genome for sequencing, such as annotated exomes (whole exome sequencing). In the near future, it is likely that an individual's genome will be sequenced not once but several times for the purpose of diagnosis, treatment, and monitoring of diseases. This necessitates appropriate and highly scalable infrastructures to store, manage, and analyze massive amounts of genomic data with a short turnaround time.

In prior work [3], we demonstrated that by leveraging the industry-standard Hadoop platform [4] for distributed data storage and parallel computation, NGS data analysis studies that previously took days to run can be completed in hours using the same widely accepted NGS software packages in their original, unmodified form. This research extends our previous proof-of-concept effort in the following directions:

1. We have developed an end-to-end NGS (DNA) data analysis pipeline in Hadoop for commercial use by GE Healthcare's SeqWright Genomic Services business [5]. We expanded our pipeline from 5 to 12 operations, from alignment of raw sequence reads to the annotation and scoring of variant calls including two paths—one using a commercially available variant caller and the other utilizing a proprietary variant caller developed at GE.
2. For each operation in our pipeline, we not only parallelize across samples but also parallelize execution within each individual sample. Our highly parallel pipeline runs on commodity hardware, and can be used on both on-premise compute clusters and in Amazon's Elastic Compute Cloud.
3. We demonstrate the commercial viability of our Hadoop-based pipeline by performing extensive performance and scalability evaluation studies. We evaluated our pipeline using sequencing workloads ranging from 10GB whole exome samples to over ½TB whole genome samples.

To date, our Hadoop-based pipeline has achieved a 30x speedup over a traditional single server implementation. Further, our pipeline scales sub-linearly with the number of samples—a doubling of the number of samples results in only a 1.05x growth in the execution time and ten times the number of samples leads to only a 2.24x growth in the execution time.

978-1-4673-6799-8/15/$31.00 ©2015 IEEE

With respect to the depth of coverage of the samples, a 2x depth increase results in only a 1.42x growth in the execution time and 4x depth results in only a 2.53x growth in runtime. With this efficient and highly scalable pipeline, GE SeqWright can now execute DNA sequencing data analyses in hours instead of days or weeks, and large cohort populations can be analyzed in parallel. Such performance improvements mean less waiting time for researchers, leading more quickly to new results, products, and publications. For clinicians and their patients, these improvements can lead to more timely diagnoses, and an earlier start to precision medical treatments.

II. NGS Data Analysis Pipeline

The NGS data analysis pipeline implemented in Hadoop may be executed via one of two potential paths. Fig 1 shows the data flow along the core steps for both paths.

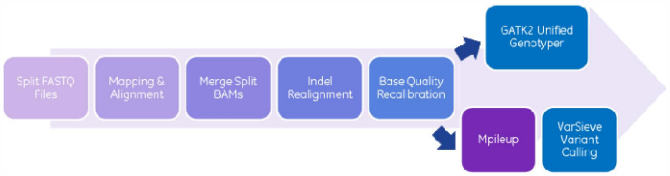

Figure 1: NGS analysis pipeline with two potential paths

The steps described below process the input (FASTQ) files generated from the sequencers using the current state-of-the-art tools to generate biologically relevant results.

Raw Sequence Quality Check: The first step of short read mapping is to perform a quality control (QC) check of the raw sequence data before proceeding with the subsequent computationally intensive steps. This step can also be used to summarize the quality of the raw data. FastQC is an open source tool which reports metrics including basic statistics, quality score distribution as a function of read position, per sequence quality scores, and sequence duplication levels [6].

Pairwise Sequence Alignment: A variety of aligners have been published in the literature that vary in speed and accuracy. BWA [7] and Bowtie2 [8] are two commonly used aligners that utilize the Burrows Wheeler Transform to index the genome and to perform rapid sequence searches. Specifically, BWA MEM (Burrows-Wheeler Alignment Maximal Exact Matches) seeds alignments using maximal exact matches and then extends those exactly-matched seeds to identify positions with minimum errors for the paired read. The results of the alignment are stored in a widely adopted SAM (sequence alignment map) file format and contain, for each read, its left mapping position on the genome, mapping quality, read sequence, the location of its paired mate (if any) and a number of sample-specific tags. BAM (binary alignment map), the bzip2-compressed version of the SAM format, is the preferred file format for storing alignments owing to its smaller size. SAM/BAM files can be interconverted without loss of information with the SAMtools [9] tool suite.

Sorting & Indexing: For downstream processing, BAM files are sorted by read mapping position and are indexed for faster search and retrieval to create sorted BAM and BAI (BAM Index) files. For each read, an optional MD tag that encodes some information about the deviation of the read from the reference sequence, if any, is inserted. SAMtools is used for all of these operations, and further, alignment statistics are generated by the SAMtools flagstat and idxstats commands.

Indel Realignment: Indels (short insertions and deletions) often lead to nearby false positive single nucleotide variants. The Genome Analysis Toolkit (GATK) offers a suite of tools which includes an Indel Realignment step [10]. This toolkit first identifies problematic regions of the chromosome that require realignment, based on the BAM file and a catalog of known indels – this is performed by the Realigner Target Creator operation. The second operation, Indel Realigner, performs a local realignment at each of the candidate sites to identify different consensus alignments that have a better score than the original alignment. The original alignments are then replaced with new ones and the BAM file is regenerated.

Base Quality Recalibration: Different sequencing platforms have different biased error profiles, which if left uncorrected, can underestimate the rate of sequencing errors. These error probabilities depend on a number of error covariates. GATK's base quality recalibration tool corrects error estimates based on a number of covariates (reported quality, machine cycle, base position) using empirical error data from the BAM file and a catalog of known variants (e.g., the dbSNP catalog [11]).

Multi Pile-up (MPileup): The SAMtools mpileup command is used to pool BAM files for subsequent variant calling. The tool first computes a recalibrated base quality score (BAQ score) that adjusts the reported base qualities at any reference position for the event of indels present in its vicinity. Pileup of read base calls and re-calibrated quality scores are then generated for each reference position covered by at least one read.

Variant Calling: Variant calling involves evaluating the evidence for a single nucleotide variant or indel at a position in the reference genome. This evidence includes the fraction of read base calls corresponding to the variant allele (compared to the reference), the quality of variant base calls, and whether the variant reads map preferentially to one strand, forward or reverse read, over the other. A statistical test is performed to evaluate the significance of the observed proportion of variant reads. Different variant calling tools employ different tests. The accuracy and sensitivity of variant detection depends upon the depth of coverage, the observed proportion of variant reads, the level of sequencing noise in the form of poor quality scores, and the level of sequencing errors.

GATK2 Unified Genotyper: When the first of the variant calling paths is used, the final step in our pipeline uses GATK2 Unified Genotyper for single nucleotide variant and indel detection on the realigned and recalibrated samples. This operation uses a likelihood-based model to infer the genotype and variant allele frequency from a sample or pool of samples.

VarSieve: Our second approach to variant calling is to use a frequentist hypothesis testing method, such as VarScan2 [12], MutaScope [13], or Shimmer [14]. We have developed a proprietary variant caller "VarSieve", which combines the approaches of VarScan2 (hypothesis testing and tumor-normal comparison) with the statistical testing method of MutaScope (binomial test) and Shimmer (report p-values for multiple testing) and the advanced filtering methods and Bayesian variant calling method of MuTect. VarSieve works on Mpileup files produced from BAM alignments, and filters each genomic locus based on a number of criteria: i) number of reads supporting the reference or variant locus, ii) base quality of reads supporting reference and variant locus, iii) skewed mapping of reads to one DNA strand at any locus, and iv) fraction of total reads that support a particular variant. Positions that pass these stringent criteria are subject to a Fisher Exact Test (as in VarScan2), a binomial test (MutaScope), or a Bayesian test (as in MuTect).

Our pipeline facilitates the detection and identification of SNPs, small indels and somatic mutations in DNA sequencing samples based on a standard reference genome. More broadly, the term DNA sequencing data analysis may also include other applications such as copy number variation and de novo sequence assembly. While these use cases require other forms of analysis (e.g., clustering, binning, classification) not covered by our pipeline, a MapReduce-based approach to parallelize analysis both across and within samples would provide high scalability for many of these use cases, as well.

III. Related Work

Given the increasing affordability of sequencing technologies and the proliferation of next-gen sequencers in life science organizations worldwide (http://omicsmaps.com), advanced data storage and computing techniques are being developed to process the petabases of genomic data expected to be generated per year [1]. The need for efficiently processing very large-scale next-generation sequencing data is well-recognized as evidenced by numerous efforts in the NGS research community to leverage supercomputers [15][25][27], grid computing [16], commodity compute clusters (equipped with Big Data technologies) [17][18], public cloud computing platforms [19], and even hardware-based acceleration (through the use of GPUs, Xeon Phis and FPGAs) [20] for increasing the throughput of complex data analysis operations.

Analysis Pipeline: Bioinformatics literature is abound with novel parallel algorithms for optimizing the performance of individual computational steps such as sequence alignment and assembly that form a part of a larger NGS analysis pipeline [21][22]. In contrast to this body of research focusing on single operations, we are only beginning to see a handful of efforts that seek to accelerate end-to-end NGS analysis pipelines at large data scales on modern computing platforms, chief among them being ADAM [18], SeqInCloud [19], Halvade [24], and SPRITE [25]. Our work is similar to these latter efforts and focuses on developing a highly scalable end-to-end NGS (DNA) data analysis pipeline for commercial use.

Commodity Hardware: Among projects that specifically address end-to-end NGS pipeline optimizations, some [25][26][27] require access to specialized hardware resources, reducing their viability from a commercialization point of view, while others [18][19][23][24] target modern commodity compute clusters built from standard off-the-shelf hardware components for running large-scale pipelines. Our pipeline builds on the popular industry-standard Hadoop platform that enables analytics to be executed in parallel across potentially thousands of compute cluster nodes via its fault-tolerant MapReduce programming model. Leveraging mainstream Big Data technologies enables us to run our pipeline on both on-premise clusters and in public cloud environments like the Amazon Web Services (AWS) cloud.

Highly Parallel Processing: The GATK suite [10] only supports multithreaded execution on a single multicore machine—the lack of distributed execution over multiple nodes limits its scalability. Our pipeline is parallelized using the MapReduce model in Hadoop. Every step in our pipeline is not only parallelized across multiple samples in a study but also within each individual sample leading to sub-linear scaling. SeqInCloud [19] provides a version of an NGS analysis pipeline using BWA and GATK tools running on the Microsoft Azure cloud. In addition to parallelizing across samples using Hadoop, they parallelize analysis of each sample by loci instead of chromosomes for maximum performance. Halvade [24] implements a highly parallelized pipeline in Hadoop using a single MapReduce job. While similar to these works, our pipeline includes more steps and an alternative path for variant calling, and is also more modular and customizable in its execution. Efforts such as [23] and the Avocado pipeline built on ADAM [18] are examples of generic parallel frameworks with optimizations specific to NGS data analysis. While capable of highly parallel processing, these frameworks focus on parallelizing the most expensive steps of the pipeline, and use I/O optimizations to speed up other data-intensive steps. In [23], the authors provide extensions to Hadoop's distributed file system that allow for tasks to be better colocated with the sample data. ADAM uses the Spark framework for in-memory data processing. While we intend to optimize for I/O access in the future, our current pipeline focuses on extracting maximal parallelism for the end-to-end pipeline using Hadoop.

Standard NGS software packages: Our pipeline is composed of standard, state-of-the-art software packages for NGS data analysis (such as BWA, SAMtools and GATK) that are widely accepted within the research community, as well as proprietary tools (such as VarSieve) developed within GE. Using our pipeline, researchers can plug-in newer or latest versions of these standard packages as and when they are made available, while continuing to benefit from the scalability Hadoop provides. This approach stands in contrast to ADAM [18], SPRITE [25] and D4M [26] where the standard packages may need to be re-implemented using programming models and languages specific to the underlying platform in order to extract maximum performance gains.

Commercial Use and Evaluation: Although research projects ([18][23][24][25]) and commercial Cloud-deployed services such as BaseSpace, DNAnexus, Bina Technologies, Appistry and Seven Bridges Genomics) demonstrate impressive performance speedups, genomic analysis service providers bring up the following concerns: (i) besides brief mentions in datasheets or other anecdotal evidence, there is a general lack of comprehensive evaluation in the literature about the scalability characteristics of NGS analysis pipelines, and (ii) performance numbers quoted by existing frameworks may skip certain analysis steps in a pipeline, or may use modified versions of standard software packages for certain steps. Besides GenBase [28], there is a lack of benchmarks for NGS data analysis, making it difficult to comprehend performance and scalability of end-to-end pipelines on different platforms.

We have developed unique data analysis capabilities within our pipeline to differentiate GE within the NGS research community and to provide value-added services beyond those available from other commercial sequencing companies. Thus, our long-term goal is to create an extensible end-to-end framework to rapidly develop novel NGS applications and deploy them for high-throughput execution.

IV. Design & Implementation

Our end-to-end pipeline is implemented in Hadoop using MapReduce, a fault-tolerant parallel computing paradigm wherein data is processed in two main stages. The Map stage involves tasks that locally process data stored on each node of the cluster, and then shuffle any resulting data such that related data points are grouped together on the same node for further processing. The Reduce stage then executes operations on the grouped data, often producing an aggregated result. We chose Hadoop as our job scheduler due to its inherent fault tolerance, transparent (to the developer) data movement, and natural support for merge (Reduce) operations. To date, we have developed the twelve jobs shown in Table I. This includes jobs that process data for the core steps in the pipeline as well as auxiliary jobs such as Split input (1) and Merge split BAM (6) that were developed to appropriately restructure (split and merge) the data to enable the highest level of parallelism in processing at each step of the pipeline.

TABLE I. Twelve MapReduce Jobs in End-to-End Pipeline

MapReduce job	Level of Parallelism
1. Split input FASTQ files (into split-files or *chunks*)	Per sample
2. FastQC on the split-files	Per chunk
3. BWA MEM aligner	Per chunk
4. Bowtie2 aligner	Per chunk
5. SNAP aligner	Per sample
6. Merge split BAM files	Per chromosome per sample
7. Indel Realignment of BAMs	Per chromosome per sample
8. Base Quality Recalibration (BQR) of realigned BAMs	Per sample
9. Index BAM files & stats	Per sample
10. MPileup	Per chromosome per sample
11. GATK2 Unified Genotyper variant caller	Per chromosome per sample
12. GE's VarSieve variant caller	Per chunk

Each job has been designed and implemented to be a functionally independent, atomic operation. This allows our pipeline to be customized (steps may be removed, replaced, or reordered) depending on the needs of different studies and samples. For instance, GE SeqWright can pick and choose from among multiple aligners (BWA MEM, Bowtie2, SNAP) by including the corresponding jobs in the pipeline. Likewise, if certain steps such as indel realignment and BQR are not applicable to a given set of samples, then SeqWright can easily customize the pipeline so as to not include those jobs.

Prior to running any MapReduce jobs from our pipeline on a Hadoop cluster, we perform marshaling tasks to ensure that all required inputs are in the right locations. We upload the input sample files (typically in compressed FASTQ format) to Hadoop's distributed file system (HDFS) on the cluster. We also ensure that each node in the cluster has available the complete reference genome, any shell scripts, and the NGS software packages executed via those scripts. As such, each job takes its input from HDFS and stores its output to HDFS, and is invoked with a set of command-line parameters.

We have designed our pipeline so as to extract the maximum level of parallelism available in processing the input data for each step. Depending on the nature of the analysis operations they perform, different jobs may lend themselves to different levels of parallelism as shown in Table I. Our jobs can be classified into different categories based on the levels of parallelism (across samples and within each sample):

Data-Parallel by Chunk: At various stages in our pipeline, large data files are split into smaller split-files or chunks to allow them to be processed in parallel. Jobs in this category (e.g., BWA MEM, Bowtie2, FastQC, VarSieve) have map tasks that process each chunk of input data in HDFS independent of each other, and produce a corresponding output chunk in HDFS. Of these jobs, VarSieve also includes a Reduce stage to combine output chunks from the Map stage. The VarSieve job takes as input a collection of mpileup files, runs the variant caller on each row of each mpileup chunk independently, to produce a collection of output vcf-split files, one per mpileup chunk. The Reduce stage takes each vcf-split file and merges and sorts them into one vcf file per sample.

Data-Parallel by Sample (large files): Jobs in this category (Split Input, SNAP, and Index BAM files) are challenging to parallelize because the underlying NGS software packages process files that are larger than the block size in HDFS, and not individual chunks. In these jobs, the bulk of the processing is performed in the Reduce stage. The Map stage simply passes file names from each input chunk to the Reduce stage. E.g., the Split Input job takes as input a collection of paired FASTQ files. For each block in HDFS, a map task simply passes the filename corresponding to that block to the Reduce stage. The Reduce stage receives only unique file names from the Map stage, and so it is able to run a single reduce task per full paired input FASTQ sample file.

Data-Parallel by Sample (small files): Similar to the previous category, these jobs also use a Map stage to collect and pass input file names to the Reduce stage which then

invokes the appropriate analysis algorithms. The difference here is that these jobs (e.g., BQR) deal with multiple smaller files as opposed to blocks of a larger file. In addition to file names, the shuffled data also include keys to indicate which of these files (the values) should be grouped together. The Reduce stage then operates on many files at once to produce a single output. The BQR job takes as input a collection of BAM files indel-realigned by chromosome. The Map stage splits each chromosome-level BAM file into a fixed number *N* of disjoint BAM files (stored temporarily in HDFS) and passes the location of each split to the Reduce stage with a number (between 1 and *N*) as key. The Reduce stage merges BAM splits with the same key, to create *N* larger BAM files per sample, each of which are recalibrated in parallel.

Data-Parallel by Sample and Chromosome: Jobs in this final category (e.g., Indel Realignment, MPileup, and GATK2 Unified Genotyper) are parallelized by both sample and by chromosome resulting in 25 parallel processes per sample (chromosomes 1→22, X, Y, and M). Chromosome 1 is usually the largest and so usually dominates the execution time for these jobs. The Indel Realignment job is parallelized using a collection of empty input files (named as BAM filenames per sample and the chromosomal region per file). The Map stage takes the list of BAM file locations referenced in the empty filenames and first uses GATK2 RealignerTargetCreator to create a list of target intervals for realignment, and then realigns those chromosome-level BAM files using IndelRealigner to produce new, realigned BAM files.

V. Single Server vs. Hadoop Comparison

We first compared the performance of the sequential and Hadoop-based parallel version of our pipeline using a public dataset of Korean lung adenocarcinoma patients [29]. That study involved sequencing 200 lung cancer and matching normal tissues, as well as transcriptome sequencing from the patients. We obtained data for two patients for whom two different datasets were available (whole exome sequencing on cancer sample and matching normal tissue) from the public short read database at the European Nucleotide Archive (http://www.ebi.ac.uk/ena). The four datasets contain paired reads, resulting in a total of eight input FASTQ files. For these experiments, these samples were aligned to the human genome g1k_v37 [8]. The total size of the eight files is 110.9GB.

For comparison, we executed the sequential pipeline on a single node of our 48-node Hadoop cluster. Each node in our cluster has 16 dual core 2.93GHz CPUs and is equipped with 384GB of RAM and 4.75TB of local disk space. Table II provides a side-by-side performance comparison of the single server vs. Hadoop-based pipeline runtimes (the latter averaged across three independent runs). As indicated in Table II, our Hadoop-based pipeline achieved over a 30x speedup over the single-server implementation, with the end-to-end pipeline execution time reducing from 3 days to a fraction of a day. The speedups observed for individual steps depend on the maximum possible parallelism for each step. For example, the BWA MEM and VarSieve steps are the most conducive to parallelism, as each of the read pairs and called variants respectively could be analyzed in parallel. On the other end of the parallelism scale, the Indel Realignment, Base Quality Recalibration and Unified Genotyper steps each ran 25 chromosomes x 4 samples = 100 processes in parallel.

Table II: Single multi-core machine vs. Hadoop cluster performance comparison. *All runtime values are in minutes, unless otherwise noted. Dashes indicate steps not required in the single server execution.*

Operation	Single Server	Average Hadoop	Hadoop Speedup
Split FASTQ Files	-	6.45	12.68x
BWA MEM	516.30	26.64	
Merge Split BAMs	-	7.63	
Fastqc	80.87	0.61	132.42x
Indel Realignment	512.03	19.54	26.21x
Base Qualty Recalib	1,138.98	33.03	25.27x
Merge BQR BAMs	-	12.04	
Indexing & Stats	39.98	1.22	
Unified Genotyper	297.37	11.42	26.03x
MPileup	88.22	3.67	24.06x
VarSieve	2,051.40	5.81	353.04x
Total Time (hours)	**78.75 hrs**	**2.60 hrs**	**30.28x**
Total Time (days)	**3.28 days**	**0.11 days**	

While not the focus of this paper, it is worth noting that we also compared the number of mapped reads, percent of target bases covered, depth of coverage, number and quality of variants called and more using both real and synthetic data. The Hadoop and local pipelines demonstrate very similar performances in terms of identifying somatic (cancer) and germline (normal) mutations in both datasets, with over 99% agreement in most instances.

VI. Scalability Performance Testing

Our next phase of measurements was for the purpose of understanding the scalability of the Hadoop-based pipeline. Our first objective was to understand how the execution time grows as the number of samples in a whole exome (20GB+/sample) study grows, to be able to process large cohort populations. Our second objective was to evaluate how the performance of the pipeline changes for whole genome samples (½TB+/sample) at increasing depths of coverage by sequencing reads. In traditional single server computing, the processing time is expected to grow linearly with the number of samples; for example, doubling the number of samples would typically double the execution time. In the Hadoop-based pipeline we can process many samples in parallel, thus we expected (and found) that the runtime grows sub-linearly.

A. Theoretical Execution Time Growth vs. Data Growth

We chose a single sample from the previous analysis to evaluate the scalability of the platform. This paired sample (total input size of 20.76 GB) was run through the pipeline thrice to generate an average execution time. From the average execution times of the one sample, we performed a theoretical scalability analysis to estimate the execution time for two samples and ten samples (shown in Table III). This analysis evaluated the number of parallel processes required at each step, and based on the number of available resources in our

cluster and the average execution time of each step; an estimated runtime was generated for two and ten samples.

It is important to keep in mind that this analysis is highly dependent on the size and configuration of the Hadoop cluster, and that different computing environments are likely to lead to different scalability results. In the best case scenario, if the compute cluster were infinitely elastic we could add as many samples as desired and the execution time would not grow at all. In the worst case scenario, if the cluster were already fully utilized when processing a single sample, then doubling the number of samples would double the execution time.

Table III: Summary results of theoretical analysis of NGS pipeline scalability from one to two and one to ten samples being executed in parallel.

# Samples	1 sample *actual time*	2 samples *time estimate*	10 samples *time estimate*
Total Time	**1.51 hrs**	**1.57 hrs**	**3.29 hrs**
Input Size	**20.76GB**	**41.52GB**	**207.6GB**
Data Size	**1x**	**2x**	**10x**
Run Time	**1.00x**	**1.04x**	**2.18x**

From the theoretical analysis, we predict that under ideal conditions the execution time would increase 4% when the data size doubles. Further, we predict that the time would grow by 118% when the data size is tenfold. This sub-linear growth in execution time is a positive side-effect of the approach Hadoop takes to schedule and execute jobs.

1) Why some execution times grow sub-linearly

Hadoop-based process execution times may grow sub-linearly with an increase in the data to be processed due primarily to underutilization of the Hadoop environment. It would be unusual for a process to consume the exact capacity of a Hadoop cluster, thus when additional data is added to be processed in parallel, portions of that additional processing will fill the excess capacity available during the run of the initial data, leaving only a subset of the new data to be processed once the initial data processing has been completed.

2) Why some execution times do not grow at all

In a similar vein, some process execution times may not grow at all, even when the number of samples increases substantially. This is because, particularly in the later steps in the NGS workflow, there are fewer opportunities for parallelism—larger clusters may be severely underutilized and have substantial excess capacity to process more samples.

The scalability estimates in Table III assume unlimited network and disk I/O bandwidth. However, as more processes are executed in parallel, there will be increased disk reading, writing, and network traffic to move data to and from HDFS. This analysis focuses purely on the computing time and overlooks the expected slowdown resulting from network and disk bandwidth limitations, which will have a small but measurable effect on the performance as the number of parallel processes and data volume grows.

B. Whole Exome Scalability Testing

Nine independent runs were executed to evaluate the whole exome sample time growth vs. data growth. The single paired sample that was used for the single sample testing was duplicated such that two copies of the same sample were used for the two-sample testing. Similarly, nine duplicates were created and used for the 10 sample test. The average results from the three tests per sample size can be found in Table IV. We observed that 2x the number of paired samples increased the execution time by only 1.05x. Further, 10x the number of samples increased the execution time by only 2.24x.

Table IV: Scalability performance comparison of NGS pipeline from one to two and one to ten samples being executed in parallel. *All values are in minutes, unless otherwise noted.*

# samples	1 sample	2 samples	10 samples
Split FASTQ Files	3.04	2.98	3.91
FastQC	0.39	0.43	0.75
BWA MEM	8.49	8.56	18.30
Merge Split BAMs	3.62	4.87	63.42
Indel Realignment	18.07	15.90	20.64
Base Qualty Recalib	22.39	25.49	44.79
Merge BQR BAMs	6.48	6.87	12.76
Indexing & Stats	0.92	0.91	1.07
Unified Genotyper	7.51	7.78	11.14
MPileup	2.18	2.19	3.04
VarSieve	3.56	3.77	9.63
Total Time (hr)	**1.51 hr**	**1.59 hr**	**3.38 hr**
Input Data Size	**20.76 GB**	**41.52 GB**	**207.6 GB**
Data (Sample) Size	**1x**	**2x**	**10x**
Run Time	**1.00x**	**1.05x**	**2.24x**

These observed results are consistent with our theoretical computed time growths from Table III. The actual execution time in Table IV is slightly worse than the theoretical (1.05x actual vs. 1.04x theoretical execution time for 2x data, and 2.24x actual vs. 2.18x theoretical execution time for 10x data), as expected, given the realities of the network and disk I/O bandwidth limitations. From these results, we conclude that not only is the Hadoop-based NGS pipeline considerably faster than a single server solution, but also scales extremely well with the number of samples being analyzed.

C. Whole Genome Depth Testing

After evaluating the scalability of the pipeline based on the number of whole exome samples, the next step was to understand how the pipeline performed based on the depth of coverage of a whole genome paired sample. We filtered a whole genome FASTQ sample from The 1,000 Genomes Project [9] to generate three separate paired samples, with depths of 10x, 20x, and 40x. We ran these samples through the pipeline independently, to understand how the performance was impacted as the depth of the sample increased.

Unlike the previous analysis, in which we demonstrated that the runtime grew sub-linearly with the number of samples, we do not expect to see sub-linear behavior in these experiments. In the previous experiments there was excess capacity in the cluster that could be utilized when running multiple independent samples in parallel. In the present use case we are increasing the size of a single sample, and so there is almost no opportunity for increased parallelism—the same number of steps will be executed, just on larger files. The only

case where there is increased parallelism is the BWA aligner, which runs on the split input files. The larger samples produce more split files that are analyzed in parallel, so a sub-linear speed-up for this step is achievable. However, given that this step makes up a small percentage of the overall runtime, we anticipate we'll see a linear or near linear growth in execution time with the growth in sample size.

As can be observed in Table V, doubling the depth of the sample less than doubled (1.42x) the execution time. This was a better than expected result, as our initial hypothesis was that the execution time would approximately double with the doubling of the sample depth. Further, quadrupling the sample depth (10x to 40x) resulted in a growth to only 2.53x the execution time, again providing results better than our initial hypothesis. When we compare the 40x depth average execution time to the 20x depth average execution time, we get a ratio of (8.36hr / 4.70hr) = 1.78x, indicating that the sub-linear growth in execution time as the sample size doubles holds relatively steady.

Thus, while we do not achieve much by way of improved levels of parallelism when the sample depth of coverage is increased, we see that the pipeline does achieve sub-linear scalability with the size of the sample being analyzed. This was an unexpected, positive discovery.

Table V: Whole genome performance comparison of NGS pipeline from 10x to 20x depth and 10x to 40x depth samples. *All times are in minutes, unless otherwise noted.*

Genome Depth	10x	20x	40x
Split FASTQ Files	10.83	22.69	62.33
FastQC	0.54	0.76	1.22
BWA MEM	23.61	23.97	41.18
Merge Split BAMs	9.92	17.94	46.64
Indel Realignment	23.38	29.47	46.89
Base Qualty Recalib	29.04	40.83	90.29
Merge BQR BAMs	18.23	31.44	71.04
Indexing & Stats	2.02	3.69	6.74
Unified Genotyper	14.34	18.95	28.48
MPileup	4.58	7.03	13.46
VarSieve	5.11	8.01	11.24
Total Time	**3.31 hr**	**4.70 hr**	**8.36 hr**
Input Data Size	**74.5 GB**	**149 GB**	**298 GB**
Data (Depth) Size	**1x**	**2x**	**4x**
Run Time	**1.00x**	**1.42x**	**2.53x**

D. Whole Genome Combined Scalability & Depth Testing

The final phase of testing assessed the scalability of the NGS pipeline in Hadoop when evaluating the whole genome paired samples. Specifically, we wished to understand if the scalability of the pipeline changed when running against considerably larger samples. The 10x depth whole genome paired sample files were used for this test phase. Six independent runs were executed to evaluate the whole genome sample time growth vs. data growth, with the single 10x depth whole genome paired file run from the previous experiment serving as the baseline. Two copies of the same sample were used for the two-sample testing. Similarly, nine duplicates were created and used for the 10 sample test.

Table VI shows the results averaged across three runs per sample size. As can be observed, doubling the number of samples results in only a 7% increase in the total computation time, exactly in line with the whole exome scalability results previously reported. Further, increasing the number of input samples tenfold increases the computing time by only 125%, again extremely consistent with the previous experimental results in Table III. Thus, we conclude that the pipeline scales sub-linearly, over a span of whole genome sample sizes.

Table VI: Scalability performance comparison of whole genome 10x depth NGS pipeline from one to two and one to ten samples executed in parallel. *All times are in minutes, unless otherwise noted.*

# Genome Samples	1 sample	2 samples	10 samples
Split FASTQ Files	10.83	10.23	15.95
FastQC	0.54	0.69	2.26
BWA MEM	23.61	23.86	84.51
Merge Split BAMs	9.92	13.78	61.67
Indel Realignment	23.38	23.18	27.68
Base Qualty Recalib	29.04	33.01	75.14
Merge BQR BAMs	18.23	21.78	67.62
Indexing & Stats	2.02	2.17	2.87
Unified Genotyper	14.34	14.62	17.39
MPileup	4.58	4.62	6.63
VarSieve	5.11	6.98	24.75
Total Time	**3.31 hr**	**3.55 hr**	**7.44 hr**
Input Data Size	**74.5 GB**	**149 GB**	**745 GB**
Data (Sample) Size	**1x**	**2x**	**10x**
Run Time	**1.00x**	**1.07x**	**2.25x**

Table VII: Big Data dataset tests of one, two and four samples executed in parallel. *All times are in minutes, unless otherwise noted.*

# Genome Samples	1 sample	2 samples	4 samples
FastQC	1.95	4.45	8.58
BWA MEM	32.48	90.98	105.32
Merge Split BAMs	52.32	106.43	170.82
Indel Realignment	86.47	121.87	144.32
Base Qualty Recalib	196.18	387.70	515.93
Merge BQR BAMs	86.83	180.94	290.39
Indexing & Stats	9.82	11.41	12.43
Unified Genotyper	42.45	50.48	54.23
MPileup	24.97	30.75	31.05
VarSieve	26.03	54.99	55.77
Total Time	**9.33 hr**	**17.33 hr**	**23.15 hr**
Input Data Size	**0.55 TB**	**1.40 TB**	**2.70 TB**
Data (Sample) Size	**1x**	**2.54x**	**4.89x**
Run Time	**1.00x**	**1.86x**	**2.48x**

E. Big Data Dataset Testing

The final set of performance tests evaluated our pipeline's scalability against very large datasets. These tests were carried out using a subset of the dbGaP dataset (phs000447.v1.p1.c2) of whole-exome samples from 149 high-risk prostate cancer patients [30]. We obtained authorized access to use this data for research purposes. Table VII shows runtimes of our

Hadoop-based pipeline in cases where 1, 2 and 4 samples from this dataset are processed at once. Note that the overall input data size in the 4-sample case is 2.7 TB.

While it took 9.33 hours to process a single 0.5 TB sample, we were able to process 4 such samples concurrently on our Hadoop cluster in less than a day. These sub-linear scaling results (4x the number of samples leads to only a 2.48x increase in processing time) demonstrate that using the same cluster resources as in previous tests, our pipeline can scale extremely well even when processing multi-terabyte samples – this is representative of the workloads that GE SeqWright expects to handle as a part of their genomic services business.

VII. CONCLUSIONS

It appears inevitable that in the near future, personal genome sequencing will become a routine practice as it gains both credibility and momentum across the entire clinical continuum ranging from early risk assessment in healthy individuals to genome-guided treatment in patients with complex diseases. The impact of NGS on precision medicine will depend on many challenges—technical, ethical, training, acceptance and adoption of the technology, and reimbursements for NGS-based tests. There will be pressure and focus on improving the sequencing technology to produce even more cost-effective and quality reads within hours. On the analysis side, progress in data storage, management, and distributed computing will play a crucial role in the adoption of NGS technologies for research and clinical applications.

This research has resulted in a complete next-generation DNA sequencing pipeline being fully implemented within the Hadoop parallel processing framework, enabling very efficient and highly scalable DNA sequencing analyses. The pipeline includes 12 different sequencing operations with two alternative paths for sequencing—one using a commercial variant caller and the other utilizing our custom variant caller.

To date, the Hadoop-based pipeline has achieved a 30x speedup over a traditional single server pipeline implementation. Furthermore, it has been demonstrated that the pipeline scales sub-linearly with the number of samples, with an increase to 2x the number of samples resulting in only a 1.05x growth in execution time. Similarly, an increase to 10x (order of magnitude) number of samples resulted in only a 2.24x growth in execution time. This sub-linear scalability allows large cohort populations to be analyzed in parallel with only relatively small increases in the overall execution time. Using the Hadoop-based implementation, a complete end-to-end pipeline execution can go from days to hours. This enables rapid turn-around times for clinical samples, enabling biomarker discovery and getting results back to clinicians and patients more quickly. The speedup allows researchers to run analyses many more times than was previously feasible, allowing the testing of multiple hypotheses, and qualitatively changing how researchers interact with genomic data.

REFERENCES

[1] M.C. Schatz and B. Langmead. The DNA Data Deluge, IEEE Spectrum, issue on Biomedical devices, Jun 2013.

[2] Z.D. Stephens, et al. Big Data: Astronomical or Genomical, PLoS Biol 13(7). doi: 10.1371/journal.pbio.1002195.

[3] Aggour KS, Sangurdekar DP, Kumar VS, Newberg LA, Kodira CD, Efficient Next-Generation DNA Sequencing in Hadoop, Proc. of the 6th Intl. Conf. on Bioinformatics and Comp. Biology pp.183-190, 2014.

[4] T. White. Hadoop, The Definitive Guide. O'Reilly Media Inc., 2012.

[5] GE Healthcare SeqWright – DNA Sequencing, Molecular Biology and Custom Genomic Analysis Services. http://www.seqwright.com/

[6] FastQC, Babraham Bioinformatics Institute, http://www.bioinformatics.babraham.ac.uk/projects/fastqc.

[7] Li H, Durbin R., Fast and accurate long-read alignment with Burrows-Wheeler transform, Bioinformatics. 2010 Mar 1; 26(5):589-595.

[8] Langmead B, Salzberg SL., Fast gapped-read alignment with Bowtie 2, Nat Methods. 2012 Mar 4; 9(4):357-359.

[9] Li H, Handsaker B, et al.; 1000 Genome Project Data Processing Subgroup, The Sequence Alignment/Map format and SAMtools, Bioinformatics. 2009 Aug 15; 25(16):2078-2079.

[10] A. McKenna, et al. The Genome Analysis Toolkit: A MapReduce Framework for Analyzing Next-generation Sequencing Data. Genome Res. 2010 Jul 19; 20(9):1297-1303.

[11] Sherry ST, Ward M-H, Kholodov M, et al. dbSNP: the NCBI database of genetic variation. Nucleic Acids Research. 2001;29(1):308-311.

[12] Koboldt DC et al, VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing, Genome Res 2012 Mar;22(3):568-76.

[13] Yost SE et al, Mutascope: sensitive detection of somatic mutations from deep amplicon sequencing, Bioinformatics 2013 Aug 1;29(15):1908-9.

[14] Hansen NF et al, Shimmer: detection of genetic alterations in tumors using next-generation sequence data, Bioinformatics 2013 Jun 15;29(12):1498-503.

[15] G. Lockwood. DNA Sequencing: Not Quite HPC yet. www.theplatform.net/2015/03/03/dna-sequencing-not-quite-hpc-yet/

[16] Sulakhe, D., et al., "GNARE: automated system for high-throughput genome analysis with grid computational backend." J Clin Monit Comput. 2005 Oct; 19(4-5): 361-9

[17] R.C. Taylor, An overview of the Hadoop/MapReduce/ HBase Framework and its Current Applications in Bioinformatics. BMC Bioinformatics, 2010 Dec 21; 11 Suppl 12:S1

[18] F. Nothaft. Scalable Genome Resequencing with ADAM and Avocado. Tech. Report No.: UCB/EECS-2015-65, UC Berkeley, 2015.

[19] N. Mohamed, et al.. Accelerating Data-Intensive Genome Analysis in the Cloud, Proc. of the 5th Intl. Conf. on Bioinformatics and Comp. Biology, 2013.

[20] Aluru S, Jammula N. A Review of Hardware Acceleration for Computational Genomics. Design & Test, IEEE: 31(1), pp. 19-30, 2014.

[21] HPC Software Libraries for Next-Gen Sequencing Analytics – Aluru Lab. http://aluru-sun.ece.iastate.edu/doku.php?id=software

[22] Wang et. al., Investigating Memory Optimization of Hash-index for Next Generation Sequencing on Multi-core Architecture., Proc. of 11th IEEE Intl. Workshop on High Performance Comp. Biology, 2012.

[23] Diao Y, Roy A, Bloom T. Building Highly-Optimized, Low-Latency Pipelines for Genomic Data Analysis. Proc. of 7th Intl. Conf. on Innovative Data Systems Research (CIDR), 2015.

[24] Decap D. et al. Halvade: Scalable Sequence Analysis with MapReduce. Bioinformatics, 2015. doi: 10.1093/bioinformatics/btv179.

[25] Rengasamy V, Madduri K. Engineering a High-performance SNP detection pipeline. Tech Report, Penn State Univ, 2015.

[26] Dodson S, Ricke DO, Kepner, J. Genetic Sequence Matching using D4M Big Data Approaches. Proc. of IEEE High-Performance Extreme Computing (HPEC), 2014.

[27] Kandel M et al, Genomic Applications on Cray Supercomputers: Next Generation Sequencing Workflow. Cray User Group conference, 2013.

[28] Taft R, et al. GenBase: A Complex Analytics Genomics Benchmark. Tech Report No. MIT-CSAIL-TR-2013-028, MIT,2013.

[29] Abecasis GR, et al., 1000 Genomes Project Consortium, An integrated map of genetic variation from 1,092 human genomes, Nature 2012 Nov 1; 491(7422):56-6

[30] Berger MF, et al., The Genomic Complexity of Primary Human Prostate Cancer. Nature, 2011. 470(7333):214-220.