2016 IEEE International Parallel and Distributed Processing Symposium Workshops

SparkScore: Leveraging Apache Spark for Distributed Genomic Inference

Amir Bahmani* Alexander B. Sibley† Mahmoud Parsian‡ Kouros Owzar§ Frank Mueller¶
*¶Department of Computer Science, North Carolina State University, Raleigh, NC
†§Duke Cancer Institute, Duke University School of Medicine, Durham, NC
‡ Illumina, Santa Clara, CA
*abahman@ncsu.edu †alexander.sibley@duke.edu ‡mparsian@illumina.com §kouros.owzar@duke.edu ¶mueller@ncsu.edu

***Abstract*—The method of the efficient score statistic is used extensively to conduct inference for high throughput genomic data due to its computational efficiency and ability to accommodate simple and complex phenotypes. Inference based on these statistics can readily incorporate *a priori* knowledge from a vast collection of bioinformatics databases to further refine the analyses. The sampling distribution of the efficient score statistic is typically approximated using asymptotics. As this may be inappropriate in the context of small study size, or uncommon or rare variants, resampling methods are often used to approximate the exact sampling distribution. We propose SparkScore, a set of distributed computational algorithms implemented in Apache Spark, to leverage the embarrassingly parallel nature of genomic resampling inference on the basis of the efficient score statistics. We illustrate the application of this computational approach for the analysis of data from genome-wide analysis studies (GWAS). This computational approach also harnesses the fault-tolerant features of Spark and can be readily extended to analysis of DNA and RNA sequencing data, including expression quantitative trait loci (eQTL) and phenotype association studies.**

I. Introduction

The advent of high-throughput technologies for DNA genotyping and sequencing has greatly accelerated the pace for medical discovery while posing new statistical and computational challenges [8], [23], [29]. Genomic association studies based on SNPs generally fall into two categories. The first is based on studying the effect of single variants with respect to a phenotype. These are referred to as variant-by-variant analyses. Taking a broader view, what may be of interest is to study the joint relationship between a set of SNPs and a phenotype. The set could be the SNPs within a biological pathway or located within a gene. These are referred to as SNP-set analyses. The latter require the calculation of the millions of variant level statistics, essentially the conduct of a variant-by-variant analysis, which are in turn aggregated within each set.

The method of the efficient score statistic [32], given its high computational efficiency and stability, and ability to incorporate complex phenotypes, serves as the backbone for a wide variety of inferential methods for analysis of genomic data (e.g., [45] [34] [25] [13] [21] [17] [30] [44] [36] [9]). These methods have been used for genomic discovery in a spectrum of diseases, identifying: *de novo* mutations associated with risk of neurodevelopmental and neuropsychiatric disorders [14]; variants associated with body mass index (BMI) in African American participants of the Women's Healh Initiative [10]; genes associated with risk of non-small cell lung cancer (NSCLC) [39]; common variants associated with chemotherapy induced neuropathy in breast cancer patients [3]; common variants associated with overall survival in pancreatic cancer patients [12]. Unlike the Wald and likelihood ratio tests [7], the efficient score statistic does not require numerical optimization of the primary model parameters. The method also enables the incorporation of baseline covariates into the analysis.

In order to conduct the inference, one must estimate the sampling distribution of the statistics. One approach is to use asymptotics, or large sample theory. That is, to consider the limit, as the sample size of the study approaches infinity, of the sampling distribution under certain assumptions. These assumptions, often called regularity conditions, are often not realized. For example, it can be shown that the type I error rate can be severely inflated for SNPs that have a low mutation rate [26]. Furthermore, for some studies the sample size may not be large enough to warrant the use of large sample methods in the first place.

Resampling methods [40] provide an alternative approach for genomic inference. They generally impose fewer assumptions than their asymptotic counterparts, at the cost of greatly increasing the computational burden of the analysis. While asymptotic approaches require that the statistics are calculated only once per analysis, resampling based approached require that this multitude of statistics is calculated repeatedly. As both the marginal score statistics and the resampling replications are calculated independently, both procedures are embarrassingly parallel, offering the potential for massive reductions in computation time.

To address the resulting computational challenge for resampling based inference, what is needed is a scalable and distributed computing approach. We stipulate that a cloud computing platform is suitable as it allows researchers to conduct data analyses at moderate costs, participating in the absence of access to a large computer infrastructure [16], [35], [37]. The pay-as-you-go model of cloud computing, which removes the maintenance effort required for a high performance computing (HPC) facility while simultaneously offering elastic scalability, makes it well suited for genomic analysis.

Apache Spark is a new computing framework that can outperform Hadoop significantly in iterative machine learning jobs if data fits into memory across a large number of compute nodes. Beyond in-memory computing, Spark has been used to interactively query a 39 GB dataset with sub-second response time [43].

Spark introduces an abstraction called resilient distributed datasets (RDDs). RDDs constitute a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. Spark also provides a caching mechanism, where users can explicitly cache an RDD in memory across machines and reuse it in multiple map-reduce-like parallel operations.

This work gives an overview of the statistical methodology needed for the aggregation of SNP-level associations into feature-level statistics, as well as the computational algorithms used to implement them in Apache Spark. We illustrate the application of this computational approach for the analysis of data from genome-wide analysis studies (GWAS). Experiments conducted with Amazon's Elastic MapReduce (EMR) on synthetic data sets demonstrate the efficiency and scalability of SparkScore, including high-volume resampling of very large data sets.

In the following, we use the terms "resampling" (repeated calculation within a sample space in statistics) and "iteration" (repeated calculation in computer science) interchangeably.

II. Method

Statistical Model

A SNP is typically represented as a pair, $(\mathrm{chr}, \mathrm{pos})$, according to its position, pos, on a chromosome, chr, with respect to a reference genome. Without loss of generality, we index the sequenced or genotyped SNPs using the integers $1, \ldots, J$. Let G_{ij} denote the genotype for patient i at locus j, and Y_i be a random variable that quantifies or qualifies the phenotype, or outcome of interest, for that patient, e.g., survival time, extent of disease, or a biomarker level. We denote the marginal null hypothesis for locus j, that the SNP j and the phenotype of interest are independent (not associated) as $H_{0j} : Y \perp G_j$. We propose to test this hypothesis using the method of efficient score. Let $U_j = \sum_{i=1}^{n} U_{ij}$ be the corresponding marginal score, where U_{ij} is the contribution of patient i to the score for locus j. If the score statistic is large in magnitude, we take this as statistical evidence for association between the SNP and the outcome.

The score statistics for individual SNPs can be combined to test for associations between the phenotype genes. A gene can be represented as a triplet, $(\mathrm{chr}, \mathrm{start}, \mathrm{end})$, where start and end are the start and end positions of the gene on chromosome chr with respect to the reference genome. Without loss of generality, we index the genes with the integers $1, \ldots, K$. Then let $\mathbb{I}_J = \{I_1, \ldots, I_K\}$ be a partition of the SNPs $1, \ldots, J$. In other words, each I_k is a non-empty subset of $\{1, \ldots, J\}$, which we will refer to as a SNP-set, containing all SNPs j whose positions lie within gene k. The corresponding null hypothesis for SNP-set k is denoted by $\mathbb{H}_k = \cap_{j \in I_k} H_j$. Rejecting this hypothesis would indicate that there is statistical evidence that at least one variant within the set is associated with the phenotype.

The SNP-set statistics are composed from the marginal score statistics of the member SNPs. One method of combining the marginal scores is the Sequence Kernel Association Test (SKAT) [42]. The SKAT statistic for SNP-set k is given by

$$S_k = \sum_{j \in I_k} \omega_j^2 W_j^2 = \sum_{j \in I_k} \omega_j^2 \left\{ \sum_{i=1}^{n} U_{ij} \right\}^2,$$

where ω_j is the weight for SNP j. For example, SNPs could be weighted by the quality of the genotyping results, their relative allelic frequency, or by the probability that a mutation at that locus is detrimental. For a review of methods for SNP-set testing, see [28], [4], or [18].

In order to gauge if the resulting statistics provide sufficient evidence to reject any of the K null hypotheses, we must estimate the sampling distribution of the K SKAT statistics from the data. To this end, we will consider two resampling based approaches. A permutation replicate for the marginal statistic U_j is obtained randomly by shuffling the phenotype pairs $\{(Y_1, \Delta_1), \ldots, (Y_n, \Delta_n)\}$ among the patients, and then updating the U_{ij} terms as $\tilde{U}_{ij}$. The terms $(\tilde{U}_1, \ldots, \tilde{U}_m)$ ultimately yield resampling replicates of the SKAT statistics $\tilde{S}_1 \ldots, \tilde{S}_K$.

An alternative method to permutation resampling is the Monte Carlo-based method proposed by Lin [20]. Here replicates are obtained by first simulating a random sample $Z_1, \ldots, Z_n$ from a standard normal distribution, and then updating the U_{ij} terms according to $\tilde{U}_{ij} = Z_i U_{ij}$. The advantages of this method, compared to permutation resampling, are that it is computationally more efficient, since it reuses the original U_{ij}, and that it allows for incorporation of baseline covariates in the analysis.

In either case, the $\tilde{S}_k$ from such replicates are then used as an empirical estimate of the distribution of the observed SKAT statistic S_k. The smaller the proportion of resampling statistics found to be greater than the observed statistic, the stronger the evidence of an association between the SNP-set and the phenotype. This proportion forms the p-value for the SKAT statistic, and the precision of the p-value is therefore directly tied to the number of resamplings performed.

Let us consider these issues within the context of a specific example that forms the basis for our experiments. Suppose that the phenotype of interest is time to death following start of a treatment regimen in a clinical trial. For each patient, periodic follow-up reports, providing updated health information, are received. At the time of data analysis, the actual time of death is only observed for those patients for whom a report of death has already been received. For the remaining patients, what is observed is not time of death but rather the length of time between the start of therapy and the last follow-up report. This is typically called the follow-up time. For the purpose of the analysis, the times of death for these patients are effectively censored at their respective last follow-up dates. The corresponding phenotype can be presented as the pair (Y_i, Δ_i), where Y_i is the observed time and $\Delta_i \in \{0, 1\}$ is the event indicator. For a dead patient, Y_i denotes time of death and Δ_i is set to 1, while for a censored patient Y_i denotes the last follow-up time and Δ_i is set to 0.

The Cox score statistic [6] is a commonly used for inference in this setting. This statistic, under the null hypothesis H_j, is given as

$$U_{ij} = \Delta_i(G_{ij} - a_{ij}/b_i),$$

where $a_i = \sum_{l=1}^{n} \mathbf{1}(Y_l \geq Y_i)G_{lj}$, and $b_i = \sum_{l=1}^{n} \mathbf{1}(Y_l \geq Y_i)$. Here, $\mathbf{1}(x)$ is the indicator function. These U_{ij} are then aggregated into U_j, and the U_j, for each $j \in I_k$, are further combined to form the SKAT statistics for each gene k. Note that b_i is invariant with respect to the SNP, as it is not indexed by j, and only needs to be calculated once per analysis.

To illustrate the computational advantage of the score test compared to the Wald and likelihood ratio tests, note that the latter would require solving the equation

$$U_j(\beta_j) = \sum_{i=1}^{n} \Delta_i \left\{ G_{ij} - \frac{\sum_{l=1}^{n} \mathbf{1}(Y_l \geq Y_i)G_{lj} \exp(\beta_j G_{lj})}{\sum_{l=1}^{n} \mathbf{1}(Y_l \geq Y_i) \exp(\beta_j G_{lj})} \right\} = 0,$$

for β_j. Given that there is no closed-form solution for this equation, numerical methods for optimization or root-finding will have to be employed. It should be noted that this optimization must be executed for every SNP in the analysis. Computational complexity aside, the use of the Walk or likelihood ratio tests, would also require that convergence of each optimization is monitored, and that corrective actions are taken in case of failure of convergence.

Computational Model

In this section, we outline the computational approach, including the algorithms for Monte Carlo and permutation resampling. As mentioned in the introduction, Apache Spark not only provides scalability and fault tolerance of map-reduce, but it also provides multiple useful operations, such as cache, join, etc.

Figure 1 shows a simple illustration of the SparkScore framework. We tested SparkScore on YARN (Yet Another Resource Negotiator) and Spark clusters. Because we only tested SparkScore on a Hadoop Distributed File System (HDFS), we only depict HDFS in Figure 1, even though SparkScore, like any other Spark-based application, could run on other non-HDFS Data Management Services (e.g., Ceph).

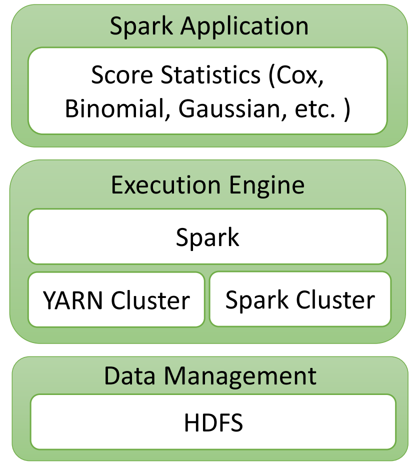

Figure 1: SparkScore Framework

Spark algorithms for calculating Cox scores and resampling statistics are shown in Algorithms 1, 2 and 3. For both the Monte Carlo and permutation methods, we first calculate the observed scores, S_k^0. Algorithm 2, after calculating S_k^0, calls Algorithm 1's steps 7 to 12 for each iteration. Therefore, Algorithm 2 is the iterative version of Algorithm 1. On the other hand, in the Monte Carlo implementation, step 8 of Algorithm 1 is different. After calculating S_k^0, the Monte Carlo algorithm caches the U RDD and reuses it in the next iterative steps. Apache Spark provides caching to explicitly cache an RDD in memory across machines and reuse it in multiple map-reduce-like parallel operations [43]. To assess the impact of caching, we conducted multiple experiments reported in Section V.

III. SYNTHETIC DATA SETS

The synthetic data sets are generated using the R [31] statistical environment. Given that they are generated for sole the purpose of assessing relative efficiencies of computational approaches, rather than their statistical operating characteristics, it is not necessary to fully account for the biological and clinical characteristics present in real data. For example, in reality, certain pairs of SNPs would be highly correlated across patients, but here they are generated independently. Also, in practice, patient survival and censoring times are generated independently and then compared to assure a fixed, known median survival time. Here, the event indicator is applied arbitrarily.

The phenotype, survival time, for each patient is drawn from an exponential distribution with parameter $\frac{1}{12}$, simulating a mean survival time of 12 months. The event/censoring status for each patient is drawn from a Bernoulli distribution with parameter 0.85, yielding an 85% event rate. For each SNP j, the genotypes, G_{ij}, for all patients are drawn from a binomial distribution with parameter $(2, \rho_j)$. Here, $\rho_j \in (0, 1)$ denotes the relative allelic frequency of the polymorphism, and is varied across SNPs. Finally, the SNP-sets are composed arbitrarily from all simulated SNPs by sampling the size of each set from an exponential distribution with parameter m/K. Here, m is the total number of SNPs being simulated and K is the desired number of SNP-sets. The resulting values are rounded down to the nearest integer, or up to 1 if they are between 0 and 1. To ensure that the computation time attributed to the analysis of each SNP is accounted for in our simulations, the SNP-set K is augmented by the SNPs not picked by SNP-sets 1 through $K-1$. As the number of SNPs included in the calculations is a critical factor in the execution time, in practical applications, any SNP not present in at least one SNP-set would be excluded from the data.

Algorithm 1: Computing SKAT Statistic S_k

Input: Genotype Matrix, Pairs of Events and Survival Times per Patient, SNP Weights, SNP-Sets
Output: HashMap<$SNPSet_k$, S_k>
1 Read input files from HDFS;
2 $RDD_{Weights_{SNP}}$ = **Map** (SNP Weight Text File) **emit** (**Key**: SNP_j, **Value**: $Weight^2_{SNP_j}$);
3 RDD_{GM} = **Map** (Genotype Matrix Text File) **emit** (**Key**: SNP_j, **Value**: (($Patient_1$, $Value_1$) ... ($Patient_n$, $Value_n$)));
4 $UnionSet_{SNPSets} = \bigcup_{k=1}^{K} SNPSet_k$;
5 RDD_{FGM} = **Filter**(RDD_{GM} based on $UnionSet_{SNPSets}$);
6 **Broadcast** Pairs of <Event Indicator, Survival Time> over all cluster nodes;
7 RDD_U = **Map**(RDD_{GM}):
  (I) **For each** $Patient_i$:
    Calculate $U[SNP_j, Patient_i]$;
  (II) **emit** (**Key**: SNP_j, **Value**: list<$Patient_i$, $U[SNP_j, Patient_i]$>);
8 $RDD_{InnerSigma}$ = **Map** (RDD_U):
  (I) Calculate $U^2_{SNP_j} = \{ \sum_{i=1}^{n} U(Patient_i, SNP_j) \}^2$;
  (II) **emit** (**Key**: SNP_j, **Value**: $U^2_{SNP_j}$);
9 RDD_{Join} = **Join** ($RDD_{Weights_{SNP}}$ and $RDD_{InnerSigma}$):
  emit (**Key**: SNP_j, **Value**: $< U^2_{SNP_j}, Weight^2_{SNP_j} >$);
10 $RDD_{SNP_{score}}$ = **Map** (RDD_{Join}):
  emit (**Key**: SNP, **Value**: $Weight^2_{SNP_j} \times U^2_{SNP_j}$)
11 **For** all $SNPSet_k$:
  Calculate $S_k = \{ \sum_{SNP_j \in SNPSet_k}^{J} SNP_{score_j} \}$;
12 **return Key**: SNP-Set, **Value**: Score;

Algorithm 2: Permutation Method

Input: Genotype Matrix, Pairs of Events and Survival Times per Patient, SNP Weights, SNP-Sets, Number of Iterations (B)
Output: HashMap<$SNPSet_k$, $counter_k$>
1 HashMap<$SNPSet_k$, S^0_k> = **Call** Algorithm 1, and calculate Observed Score (S^0_k) ;
2 Generate B random shufflings of the pairs of <Event Indicator, Survival Time>;
3 **For** $b = 1$ to B:
  (I) Recalculate step 6 to 12 of Algorithm 1, iterating over the shufflings of pairs of events and survival times to get S^b_k;
  (II) **For** all $SNPSet_k$, **if** S^b_k >= S^0_k **then** increment $counter_k$;

Algorithm 3: Monte Carlo Method

Input: Genotype Matrix, Pairs of Event and Survival Time per Patient, SNP Weights, SNP-Sets, Number of Iterations (B)
Output: HashMap<$SNPSet_k$, $counter_k$>
1 HashMap<$SNPSet_k$, S^0_k> = **Call** Algorithm 1, and calculate Observed Score (S^0_k);
2 **Cache** RDD_U from Algorithm 1;
3 Generate B sets of n random samples from a Normal$(0, 1)$ distribution to serve as Monte Carlo weights, $MCWeight_{Patient_i}$;
4 **For** $b = 1$ to B:
  (I) As a modification of Step 8 in Algorithm 1, $RDD_{InnerSigma}$ = **Map** (RDD_U):
    (a) Calculate $\sum_{i=1}^{n} U(Patient_i, SNP_j) \times MCWeight_{Patient_i}$;
    (b) **emit** (**Key**:SNP_j, **Value**: $U^2_{SNP_j}$);
  (II) Continue steps 9 to 12 of Algorithm 1;
  (III) **For** all $SNPSet_k$, **if** S^b_k >= S^0_k **then** increment $counter_k$;

Again, while these synthetic data may not fully reflect biological or clinical features of real data, the size and format used in our simulations are representative of those found in actual data, ensuring realistic and relevant algorithm execution times.

IV. Experimental Setup

We utilize Amazon's EMR to conduct the experiments. We create clusters of $m3.2xlarge$ Amazon EC2 instances. Table I provides information about $m3.2xlarge$. Apache Spark with YARN is supported on Amazon EMR clusters.

Table I: m3.2xlarge - Amazon EC2 Instances

Processors	vCPU	Mem (GiB)	Storage (GB)
Intel Xeon E5-2670 v2 (Ivy Bridge)	8	30	2×80

To assess the scalability of the proposed Spark algorithms, we conduct three types of experiments. In Experiment A, we test the scalability and sensitivity of the Monte Carlo method relative to the permutation method. In Experiment B, we test the impact of software caching provided by Apache Spark on the Monte Carlo method. And in Experiment C, we prototype and evaluate selected auto-tuning capabilities using SparkScore.

To asses runtime predictability and report the standard deviation, selected configurations of Experiments A and B are run five times each, and the results are summarized in Tables III and V. However, due to funding limitations and the significant runtimes required for the permutation method, other experiment configurations are run only twice.

In the following Figures and Tables, the zero iteration case represents the execution time of calculating S^0_k using Algorithm 1. The additional iterations, S^1_k to S^B_k represent the statistical resamplings using Algorithm 2 or Algorithm 3.

Moreover, for the Monte Carlo method, software caching is enabled, unless explicitly stated otherwise.

V. Results and Analysis

A. Scalability and Sensitivity

Our first experiment focuses on the scalability of the Monte Carlo method compared to the permutation method. Input parameters of this experiment are shown in Table II. Figure 2 depicts the performance of the two methods over different numbers of iterations. The x- and y-axes denote the number of iterations and execution time in seconds, respectively. As we increase the number of iterations, we can see that the Monte Carlo method significantly outperforms the permutation method. For 16 iterations, the run-time of Monte Carlo is an order of magnitude faster than that of permutation. Also, the execution time of Monte Carlo with 10,000 iterations is still less than permutation with 16 iterations. Table III summarizes the mean and standard deviation of the runtimes for five executions of each method for the specified numbers of iterations. Note that the standard deviations remain small relative to the execution times, indicating high predictability of the runtimes.

Table II: Input Parameters for Experiment A

Patients	SNPs	SNP-Sets	Avg. # SNPs per SNP-Set	Nodes
1000	100000	1000	100	6

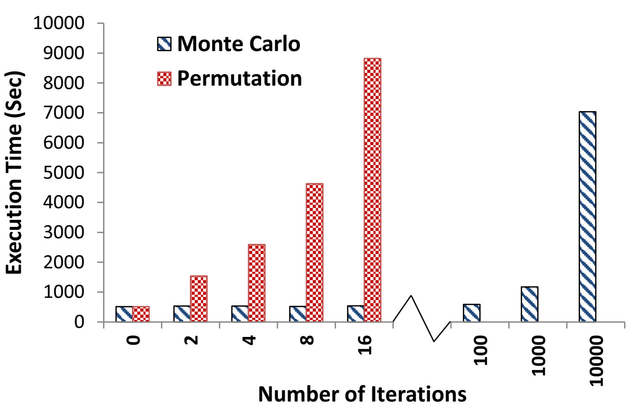

Figure 2: Scalability - Monte Carlo vs. Permutation Resampling

Figure 3 depicts the sensitivity of the methods under different numbers of SNPs and iterations. In this experiment, the number of iterations $\times$ number of SNPs is constant. Just as in the previous experiment, Monte Carlo outperforms permutation in terms of performance. However, within each method performance is quite similar for the three different configurations, with each resulting in the same amount of work.

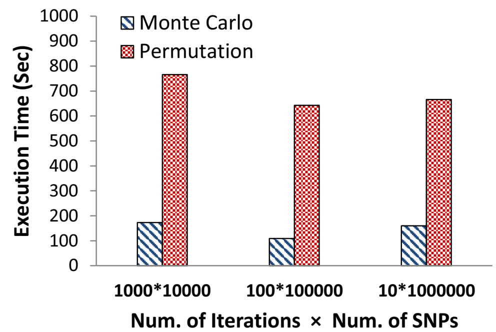

Figure 3: Sensitivity - Monte Carlo vs. Permutation Resampling

B. Caching

To assess the impact of caching in our framework, we test two sets of data simulation parameters. The number of SNPs is $10K$ in the one simulation and $1M$ in another. Input parameters are shown in Table IV. The results in Figures 4 and 5 indicate a significant impact of caching on Monte Carlo. The x- and y-axes denote the number of iterations and execution time in seconds, respectively. The y-axis is logarithmic in scale for Figure 4. For a genotype matrix with $10K$ SNPs, the cached version of Monte Carlo for $10,000$ iterations is faster than for 200 iterations of the method without caching (Figure 4). In Figure 5, for a genotype matrix with $1M$ SNPs, the cached version of the Monte Carlo for 1000 iterations is faster than 10 iterations for Monte Carlo without caching. Table V summarizes the mean and standard deviation of the runtimes for five executions of each number of iterations, with and without caching.

Table IV: Input Parameters for Experiment B

Patients	SNPs	SNP-Sets	Avg. # SNPs per SNP-Set	Nodes
1000	10K	1000	100	18
1000	1M	1000	1000	18

C. Auto-tuning

We investigate the benefit of performance optimization using techniques of auto-tuning in two ways. In the first, we consider strong scaling, where the number of tasks per node is changed while the program input size is constant. Input parameters are reported in Table VI and the results are shown in Figure 6. We observe that by increasing the amount of resources for the same workload, the execution time is reduced significantly. The execution time of 18 nodes for 20 iterations is two orders of magnitude smaller than that for 6 nodes.

With Apache Spark running on a YARN cluster in EMR, for the second investigation we consider three run-time flags of Apache Spark: (1) the number of executors (containers), (2) the amount of memory per executor, and (3) the number of cores per executor. The input parameters are shown in

Table III: Average Runtimes and Standard Deviations for Experiment A

Iterations	0	2	4	8	16	100	1000	10000
Monte Carlo Avg.	509.4	532.2	532.4	516.4	542.8	590.4	1170.8	7036.6
Monte Carlo STDV	9.65	23.15	19.26	17.54	12.23	16.89	54.1	40.29
Permutation Avg.	509.4	1535.2	2594.4	4628.4	8818.6	N/A	N/A	N/A
Permutation STDV	9.65	74.77	48.64	132.67	344.61	N/A	N/A	N/A

Table V: Average Runtimes and Standard Deviation for Experiment B

Iterations	0	10	100	200	300	400	500	600	700	800	900	1000	10000
Caching Avg.	94	101	132	140.4	163.6	178.4	188.2	214.8	225.5	241.8	257.4	283	1928.6
Caching STDV	8.51	4.89	24.28	3.64	9.09	7.53	6.76	12.29	7.25	7.66	10.21	13.58	138.35
NoCache Avg.	94	641.4	5418	10709	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
NoCache STDV	8.51	34.88	78.19	62.14	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

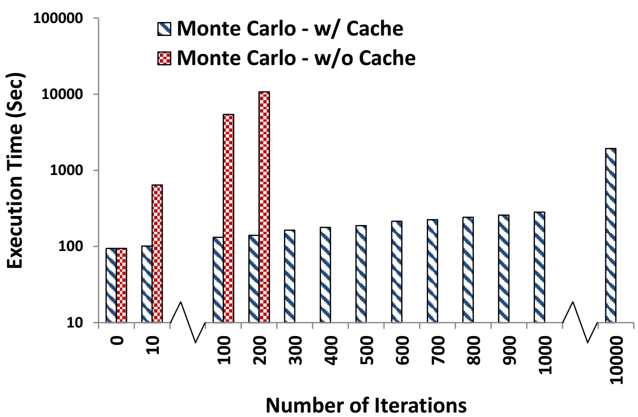

Figure 4: Monte Carlo w/ and w/o Caching - 10K SNPs

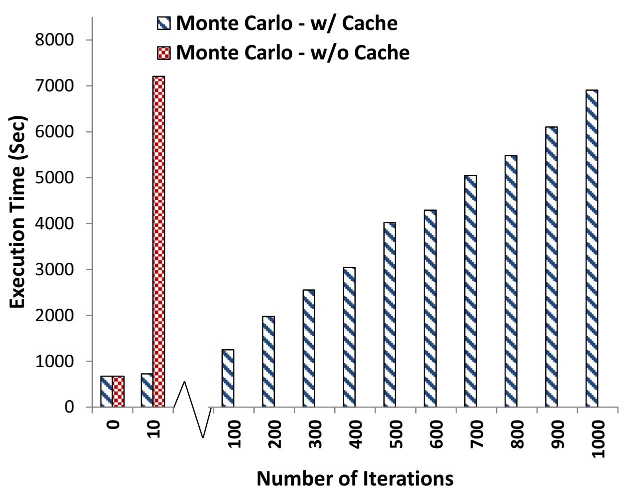

Figure 5: Monte Carlo w/ and w/o Caching - 1M SNPs

Table VI: Input Parameters of the Strong Scaling Investigation

Patients	SNPs	SNP-Sets	Avg. # SNPs per SNP-Set	Nodes
1000	1M	1000	1000	6
1000	1M	1000	1000	12
1000	1M	1000	1000	18

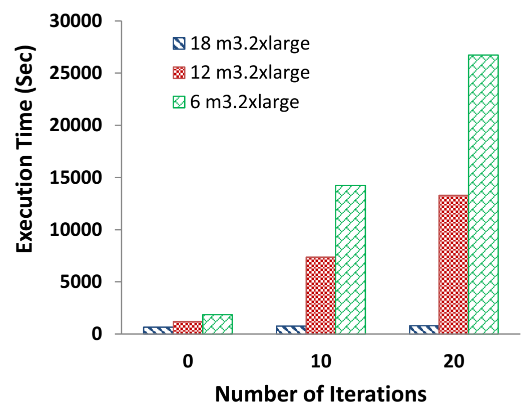

Figure 6: Strong Scaling - 1M SNPs

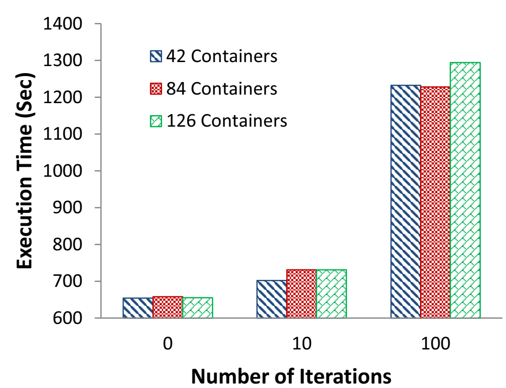

Figure 7: Apache Spark Run-time Properties on YARN Cluster - 1M SNPs

Tables VII and VIII. Figure 7 shows that the performance difference for different numbers of containers using a constant number of cluster nodes is almost negligible.

Table VII: Input Parameters of SparkScore - Auto-Tuning Investigation

Patients	SNPs	SNP-Sets	Avg. # SNPs per SNP-Set	Nodes
1000	1M	1000	1000	36

Table VIII: Input Parameters of the Apache Spark Run-time Properties on YARN Cluster - Auto-Tuning Investigation

Containers	Amount of Memory per Container (GiB)	Cores per Container
42	10	6
84	10	3
126	8	2

VI. Related Work

BlueSNP [11] is an R extension package for conducting GWAS on Hadoop clusters. A scalable implementation of SNP-Pair testing for genetic association is reported in [15]. There they design a parallel implementation of statistical correlations between particular loci in the genome of an individual plant and the expressed characteristics of that individual. The work is implemented in MPI and OpenMP. A hybrid map-reduce and MPI library is proposed [38]. This approach aims to find a middle ground between a deep redesign and an existing sequential algorithm with MPI calls. They mention that the price for this flexibility is a lack of fault tolerance due to the underlying MPI execution model.

Hadoop and Spark have been extensively used for DNA and protein sequence alignment and mapping [24] [33] [27]. SparkSeq [41] performs in-memory computations on the Cloud via Apache Spark. It covers operations on Binary Alignment/Map (BAM) and Sequence Alignment/Map (SAM) files [19], and it supports filtering of reads summarizing genomic features and basic statistical analyses operations. AzureBlast [22] is a parallel BLAST (Basic Local Alignment Search Tool [2]) engine on the Windows Azure cloud platform. BLAST searches a database of subject sequences and discovers all the local similarities between the query sequence and subject sequences. In AzureBlast, the input sequences are divided into multiple partitions and distributed among worker instances. After workers have processed all data partitions, the results are merged together. The scalability potential of the the Burrow-Wheeler Aligner DNA mapping algorithm [5] is analysed in [1]. The paper compares the performance of three implementations: native cluster-based, Hadoop, and Spark versions.

VII. Conclusion and Future Work

This work describes SparkScore, a set of distributed computational algorithms, implemented in Apache Spark, that leverage the embarrassingly parallel nature of asymptotic and resampling inference on the basis of efficient score statistics in the context of genomic inference. We evaluated the scalability of SparkScore under different sets of experiments on the AWS cloud. Experiments showed that Apache Spark features such as caching could significantly reduce the execution time. We plan to further investigate Apache Spark parameter options for SparkScore for the purpose of tuning.

Acknowledgment

In memory of Someyra Nazemi Gelian, a graduate student at NC State who battled to the end with cancer and ultimately inspired this work. This research is partially supported by grants from the National Science Foundation, award numbers 0988311 (FM) and 1217748 (FM), and by a grant from the National Institutes of Health, award number P01CA142538 (KO). The costs for using the Amazon Web Services (AWS) to conduct the experiments are covered by AWS in Education Research grant awards (AB, KO, FM). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

[1] Zaid Al-Ars and Hamid Mushtaq. Scalability potential of BWA DNA mapping algorithm on apache spark.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

[3] R. M. Baldwin, K. Owzar, H. Zembutsu, A. Chhibber, M. Kubo, C. Jiang, D. Watson, R. J. Eclov, J. Mefford, H. L. McLeod, P. N. Friedman, C. A. Hudis, E. P. Winer, E. M. Jorgenson, J. S. Witte, L. N. Shulman, Y. Nakamura, M. J. Ratain, and D. L. Kroetz. A genome-wide association study identifies novel loci for paclitaxel-induced sensory peripheral neuropathy in CALGB 40101. *Clin. Cancer Res.*, 18(18):5099–5109, Sep 2012.

[4] Saonli Basu and Wei Pan. Comparison of statistical tests for disease association with rare variants. *Genetic Epidemiology*, 35(7):606–619, 2011.

[5] Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. 1994.

[6] D.R. Cox. Regression models and life tables (with discussion). *J. Roy. Statist. Soc. Ser. B*, (34):187—220, 1972.

[7] D.R. Cox and D.V. Hinkley. *Theoretical Statistics*. Chapman and Hall/CRC, 1974.

[8] Sara El-Metwally, Taher Hamza, Magdi Zakaria, and Mohamed Helmy. Next-generation sequence assembly: Four stages of data processing and computational challenges. *PLoS Comput Biol*, 9(12):e1003345, 12 2013.

[9] S. S. Han, P. S. Rosenberg, A. Ghosh, M. T. Landi, N. E. Caporaso, and N. Chatterjee. An exposure-weighted score test for genetic associations integrating environmental risk factors. *Biometrics*, 71(3):596–605, Sep 2015.

[10] Y. J. Hu, Y. Li, P. L. Auer, and D. Y. Lin. Integrative analysis of sequencing and array genotype data for discovering disease associations with rare mutations. *Proc. Natl. Acad. Sci. U.S.A.*, 112(4):1019–1024, Jan 2015.

[11] Hailiang Huang, Sandeep Tata, and Robert J. Prill. BlueSNP: R package for highly scalable genome-wide association studies using hadoop clusters. *Bioinformatics*, 29(1):135–136, 2013.

[12] F. Innocenti, K. Owzar, N. L. Cox, P. Evans, M. Kubo, H. Zembutsu, C. Jiang, D. Hollis, T. Mushiroda, L. Li, P. Friedman, L. Wang, D. Glubb, H. Hurwitz, K. M. Giacomini, H. L. McLeod, R. M. Goldberg, R. L. Schilsky, H. L. Kindler, Y. Nakamura, and M. J. Ratain. A genome-wide association study of overall survival in pancreatic cancer patients treated with gemcitabine in CALGB 80303. *Clin. Cancer Res.*, 18(2):577–584, Jan 2012.

[13] I. Ionita-Laza, S. Lee, V. Makarov, J. D. Buxbaum, and X. Lin. Sequence kernel association tests for the combined effect of rare and common variants. *Am. J. Hum. Genet.*, 92(6):841–853, Jun 2013.

[14] Y. Jiang, Y. Han, S. Petrovski, K. Owzar, D. B. Goldstein, and A. S. Allen. Incorporating functional information in tests of excess de novo mutational load. *The American Journal of Human Genetics*, 97(2):272–283, 2015.

[15] Lars Koesterke, Dan Stanzione, Matt Vaughn, Stephen M Welch, Waclaw Kusnierczyk, Jinliang Yang, Cheng-Ting Yeh, Dan Nettleton, and Patrick S Schnable. An efficient and scalable implementation of SNP-pair interaction testing for genetic association studies. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 523–530. IEEE, 2011.

[16] Ben Langmead, Kasper D Hansen, Jeffrey T Leek, et al. Cloud-scale RNA-sequencing differential expression analysis with Myrna. *Genome Biol*, 11(8):R83, 2010.

[17] S. Lee, M. J. Emond, M. J. Bamshad, K. C. Barnes, M. J. Rieder, D. A. Nickerson, D. C. Christiani, M. M. Wurfel, and X. Lin. Optimal unified approach for rare-variant association testing with application to small-sample case-control whole-exome sequencing studies. *Am. J. Hum. Genet.*, 91(2):224–237, Aug 2012.

[18] Seunggeung Lee, Goncalo Abecasis, Michael Boehnke, and Xihong Lin. Rare-variant association analysis: Study designs and statistical tests. *The American Journal of Human Genetics*, 95(1):5 – 23, 2014.

[19] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[20] D. Y. Lin. An efficient Monte Carlo approach to assessing statistical significance in genomic studies. *Bioinformatics*, 21(6):781–787, 2005.

[21] D. Y. Lin and Z. Z. Tang. A general framework for detecting disease associations with rare variants in sequencing studies. *Am. J. Hum. Genet.*, 89(3):354–367, Sep 2011.

[22] Wei Lu, Jared Jackson, and Roger Barga. Azureblast: a case study of developing science applications on the cloud. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 413–420. ACM, 2010.

[23] Jason H. Moore, Folkert W. Asselbergs, and Scott M. Williams. Bioinformatics challenges for genome-wide association studies. *Bioinformatics*, 26(4):445–455, 2010.

[24] Matti Niemenmaa, Aleksi Kallio, André Schumacher, Petri Klemelä, Eija Korpelainen, and Keijo Heljanko. Hadoop-bam: directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, 28(6):876–877, 2012.

[25] K. Owzar, Z. Li, N. Cox, and S. H. Jung. Power and sample size calculations for SNP association studies with censored time-to-event outcomes. *Genet. Epidemiol.*, 36(6):538–548, Sep 2012.

[26] Kouros Owzar, Zhiguo Li, Nancy Cox, and Sin-Ho Jung. Power and sample size calculations for SNP association studies with censored time-to-event outcomes. *Genetic Epidemiology*, 36(6):538–548, 2012.

[27] Aisling O'Driscoll, Jurate Daugelaite, and Roy D Sleator. 'big data', hadoop and cloud computing in genomics. *Journal of biomedical informatics*, 46(5):774–781, 2013.

[28] Wei Pan. Asymptotic tests of association with multiple SNPs in linkage disequilibrium. *Genetic Epidemiology*, 33(6):497–507, 2009.

[29] Yiming Qin, Hari Krishna Yalamanchili, Jing Qin, Bin Yan, and Junwen Wang. The Current Status and Challenges in Computational Analysis of Genomic Big Data. *Big Data Research*, 2(1):12–18, 2015. Special Issue on Computation, Business, and Health Science.

[30] L. Qu, T. Guennel, and S. L. Marshall. Linear score tests for variance components in linear mixed models and applications to genetic association studies. *Biometrics*, 69(4):883–892, Dec 2013.

[31] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[32] C.R. Rao. Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. *Proceedings of the Cam- bridge Philosophical Society*, (44):50–57, 1948.

[33] G. Sudha Sadasivam and G. Baktavatchalam. A novel approach to multiple sequence alignment using hadoop data grids. In *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*, page 2. ACM, 2010.

[34] D. J. Schaid. General score tests for associations of genetic markers with disease using cases and their parents. *Genet. Epidemiol.*, 13(5):423–449, 1996.

[35] Michael C. Schatz, Ben Langmead, and Steven L. Salzberg. Cloud computing and the DNA data race. *Nature biotechnology*, 28(7):691–693, 2010.

[36] I. D. Shterev, S. H. Jung, S. L. George, and K. Owzar. permGPU: Using graphics processing units in RNA microarray association studies. *BMC Bioinformatics*, 11:329, 2010.

[37] Lincoln D. Stein et al. The case for cloud computing in genome informatics. *Genome Biol*, 11(5):207, 2010.

[38] Seung-Jin Sul and Andrey Tovchigrechko. Parallelizing BLAST and SOM algorithms with MapReduce-MPI library. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 481–489. IEEE, 2011.

[39] Zhaoxi Wang, Yang Zhao, Li Su, Seunggeun Lee, Xihong Lin, and David C. Christiani. Abstract 2547: Targeted sequencing of 5p15 locus defined by non-small cell lung cancer (nsclc) gwas using next-generation sequencing. *Cancer Research*, 73(8 Supplement):2547, 2013.

[40] P.H. Westfall and S.S. Young. *Resampling-Based Multiple Testing: Examples and Methods for P-Value Adjustment*. A Wiley-Interscience publication. Wiley, 1993.

[41] Marek S. Wiewiórka, Antonio Messina, Alicja Pacholewska, Sergio Maffioletti, Piotr Gawrysiak, and Michał J. Okoniewski. Sparkseq: fast, scalable, cloud-ready tool for the interactive genomic data analysis with nucleotide precision. *Bioinformatics*, page btu343, 2014.

[42] Michael C. Wu, Seunggeun Lee, Tianxi Cai, Yun Li, Michael Boehnke, and Xihong Lin. Rare-variant association testing for sequencing data with the sequence kernel association test. *The American Journal of Human Genetics*, 89(1):82 – 93, 2011.

[43] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010.

[44] P. Zeng, Y. Zhao, H. Li, T. Wang, and F. Chen. Permutation-based variance component test in generalized linear mixed model with application to multilocus genetic association study. *BMC Med Res Methodol*, 15:37, 2015.

[45] F. Zou, J. P. Fine, J. Hu, and D. Y. Lin. An efficient resampling method for assessing genome-wide statistical significance in mapping quantitative trait Loci. *Genetics*, 168(4):2307–2316, Dec 2004.