Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ Date: _____

Xiaobo Sun

The Applications of NoSQL Systems and Ensemble Learning in Managing, Processing and Analyzing Big Omics Data

By

Xiaobo Sun
Doctor of Philosophy

Computer Sciences and Informatics

Zhaohui Qin
Advisor

Hao Wu
Committee member

Joyce Ho
Committee member

Peng Jin
Committee member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

The Applications of NoSQL Systems and Ensemble Learning in Managing, Processing and Analyzing Big Omics Data

By

Xiaobo Sun
M.S., Emory University, 2014

Advisor: Zhaohui Qin, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Sciences and Informatics
2018

Abstract

The Applications of NoSQL Systems and Ensemble Learning in Managing, Processing and Analyzing Big Omics Data

By Xiaobo Sun

The development of high-throughput genomics technologies has resulted in massive quantities of diverse omics data. However, existing dataset search tools rely almost exclusively on the metadata. To overcome this barrier, we have developed Omicseq, a novel web-based platform that facilitates the easy interrogation of omics datasets beyond just metadata. The core component of Omicseq is trackRank, a novel algorithm for ranking omics datasets that fully uses the numerical content of the dataset to determine relevance to the query entity. The Omicseq system is supported by a scalable NoSQL database that hosts a large collection of *processed* omics datasets, and provide a web-based user interface for searching and queries.

In addition, operations on big omics data can be a challenge for traditional single machine based methods. For example, sorted merging of a large number of Variant Call Format (VCF) files are frequently encountered in large scale whole genome sequencing projects. We custom design optimized schemas for Hadoop (MapReduce), HBase and Spark, to perform sorted merging of massive genome-wide data. These schemas all adopt the divide-and-conquer strategy to split and conquer tasks in an ordered, parallel and bottleneck-free way. Our experiments on merging VCF files suggest that all three schemas either deliver a significant improvement in efficiency or render much better strong/weak scalabilities over traditional methods such as the VCFTools, thus providing generalized scalable schemas for performing sorted merging on genetics and genomics data using these Apache distributed systems.

Methylation level changes of CpG sites are associated with specific diseases such as Alzheimer's disease. However, quantifying these changes across the whole genome remains a challenge, especially for those not covered by the array-based technologies. In this study, we develop an ensemble feature selection and classification model to identify the most relevant features to CpG methylation level changes in a specific disease from a comprehensive collection of genome-wide precomputed epigenomic profiles, and to predict methylation level changes at CpGs beyond the array. Therefore, it provides insights to the mechanism behind CpG methylation level changes in a specific disease as well as an approach to evaluate an individual's risk exposures to it.

The Applications of NoSQL Systems and Ensemble Learning in Managing, Processing and Analyzing Big Omics Data

By

Xiaobo Sun
M.S., Emory University, 2014

Advisor: Zhaohui Qin, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Sciences and Informatics
2018

Acknowledgements

First of all, I would like to give special thanks to my advisor, Dr. Zhaohui Qin, for his patient, tireless and wise guidance on the path of my Ph.D. studies. Only with his solid supports, can I step into the fields of both computer sciences and statistics and conduct the researches that intrigue me most. I have benefitted a lot from his integrity, consistency and rigorous scholarship which makes him far beyond just an academic advisor to me. I also would like to give my sincere thanks to my dissertation committee members, Dr. Hao Wu, Dr. Jocey Ho, and Dr. Peng Jin. Dr. Hao Wu gave me many valuable feedbacks on the dissertation and my presentation skills. Dr. Jocey Ho inspired me on many cutting-edge machine learning approaches and helped me out of puzzles during the my third studies. Dr. Peng Jin kindly supported my research financially and also brought me many thoughts from a biomedical perspective. I also give many thanks to Dr. Fusheng Wang at the StonyBrook university for his advising on my first and second studies, his supervision of my first Ph.D. qualifying exam and his kindness of inviting me to international conferences VLDB in 2017.

Secondly, I would like to thank Dr. Vaidy Sunderam, Dr. Eugene Agichtein, Dr. Li Xiong, and Dr. James Lu for offering me the opportunity to study at the Department of Computer Sciences and Mathematics in the past five years and their supports for my smooth transition from biomedical sciences to computer sciences.

Finally, I would like to deeply thank my parents who have always backed me up no matter what difficulties I have met.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Big Omics Data

The rapid development of genomics technologies, such as microarrays and massively-parallel DNA sequencing, have dramatically increased the speed of generating and size of experimental data in the life sciences and propelled the genetics studies into the Big Data era. Big Data can be defined as datasets which have a much larger size than what a traditional relational database is capable of managing and handling efficiently [1]. More formally, Big Data can be defined from three versus levels---volume, variety, and velocity [2]. The first V refers to the large size of datasets, and calls for scalable data storage, query and retrieval techniques; The second V refers to the heterogeneous data structure and format from various data resources, and calls for more flexible data integrating techniques. The third V refers to the increasing new data generating rate, and calls for versatile and efficient data retrieving and preprocessing technologies. These massive genomics data provide new information in an unprecedented scale and offer an attractive new source for biomedical knowledge. By design, these genome-wide profiling data such as ChIP-seq [3-5] and RNA-seq [6] offer unbiased, genome-wide, and often base-pair resolution information from transcription factor binding to histone modification. Scientific publications reporting these data often focus their attentions on higher level analyses that reflect important, but limited, aspects

of such data. Additionally, over the past ten years, millions of variants at genomic loci and single-nucleotide polymorphism (SNPs) have been identified by using high-density SNP arrays and DNA re-sequencing [7]. On average, more than 4 million genetic variants are identified, among which at least 100,000 are novel, per individual under investigation via deep sequencing [8]. It is now not uncommon to conduct deep sequencing in a genome-wide association studies (GWAS) involve hundreds or even thousands of individuals which generates terabytes or even petabytes of data to detect genetic variants that are associated with complex traits and common diseases in the population [9].

Furthermore, specific genomic information can be derived from sequencing data using computational and statistical methods. For example, PhastCons score [10], degree of centrality in the protein-protein interaction network, or the probability of being loss of function intolerant (pLI) of a gene, proposed by the Exome Aggregation Consortium (ExAC) [11]. Like above-mentioned genome-wide profiling data, such data also provide useful information from different perspectives. The challenge is how best to process and analyze large catalogues of genomic data. In a perspective article which describes NIH's vision of Big Data to Knowledge (BD2K) [12], Margolis et al. pointed out that "A fundamental question for BD2K is how to enable the identification, access, and citation of (i.e., credit for) biomedical data". In Eric Green's presentation on "NIH and Biomedical 'Big Data'", the first and second "major' problems to solve" for big data are "Locating the data" and "Getting access to the data". However, currently, such data is typically scattered and sequestered in data repositories that obstruct conveniently locating, retrieving, processing and interrogating the data. Consequently, it becomes increasingly important to efficiently collect, integrate, store, analyze, and display these data. For example, the RCSB Protein Data Bank [13]

searches and integrates data over 20 sources covering genomic, proteomic and disease relationships for providing queries and analyses on three-dimensional structure data of biological macromolecules. The Bio2RDF [14] is a mashup system which collects documents from public bioinformatics databases, such as Kegg, PDB,MGI, HGNC and so on, and makes them available through in standard RDF format.

Relational database systems, represented by MySQL, PostgreSQL and SQLite, have been applied in the field of bioinformatics for data management and analyses for a long time. It has the advantages of simplicity, fast data access, efficient storage, support for complex queries as well as properties of atomicity, consistency, isolation and durability (ACID). However, as bioinformatics data, especially genomic and proteomic data, grow in both size and complexity, a relational database becomes incapable of efficiently managing and handling Big Data due to its inherent drawbacks such as the requirements of rigid data schemas, indices building, and data normalization and isolation. These drawbacks lead to extra costs of data preprocessing, slow inserting and updating operations, disk I/O bottleneck, poor scalability and data availability in face of unstructured data in size of terabytes and petabytes. The advent of NoSQL solutions relieves these limitations of the relational database system to some extent. Compared to relational database systems, NoSQL database systems differ in several aspects: First, they can adopt a variety of models of how the data is stored, such as key/value stores represented by memcached, relationship stores represented by Neo4j, column-oriented stores represented by HBase and Cassandra, and document-oriented stores represented by MongoDB; Second, they can choose to whether store the data in memory or to persist the data to permanent storage; Third, they usually loose the requirements of strict consistency and normalization by storing multiple copies of data, which

affects the latency on handling large number of concurrent read and write requests. However, eventual consistency must be guaranteed by all NoSQL databases. Fourth, they usually have limited supports for transaction, atomic read-update-write and secondary index. Fifth, they can have different physical models, either single machine or distributed. And if distributed, most of them have a transparent load balancing system. Additionally, many distributed NoSQL databases have a better scalability than sharded relational databases. Sixth, they usually have a graceful failure handling system for both task and data failures.

Not only are the relational database systems inefficient for storing large scale of data, but also for large scale of processing and ad hoc analysis. This is because disk seek time is improving in a slower pace than disk transfer rate. Therefore, if the disk seek dominates the data access pattern, it will take much longer to read or write large scale of data than streaming through it, which operates at the disk transfer rate. Most relational databases adopt the B-tree-like data structure for indexing and storing data, which optimizes the query and update speed of small random data. However, data access pattern based on B-tree are dominated by disk seek, thus when the whole dataset is analyzed or processed in batch, the efficiency dramatically degrades. To address this problem, some distributed systems running on a cluster of computing nodes, represented by MapReduce, have been recently developed. These systems are designed for batch processing dataset in size of petabytes, and particularly efficient for tasks that write once and read many times. Additionally, they work well on semi-structured or unstructured data since they adopt the schema-on-read model instead of schema-on-write model used by relational database. Furthermore, they give up the data normalization, which is required by relational database to retain data integrity and remove redundancy, in return for data locality and high-speed streaming reads and writes. Finally,

they scale linearly with the data and cluster size. And some of them, such as Spark, allow dynamically adding/removing computing nodes at runtime.

In bioinformatics, researchers have already started to embrace these distributed systems, especially those developed by Apache foundation, to manage and process large amount of high throughput [15] omics data. For example, the Cancer Genome Atlas project makes use of the Hadoop framework to split genome data into chunks distributed over the cluster for parallel processing[1, 16]. The CloudBurst [17], Seal [18], Hadoop-BAM [19] and Crossbow software [20] take advantage of the Hadoop framework to accelerate sequencing read mapping, aligning and manipulations as well as SNP calling. The Collaborative Genomic Data Model (CGDM) [21] uses HBase to boost the querying speed for the main classes of queries on genomic databases. The MetaSpark [22] utilizes Spark's distributed data set to recruit large scale of metagenomics reads to reference genomes, achieves better scalability and sensitivity than single-machine based programs [23]. Industry cloud computing vendors such as Amazon [24] and Google [25] are also beginning to provide specialized environments to ease genomics data processing in the cloud.

Although numerous Apache distributed system-based applications have already been developed for processing and analyzing large scale of genomics data including ADAM [26],VariantSpark [27], SparkSeq [28], Halvade [29], SeqHBase [30] among others, we believe there are still many opportunities in biomedical data analyses to take advantage of distributed systems as data becomes larger and more complex. For example, any data manipulation problem that involves the key-value model, such as union, intersection, grouping and aggregation, sorting and shuffling, naturally fit

into the framework of these systems. Consequently, via deliberately designed workflow and execution plan, these problems can be solved in an efficient and scalable manner.

Epigenetics is a branch of omics studies which studies on inheritable non-genetic markers that are both stable through cell division and variable in response to cellular and environmental stimuli. Epigenetic markers may change within an individual over time and have been shown to exhibit cell-type specificity [31-33]. DNA methylation is the best know and extensively studied epigenetic marks and is involved in a variety of cellular processes such as cell differentiation, cancer progression, chromosome instability and son on [34]. DNA methylation usually occurs at the 5-GC-3 sites where a methyl group is added to the cytosine, which renders such sites mutation hotspots and an important role in regulating DNA replication, transcriptional activity, and disease susceptibility. Such an important role is reflected in the observations that diseased and normal individuals can show differential DNA methylation levels (either up- or down- regulated) at specific CpG sites [35]. Across the human genome, there are about 28 million single CpG methylation sites. The current golden standard for quantifying the DNA methylation levels is the whole-genome bisulfite sequencing (WGBS), which quantifies DNA methylation levels at ~26 million CpG sites in the human genome. However, WGBS also has from several limitations such as its prohibitively expensive costs, conversion bias and experimental feasibility in particular genomic regions. As a more economical alternative to the WGBS, methylation microarrays such as the Illumina HumanMethylation450 BeadChip measures the DNA methylation level at ~482,000 preselected CpG sites which consists of 2% of the total CpG sites. So if knowledge and mechanisms, such as the relationship between CpG methylation status and regulatory events and other epigenomic features, can be learnt from these preselected CpG sites, they can be applied to

predict the methylation level of the rest 98% unassayed sites and evaluate an individual's risk of exposure to diseases or propensity to phenotypes.

The powerful high-through put technologies including the next-generation sequencing (NGS) experimental assays have been developed to comprehensively survey the entire genome for regulatory events. Representative experiments of these types of assays include: Coupling chromatin immunoprecipitation and next generation sequencing (ChIP-Seq) to identify in vivo binding of transcription factors and histone marks; DNase I hypersensitive sites sequencing (DNase-seq) [36, 37] and formaldehyde-assisted isolation of regulatory elements sequencing (FAIRE-seq) [38], both for identifying open chromatin regions. Given the importance of such regulatory information, large international consortia, like the Encyclopedia of DNA Elements (ENCODE) [39] and the Roadmap Epigenomics Mapping Consortium (REMC) [40] have been formed to systematically conduct these experiments to identify functional elements with regulatory activities across hundreds of cell lines/tissues. These datasets, with a total size from hundreds of terabytes to petabytes, offer a great opportunity to link epigenomic variants such as CpG methylation status to regulatory elements, including TF binding, histone modification, and open chromatin.

1.2 Outline

In this paper, our first study goal is how best to locate, collect, store, process, analyze and access large catalogues of omics data. The second study goal is how to apply the recent distributed batch

processing systems and cloud technologies to address a specific type of omics data manipulation that involves complex and expensive operations. The third goal is how to use machine learning methods to discover biomedical knowledge from epigenomics data. In chapter 2, we introduce a ranking algorithm suite, trackRank, to identify and rank omics datasets according to their significance for a give query gene. This algorithm is implemented within a web-portal interface named OmicSeq, which provides services of searching, browsing and viewing ranked omics data, and is supported by an underlying high performance NoSQL database. In chapter 3, we implement three working schemas using three Apache distributed systems, Hadoop, HBase and Spark, to address the problem of sorted merging of omics data in a more efficient and scalable way. Their performances are also compared with traditional single machine-based and HPC-based methods. In chapter 4, we develop an ensemble learning model to identify disease-specific the methylation level change of CpG sites by leveraging a comprehensive collection of genome-wide epigenomic profiles across cell types and factors, along with other static genomic features.

Chapter 2

OmicSeq: A web-based search engine for exploring omics datasets

2.1 Introduction

Data sharing policies promulgated by the United States National Institutes of Health and adopted elsewhere have caused genome-wide profiling experimental data to accumulate rapidly in public repositories such as GEO [41], the Sequence Read Archive (SRA) [42] and ArrayExpress [43]. The NCBI Gene Expression Omnibus (GEO) has archived 80,690 experimental studies that comprise more than 2,086,234 samples since 2001 (accessed 1/10/2017 http://www.ncbi.nlm.nih.gov/geo/) [41]. In addition to individual investigators, large consortia such as the 1000 genomes [44], the Cancer Genome Atlas (TCGA) [45], International Cancer Genome Consortium (ICGC) [46], the ENCyclopedia Of DNA Elements (ENCODE) [39], modENCODE [47, 48] and Roadmap Epigenomics projects [49] are specifically tasked to provide high-quality genome-wide profile data as resources for the research community. Such a large volume of public omics data represents a tremendous resource for biomedical research because these genome-wide profiling data offer unbiased, genome-wide coverage. Therefore a dataset

generated in one study can be, and in many cases is explicitly intended, to be used for completely different studies that may constitute secondary or even tertiary analyses of the original data.

Our inability to search through massive, disparate datasets poses a major barrier to biological and biomedical research. Despite the obvious potential and promise of using omics data to address important research questions, unlike text-based data, currently it is very difficult to explore and query omics data. We believe the primary challenge facing life sciences research is the limitation in existing informatics solutions. First, genomics data is typically sequestered in disparate data repositories that impair the process of conveniently locating, retrieving, processing, and interrogating the data. Second, most data stored in public data repositories are unprocessed which means that users must process these data using arbitrary methods prior to use. This often presents significant challenges for biomedical researchers who neither have, nor have access to, bioinformatics expertise. Due to these issues, we believe most of the public omics data are under-used and, as a result, opportunities for novel discovery are being missed. Third, the ability to identify datasets of interest is very limited, given that existing search methods are usually limited to metadata only. Finally, it is a daunting task to rank and select among datasets in terms of their relevancy to a query term among a collection of diverse data types.

The search engine is perhaps the most useful tool to explore the internet. We believe a search engine [50] is also critically important for exploring the massive collection of omics datasets. For the internet search engine, the key lies in its ability to rank websites in terms of their relevancy to the query term. For example, the success of Google is largely attributed to the pageRank algorithm [51], and we believe the ranking idea is also the key to effectively explore massive genomics data.

To address this challenge, we have developed a novel informatics infrastructure named Omicseq that allows users to view, browse, and search processed, ready-to-use omics data. Our platform includes two parts: 1) a scalable and elastic database storing *processed* omics, or genome-wide profiling datasets such as ChIP-seq, RNA-seq and DNase-seq; 2) a web interface to access, browse and search for omics data. Akin to an internet search engine, given a query entity (e.g. gene or gene set) will help researchers easily identify important data with a ranked listing of the most relevant genomic data sets related to the gene or gene set. The overarching goal of the Omicseq project is to develop enabling technologies to facilitate data-driven biomedical research that makes optimal maximum use of existing public omics resource.

2.2 Methods

Processing Different Data Types

For each dataset, we first retrieve sample metadata and source file information and import them into a metadata database that will complement the subsequently processed data In cases wherein pre-processed data are directly available from the source, such as level 3 data in TCGA, we can download and import the data directly. Otherwise, we next download raw data (typically in FASTQ format) and subject them to a data processing workflow that makes use of third-party softwares such as SRATOOLKIT ([52]), bowtie2 ([53]) and TCGA assembler ([54]) to transform raw data into appropriate formats such as SAM, and BED. Subsequently, depending to the type of experiment, we call upon appropriate pipelines to calculate gene rank and percentile statistics which are maintained in the database.

ChIP-seq

Reads from ChIP and input samples will be mapped to the hg19 human reference genome separately using BWA [55, 56]. We calculate and save two types of scores, one for gene bodies and one for promoters (defined to be 5kb up- and down-stream of the transcription start sites (TSS)). This is because the promoter region is of interest to study TF binding, and the gene body is of interest for some histone marks such as H3K36me3. For each gene, the promoter score is calculated as the difference between ChIP read count and input read count, the gene body score is calculated as the difference between ChIP read count and input read count normalized (divided) by the length of the coding region. Next the two sets of scores are ranked and converted to percentiles separately.

DNase-seq

Similar to ChIP-seq data, we record the difference between numbers of reads from DNase sample and the control sample in the promoter regions of all the genes. Then the read count differences are sorted and converted to percentile.

RNA-seq

We use the transcript abundance measure represented by read per million per kilobases mapped (RPKM) value [6] as the gene-based score. We choose RPKM since it is a widely-used gene expression measure. We are aware of alternative measures such as TPM [57] and RSEM measures [58]. We plan to provide such alternative measures in the future release of Omicseq.

Copy number variation

Copy number variation (CNV) data is processed the same way as in the CNV analysis pipeline used by the TCGA consortium

(https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/CNV_Pipeline/) in which the estimated copy number for the coding region of each gene was transformed into the segment mean value, which is defined as log2(copy-number/ 2). Normal copy number (diploid regions) will have a segment mean of zero, amplified regions will have positive values, and deletions will have negative values.

Methylation

Currently, all genome-wide methylation profiling data stored in Omicseq are obtained from array-based technology (such as the Infinium HumanMethylarion 450 BeadChip array from Illumina Inc, (San Diego, CA)). The measure of methylation is taken as the beta values, calculated from array intensities as Beta=M/(M+U) using the minfi package [59]. For each gene, the average methylation measure of CG sites falling into its promoter region (5kb upstream or downstream of the TSS) is used as the quantitative measure for each gene and subsequently converted into percentiles after ranking.

GRO-seq

The global run-on-sequencing (GRO-seq) assay [60] is a sequencing-based mainly used to evaluate promoter-proximal pausing on all genes. We calculate the pausing index for each gene and treat it as the gene-based measure, and subsequently converted into percentiles after ranking.

Microarray gene expression

For microarray data, typically processed gene-level gene expression data are available from the lab where the experiment is done. We take these measures, sorted and then converted them to percentiles.

Somatic mutations

The number of single nucleotide somatic mutations found within the coding region of a gene is tallied, sorted and converted to percentiles.

DNA sequence motif

The number of a specific type of motif (such as CTCF) detected that locate within the promoter region (within 5kb upstream and downstream of the TSS) for a gene is taken as the gene-based measure. These values are then sorted and converted to percentiles.

Summary statistics

Each of the tracks we described so far represent a single sample. In some cases, the study-level summary statistics is of more interest. The average measurement from all samples in a study (for example, the average gene expression level among all prostate cancer tumor samples in TCGA) is taken as the measurement. These summary statistics are then sorted and converted to percentiles.

TrackTrank Algorithm

For omics datasets, a typical query entity is either a gene name or a set of gene names. Hence the key is to effectively rank omics datasets of diverse types given a query. As described above, for each dataset (a track), regardless of the data type, we can obtain a numerical and continuous score (such as promoter region read counts for ChIP-seq, RPKM values for RNA-seq) for each gene. In order to make the scores comparable across tracks, within each track, we convert all the scores into percentiles, which is straightforward for a set of fixed features (genes). Here we present a detailed view of the underlying trackRank component which is at the heart of the Omicseq system.

trackRank

Now suppose gene A is selected as the query gene, we retrieve the percentiles of gene A in each and every track, next we rank all these percentiles collected from all tracks for gene A and do another round of sorting. The track in which gene A has the lowest percentile is ranked the highest, the track in which gene A has the second lowest percentile ranked the second and so on. The procedure is illustrated in Figure 2.1. To speed up the search, we pre-calculate the percentile for each gene and pre-sort them within each dataset. Therefore only one round of sorting is needed to return the query result.

trackRank set

In many studies, it is often of interest to find the biological properties for a set of genes. For example, a group of genes located in the same region of the genome or belonging to the same biological pathway.

When querying multiple genes, the key is to estimate the ranking percentile for an arbitrary set of genes. To achieve this, we derive a statistical-sound approach to derive combined ranks for a group of genes, and thus assess the significance of the query gene set in each track. Suppose the query is a group of k genes. There are N tracks in total. We denote their individual percentiles in the jth track as x_{ij}'s, , $i = 1, \dots, k,\ j = 1, \dots, N$. It is reasonable to assume that $x_{ij} \sim iid\ Uniform[0,1]$. Let $y_j = \sum_{i=1}^{k} x_{ij}$, when k is large (say, $k > 10$, we can use Normal approximation to calculate the percentile of y_j courtesy of the central limit theorem (CLT). We have $\sqrt{k}\left(\frac{y_j}{k} - \frac{1}{2}\right) \sim N\left(0, \frac{1}{12}\right)$. All we need to do next is to rank all the tracks based on y_j's

2.3 Results

The Current Release of Omicseq Search Engine

We provide a Web based portal (http://www.omicseq.org) as the interface to provide query capability on individual genes or pathway with the intent to accommodate more general query types in the future. This website is free and open to all users and there is no login requirement.

Currently, Omicseq contains 50,484 unique, high quality genome-wide profiling datasets. The vast majority of which are from Human and the rest are from Mouse. The majority of these data is collected from large international consortia including: 36,694 datasets from TCGA, 3,935 from ENCODE and 2,331 from Roadmap Epigenome, 2,079 from Cancer Cell Line Encyclopedia (CCLE) [61], 661 from ICGC, 660 from GEUVADIS [62]. We included diverse data types including ChIP-seq, RNA-seq, DNase-seq, copy number variation, methylation, GRO-seq, microarray gene expression, DNA motifs and summary-level data. More data types will be added later. Currently, for query terms we allow a single gene name or a known pathway name. There are 32,745 Human gene names (from RefSeq version hg19) in our database. Gene queries are flexible as we accept gene names as well as known aliases. For example OCT4, an alias for POU5F1 will be recognized. There are 10,023 pathways (from mSigDB) stored in Omicseq that may be queried. Users can select them from a list arranged in alphabetical order or enter text corresponding to the pathway of interest. Partially typed gene or pathway names will be auto-completed for the convenience of the user. We also begin to provide gene search capability on mouse data. There are 30,493 Mouse gene names (from RefSeq version mm9) in our database. More mouse data and the pathway search capability will be added in the next release.

After a user submits a search, a result page will be displayed. The result page lists top 50 most relevant datasets based on the corresponding trackRank algorithm. For each dataset, corresponding metadata is also displayed, including cell, factor/experimental condition, and another "MetaData" link to a popup window to display more comprehensive metadata about the dataset. Users will also be able to find corresponding publications by clicking "PubMed" button. A link named "GEO" will point to the corresponding GEO page if available. For the convenience of the users, we also supply dozens of external links about the query gene such as PubMed, Wikipedia, WikiGenes [63]. At the bottom of the result page, users can click on the "D" button to view a pie chart that show the breakdown of data types among the top ranked tracks (in which the query gene has a percentile that is less than 1%).

There will also be an option / popup button that allows users to specify advanced search options. For example, the search can be constrained to a specific experiment type, such as ChIP-seq, RNA-seq or DNase-seq. It will also be possible to constrain searches to a specific data sources such as the ENCODE [39], TCGA [45] or Roadmap Epigenomics [64], or combination of multiple ones. The default is to search all data sources. Querying selected cell lines, such as MCF7, LNCaP is also supported.

We believe the Omicseq system is fairly intuitive to use. Nevertheless, we have created a tutorial document with detailed instructions. This tutorial page (http://www.omicseq.org/tutorial.htm) can be easily accessed from any page within the Omicseq web portal. In addition, we have created a video tutorial which can be accessed at https://youtu.be/tfmjh6ADVu0.

Our recent testing indicates that a typical gene-based search takes only 0.1 seconds with a nine node MongoDB setup. In general, loading/ingesting data into our database takes significant time though by implementing a parallel loading strategy, we are able to achieve a loading metric of 7.3 seconds per dataset. Our performance study also demonstrates that MongoDB based indexing is highly scalable on query performance and loading performance. Given that data loading is an incremental one-time cost, we expect data ingestion to be prompt.

System Architecture

The Omicseq system is based upon the SpringMVC architecture using Spring 4.0 and Servelet 3.0 techniques. It runs as a cluster of databases and web servers on Amazon Web Services Elastic Computing instances, which can easily scaled in response to demand. The database server cluster is built on a sharded TokuMX database with the supporting web servers using Apache Tomcat for the user interface application. As shown in Figure 2.2, the overall system consists of seven major subsystem components including: 1) the webcrawler system (under construction); 2) the source data download system; 3) the data processing system; 4) the cache system; 5) the database system; 6) The task-scheduling system; and 7) the web application service system. They work collaboratively to collect and process data from multiple data sources, and provide query services to users.

Considering the large size of data, two mechanisms were adopted to optimize the overall system performance. First, we implemented a task scheduling system initiated at the severs' startup which manages the assignment and reclamation of computing resources to different tasks such as

webpage crawling, data downloading and processing tasks, allowing them to be performed in batch. Second, a memcached system has been implemented on a separate server to alleviate database workload and boost system performance and improve query speed. With memcached, frequently queried data are preloaded into the memory at the server's starting up. In addition, intermediate data generated between data processing steps are also cached for later use. So that when a request for such data is issued, either by users or internal system components, the memcached system is first consulted to avoid redundant database lookups. If the data is not found then the query would be routed to the database. This approach significantly reduces the load on the database system and improves query speed.

Omicseq Web Server

The Web portal is based on the well-performing Apache Server with tight integration of the Tomcat Application Server. For purposes of scalability, high availability, and security the infrastructure has been deployed on Amazon Web Services Elastic Computing environment. The Web portal application has two layers: application logic layer and presentation layer. The application logic layer provides the mapping of queries into various backend database operations or computation operations, caching management of repeated queries, and search job management of multiple query requests. The application logic layer also provides post-processing of data, for example, the generation of images from returned data to be visualized. We use JFreeChart library (http://www.jfree.org/jfreechart/) to produce images. Many common operations in the application layer are implemented as RESTful WebAPIs to facilitate easy integration with various applications. REST is a software architecture style for distributed hypermedia systems such as the Web. REST is lightweight in representation, and can be used to build applications easily. In RESTful Web

Services, data are viewed as resources, which can be identified by their URIs. Normally implemented on the HTTP, RESTful Web Services are very efficient for transporting data over the Web. We define a set of common queries as RESTful Web Services to allow flexible integration of the search engine into users' applications.

The presentation layer provides the actual interfaces for interactive searching and visualization of query results. The front page provides a search box where a user can type in a keyword and select the corresponding query type to start a search. Auto-completion is implemented using an Ajax-based request based on the JQuery framework. The Ajax request is supported by a RESTful Web API which generates possible completed keywords based on a collection of keywords related to common names or IDs such as gene names stored in the indexing database.

The online search portal is built on top of a NoSQL databases (a MongoDB is used to manage index data for gene search). The web portal provides modeling, managing and searching of results from the databases on top of an Apache Web Server integrated with Tomcat Application Server.

Database

The database system consists of a cluster of five nodes and can be functionally divided into two components. The first component is a single, unsharded database built on MongoDB 2.4.9. That stores the extracted metadata of the crawled raw data. For example, metadata information including sample ID, source type, data-processing state are stored in a collection indexed on sample ID. The second component is a distributed sharded database built on TokuMX 2.0.1 that extends the MongoDB protocol but with additional performance-enhancing features such as

improved multithread reading and writing, high throughput transaction support, and efficient creation and management of clustering indexes. This component includes three Shard servers where the bulk of data are maintained, a Configuration server which holds metadata about which shards hold what data, and a Service server which acts as a proxy to route client requests to the appropriate shard (Figure 2.2). The shard servers holds 600 million records of gene rank, miRNA rank and etc. In such a large data scenario, the choice of shard key is a critical step in optimizing the system performance. Since almost all client queries are based on gene ID or miRNA ID, these are used as a primary shard key to horizontally split the data of corresponding collection into separate shards. The internal balancer of TokuMX automatically balances the data load among shard servers. In addition, since some queries may be based on sample ID or need sorting according to percentile value, clustering indexes are also built on these fields which significantly improve range queries as well as data migration among shards.

User Cases

As an example, It has long been established that prostate-specific antigen (PSA), encoded by KLK3 gene, is an important biomarker for prostate cancer [65]. When we query the KLK3 gene, it turns out that RNA-seq data on prostate tumor samples from TCGA and ICGC show up on top. Hence users who queries KLK3 will immediately recognize the importance of this gene for prostate cancer. Similarly, a query of the HER2 gene (ERBB2) identified hundreds of RNA-seq data from tumor samples of breast cancer patients in TCGA show up on top. As another example, when query PTEN, the top ranked datasets are dominated by CNV data show strong deletion, which clearly indicates that PTEN is a tumor suppressor. Search results for these three genes can be found in Figure 2.3.

It is well known that many of the genes in the genome received very little attention in the literature. When query all genes in the human genome against PubMed recently, we found about 30% of the genes are not mentioned in any paper, and the median number of publications is only six for all human genes. For example, for non-coding RNA (ncRNA) SLCO4A1-AS1, a PubMed query returns zero record. But an Omicseq search return 278 datasets in which this ncRNA gene is within the top 1% among all genes in the genome. Among the top ranked datasets, there are multiple ChIP-seq datasets of histone marks H3K4me1 and H3K36me3, both show significant enrichment in the promoter region of this gene in cancer cell lines HepG2 and HeLa-S3. There are also 205 CNV datasets, from various tumor samples collected by TCGA included among the top ranked datasets. All these datasets point to potential functional connection of this ncRNA gene with cancer.

2.4 Discussion

Literature is the dominating source of biomedical knowledge today. The explosion of massive genomics data offers an attractive, alternative source for biomedical knowledge since these assays interrogate a large number of genes which provides a somewhat unbiased (no vetting from investigators), comprehensive view of the genome. As a result, genome-wide profiling experiments are supplementing the traditional hypothesis-driven research paradigm with a data-driven paradigm. However, key informatics infrastructure needs to be developed in order to make these datasets a truly useful resource. One of the major aims of the recent BD2K (https://datascience.nih.gov/bd2k) efforts (e.g., bioCADDIE) is focused on making biomedical data searchable and reusable to speed up discovery [66]. Here we describe our attempt to develop such an informatics infrastructure to fulfill this aim. The ultimate overarching goal of Omicseq is

to enhance findability of omics datasets, to facilitate re-utilization, or re-purposing of existing data for secondary, tertiary analyses.

Genome-wide profiling experiments offer great opportunities for “data-centric” knowledge discovery, which we believe represents an innovation that significantly improves upon, and effectively augments, the traditional knowledge discovery approach relied upon in text mining. In this sense, Omicseq provide a novel tool that can facilitate discoveries using existing and emerging genomic datasets.

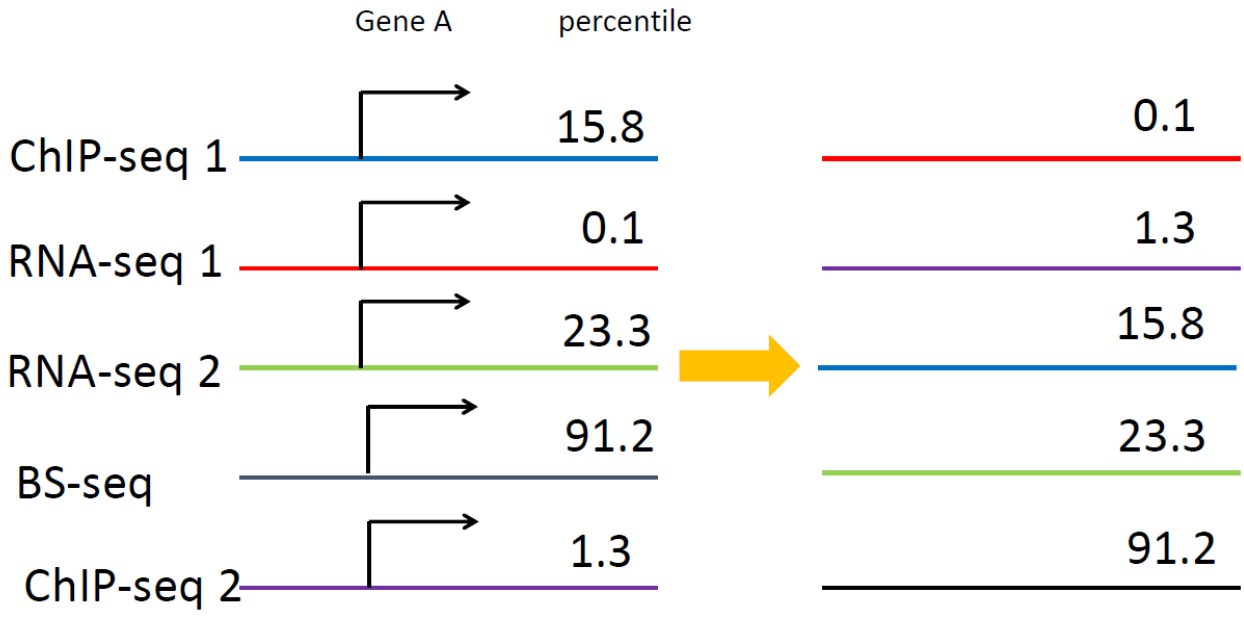


Figure 2.1: Illustration of ranking genomic datasets of different types.

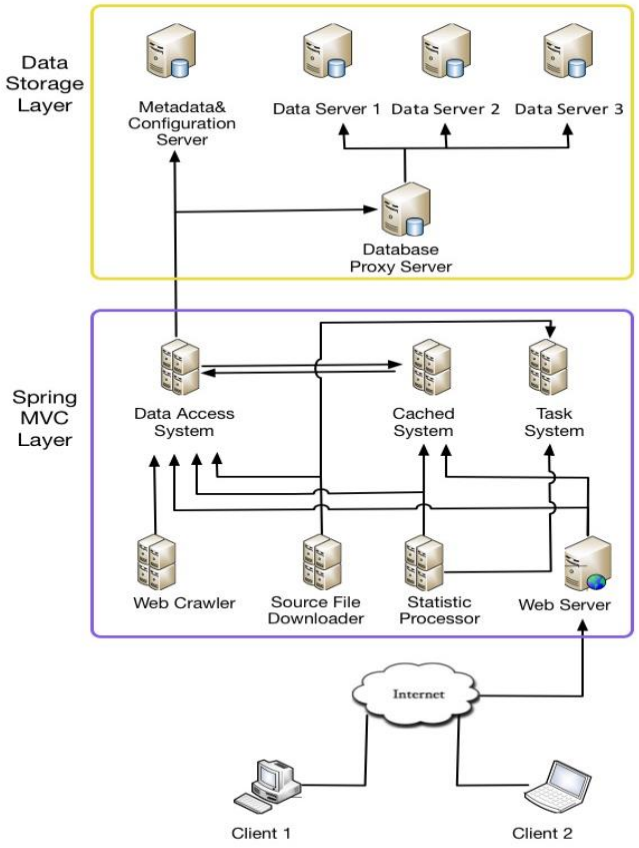


Figure 2.2: The architecture of Omicseq web system. The system consists of the data storage layer (in the yellow square) and the Spring MVC web service layer (in the purple square). In the storage layer, processed omics data are saved in sharded TokuMX database with three shard nodes and one access proxy node. Metadata are saved on another mongoDB database server. In the MVC layer, on one hand, the web crawler, source file downloader and statistical processor work together to load data into the database. On the other hand, front-end web server delegates users' query to the corresponding service which in turn retrieves relevant data from cached system or database system. The web server uses the retrieved data to generate the query results and displays them on the result page.

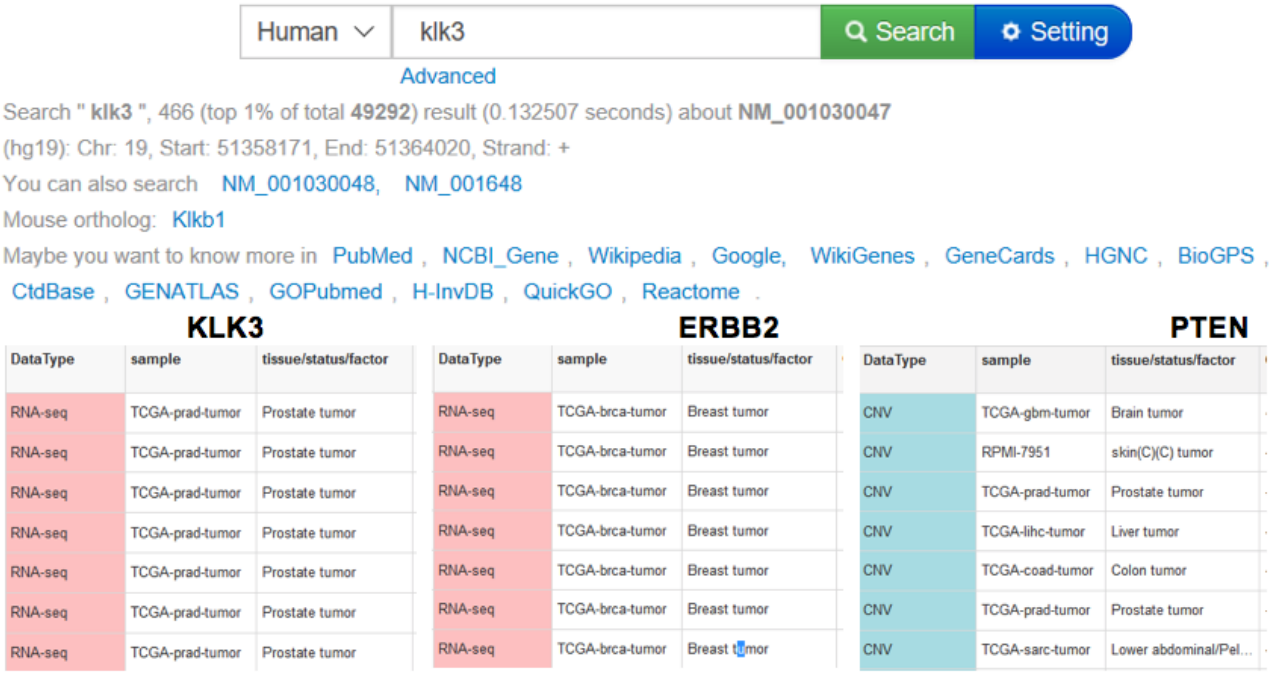


Figure 2.3: Search interface and result pages for KLK3, ERBB2 and PTEN. Most relevant datasets are displayed with links to metadata and paper, etc.

Chapter 3

Optimized Distributed Systems Achieve Significant Performance Improvement on Sorted Merging of Massive VCF Files

3.1 Introduction

Studies like Genome Wide Association Studies (GWASs), Whole Genome Sequencing (WGS) and whole exome sequencing (WES) studies have produced a massive amount of data. The ability to efficiently manage and process such massive data becomes increasingly important in a successful large scale genetics study [1, 26, 67]. Single machine based methods are inefficient on processing such big data due to the prohibitive computation time, I/O bottleneck, and CPU and memory limitations. Traditional HPC techniques based on MPI/OpenMP also suffer from limitations such as not allowing addition of computing nodes at runtime, shortage of a fault-tolerant and high available file system, inflexibility of customizing the computing environment without administrator permission of the cluster [1, 68]. It becomes increasingly attractive for investigators to take advantage of more powerful distributed computing resources or the cloud to

perform data processing and analyses [1, 69]. Apache Foundation has been a leading force in this endeavor and has developed multiple platforms and systems including Hadoop [70, 71], HBase [72] and Spark [73]. All these three Apache platforms have gained an increasing popularity in recent years, and have been endorsed and supported by major vendors such as Amazon Web Services (AWS).

One particular problem that could be possibly efficiently addressed by using a distributed system is sorted merging, which is a ubiquitous operation in processing genetics and genomics data. As an example, in WGS, variants identified from individuals are often called and stored in separate VCF files. Eventually these VCF files may need to be merged (into a VCF or TPED file) as required by downstream analysis tools such as PLINK [74] and BlueSNP [75, 76]. Either a VCF or TPED file requires data to be sorted by genomic locations, thus these tasks are equivalent to the well-known sorted full-outer-joining problem [77, 78]. Currently, they are handled by software such as VCFTools [79] and PLINK, which become very cumbersome even in the face of a moderate scale of genomic data. The main reason is that these tools adopt the multiway-merge-like method [80] with a priority queue as the underlying data structure to ensure the output order. Although such a method only requires one round of read through of the input files, a key deficiency is that it can only have one consumer to access items from the data queue, which literally makes it sequential on writing. This problem cannot be eliminated even if the multiway-merging is implemented as parallel processes due to I/O saturation, workload imbalance among computing units, and memory limitation. Therefore, these single-machine based tools are inefficient and time-consuming when handling large datasets.

In this study, we use the case of sorted-merging multiple VCF files to demonstrate the benefits of using Apache distributed platforms. However, simply running sorted merging on such distributed systems runs into problems of bottlenecks, hotspots and unordered results commonly seen in parallel computations. Rather, we believe working schemas custom designed for each specific distributed platform are required to unleash their full potentials. To overcome the limitations of single-machine, traditional parallel/distributed, and simple Apache distributed system based methods, we propose and implement three schemas running on Hadoop, Spark and HBase respectively. We choose these three platforms because they are representative cloud distributed systems providing data partitioning based parallelism with distributed storage, data partitioning based parallelism with in-memory based processing, and high dimensional table like distributed storage, respectively. Hadoop [70] is the open source implementation of MapReduce [71] based parallel key-value processing technique, and has the advantage of transparency and simplicity. HBase [72] is a data warehousing platform which adopts Google's BigTable data storing structure [81] to achieve high efficiency in storing and reading/writing large scale of sparse data. Spark [73] introduces the concept of Resilient Distributed Dataset (RDD) and Directed Acyclic Graph (DAG) execution to parallel key-value processing, thus enabling fast, robust and repetitive in-memory data manipulations. Specifically, our schemas involve dividing the job into multiple phases corresponding to tasks of loading, mapping, filtering, sampling, partitioning, shuffling, merging and outputting. Within each phase, data and tasks are evenly distributed across the cluster, enabling processing large scale of data in a parallel and scalable manner, which in turn improves both speed and scalability.

3.2 Methods

Overview

The benefits of using the three Apache distributed platforms to perform sorted merging are four-fold when compared to using the multiway-merge method [80], a relational database based approach, or a HPC framework. First, with genomic locations as keys and genotypes as values, it is readily transformed into the key-value model in which all three platforms offer a rich set of parallel operations. Second, data in VCF files are semi-structured. This type of data ideally fit for all three platforms which allow defining the schema during data loading, avoiding the preprocessing of raw data into a rigid schema as in a relational database. Third, all these platforms provide built-in efficient task coordination, high fault tolerance, data availability and locality which are absent in the traditional HPC framework. Fourth, the merged results are directly saved onto a distributed file system such as HDFS or Amazon S3 which can be directly used for subsequent cluster-based GWAS or WGS analytical tools such as BlueSNP.

Despite these advantages, simply performing sorted merging on these Apache distributed systems will not deliver expected results for the following reasons. First, it can lead to globally unsorted results. Hash-based shuffling of input data is the default mechanism for distributing data to parallel working units in the system. However, shuffling will lead to globally unsorted results. Second, bottlenecks and hotspots can happen during the processing in the cluster. Bypassing the hashing based shuffling can lead to unbalanced workloads across the cluster, result in straggling computing units which become the bottlenecks for response time. In addition, for parallel loading of presorted data into HBase, data being loaded from all the loading tasks access the same node simultaneously while other nodes are idling, leading to an I/O hotspot. Third, sampling costs could become

prohibitive. Although Hadoop provides a built-in utility named *total-order-merging* [77] to achieve both workload balance and global order, it involves transferring to and sampling all the data on a single node. The communication costs over the network and disk I/O can be prohibitive when data size is very large. In the following sections, we will illustrate how our custom designed schemas are able to overcome these limitations in detail.

Data Formats and Operations

In a typical WGS experiment, data analysis often starts from individual genotype files in VCF format [82]. A VCF file contains data arranged into a table consisting of eight mandatory fields including chromosome (CHROM), the genomic coordinate of the start of the variant (POS), the reference allele (REF), a comma separated list of alternate alleles (ALT), among others. In our experiments, we use a dataset consisting of the VCF files of 186 individuals [9] generated from Illumina's BaseSpace software (Left tables in Figure 3.1). Each VCF file has around 4-5 million rows, each row contains information on one of the individual's genomic variants. Each VCF file is about 300 megabyte in size. In an attempt to protect the privacy of the study subjects, we apply the following strategy to conceal their real genetic variant information contained in the VCF files: we first transform each original genomic location by multiplying it with an undisclosed constant real number, taking the floor integer of the result, and then add another undisclosed constant integer number.

It is common that multiple VCF files need to be merged into a single TPED file for analysis tools such as PLINK. A TPED file resembles a big table, aggregating genotypes of all individuals under investigation by genomic locations (Right table in Figure 3.1). The merging follows several rules.

First, each record is associated with a data quality value in the "filter" column, which records the status of this genomic position passing all filters. Usually only qualified records with a "PASS" filter value are kept. Second, genotypes in VCF files are stored in the form of allele values, where 0 stands for the reference allele, 1 stands for the first mutant allele, 2 stands for the second mutant allele, and so on. Allele values must be translated into corresponding types of nucleotides in the TPED file. Third, all individuals need to have a genotype for genomic locations that appear in at least one VCF file. The default genotype for a missing value is a pair of homozygous reference alleles. The merging of multiple VCF files to a single VCF file follows the rules as: First, "ALT" and "INFO" columns of a genomic location in the merged file are set as the concatenated values of the corresponding columns on that location from all input files with duplicated values removed. Second, the "QUAL" column of a genomic location in the merged file is set as a weight-averaged quality value of all individuals on that location. Third, a genomic location is kept only when it appears in at least one input file and has a "FILTER" value that equals to "PASS". Fourth, if an individual does not have allele values on a genomic location in the input file, their missing allele values are designated as "." in the merged file.

For our Apache cluster-based schemas, the merging of VCF files into a TPED file and the merging of VCF files into a single VCF file differ only in the value contents of the key-value pairs, so they should have the same scalability property. Although we implement the applications of both merging types using our Apache cluster-based schemas, which are available on our project website, we focus our experiments on the merging of multiple VCF files to a TPED file and only evaluate the execution speed of the merging of multiple VCF files to a single VCF file with VCFTools as the benchmark.

MapReduce (Hadoop) Schema

This schema is built on Hadoop's underlying model MapReduce and running on Hadoop clusters. MapReduce [71] is a parallel computing model based on a *split-apply-combine* strategy for data analysis, in which data are mapped to key-values for splitting (mapping), shuffling and combining (reducing) for final results. We use Apache Hadoop-2.7 as the system for our implementation. Our optimized schema consists of two MapReduce phases, as shown in Figure 3.2 (the pseudocodes are shown in Figure 3.11).

1) First MapReduce phase.

Raw data are loaded from HDFS into parallel mappers to perform the following tasks: First, unqualified data are filtered out and qualified ones are mapped to key-value pairs. The mapper output key is the genomic location and output value is the genotype and individual ID. Second, Key-value pairs are grouped together by chromosomes and temporarily saved as compressed Hadoop sequence files [83] for faster I/O in the second MapReduce phase. With this grouping, we only need to merge records from selected chromosomes of interest rather than from all of them. Meanwhile, these records are sampled to explore their distribution profile of keys along chromosomes to determine boundaries. The boundaries are determined such that there is an approximately equal number of records within each segment. Because all records falling in the same segment will be assigned to the same reducer in the later phase, boundaries calculated in this way ensure that the workload of each reducer is balanced. There are two rounds of samplings. The first one happens in each mapper with a pre-specified sampling rate, which in our case is set to be 0.0001. Sampled records are then separated and distributed to different reducers in this phase by

chromosomes, where they are sampled again with a rate equal to the reciprocal of the number of input files. This second sampling effectively limits the number of final sampled records even in the face of a very large number of input files. Because the number of reducers instantiated in the second phase is decided by the number of sampled records, we can therefore avoid launching unnecessary reducers thus reducing task overheads.

***2) Second MapReduce phase*.** In this phase, multiple parallel MapReduce jobs are created, and each job specifically handles all records of a single chromosome outputted as sequence files in the first phase. Within each job, a partitioner redirects records to the appropriate reducer by referring to the splitting boundaries from the previous phase, so that records falling in between the same pair of boundaries are aggregated together. Finally, each reducer sorts and merges aggregated records by genomic locations before outputting them to a TPED file. In this way, globally sorted merging can be fulfilled.

HBase Schema

HBase [72] is a column-oriented database where data are grouped into column families and split horizontally into regions spreading across the cluster. With this data storing structure, it supports efficient sequential reading and writing of large-scale data as well as fast random data accessing. Also, HBase is storage efficient because it can remember null values without saving them on disk. These features make HBase an ideal platform for managing large, sparse data with relatively low latency which naturally fits the sorted merging case. We use the HBase-1.3 as the system for our implementation. As shown in Figure 3.3, our optimized HBase schema is divided into three phases as discussed next (refer to Figure 3.12 for pseudocodes).

1) Sampling phase

The main challenge of HBase lies in that it is not uncommon to find that one server of the cluster becomes a computational hotspot. This can happen when it starts loading a table from a single region hosted by a single node. Therefore, we need to presplit the table into regions of approximately equal size before loading. The sampling phase is introduced to determine reasonable presplitting regional boundaries. The total number of regions is set to be half of the number of input files so that the size of each region is approximately 1GB. Meanwhile, mappers of this phase also output qualified records as compressed Hadoop sequence files on HDFS which are used as inputs in the next phase. In addition, filtering and key-value mapping also take place in this phase.

2) Bulk loading phase

Even when the table has been presplit evenly, the hotspot problem of loading sorted inputs is not yet fully solved because sorted records are loaded sequentially and at any instant they still access the same region and server, which necessitates the adding of this phase. During the bulk loading, the key and value of each record produced from the previous phase is converted into HBase's binary row-key and column-value respectively, and saved into a HFile, HBase's native storage format. The row-key here is in the form of chromosome-genomic location, and column-value refers to reference allele, individual ID and genotype. The bulk loading populates each HFile with records falling in the same pair of presplit regional boundaries. Because HFiles are written simultaneously by parallel mappers/reducers, all working nodes are actively involved and the regional hotspot is thus circumvented. Upon finishing writings, the HBase can readily load HFiles

in parallel into the table by simply moving them into local HBase storage folders. This procedure is therefore at least an order of magnitude faster than the normal loading in which data are loaded sequentially via HBase servers' I/O routines. The order of records in the table is guaranteed because they are internally sorted by writing reducers and HBase's Log-Structured Merge-tree [84]. It is noteworthy to mention that VCF records are sparse, thus HBase is very storage-efficient.

3) Exporting phase

A scan of a specified genomic window is performed on the table. It involves launching parallel mappers each receiving records from a single HBase region, filling in missing genotypes, concatenating records with the same row-key, and outputting final results into TPED files.

Spark Schema

Spark [73] is a distributed engine that embraces the ideas of MapReduce and RDD. It can save intermediate results in the form of RDD in memory, and perform computations on them. Also, its computations are lazily evaluated, which means the execution plan can be optimized since it tries to include as many computational steps as possible. As a result, it is ideal for iterative computations such as sorted merging. We implement our optimized Spark schema on Spark-2.1. It has three stages described below and shown in Figure 3.4 (refer to Figure 3.13 for pseudocodes).

1) RDD preprocessing stage

This stage involves loading raw data as RDDs, filtering, and mapping RDDs to paired-RDDs with keys (chromosome and genomic position) and values (reference allele, sample ID and genotype). This stage ends with a sorting-by-key action which extends to the next stage.

2) Sorting and merging stage

The sort-by-key shuffling repartitions and sorts PairRDD records so that records with the same key are aggregated together, which are then merged into TPED format and converted back to RDD records for outputting. However, Spark's native family of group-by-key functions for merging should not be used here because their default partitioner is hash-based and different from the range-based partitioner used by previous sort-by-key function. Consequently, the merged results would be reshuffled into an unsorted status. We therefore optimize the merging to bypass these functions and be performed locally without data reshuffling to ensure both order and high speed.

3) Exporting stage

In this stage, merged RDD records are saved as TPED files on HDFS.

Execution parallelism has an important impact on the performance. To maximize performance, the number of parallel tasks is set to the number of input files. In this way, the data locality is maximized and each task is assigned a proper amount of work. In addition, unlike using MapReduce or HBase, when performing sorting by keys, no explicit sampling is needed because Spark keeps track of the number of records before determining repartition boundaries.

Parallel Multiway-Merge and MPI-based High Performance Computing Implementations

Since many bioinformatics researchers have access to a traditional in-house HPC cluster or stand-alone powerful server (with cores $\geq$ 16 and memory $\geq$ 200G) rather than a heterogeneous cloud-based cluster, we also implement a parallel multiway-merge program running on a single machine

and a MPI-based (mpi4py v3.0) "single program, multiple data (SPMD)" program running on a HPC cluster as benchmarks. We choose to implement multiway-merge, because many existing bioinformatics tools, including VCFTools and PLINK, adopt it as the underlying algorithm for sorted merging. Multiway-merge is highly efficient on single machine as it requires only one scan of sorted input files, so it can theoretically run at the speed of disk I/O.

Generally, there are two types of parallelism---data parallelism and task parallelism. The former splits data horizontally into data blocks of roughly equal size (the size of genomic intervals in our case) before assigning them to all available processes; the latter assigns a roughly equal number of input files to each process. For parallel multiway-merge, we choose data parallelism because the implementation of task parallelism would be the same as the HPC-based implementation running on a single node. We will discuss data parallelism in a later portion of this paper. Perhaps the most difficult part of data parallelism is that we do not know the data distribution across all input files, which usually leads to the problem of workload imbalance among processes. If we pre-sample all the input files to estimate the record distribution, then a full scan of the input files is required which will almost certainly take more time than the single-process multiway-merge method. As a compromise, we assume that the distributions of SNP locations in all VCF files are uniform and the input files can be split into regions of approximately equal sizes. The total number of regions are set to be the number of concurrent processes so that each region is specifically handled by a process. To avoid seeking of a process's file reader to its starting offset from the beginning of the file, we take advantage of the Tabix indexer [85], which builds indices on data blocks of the input file and place the reader's pointer directly onto the desired offset. One important aspect of the Tabix indexer is that it needs the input file to be compressed in bgzip format which

is not supported by Hadoop, HBase and Spark. The compression and decompression of a file in bgzip format can be much faster than in bz2 format used in our cluster-based schemas, single multiway-merge and HPC-based implementations, so parallel multiway-merge can run much faster than other methods/schemas when input size is small.

For the HPC-based implementation, we adopt the task parallelism (Figure 3.5) to avoid sampling and workload imbalance. Otherwise the workflow of HPC-based implementation is the same as that of the MapReduce-based schema with the same operations and the same order: sampling in parallel, dividing the dataset into splits of equal size, and assigning splits to processes to do the merging. But this implementation is without data locality offered by HDFS and task coordination offered by YARN and thus has a performance no better than the MapReduce-based schema. Specifically, input files are shared across all nodes in the cluster via a Network File System (NFS). In the first round, each core/process fetches roughly the same number of files from the NFS and performs multiway-merging locally. In the following rounds, we adopt a tree-structured execution strategy. In the second round, processes with even ID numbers (process id starts from 0) retrieve the merged file from its adjacent process to the right, which are then merged with its local merged file. Processes with odd ID number are terminated. In the third round, processes with ID divisible by 4 retrieve the merged file from its adjacent process to the right in the second round to merge with its local merged file. This process continues until all the files are merged into a single file for a total number of rounds of $\log(n)$, where n is the number of input files.

Strong and Weak Scalabilities

In this study, we quantify scalability by measuring computing efficiency in tests of strong and weak scalabilities. We define efficiency as the average time costs of processing a file per core:

Efficiency = $(T_b*C_b/N_b) / (T_i*C_i/N_i)$

where T_b is the baseline running time, C_b is the baseline number of cores, N_b is the baseline number of input files, T_i is the current running time, C_i is the current number of cores, N_i is the current number of input files. We also incorporate the parallel multiway-merge and MPI-based HPC implementations as benchmarks in the tests.

For the strong scalability test, we fix the number of input files at 93 and increase the computing resources up to 16-fold from the baseline. The baseline is a single node (4 cores) for all methods/schemas except for the parallel multiway-merge in which only a single core is used because it can only run on a single machine. For the weak scalability test, we increase both computing resources and input size at the same pace. The ratio is ten file/core for parallel multiway-merge and ten file/node for all others.

3.3 Results

We conduct experiments of Apache cluster-based schemas using Amazon's Elastic MapReduce (EMR) service and experiments of the HPC-based implementation using MIT's StarCluster™ toolkit which launches an AWS openMP virtual private cluster (VPC). Within both infrastructures, we choose EC2 working nodes of m3.xlarge type, which has four High Frequency Intel Xeon E5-2670 v2 (Ivy Bridge) Processors and 15GB memory. We conduct experiments of parallel multiway-merge on a single EC2 r4.8xlarge instance with 32 High Frequency Intel Xeon E5-2686

v4 (Broadwell) processors and 244 GB memory. We use a dataset consisting of 186 VCF files [9] generated from Illumina's BaseSpace software.

Overall Performance Analysis of Clustered-based Schemas

Our primary goal is to explore the scalabilities of the three schemas on input data size and available computing resources, namely CPUs. To achieve this, in this experiment we adjust the number of input files from 10 to 186, with an approximate total uncompressed size from 2.5 G to 40 G, and use a varying number of working nodes from 3 to 18, namely 12 to 72 cores.

As Figure 3.6 shows, for all three schemas, given a fixed number of cores, the execution time increases at a slower pace than that of the input size. On the one hand, the increase of execution time is more obvious with fewer cores because each core is fully utilized. As the number of input files increases, so does the number of parallel tasks assigned to each core. For example, given 12 cores, as the number of input files increases from 10 to 186 (18.6 fold), the execution time increases from 739 to 4,366 seconds (~5.9 fold) for the MapReduce schema, from 375 to 5,479 seconds (~14.6 fold) for the HBase schema, and from 361 to 1,699 seconds (~4.7 fold) for the Spark schema. On the other hand, with relatively more cores such as 72, this linear increasing trend is less pronounced because there are more cores than tasks so that all cores are assigned at most one task. We also notice that when input size is small or moderate, the Spark schema does not always show a consistent improvement in terms of execution time when using more cores. This is reflected, for example, in the intersection of curves occurred between 24 and 72 cores in Figure 3.6c. This phenomenon is attributed to the limitation of Spark’s internal task assignment policy which gives rise to the possibility that some nodes are assigned more than one tasks while others remain idle.

Strong and Weak Scalabilities of Apache Cluster-based Schemas and Traditional Parallel Methods

Figure 3.7 shows the results of the strong scalability. In accordance with the Amdahl's law [86], all schemas/methods show degraded efficiency with increasing computing nodes/cores. Parallel multiway-merge has the steepest degradation because the more parallel processes, the higher likelihood of workload imbalances among them. In addition, disk I/O reaches saturation as more processes write simultaneously. Furthermore, to achieve data parallelism and optimize execution speed, we use Tabix indexer to index data blocks of input files. While reading, each process needs to maintain a full copy of file descriptors, indices and uncompressed current data blocks of all input files in memory. When both the number of processes and input files are large, great pressure is placed on the memory management. For instance a test with 93 files and 16 processes requires over 100GB memory, which results in a very long memory swap and garbage collection (GC) time. In contrast, the MapReduce-based schema has the best efficiency. Surprisingly, its efficiency even improves when the number of cores doubles from the baseline. This is because it has many parallel tasks in its second MapReduce phase, and when the core allowance is low, the overheads of repetitive task launching and terminating on a single core become non-negligible. Consequently, as the number of cores starts to increase, the actual proportion of overheads in the total running time decreases, leading to an improved efficiency. Nonetheless, as the number of cores further increases, the unparalleled parts of the schema gradually dominate the total running time, leading to a reduced efficiency.

For the weak scalability test (Figure 3.8), all methods/schemas follow Gustafson's law [87] to have a much better efficiency than in the strong scalability test. Meanwhile, for the same reasons as the strong scalability, the MapReduce-based schema enjoys the best efficiency while the HPC-based implementation has the worst. This is because, for the HPC-based implementation, as the number of input files increases, the total number of merging rounds also increases, leading to a significantly reduced efficiency. Finally, all three Apache cluster-based schemas demonstrate significantly better weak scalability than the other two traditional parallel methods.

The Anatomic Performances Analysis of Apache Cluster-based Schemas

Another important goal of our study is to identify potential performance bottlenecks, so we evaluate the execution time of each phase/stage of all three schemas. Figure 3.9 shows the trends of the anatomic computing time spent on merging increasing number of VCF files (from 10 to 186) using 48 cores. For the MapReduce schema (Figure 3.9a), its two phases account for a comparable proportion of total time and both show a linear or sublinear scalability. The reason the time cost of the first phase between 40 and 93 input files remains flat is because both runs use two rounds of mappers. As the number of files doubles to 186, four rounds of mappers are required which results in about a 2-fold increase in the time cost as expected. For the three phases of the HBase schema (Figure 3.9b), they generally scale well with the input data size. Meanwhile, the second phase becomes more dominant with more input files owing to the larger amount of shuffled data during the writing of HFiles. However, we do not consider it as a bottleneck since all tasks of this phase are parallelized with no workload or computational hotspot. We do not observe an obvious super-linear (relative to input data size) increment pattern from the figure either. Finally, Figure 3.9c shows the time costs of three stages of the Spark schema. They show a uniform increasing

trend with the number of input files. Among them, the second one takes up a considerable proportion of the total execution time as it has a relatively expensive sort-by-key shuffling operation. Although no data is shuffled in the first stage, its time lapse is close to that of the second stage. This is because at the end of the first stage, data are sampled for determining the boundaries used by sort-by-key's range partitioner. This operation demands a considerable execution time because it scans all the data and balances them if necessary.

Given that no super-linear increasing trend is observed in running time for all phases/stages of the three schemas and they generally scale well on the input data size, we reach the following conclusion: Although the performances of these schemas might degrade to some extent when dealing with even larger data size due to overheads such as data transmission over network, we do not expect to see any significant bottleneck.

Execution Speed Comparisons Among Traditional Methods and Apache Cluster-based Schemas

Another intriguing question is: How does the speed of the Apache cluster-based schemas compare to single machine based and traditional parallel/distributed methods/applications on merging multiple VCF files into a single VCF or TPED file? To answer this question, we choose the widely-used VCFTools (v4.2) and a single-process multiway-merge implementation as single-process benchmarks and parallel multiway-merge and HPC-based implementations as parallel/distributed benchmarks, which are the same as in the experiments of strong and weak scalabilities above.

In the first experiment, we merge 40 VCF files into one VCF file using VCFTools as the benchmark. As shown in Table 2, VCFTools takes 30,189 seconds while the fastest Apache cluster-based schema, the MapReduce-based, takes only 484 seconds using 72 cores, which is 62-fold faster. In the second experiment (Figure 3.10), we test the time costs of merging of VCF files into a TPED file using single/parallel multiway-merge and HPC-based implementations as benchmarks. The single multiway merger is run on a node with the hardware configuration (4 cores and 15G memory) identical to the nodes on which the Apache cluster-based schemas are run. The parallel multiway merger is run on a node with a maximum of 18 simultaneously running processes. The HPC-based implementation is run on a 18-node cluster with the same hardware configurations as the cluster of Apache cluster-based schemas. Initially, with ten input files, the parallel multiway-merge (~30 seconds) is much faster than all the other methods; it is about 7.3-fold faster than the fastest Apache cluster-based schema (MapReduce, 221 seconds). On the other hand, the slowest method is the single-process multiway merger which takes 620 seconds to finish and is about 2.8-fold slower than the MapReduce-based schema. It is worth mentioning that in this test the parallel multiway-merge is essentially the same as the single-process multiway-merge, and the speed difference (~378 seconds) between them is the result of a different compression format (bz2 vs bgzip) of the input files as explained above. As we gradually increase the number of input files to 186, the difference in speed between the fastest overall method (parallel multiway merger, 602 seconds) and the fastest Apache cluster-based schema (MapReduce, 809 seconds) reduces to about 1.3-fold, while the difference between the slowest overall method (single multiway merger, 13,219 seconds) and the MapReduce-based schema increases to 16.3-fold. In addition, all three Apache schemas significantly outperform the HPC-based implementation. As explained in the strong and

weak scalabilities section, we expect that the larger the input size, the faster the Apache cluster-based schemas run compared to the other traditional methods.

We also compare the time cost among the three schemas (Figure 3.10). It turns out that they have a comparable speed. More specifically, the MapReduce schema performs the best if enough cores are available and the input file size is large; the HBase schema performs the best with moderate input file size; the Spark schema performs the best with large input file size and a limited number of cores. The rationale behind our observation is that, when the number of cores is sufficient, the MapReduce-based schema can make the most use of the available computing resources because it runs a constant 25 parallel jobs (one for each of chromosomes 1-22, X Y and M (Mitochondria)) in its second phase. In contrast, the Spark-based schema has fewer tasks whose number equals to the number of input files in order to achieve maximum data-task locality. When the input data size is moderate, the HBase-schema triumphs due to its internal sorting and relative compact storage format of intermediate data. When the input data size is large and computing resource is relatively limited, the Spark-based schema outperforms the other two owing to its least number of data shuffling (only one), execution plan optimization, and ability to cache intermediate results in memory. We caution that the computing time may fluctuate depending on the distribution of genomic locations in the input files as well as the data loading balance of the HDFS.

3.4 Discussion

In this report, we describe three cluster-based schemas running on the Apache Hadoop (MapReduce), HBase and Spark platforms respectively for performing sorted merging of variants

identified from WGS. We manage to show that all three schemas are scalable on both input data size and computing resources, suggesting that large scale sequencing of variant data can be merged efficiently given computing resources that are readily available in the cloud. Furthermore, the three schemas show better strong and weak scalabilities than traditional single machine-based parallel multiway-merge and cluster-based HPC methods owing to the absence of I/O bottleneck, better workload balance among nodes, less pressure on memory, as well as data locality and efficient task coordination mechanisms provided by HDFS and YARN. Finally, we show that even with a moderate-sized cluster and input data, all three schemas significantly outperform the broadly-used, single-machine based VCFTools, single-process multiway-merge and HPC-based implementations. Although initially the parallel multiway-merge implementation is much faster than the Apache schemas owing to its advantage of local I/O and light compression of input files, its poor scalability reduces this difference as the number of concurrent processes and input files increases. Consequently, we expect that the Apache cluster-based schemas eventually outperform the parallel multiway-merge when merging a much larger scale of data using a larger number of cores.

Unlike normal merging, efficient sorted merging of many large tables has always been a difficult problem in the field of data management. Multiway-merge is the most efficient single-machine based method for sorted merging, but its performance is limited by the disk I/O [88]. Sorted merging also places challenges to distributed system based solutions because neither the efficient hash-based merging nor caching the intermediate table in shared memory is feasible [89]. Although a utility named *total-order-joining* is provided by the Hadoop for addressing this problem, it suffers from both network communication and local disk I/O bottlenecks, thus is not

scalable [77, 90]. In contrast, our schemas divide this problem into different phases/stages of tasks each conquered in parallel to bypass these bottlenecks and achieve maximum parallelism. Furthermore, in addition to merging sequencing variant data, the schemas can be generalized for other key-based, sorted merging problems that are frequently encountered in genetics and genomics data processing. As an example, they can be slightly modified to merge multiple BED format files such as ChIP-seq peak lists [91] and other genomic regions of interest. Other potentially useful features include: 1) Unlike traditional sorted merging algorithms which usually require presorted inputs for a better performance, our schemas are free of such a requirement; 2) Our implementations automatically take care of multi-allelic positions which are frequent in large scale VCF flies by retaining the information of all alleles until the merging actually occurs.

Finally, in light of different features and specialties of the three platforms, each of the three schemas we developed has its own advantages and disadvantages in different application scenarios as summarized in Table 3.1. For example, the MapReduce schema is good for a static one-time, non-incremental merging on large-size data provided sufficient cores are available since it has the most parallel jobs, the least overheads, and the most transparent workflow. The HBase schema, supported by data warehousing technologies, fits for an incremental merging since it does not need to re-merge existing results with new ones from the scratch only if the incremental merging is performed on the same chromosomes. Also, it provides a highly-efficient storage and On-Line Analytical Processing (OLAP) on merged results. The Spark schema is ideal for merging large scale of data with relatively limited computing resources because it has the least data shuffling and keeps intermediate results in memory. A bonus brought by Spark is that subsequent statistical

analyses can be carried out directly on the merged results using its rich set of parallel statistical utilities.

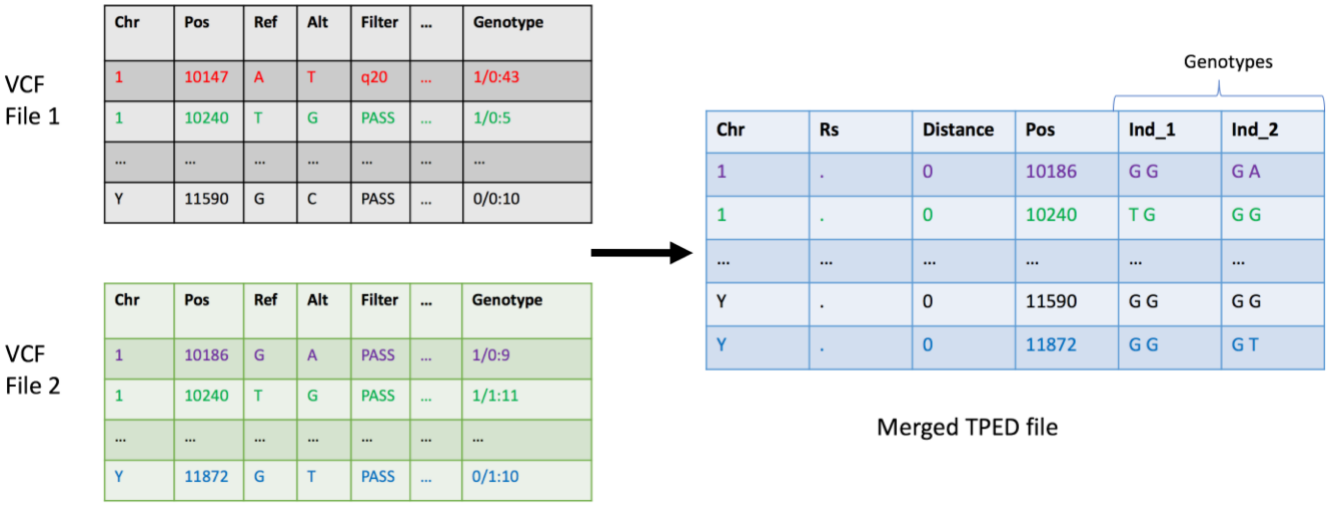


Figure 3.1: Converting VCF files to TPED. Left tables are input VCF files. Right table is the merged TPED file. Records are filtered out if their Filter value does not equal to 'PASS' (Pos 10147). Individual genotypes with the same genomic location that exist in any VCF file are aggregated together on one row. The resulting TPED file thus has an inclusive set of sorted genomic locations from all VCF files.

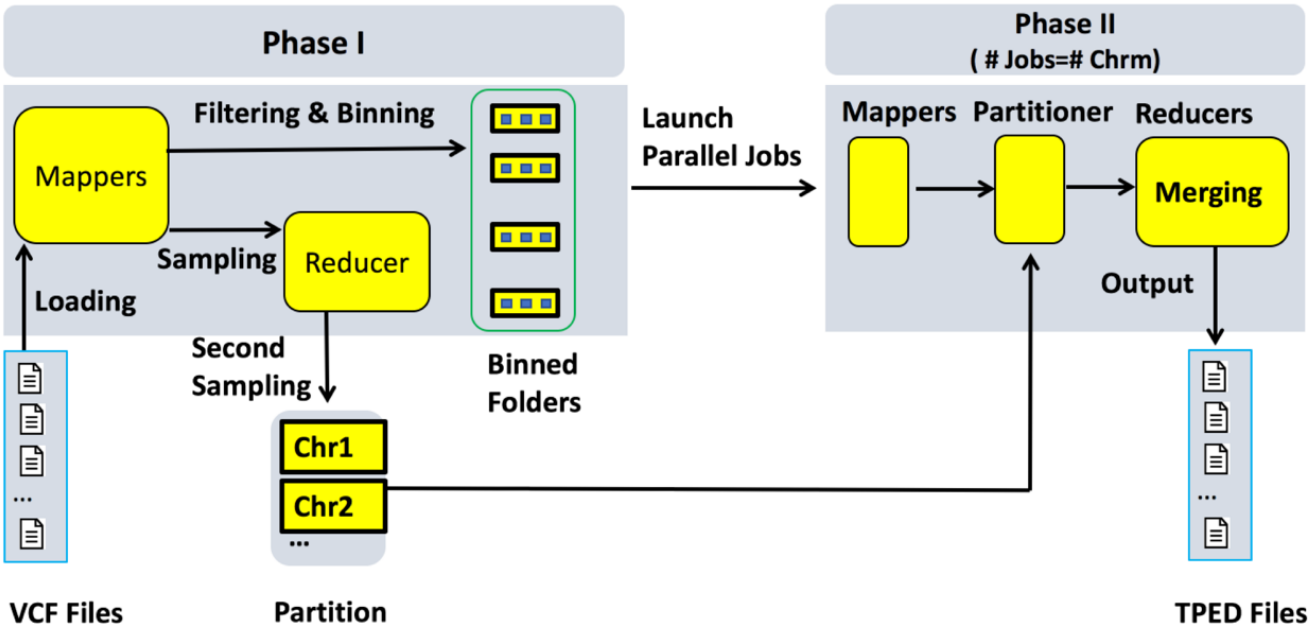


Figure 3.2: The workflow chart of MapReduce schema. It consists of two phases: In the first phase, input VCF records are filtered, grouped by chromosomes into bins, and mapped into key-value records. Two samplings are performed to generate partition lists of chromosomes. In the second phase, parallel jobs of specified chromosomes are launched. Within each job, records from corresponding bins are loaded, partitioned, sorted and merged by genomic locations before being outputted as TPED files.

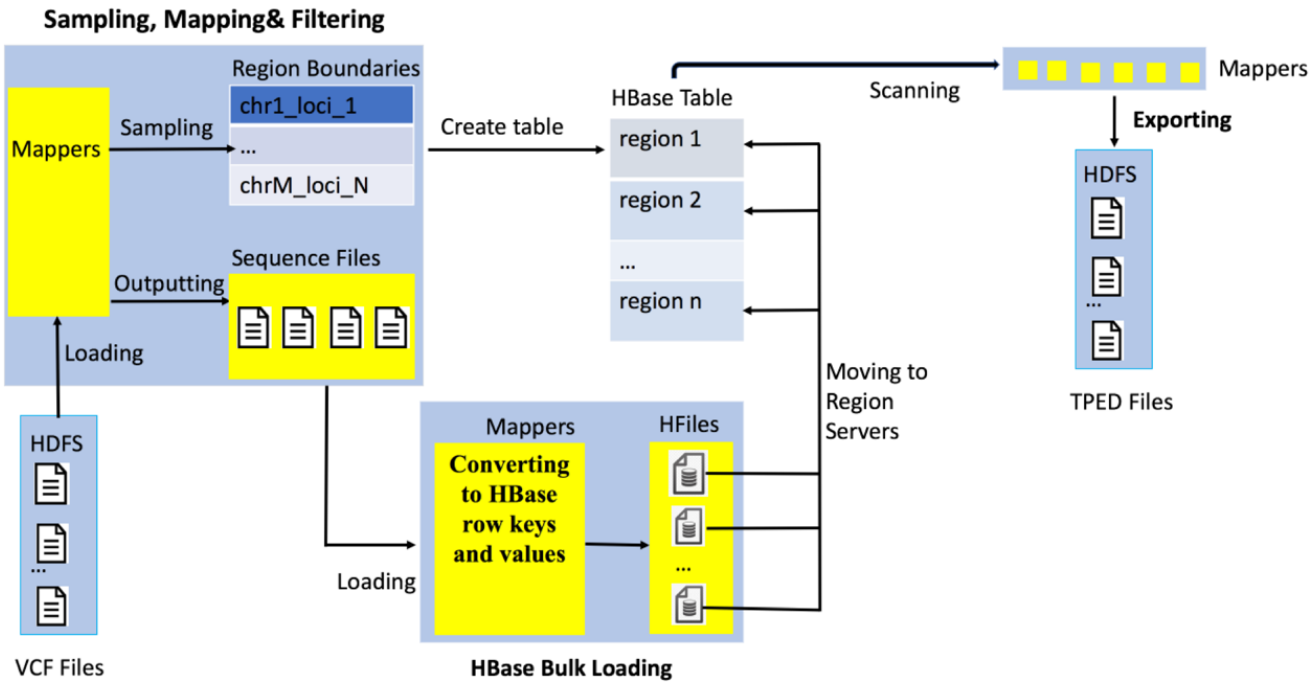


Figure 3.3: The workflow chart of HBase schema. The workflow is divided into three phases. The first one is a sampling, filtering and mapping phase. A MapReduce job samples out VCF records whose genomic positions are used as region boundaries when creating the HBase table. Only qualified records are mapped as key-values and saved as Hadoop sequence files. The second phase is HBase bulk loading in which a MapReduce job loads and writes records outputted from the previous phase, aggregating them into corresponding regional HFiles in the form of HBase's row key and column families. Finished HFiles are moved into HBase data storage folders on region servers. In the third phase, we launch parallel scans over regions of the whole table to retrieve desired records which are subsequently merged and exported as TPED files.

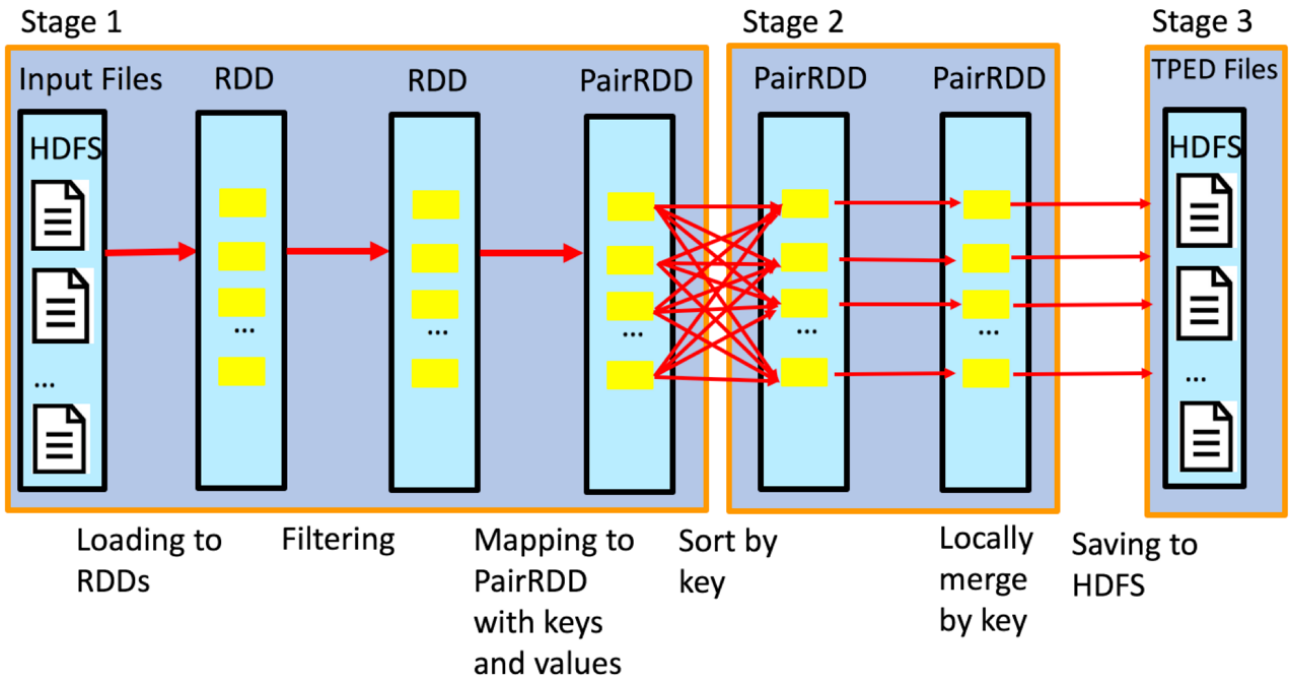


Figure 3.4: The workflow chart of Spark schema. It is a single Spark job consisting of three stages. In the first stage, VCF records are loaded, filtered, and mapped to PairRDDs with keys of genomic position and values of genotype. The sort-by-key shuffling spans across the first two stages, sorting and grouping together records by keys. Then grouped records with the same key are locally merged into one record in TPED format. Finally, merged records are exported as TPED files.

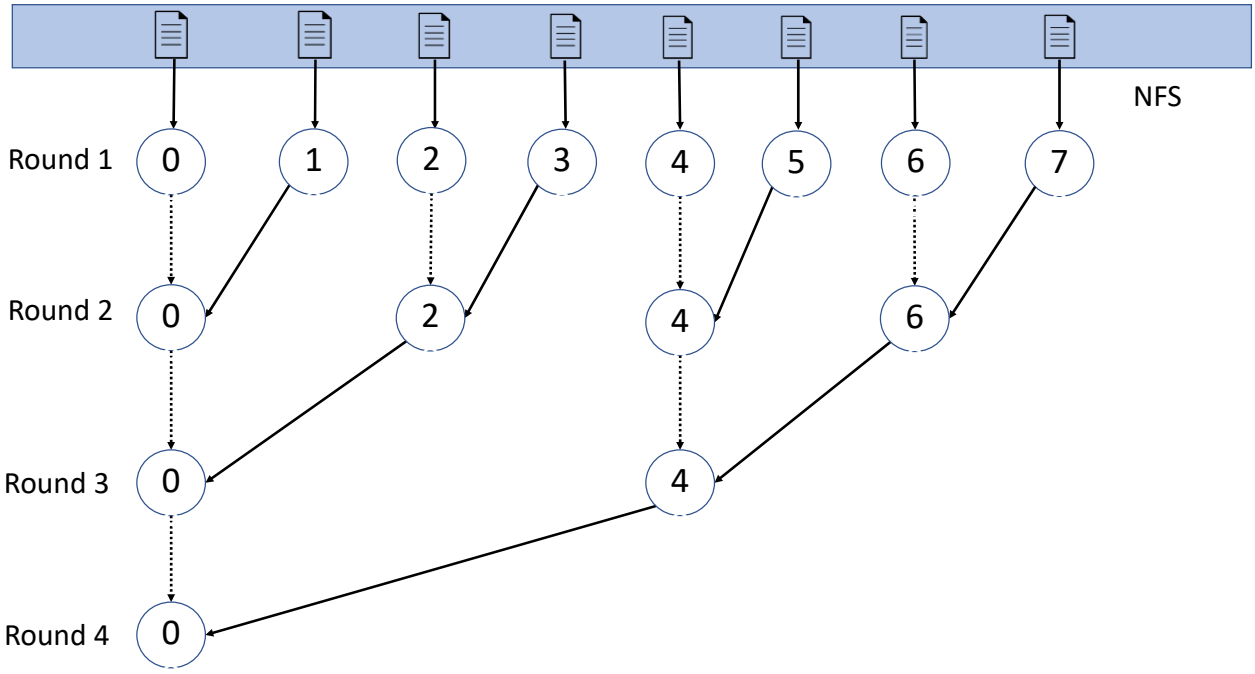


Figure 3.5: The execution plan of HPC-based implementation. The execution plan resembles a tree with branches. In the first round, each process is assigned an approximately equal number of files to merge locally. In round 2, even-numbered process retrieves the merged file of its right adjacent process to merge with its local merged file. In round 3, processes whose ID can be fully divided by 4 retrieves merged file of its right adjacent process in the second round and do the merging. This process continues recursively until all files are merged into a single file (round 4).

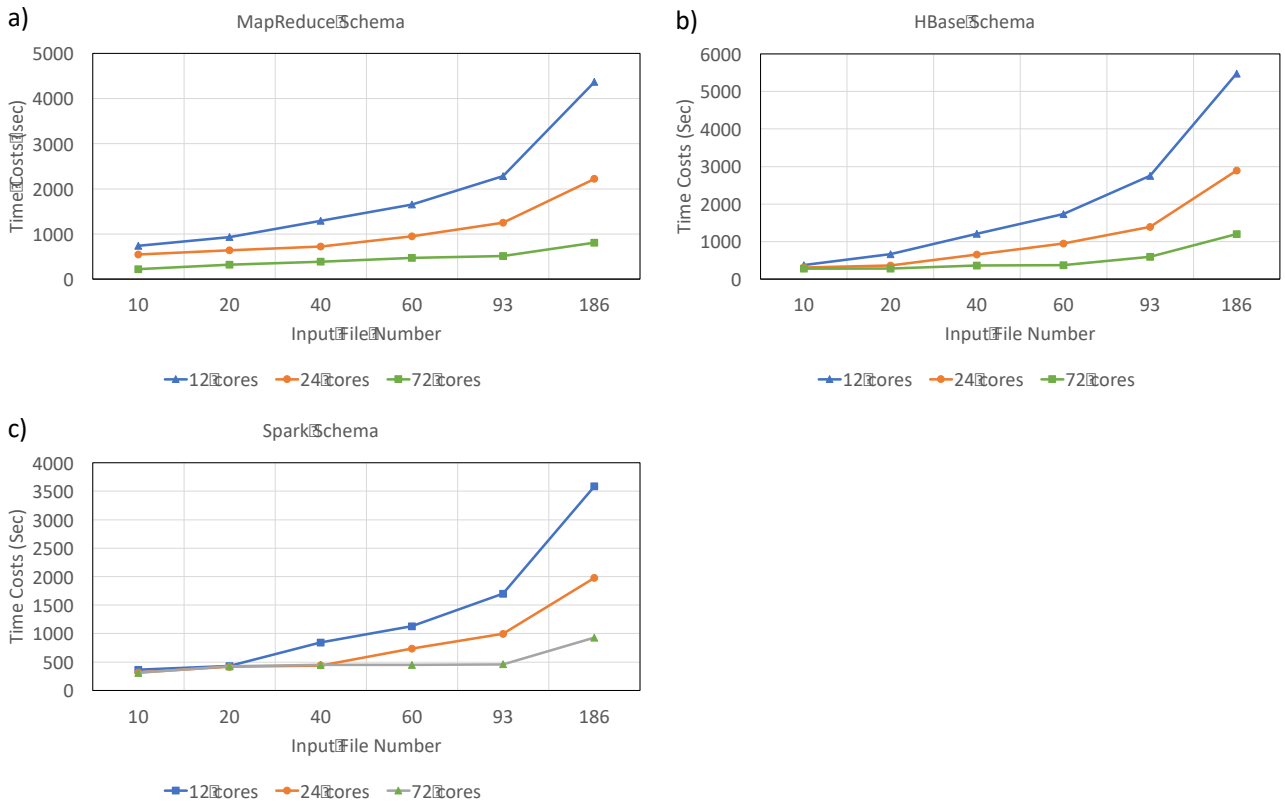


Figure 3.6: The scalability of Apache cluster-based schemas on input data size. A. MapReduce schema. B. HBase schema. C. Spark schema. As the number of input files increases from 10 to 186, the time costs of all three schemas with 12, 24 or 72 cores increase in a slower pace than that of the input data size, especially when the number of cores is relatively large. The HBase schema with 12 cores has the largest increase (from 375 to 5,479 seconds, ~14.6 fold).

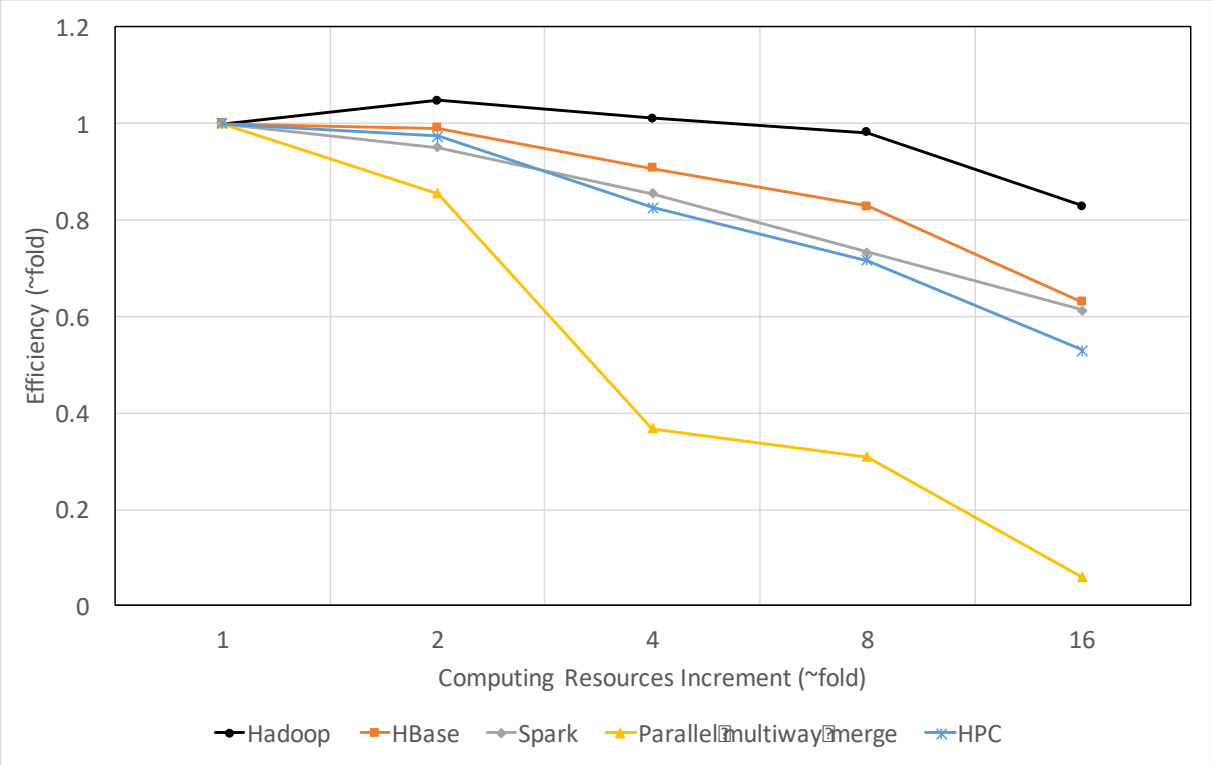


Figure 3.7: Comparing the strong scalability between traditional parallel/distributed methods and Apache cluster-based schemas. We fix the number of files at 93 and increase the number of nodes/cores. The baseline for the parallel multiway-merge is one single core, while for the others is one single node (4 cores). All methods/schemas show a degraded efficiency as computing resources increase 16 fold from the baseline. Specifically, the efficiency of MapReduce-, HBase-, Spark-based schemas drops to 0.83, 0.63 and 0.61 respectively, while the efficiency of parallel multiway-merge and HPC-based implementations drops to 0.06 and 0.53 respectively.

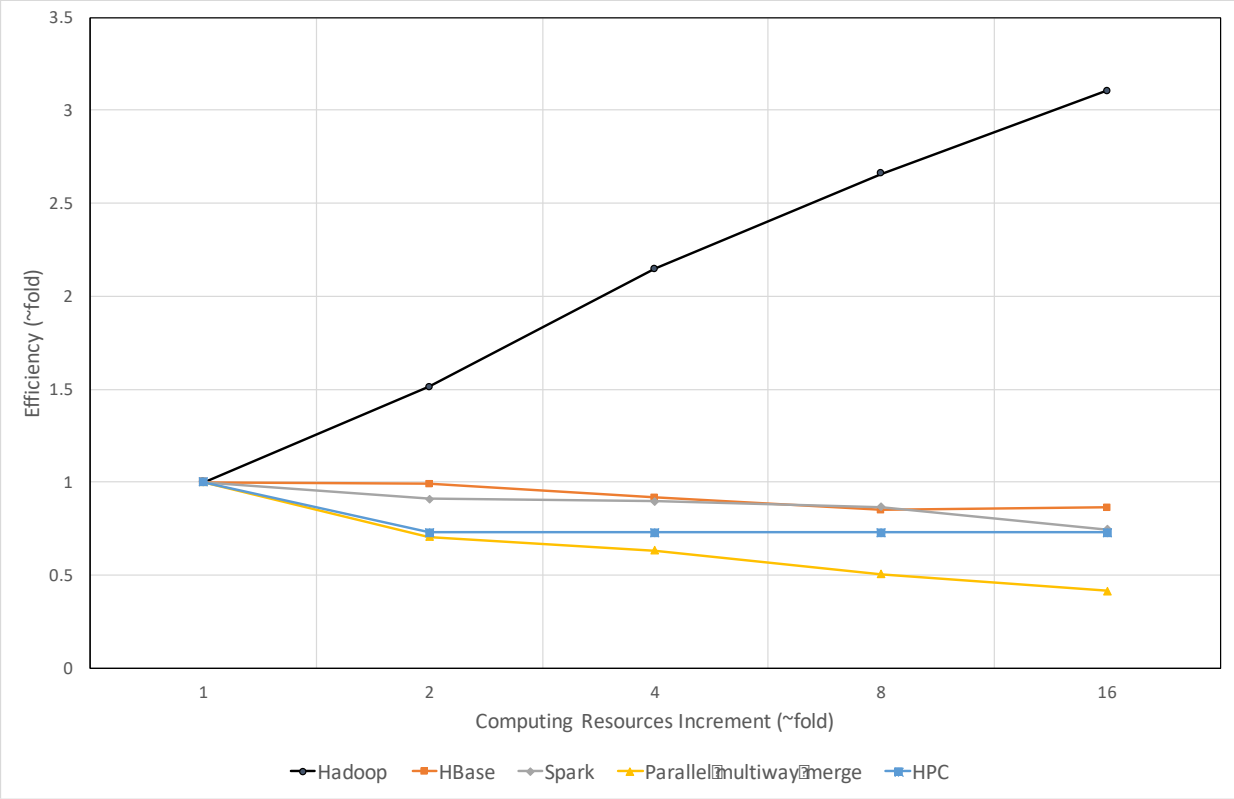


Figure 3.8: Comparing the weak scalability between traditional parallel/distributed methods and Apache cluster-based schemas. We simultaneously increase the number of cores and input data sizes while fixing the ratio of file/core (parallel multiway-merge) or file/node (all others) at ten. The baseline is the same as in the test of strong scalability. All but the MapReduce-based schema have degraded efficiency, among which the HPC-based implementation has the steepest degradation. Specifically, when computing resource increases 16 fold from the baseline, the efficiency of MapReduce-, HBase- and Spark-based schemas changes to 3.1, 0.87 and 0.75 respectively, and for parallel multiway-merge and HPC-based implementations, the efficiency reduces to 0.42 and 0.35 respectively.

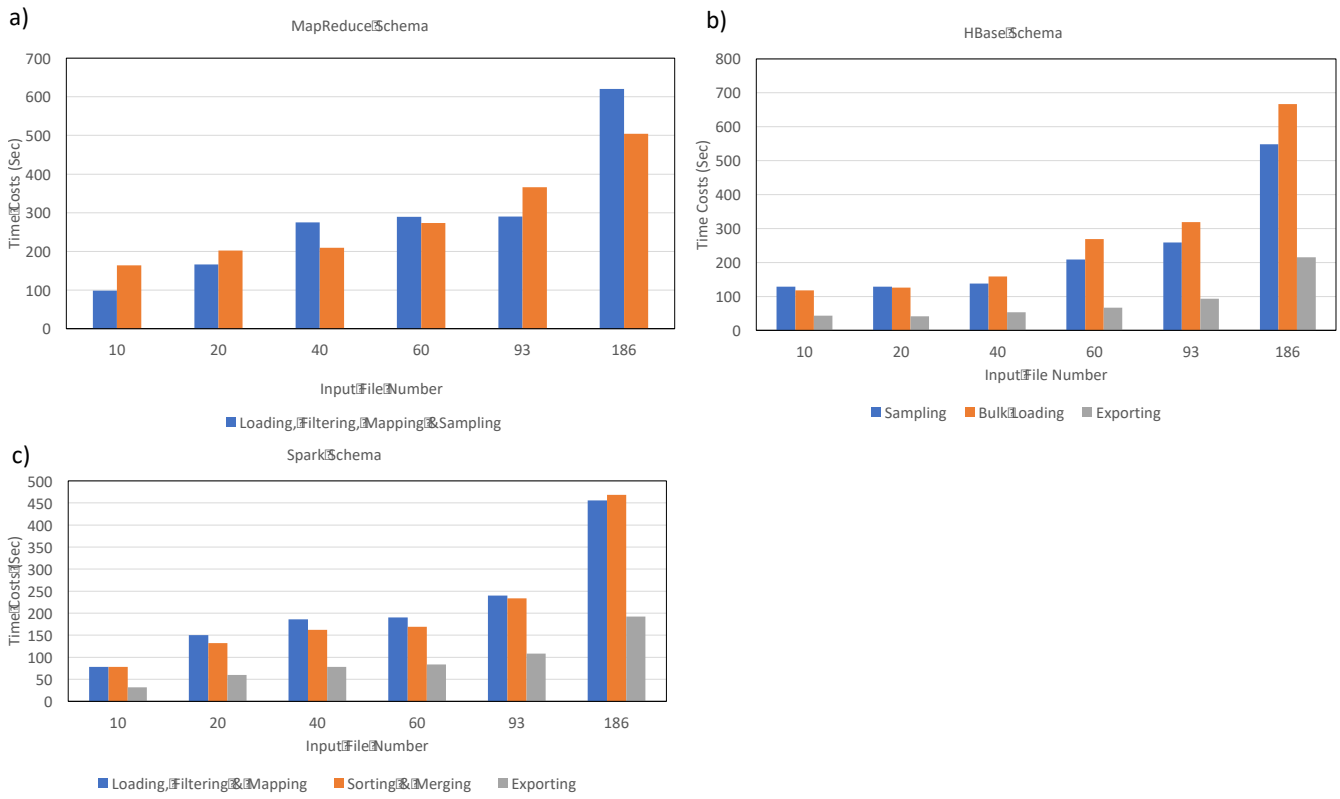


Figure 3.9: The performance anatomy of cluster-based schemas on increasing input data size. The number of cores in these experiments is fixed at 48. Time costs of all phases of the three schemas have a linear or sub-linear correlation with the input data size. **a)** MapReduce schema: The two MapReduce phases have a comparable time cost, increasing 6.3- and 3.1-fold respectively as the number of input files increases from 10 to 186. **b)** HBase schema: The time spent in each phase increases 4.2-, 5.6- and 5.0-fold respectively as the number of input files increases from 10 to 186. The bulk loading and exporting phases together take up more than 80% of total time expense. **c)** Spark schema: The time cost increases 5.8-, 6.0- and 6.0-fold respectively for the three stages as the number of input files increases from 10 to 186 files. Like the HBase schema, the first two stages of the Spark schema together account for more than 80% of the total time cost.

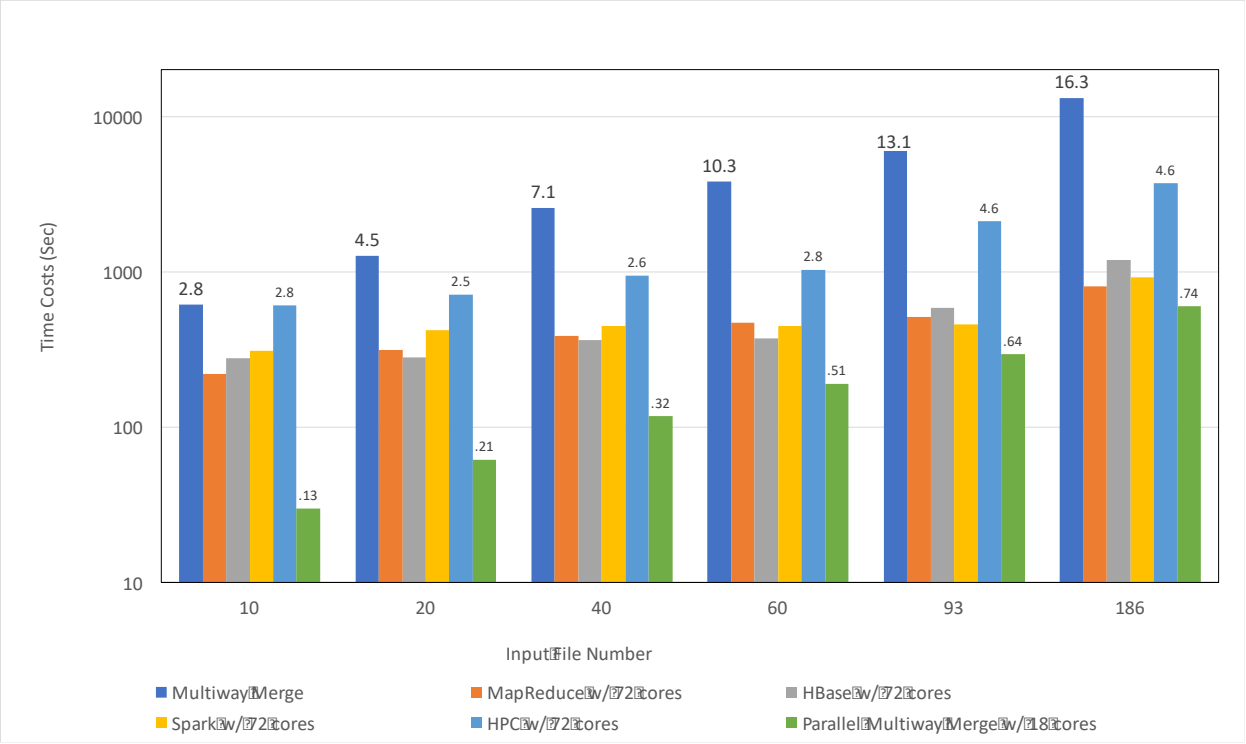


Figure 3.10: Execution speed comparison among Apache cluster-based schemas and traditional methods. Firstly, we compare of the speeds of the three Apache schemas with that of three traditional methods which are single-process multiway-merge, parallel multiway-merge and HPC-based implementations. As the number of input files increases from 10 to 186, the speeds of Apache cluster-based schemas improve much more significantly than traditional methods. The numbers in the figures indicate the ratio of the time cost of each traditional method to that of the fastest Apache cluster-based schema. Secondly, we compare the processing speed among the three Apache cluster-based schemas which are comparable to each other regardless of the input data size. The MapReduce schema performs the best in merging 10 and 186 files; The HBase schema performs the best in merging 20, 40 and 60 files; The Spark schema performs the best in merging 93 files.

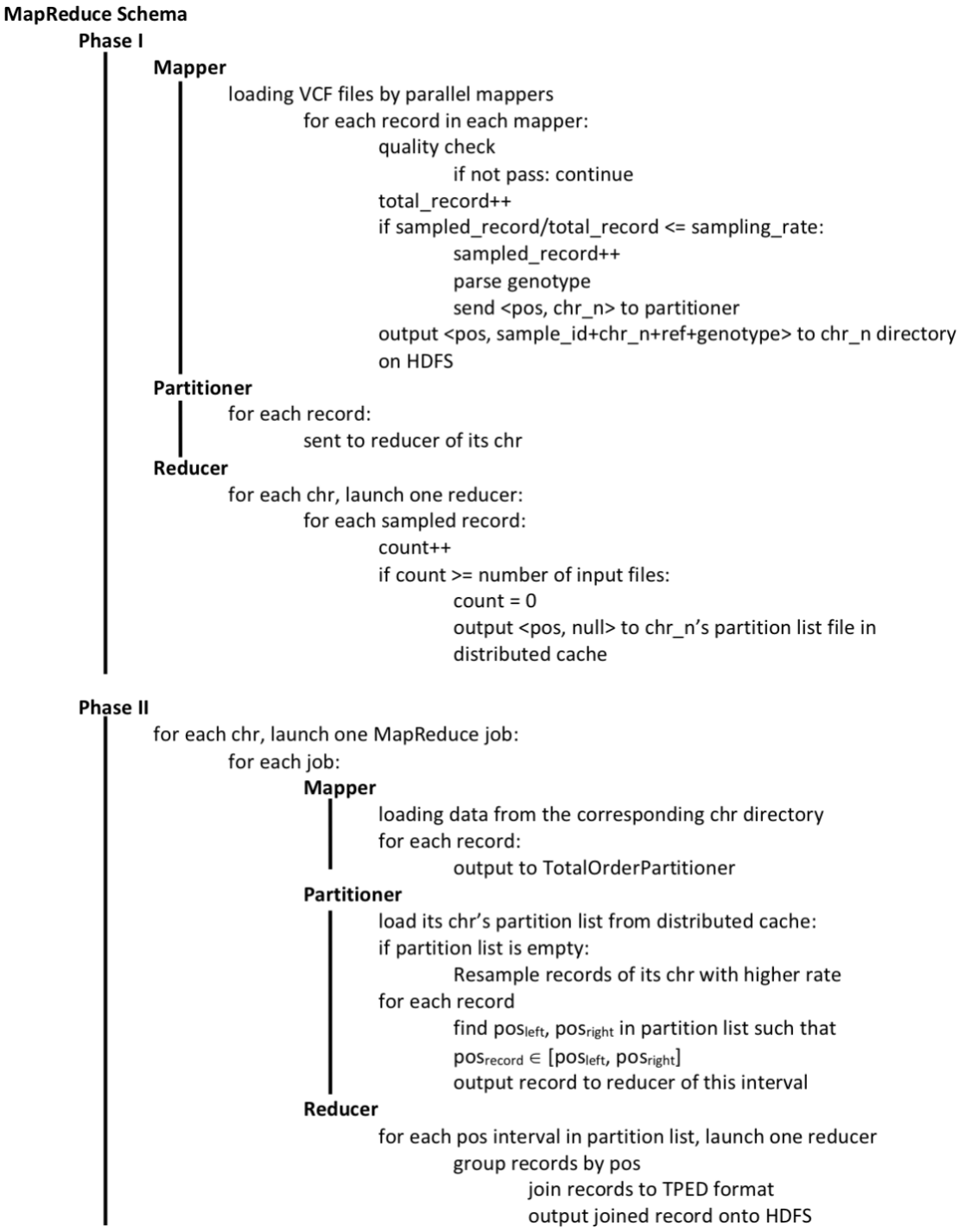


Figure 3.11: The MapReduce Schema

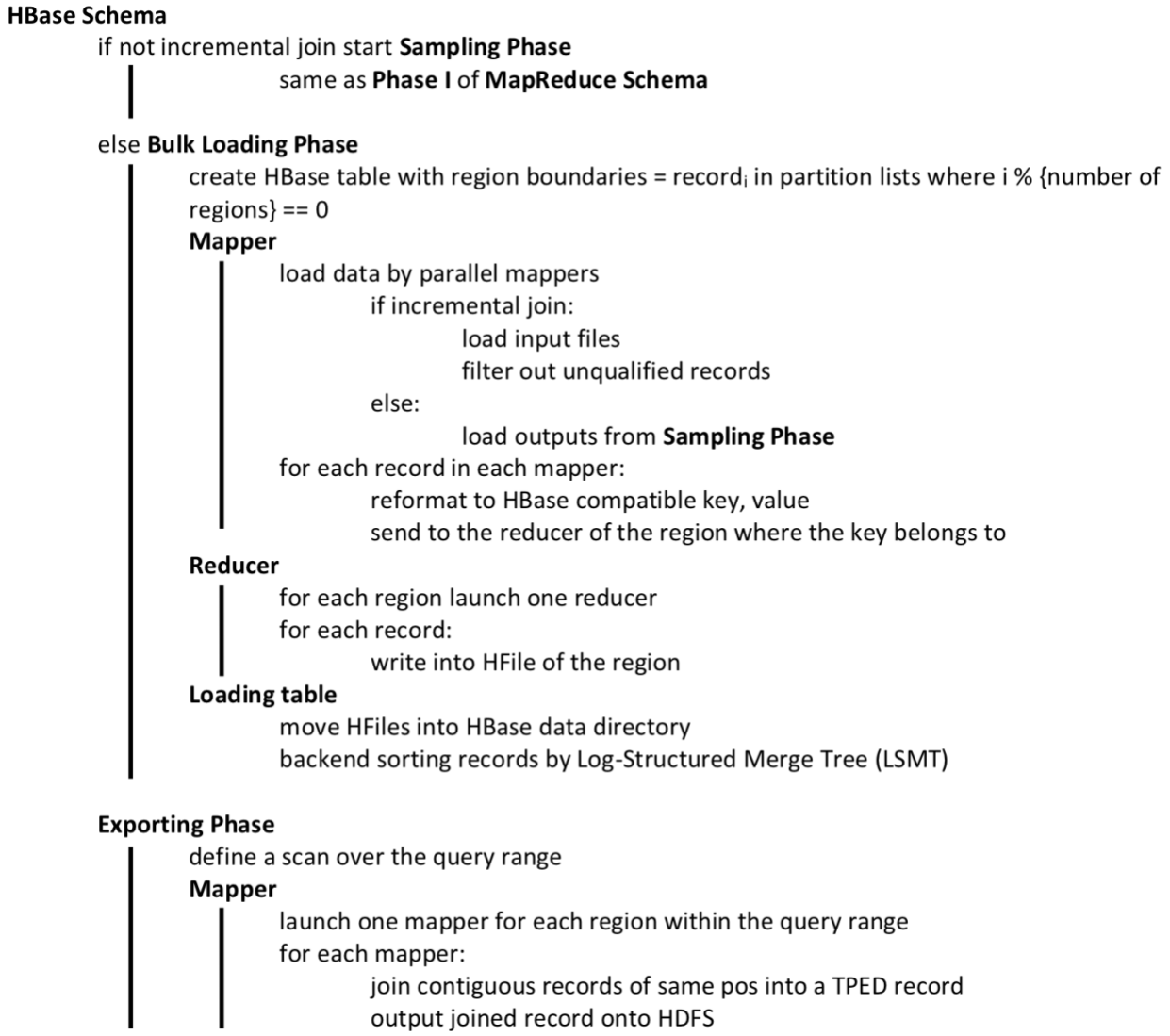


Figure 3.12: The HBase Schema

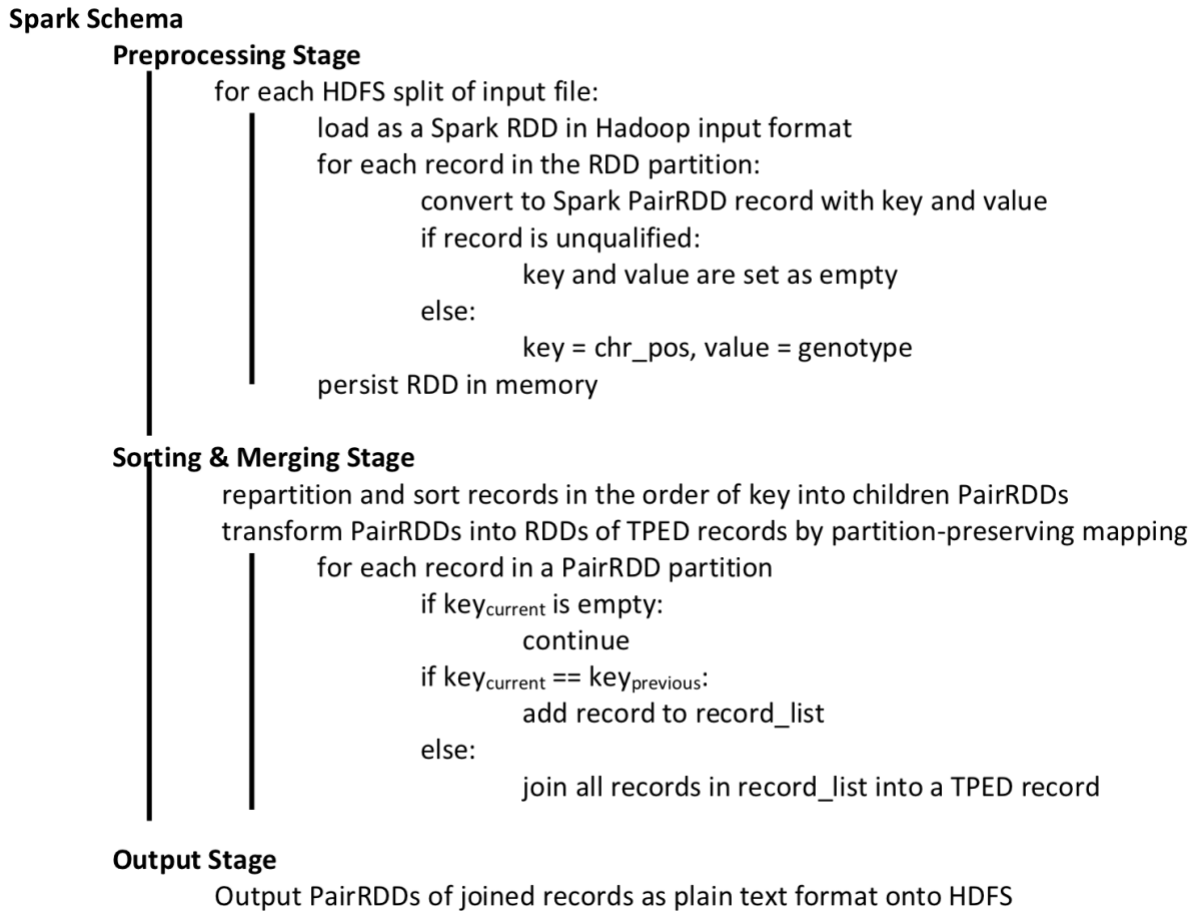


Figure 3.13: The Spark Schema

Table 3.1. Performance comparisons of VCTools versus MapReduce-, HBase- and Spark-based schemas

	VCFTools	MapReduce	HBase	Spark
Time cost (seconds)	30,189	484	577	596
Fold (faster)	-	62.4	52.3	50.7

Table 3.2. Pros and Cons of MapReduce, HBase and Spark schemas

Schemas	***Pros***	***Cons***
MapReduce	• Good for large input size and sufficient computing resources. • Simple architecture and least overheads given sufficient computing resources. • Best parallelism • Good for one-time merging. • Performance is stable.	• Merging is not incremental. • Much overheads when computing resources are limited
HBase	• Good for intermediate input size (>=20 and <=100). • Supports incremental merging. • Supports On-Line	• Users must determine region number in advance. • Has most local I/O. • Complex performance tuning.

	Analytical Processing (OLAP). • Best storage efficiency.	
Spark	• Good for large input size (>100) and relative limited computing resources. • Keeps intermediate results in memory and least local I/O. • Good for subsequent statistical analysis on merged results.	• Possibly weakened data locality during loading. • Slight unstable performance when computing resources exceeds needs of input size. • Actual execution plan is not transparent. • Complex performance tuning.

Chapter 4

Ensemble Learning of Changes of CpG Methylation Levels

4.1 Introduction

Decades of epidemiological research has shown that environmental exposures to pollutants and toxicants during fetal development can lead to both short term (e.g. reduced fetal growth) and long-term (e.g. impaired neurodevelopment) consequences on the child's health [92, 93]. But the mechanism by which the environmental factors exerts their impacts is insufficiently understood. Epigenetic mechanisms, in particular, DNA methylation variation, have been shown to hold substantial potential for improving our understanding of the underlying biological mechanisms [94].

DNA methylation is a covalent modification, heritable by somatic cells through division. 5-Methyl-cytosine represents 2–5% of all cytosines in mammalian genomes and is found primarily on CpG dinucleotides [95]. In recent studies, promising findings have been made linking DNA methylation to molecular response to environmental exposures [96-101]. For example, long-term exposure to PM_{10} has been linked to methylation in both Alu and LINE-1 [102]. The non-uniform distribution of CpG sites across the human genome and the important role of methylation in

cellular processes imply that characterizing genome-wide DNA methylation patterns is necessary for a better understanding of the regulatory mechanisms of this epigenetic phenomenon [103]. Epigenetic alteration has been recognized as playing an important role in the pathogenesis of many human diseases. In particular, difference in DNA methylation has been observed between cases and healthy controls in relevant tissues [104, 105].

So far, hundreds of EWASs have been conducted. Due to the high cost of the whole genome bisulfite sequencing (WGBS) assay [95, 106], all large scale EWASs have been conducted using array-based technologies [107-109]. The most frequently used is Illumina's Infinium HumanMethylation 450K BeadChip, measuring methylation level at about 482,000 CpG sites genome-wide [107]. The next generation EPIC array doubles this number. However, the total number of CpG sites in the human genome is approximately 28 million. That is, the 450K array only covers about 1.5% of all CpG site in the human genome, and are biased towards promoters and strongly underrepresented in distal enhancers [110]. Hence the "epigenome-wide" claim on array-based methylation profiling is rather misleading. The high cost of the WGBS continues to push researchers to use array-based technologies to conduct population-level studies such as EWAS. The low coverage of the whole methylome means lost opportunities to uncover the underlying biological mechanism in response to environmental exposure. This is similar to the problem of incomplete coverage of the genotyping array scientists faced when conducting GWAS. Therefore, methods that improve the scope of methylation profiling beyond the array will have a significant impact, especially for those that can infer/predict DNA methylation status from other precomputed or latent information in the big omics data.

Datasets generated by next-generation sequencing experiments such as ChIP-Seq, DNase-Seq, FAIRE-Seq, RNA-Seq, and ATAC-Seq are stored in large international consortium such as ENCODE and REMC. These datasets can be leveraged to systematically identify functional elements with regulatory activities across hundreds of cell lines and tissues. For example, ChIP-Seq detects protein-DNA interactions and can be used in the identification of transcription factor and other chromatin-associate protein bindings. DNase-Seq measures the sensitivities of genome-wide regions to DNase I for identifying the locations of regulatory regions. FAIRE-Seq and ATAC-Seq are similar to DNase-Seq in that they also identify accessible DNA regions in the genome. RNA-seq measures the level of cellular transcriptome which reveals genomic mutations/SNPs and/or differences in gene expression. Therefore these big omics data offer an opportunity to reveal the latent linkage between epigenomic/genomic variants and regulatory elements, including TF binding, histone modification, and open chromatin. Features selected from a comprehensive collection of these datasets, marks associated with repressed chromatin for example, have been reported to facilitate the accurate and robust recognition of non-coding disease specific risk variants under multiple testing scenarios [111]. Based on these resources, several computational tools have been developed for identifying non-coding risk variations. The Genome-wide Annotation of Variants (GWAVA) generates genome-wide metrics for distinguishing disease-implicated variants from benign variants [112]. The Combined Annotation–Dependent Depletion (CADD) project integrates many diverse annotations into a single C score for each SNP variant [113]. The DANN project overcomes the limitation of linear representation of the data of CADD and use a deep neural network (DNN) based algorithm for identifying genetic variants responsible for diseases [114]. The GenoCanyon project models the non-coding variant with a two-component mixture model (risk or benign) [15]. The Eigen project is similar to

GenoCanyon but adopts a more sophisticated two-component mixture model in the model-fitting process [115].

Although the above experiments and projects generate a large number of genome-wide genomic and epigenomic annotations and scores which provide potentially valuable information to infer DNA methylation levels, these annotations are disease/phenotype neutral, we yet do not know which combination of annotations is more associated with altered DNA methylation level in a specific disease. To address this problem, in this study, we develop an ensemble feature selection method to select the most relevant subset of annotations and scores which can be used to distinguish disease/phenotype-associated CpG sites and the background CpG sites. We also develop an ensemble learning model which is trained with labels converted from summary level data (association p-values) from array-based EWAS and selected annotations and scores as features, and applied to identify CpG sites beyond the array that are likely to show trait-associated changes.

4.2 Methods

Overview

We first collect summary level data (association p-values) from EWASs for each CpG site in the 450K array based on which we label each site as either positive and negative. We treat the learning task as a classification problem that consists of four steps: (1) construct an experimental dataset for each trait, for example, a particular type of environmental exposure; (2) collect a massive set

of external public omics datasets and process them as features; (3) develop a feature selection procedure; (4) develop an ensemble learning method to classify CpG sites into trait-associated or not. Eventually, we utilize the learned model to score the entire genome to identify novel trait-associated CpG sites not on the array for each type of environmental exposure. The workflow of the proposed method is illustrated in Figure 4.1.

Datasets and Prediction Features

We have summary level data from two studies on Alzheimer's disease and placenta with exposure to arsenic respectively. In the first study, the methylation profiles of a total of 415,848 CpGs in a collection of 708 prospectively collected autopsied brain samples are assessed using the Illumina HumanMethylation450 beadset. The correlations of the methylation levels of these CpGs with post-mortem, neuropathologic diagnosis of AD are quantified as P-values. In addition, the direction and amount of altered methylation levels are also calculated. The second study is the Rhode Island Child Health Study (RICHS) cohort, which enrolls mother-infant pairs with nonpathologic pregnancies in Providence, Rhode Island. Placental tissue and cord blood samples from 347 individuals have been collected and profiled using the 450K array. Additionally, data on a set of environmental exposures have also been collected in all these samples through subject biomarkers (toenail or placenta). Among them, arsenic exposure is defined as the amount of Arsenic found in maternal toenail. The P-value in this study refers to the correlation between the methylation level changes of CpGs and the arsenic levels within maternal urine, postpartum toenail samples and placenta.

The features used in this study include epigenomic annotations and functional annotation scores (Table 4.1). So far, we have collected 1806 epigenomic features related to histone modification (1002), TF binding (571), open chromatin (184) and RNA Pol II/III binding (49), spanning 261 cell lines. These features are constructed to cover the entire genome in 200 base-pair resolution (one value in each 200bp bin), representing local epigenomic profiles around every CpG site. The functional annotation scores are collected from multiple computational tools that have been developed to quantify the deleterious potential of a genomic locus. These computational tools include CADD (6) [113], GenoCanyon (4) [15], Eigen/EigenPC (10) [115], and DANN (4) [114]. We also add data from additional important data types including: 1. ATAC-seq (75), a powerful new technology to profile nucleosome landscape genome-wide; 2. Total RNA-seq (47) where transcription of non-mRNA can be better detected; 3. Distance to the nearest transcription start site (TSS).

Experimental Dataset Construction

In this study, there are two major statistics for evaluating the methylation level of a CpG site. The first one is the β value which quantifies single-CpG-site methylation levels. It equals to the proportion of probes for this CpG site that are methylated, which is computed as the methylated probe intensity divided by the sum of both the methylated and unmethylated probe intensities; thus, β ranges from zero (the CpG site is unmethylated) to one (the CpG site is fully methylated) [31]. The second statistic is P value of a CpG site which is the possibility of the null hypothesis that a positive sample has the same methylation level as a negative sample on this CpG site is valid.

For the dataset of Alzheimer's disease, we only have the P values of the top 71 statistically significant CpG sites, that is the 71 sites where normal and diseased people most likely have

different methylation levels. Consequently, we take all the 71 sites as the positives while all others as negative controls. During the construction of the experimental dataset, we included all 71 positive sites, and for each positive site, we sample 20 negatives site with the closest β values. Considering the fact that we are more interested in positive sites and the imbalance ratio of positive to negative sites, we assign a weight of 1 to each negative site and a weight of -log(P-value) ^1.5. As a result, the largest weight ratio between positives and negative controls is 129.6.

For the dataset of placenta with arsenic exposure, we first set two P-value thresholds for defining positive and negative control sites. Any site with a P-value less or equal to the 3e-4 are considered as positive sites (278) while with a P value equal or larger than 0.1 are considered as negative control sites (347,316). The purpose of this step is trying to set a distinguishable boundary between the positives and the negatives. Next, the experimental dataset construction adopts a similar strategy as that of the dataset of Alzheimer's disease except we select 10 instead of 20 negatives for every positive site into the experimental dataset since we have much more positives this time. In addition, we assign each site a weight of -log(P-value) ^1.5, and the maximum weight ratio between positives and negatives is 102.

Feature Selection

Initially we have a comprehensive collection of 1953 genomic/epigenomic annotations and functional annotation scores from multiple sources and computational tools as predicting features in the model. However, all these annotation and scores are disease/trait neutral in that variants associated with all diseases/traits are included. Therefore it is desirable to separate relevant features from others for a particular disease/trait to reduce noise and boost computational

performance. Furthermore, considering the limited size of CpG sites in the human genome, naively including all the features in the training of the model would lead to the "small N large P" problem and thus considerable overfittings. Consequently, a dimension reduction and feature selection step is essential before the model training.

Feature selection algorithms of classification can be divided into three categories – filter models, wrapper models and embedded models. Among the three models, filter models evaluate feature importance without referring to a specific algorithm. They calculate a ranking score for each feature in the feature space based on certain criteria such as information gain and pick up features with highest scores. Although methods of this model type are usually computational efficient, they are independent of the following classification model and thus do not take into account the biases of the classifier, which results in a relatively low performance. Wrapper models take the classifier bias into consideration by repetitively adding/removing features into/from the selection set which is evaluated by the classifier. Nonetheless, the repetitively running of the classification algorithm is computationally expensive especially for a large feature space or complicated algorithms such as DNN. Embedded models have both advantages of filter and wrapper models, the computationally efficiency and the interaction with the classification algorithm, by embedding the feature selection with the classifier construction. Therefore, we choose four embedded methods to build up our ensemble feature selection model, which are random forest, XGBoost [116], L2-regularized logistic regression and L2-regularized linear support vector classifier (SVC) respectively.

Features are normalized before applying each component feature selection method. For L2-regularized logistic regression and linear SVC, features with a coefficient statistically significantly different from 0 are selected. For random forest, each feature is given a score which is a weighted average of impurity reduction on tree nodes that use it. The top 100 features with largest impurity reduction scores are selected. For XGBoost, it measures the number of times a feature appears in all the trees of the model which implies the usefulness of this feature in building the model. The top 100 features with largest numbers are selected. Finally, the ensemble method keeps only features selected by at least two methods while discard the others.

Feature Engineering

In order to further reduce the dimensionality of feature space while still retaining most useful information, we engineer new features in the form of linear or non-linear combinations of the original feature space using sparse autoencoder. Autoencoder is a type of artificial neural network capable of learning efficient representations of the input data, called codings, without any supervision. Codings usually have a much lower dimensionality than the input data, making autoencoder a dimensionality reduction technique. Sparse autoencoder [117, 118] adds an additional term to the cost function which suppress the number of active neurons in the coding layer. As a result, each remaining active neuron typically represents an informative artificial feature. The activity of neurons in the coding layer is quantified as *sparsity*. The sparse autoencoder is trained with a specified target sparsity, which usually is a number between 0 and 1. Typically, using the sigmoid activation function in the coding layer, the codings are also within the range of 0 and 1. The Kullback-Leibler divergency is used to measure the divergence between the mean of the actual codings and target sparsity, and is added as the additional term to the cost

function. Upon finishing the training, the codings from most active neurons are taken as artificial features used in the following machine learning models.

In this study, in order to find the most appropriate hyperparameter setting for the sparse autoencoder, we use Tree-of-Parzen estimator [119] to search the hyperparameter space of the autoencoder which include the target sparsity, sparsity weight, tensor structure, L2-regularization weight, number of training epochs. We also adopts the He-initialization technique [120] for initiating tensor weights, and batch-normalization to address the problem of vanishing gradient.

Data Visualization

In order to check the quality of the data and features, especially the separability of positive and negative CpG sites within the feature space, we leverage the t-Distributed Stochastic Neighbor Embedding (t-SNE) [121] to reduce the dimensionality of features and plot them in 3-D space. t-SNE is a type of manifold learning which reduces dimensionality while trying to keep similar instances close and dissimilar instances apart. It is mostly used for visualization, in particular to visualize clusters of instances in high-dimensional space.

Model Selection and Hyperparameter Tuning

We build the ensemble model by choosing a subset of models from 6 models with model complexity from low to high, in the hope that different types of errors made by each individual model could be offset by other models to achieve a better overall performance. The six models are: 1. Logistic regressor with L2-regularization; 2. Random forest; 3. Support vector classifier (SVC); 4. XGBoost classifier; 5. Multilayer perceptron in Scikit Learn package; 6. Deep neural

network (DNN) in Tensorflow. To select the best subset of models, we individually train and validate each model using stratified 10-fold cross-validation. Considering the imbalance between positive and negative CpGs and the fact that we are more interested in accurately identifying as many positive CpGs as possible, we adopt F-score as our evaluation metric. In addition, we also evaluate the relative complexity of each model to the dataset by plotting learning curves, and find that the logistic regressor underfits the problem while all other models overfit it. Finally, logistic regressor, random forest, XGBoost, SVC and DNN are selected into the ensemble model. We choose logistic regressor because its underfitting property can alleviate the ensemble model from too much overfittings. Multilayer perceptron in Scikit learn is left out because, unlike DNN, it does not adopt techniques such as He initialization and batch normalization in its implementation to address the problem of varnishing gradients, which leads to unconverged solutions. In addition, it cannot set a neuron dropout rate which is an additional regularization term for reducing overfittings. Furthermore, it does not allow sample weights during the training without which all CpG sites tend to be predicted as the dominant negative class.

After the model selection, we use the Tree-of-Parzan method in Hyperopt [119] and 5-fold cross validation to search the hyperparameter space of each component model for the best parameter setting out of 50 adaptively-tuned settings. The evaluation metric used here is F-score. Depending on the number of hyperparameters to be tuned and the size of the search space, the process searching can be very slow especially for DNN with many layers and neurons as well as XGBoost with many trees. In order to speed up the training, we parallelize this training by simultaneously training each component model on multiple cores and dump the best hyperparameters onto the disk. Upon finishing the searching, the ensemble model loads best hyperparameters (Table 4.2) for

corresponding component models and re-train them with the whole training dataset without cross validation to obtain fitted and ready-to-use estimators.

Prediction and Model Evaluation

The final step of the ensemble model is the predictions on the test dataset and evaluations of predicted results. Each component model assigns a probability of being positive to every test CpG site. For example, the logistic regressor utilize the sigmoid function to calculate such a probability; For random forest, it is the ratio of the positive CpGs to total CpGs in the leaf node where a CpG belongs to. For SVC, the probabilities are calibrated using Platt scaling which is a logistic regression on the SVM's scores fitted by an additional cross-validation on the training data [122]. The ensemble model adopts the soft-voting method to obtain the predicted label for each CpG site, namely adding all the class membership probability estimates generated by component classifiers together and assigning the class label with the highest summed-up score.

The predicted results of the ensemble model as well as its component models are evaluated using metrics including F-score, recall, precision, area under curve (AUC) and accuracy, and curves including receiver operating characteristic (ROC) curve and precision-recall curve. The confusion matrix is also plotted to give direct visualization of the qualities of predicted results of each model.

4.3 Results

Selected Features

The features selected by at least two feature selection methods are shown in Table 4.3 in which the column named '*n*' indicates how many methods select the feature on the row. The differences of mean values of the selected features between positive and negative CpGs are calculated and subjected to the t-test with the null hypothesis that there is no difference between the two means. For the dataset of Alzheimer's disease, 37 features are selected among which are those reported to be correlated with the development of AD such as H3K4me1 [123] and H3K27ac [124], and those correlated with brain development such as H3K36me3 in cortex derived neurosphere cultured cells. For the RICHS dataset, 41 features are selected among which we found many features related to placenta and fetal development such as H3K27me3 in fetal heart and placenta smooth chorion. These preliminary findings provide biological supports to our feature selection method.

Predictions on CpG Methylation Level Changes on Alzheimer's Disease

The ensemble model is trained on 90% of the total data and tested on the rest 10%. The test dataset contains 7 positive CpGs and 149 negative control CpGs. The result metrics of the ensemble method and its component methods trained with either selected or engineered features, including AUC, recall, precision, accuracy and F-score, are shown in Table 4.4. The plots of ROC and precision-recall curves of the ensemble method and its component methods are shown in Figure 4.2. When trained with selected features, the ensemble method achieves a AUC of 0.75, a recall of 0.57, a precision of 0.29 and a F-score of 0.38. In other words, the ensemble model predicts 14 positive CpG sites among which 4 are true positive. The best component model is XGBoost which has the best recall-precision balance and the highest F-score, 0.4. The worst component model is DNN which has the lowest F-score, 0.1. In fact, DNN classifies all CpG sites as positive ones because the sample weight ratio of positive to negative sites might be too high for DNN. In addition,

due to the extreme slow training of the DNN model, we have to limit the number of neurons in each layer which potentially jeopardizes the performance of DNN. When trained with engineered features, the performances of all models but the DNN model degrade. One possible explanation is that engineered features in this study are outputs from DNN-like sparse autoencoder, which might introduce the bias-model which favors the DNN model but not others.

Predictions on CpG Methylation Level Changes on Placenta with Arsenic Exposure

Similar to the dataset of Alzheimer's disease, we split the RICHS dataset into training (90%) and test (10%) sets. The test dataset contains 28 positive CpGs and 273 negative control CpGs. The result metrics of the ensemble method and its component methods, including AUC, recall, precision, accuracy and F-score, are shown in Table 4.5. The plots of ROC curve, precision-recall curve and confusion matrices of the ensemble method and its component methods are shown in Figure 4.3. When trained with selected features, the ensemble method achieves an AUC of 0.53, a recall of 0.07, a precision of 0.67 and a F-score of 0.13. In other words, the ensemble model predicts 3 positive CpG sites among which 2 are true positive. The best component model is random forest classifier which has the best recall-precision balance and the highest F-score, 0.25. Notably, it achieves a precision of 1.0 and a recall of 0.14, correctly predicting 4 out of 28 total positives. The worst component model is DNN which has the lowest F-score, 0, because it classifies all CpG sites as negative ones. When trained with engineered features, similar to the dataset of Alzheimer's disease, the performances of all models but the DNN model degrade. DNN model remains the same performance as it still labels all sites as positives.

4.4 Discussion

In this study, we report an ensemble learning based strategy to extend the coverage of EWAS beyond the limit of array-based technologies. Our strategy requires that EWASs have already been performed using array-based technology but does not require individual-level methylation data. The goal here is to classify whether a CpG site is trait-associated using a rich set of external, precomputed epigenomics profiles collected from public resources such as ENCODE. A previous work named DIVAN adopts a similar strategy and is capable of identifying new disease-associated sequence variants beyond those found by Genome-Wide Associate Studies (GWASs) [111]. In DIVAN, a machine learning approach is implemented using 1806 features which cover a comprehensive collection of epigenomic/genomic annotations. It has been tested on 45 diseases/traits and obtains a cross validation AUC of ROC ranged from 0.65 to 0.88 with median 0.74. In our study, we also collect the same set of epigenomic/genomic annotations plus additional functional annotation scores given by multiple computational tools and functional genomics/epigenomics profiles such as ATAC-seq. To reduce overfittings, our ensemble feature selection method reduces the original 1953 features to 37 features for the dataset of Alzheimer's diseases and to 41 features for RICHS dataset. Among these features, many are disease/trait associated such as H3K4me1 in the dataset of Alzheimer's disease and H3K27me3 in fetal heart development in the RICHS dataset. Furthermore, in order to represent the most useful information from the selected features, we implement a sparse autoencoder to artificially engineer new features which are outputted from most active coding neurons and essentially linear/nonlinear combinations of input features. We next show that our ensemble model which consists of five individual models with different levels of complexity is able to obtain a recall of 0.57, a precision of 0.29 and an AUC of 0.75 for the dataset of Alzheimer's disease, and a recall of 0.07 and a

precision of 0.67 and an AUC of 0.53 for the RICHS dataset. Taking into account the imbalance between positive and negative CpG sites in the training set, these results suggest that the selected features do provide information about the methylation level changes at some CpG sites, and the ensemble model is capable of predicting methylation level changes at some CpG sites.

Generally speaking, the significances of our study are two-fold: 1. *Identify new trait-associated CpG sites beyond the 450K array:* The trade-offs between the high cost, high coverage WGBS and low cost, low coverage array-based technologies have always been a dilemma for researchers studying on the DNA methylation status across the genome. Although most researchers choose the latter for economic considerations, the low coverage of the whole methylome means lost opportunities to uncover the underlying biological mechanism in response to environmental exposure. This is similar to the problem of incomplete coverage of the genotyping array scientists faced when conducting GWAS. Therefore, our model offers a promising way to computationally extend the coverage of the CpG sites in the genome. *2. Explore mechanisms behind EWAS findings*: Although there is a growing evidence linking environmental exposures and DNA methylation, questions remain regarding the mechanisms through which these exposures impact epigenetic features, and particularly what underlies the specificity for their association with specific genes or targeted genomic regions. For example, a recent hypothesis of transcription factor occupancy driving this specificity has been put forth [103], but could be more broadly examined with more complete genome-wide data. By utilizing rich and diverse cell type specific epigenomic data as features in our machine learning approach, we can provide new evidence to understand these underlying mechanisms and spur research to explore those mechanisms in an experimental setting.

In our study, there are two major caveats and their corresponding remedies or future directions. First, no individual-level data is utilized. All the features used in the machine learning model is derived from external datasets obtained from seemingly unrelated cell lines or tissue types. We still think the strategy will work because biological connections may be hidden in the combinatorial pattern contained in the features we collected. The strength of this approach is the ability to make use of widely available results from existing EWAS without needing the underlying individual-level data. When individual-level data is available, we will design different strategies to utilize them. To be more specific, we will obtain individual-level methylation data from EWASs, with optional single or multiple types of individual-level omics profiling data (such as gene expression or chromatin accessibility). Then we will develop machine learning methods to impute methylation levels in genome-wide CpG sites beyond the array, then conduct real EWAS. Another potential caveat is that there are actually two types of CpG sites that are trait-associated, either the methylation level is higher among the cases than the controls or vice versa. It is possible that the epigenomic patterns around these two types of trait-associated CpGs are different. As an alternative strategy, we will separate these two types of CpGs and train two different classifiers. Or, we will consider this as a three-state classification problem: high, equal or low. We will train a multi-class (three-state) classification model accordingly.

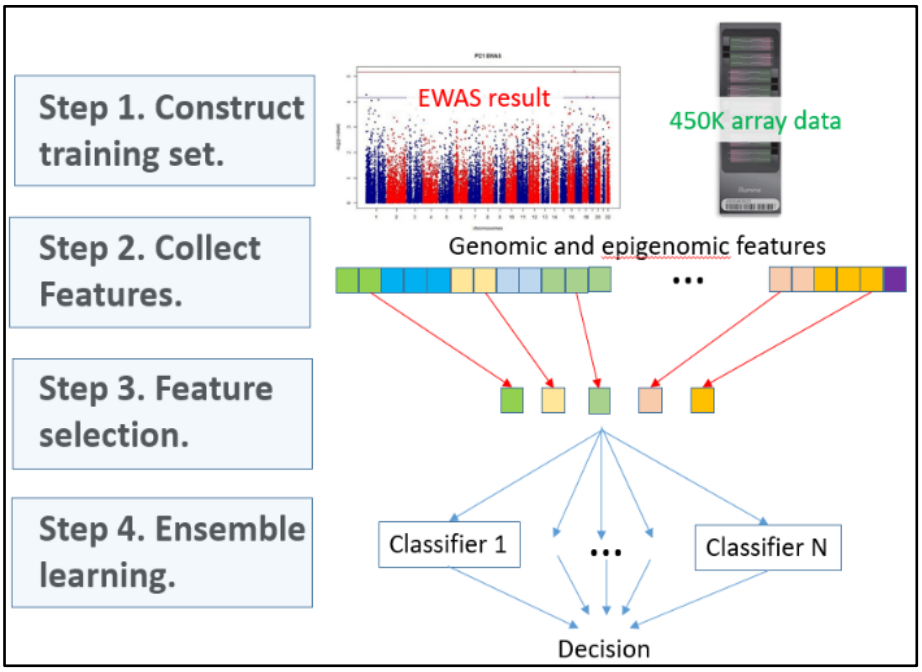
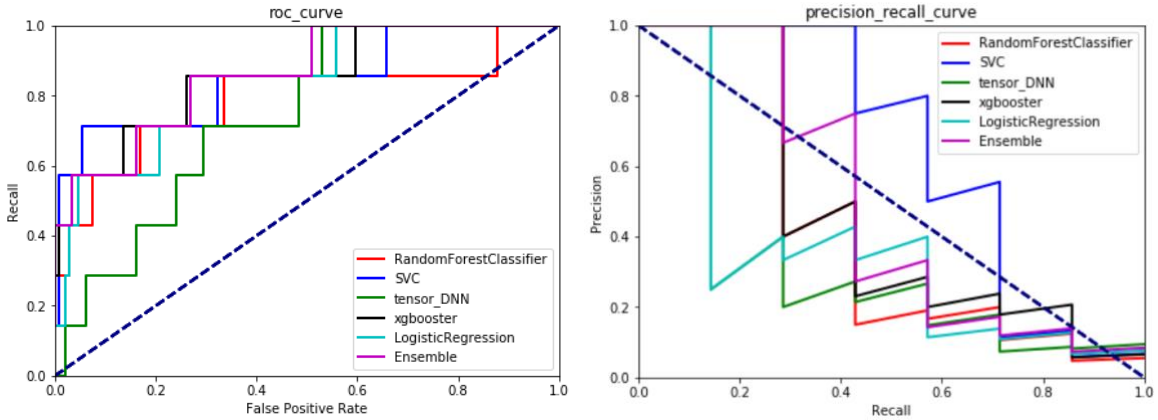


Figure 4.1: Illustration of the ensemble learning workflow.

a)



b)

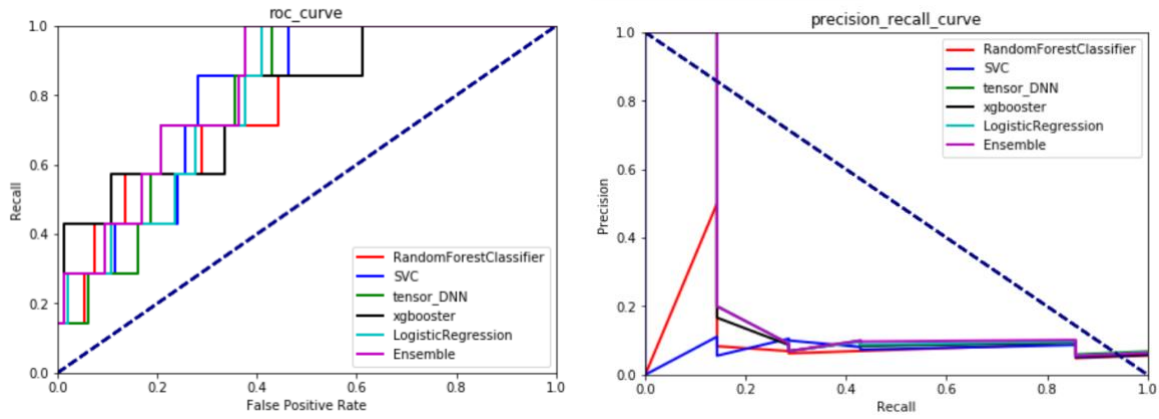
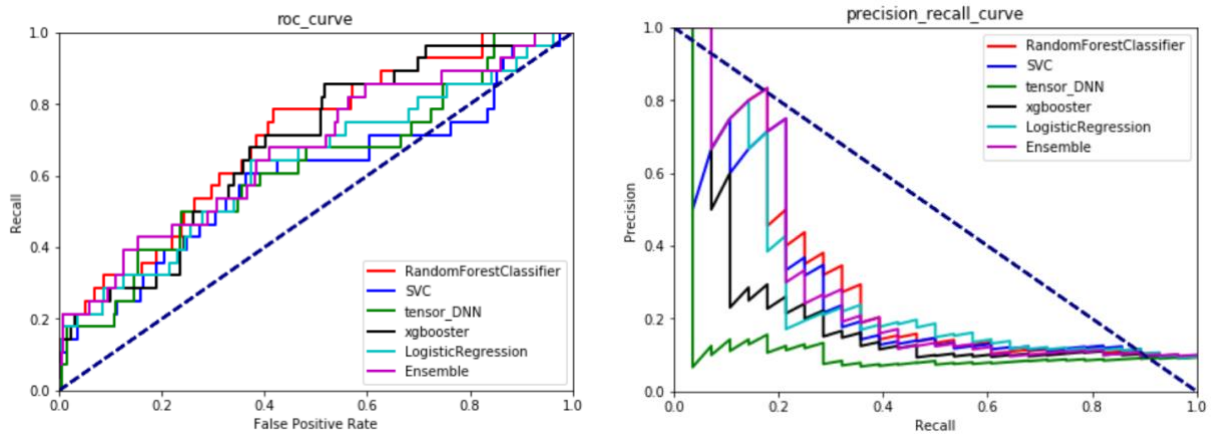


Figure 4.2: ROC and precision-recall curves of the ensemble model and its component models in evaluations of predictions on the dataset of Alzheimer's disease. a) The models are trained with selected features. b) The models are trained with engineered features.

a)



b)

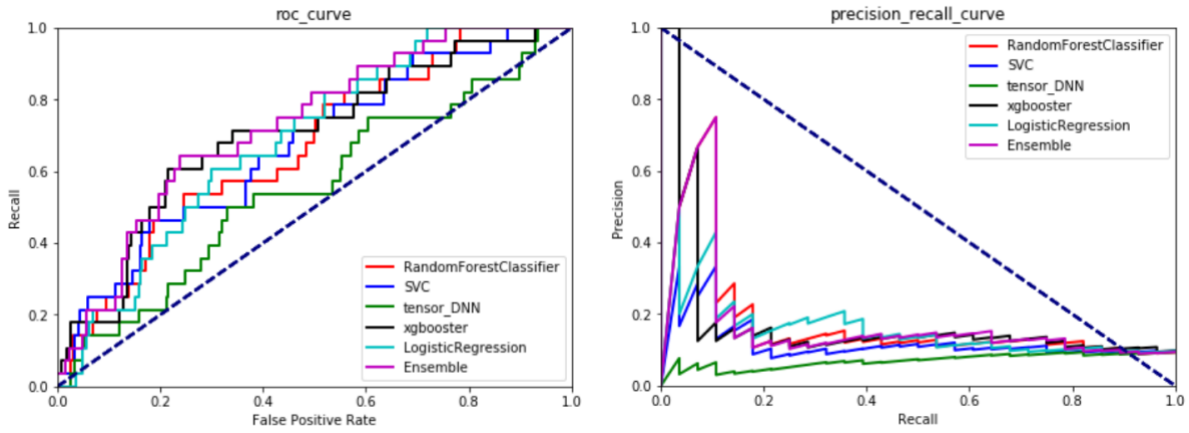


Figure 4.2: ROC and precision-recall curves of the ensemble model and its component models in evaluations of predictions on the RICHS dataset. a) The models are trained with selected features. b) The models are trained with engineered features.

Table 4.1. Summary of feature categories in the ensemble learning model

Data source	Features
REMC DNase	73
REMC Histone	735
ENCODE DNase	80
ENCODE FAIRE	31
ENCODE TF(HAIB)	292
ENCODE TF(SYDH)	279
ENCODE Histone	267
ENCODE RNA Polymerase	49
ENCODE RNA-seq	75
ENCODE ATAC-seq	47
GenoCaynon	4
Eigen	10
DANN	4
CADD	6
Distance to nearest TSS	1
Total	1953

Table 4.2. Best hyperparameters for the Alzheimer's disease and RICHS datasets trained with selected or engineered features.

Methods	**Hyperparameters**	**AD + selected features**	**AD + engineered features**	**RICHS + selected features**	**RICHS + engineered features**
Random forest	min_samples_leaf	5	5	2	1
	max_depth	12	20	29	27
	min_samples_split	28	29	5	22
	n_estimators	163	312	298	43
Logistic regression	C	9.99	7.27	6.26	5.86
Support vector classifier	C	0.95	0.67	0.93	0.87
	gamma (Gaussian kernel)	0.57	0.04	1	0.28
XGBoost	max_depth	13	6	9	16
	n_estimators	1562	1406	3443	1182
	reg_lambda	30.93	59.9	85.8	53.75
	gamma	1.45	2.97	3.32	1.01
Deep neural network	l2_reg	0.016	0.0076	0.016	0.027
	drop_out	0.4	0.21	0.42	0.15
	steps	1911	1254	1963	1175
	batch_size	30	30	30	30
	batch_normalization	True	True	True	True
	layer_structure	(185,111,37)	(84,70,56,42,28)	(140,105,70,35)	(96,80,64,48,32)

Table 4.3. Selected features for the Alzheimer's disease dataset and RICHS dataset.

AD				RICHS			
Features	**t-statistic**	**p-value**	**n**	**Features**	**t-statistic**	**p-value**	**n**
Fetal_Placenta-H3K4me1	5.20358342	2.22E-07	2	H1_Derived_Mesenchymal_ Stem_Cells-H2BK12ac	-0.1610283	0.87208195	2
K562-XRCC4	-1.3239177	0.1857251	2	Gliobla1.0	-0.0392104	0.96872527	2
CD14_Primary_Cells-H3K4me1	6.33609316	3.08E-10	2	Rectal_Smooth_ Muscle-H3K4me1	-2.279245	0.0227223	2
H9-H4K8ac	5.1906512	2.37E-07	2	Fetal_Adrenal_ Gland-H3K4me1	-1.518929	0.12888553	2
CADD_max_raw	3.82401969	0.00013648	2	H1_BMP4_Derived_ Trophoblast_Cultured_ Cells-H2BK12ac	-0.674831	0.49983502	2
CADD_avg_phred	4.03067273	5.83E-05	2	GM19099-NFKB	0.48085836	0.63065217	2
Breast_Luminal_Epithelial _Cells-H3K36me3	2.77074118	0.0056595	2	H1-hESC-TCF12	2.97084301	0.00299336	2
Chorion	0.82108474	0.41172394	2	Adult_Liver-H3K4me1	-2.2823573	0.02253781	2
NHDF-Ad-H3K9me3	-2.6226594	0.00880993	2	Melano	-0.375448	0.70735379	2
Monocytes-CD14+_RO01746-H3K4me1	7.02855679	3.11E-12	2	Fetal_Lung-H3K27me3	5.32037707	1.11E-07	2
Left_Ventricle-H3K4me1	6.8340975	1.18E-11	2	iPS-15b-H3K27me3	4.87868154	1.12E-06	2
Skeletal_Muscle-H3K27ac	6.74522791	2.15E-11	2	eigen_max_pc_raw	-1.4624618	0.14371923	2
Fetal_Adrenal_Gland-H3K27ac	5.62169709	2.24E-08	2	H1_Derived_Neuronal_ Progenitor_Cultured_ Cells-H3K36me3	-0.3431681	0.73149597	2
Fetal_Muscle_Leg-H3K4me1	4.87399332	1.21E-06	2	Chondrocytes_from_ Bone_Marrow_Derived_ Mesenchymal_Stem_Cell_ Cultured_Cells-H3K9me3	1.95771415	0.05035596	2
CADD_max_phred	4.30146657	1.80E-05	2	HeLa-S3-H3K9me3	-0.3023076	0.76243848	2
H1_BMP4_Derived_ Mesendoderm_Cultured_	4.56385187	5.42E-06	2	CD34_Primary_ Cells-H3K36me3	-2.9790253	0.00291481	2

Cells-H3K18ac							
ENCFF305PRP_ATAC_counts	-1.8885184	0.05914249	2	H1-H2BK15ac	3.36208581	0.00078324	2
Medullo_D341	0.43051393	0.66688154	2	Skeletal_ Muscle-H3K4me1	-2.7059408	0.00684958	2
Bone_Marrow_Derived_ Mesenchymal_Stem_ Cell_Cultured_ Cells-H3K9me3	-3.0925943	0.00201946	2	HeLa-S3-EZH2	5.14962324	2.78E-07	2
ENCFF205KDV_ATAC_counts	-1.4705423	0.14161739	2	CD4+_CD25-_ CD45RA+_Naive _Primary_Cells-H3K4me1	2.40015019	0.01644881	2
Pancreatic_Islets-H3K27me3	-1.4858322	0.13752634	2	CADD_max_raw	-0.1259418	0.89978638	2
Spleen-H3K27ac	1.24982085	0.21155319	2	hESC_Derived_CD184+ _Endoderm_Cultured_ Cells-H3K27me3	3.22083294	0.00129182	2
IMR90	-0.0045355	0.99638179	2	Stomach_Smooth_ Muscle-H3K27me3	5.02015445	5.46E-07	2
ENCFF780JBA_ATAC_counts	-2.1008124	0.03581818	2	GM12878-ZNF274	-2.8301304	0.00468379	2
GM128921.0	0.22920766	0.81873768	2	iPS-15b-H3K4me3	-1.3032243	0.192598	2
Fetal_Muscle_Leg-H3K27ac	4.1965936	2.86E-05	2	Penis_Foreskin_ Keratinocyte_ Primary_Cells-H3K4me1	0.79892264	0.42439839	2
H1_Derived_Neuronal_ Progenitor_Cultured_ Cells-H3K14ac	5.0026405	6.30E-07	2	FibroP	-0.4205563	0.67410921	2
CD20+-H3K4me2	-0.7676378	0.442819	2	ECC-1	-0.6434807	0.51996133	2
Medullo	-0.3698835	0.71151971	2	CADD_avg_phred	0.74310342	0.45747714	2
Neurosphere_Cultured_Cells_ Cortex_Derived-H3K36me3	-0.3436121	0.73118452	2	Fetal_Adrenal_ Gland-H3K27me3	4.20268252	2.71E-05	2
Penis_Foreskin_Fibroblast_ Primary_Cells-H3K9me3	-2.4866536	0.0129994	2	ENCFF585YAB_ ATAC_counts	-1.0850521	0.27798557	2
IMR901.0	1.28762954	0.19806667	2	Fetal_Intestine_ Large-H3K27me3	5.23566223	1.76E-07	2
Adipose_Nuclei-H3K4me1	5.35090994	1.01E-07	3	H9-H3K4me1	2.61588164	0.00894426	2

Psoas_Muscle-H3K27ac	6.18112886	8.12E-10	3	Muscle_Satellite_ Cultured_Cells-H3K27me3	4.10992507	4.06E-05	2
H1-hESC-BRCA1	-1.4030519	0.16080112	3	CD4+_CD25-_CD45RO+_ Memory_Primary_ Cells-H3K27me3	6.87523956	7.50E-12	3
Right_Atrium-H3K27ac	6.3575321	2.69E-10	3	GM12878-H3K4me1	3.39412924	0.00069739	3
genocanyon_score	4.96931437	7.46E-07	4	H1-H3K14ac	2.66835178	0.00766338	3
-	-	-	-	HeLa-S3-H3K27me3	6.16984296	7.75E-10	3
-	-	-	-	Fetal_Heart-H3K27me3	6.45699049	1.24E-10	4
-	-	-	-	genocanyon_score	1.57156253	0.11615729	4
-	-	-	-	Placenta_Chorion_ Smooth-H3K27me3	5.08545465	3.89E-07	4

Table 4.4. Test results of the ensemble learning model on the dataset of Alzheimer's disease.

	Selected features					Engineered features				
	ROC AUC	Recall	Precision	Accuracy	F-score	ROC AUC	Recall	Precision	Accuracy	F-score
Ensemble Model	0.75	0.57	0.29	0.92	0.38	0.67	0.43	0.18	0.88	0.25
Logistic Regressor	0.79	0.85	0.13	0.73	0.22	0.72	0.71	0.1	0.72	0.19
Random Forest	0.68	0.43	0.25	0.92	0.32	0.62	0.29	0.2	0.92	0.24
SVC	0.75	0.71	0.14	0.79	0.23	0.65	0.43	0.13	0.85	0.2
XGBoost	0.7	0.43	0.38	0.94	0.4	0.69	0.43	0.27	0.92	0.33
DNN	0.53	1	0.05	0.1	0.09	0.69	1	0.07	0.41	0.13

Table 4.5. Test results of the ensemble learning model on the RICHS dataset.

	Selected features					Engineered features				
	ROC AUC	Recall	Precision	Accuracy	F-score	ROC AUC	Recall	Precision	Accuracy	F-score
Ensemble Model	0.53	0.07	0.67	0.91	0.13	0.52	0.04	1	0.91	0.07
Logistic Regressor	0.58	0.21	0.27	0.87	0.24	0.55	0.71	0.1	0.72	0.19
Random Forest	0.57	0.14	1	0.92	0.25	0.52	0.04	1	0.91	0.07
SVC	0.57	0.18	0.29	0.88	0.22	0.51	0.04	0.2	0.9	0.06
XGBoost	0.56	0.14	0.44	0.9	0.22	0.54	0.11	0.38	0.9	0.17
DNN	0.5	0	0	0.91	0	0.5	0	0	0.91	0

Chapter 5

Summary

Innovations in sequencing technology have resulted in a deluge of high quality genomic data. The massive collection of genomics data is analogous to the Internet wherein extremely large amounts of diverse information are stored in heterogeneous formats. To facilitate scientific discoveries in bioinformatics using existing and emerging genomic datasets, we develop a ranking algorithm suite, trackRank, to identify and rank genomics datasets relative to their significance to a given query gene or pathway. For every genomic data type such as ChIP-seq and RNA-seq, the trackRank convert the raw data into scores that are comparable across different experiments and data types. We also develop a data computing and management infrastructure backed by NoSQL systems. This system includes a data warehouse infrastructure that manages genomics datasets, analytical results and metadata, with automatic data collection and integration from diverse data sources. It also includes high performance indexing databases involve MongoDB and Memcached to enable near real-time response of various query types. And finally, it includes a web based search portal to provide ranking based search of genome datasets, summary of data sets identified to contain statistically significance results related to the query.

Since we are pooling diverse datasets from multiple sources, maintaining high quality standards among the datasets is critically important and we are taking this seriously. We plan to add quality

metrics for datasets collected from places other than large international consortia (where data quality is typically high). In addition, we have deployed a "crowd sourcing" type of approach which allows users to leave comments on the data quality. These comments can alert future users about the potential quality issues so cautions can be exercised to avoid making conclusions based on faulty data. Despite the significant effort put into Omicseq, it is still (and will continue to be) a constant work in progress. So far, datasets are collected manually. This is manageable for collecting datasets *en masse* from those large consortia. For public data repositories such as GEO, the format, file type and annotations vary substantially. We aim to collect data more efficiently and will use a web crawler system to discover and collect newly emerging omics data across multiple sources.

In the second project, we develop three Apache cluster-based schemas, running on Hadoop, HBase and Spark respectively, for performing sorted merging on large volume of omics data. Particularly, we use the case of merging VCF files into a single TPED or VCF file as an example. Compared to traditional single machine based method such as multiway-merge or parallel computing methods such as HPC-based merging, the three schemas optimize the parallel execution of each step, minimize the waste of computing resources such as cores and memory, and maintain a better workload balance among cluster computing nodes. As a result, they are free of bottleneck and scalable on both input data size and the number of cores, and have much better strong and weak scalabilities. Our results imply that no matter how large the data size is, we can finish such a merging within a reasonable time provided we have sufficient computing resources which are now attainable via merchant cloud such as Amazon and Google. These schemas can be readily modified or extended to merge other types of omics data other than VCF files, by just changing the steps

generating contents of key-value pairs. The merging of multiple BED format files such as ChIP-seq peak lists [91] is such an example.

In addition to the simple sorted merging, there are some other more complicated types of merging of omics data. For example, the single machine based BEDTools [125] provides a utility to merge overlapping or "book-ended" features in an interval file into a single feature which spans all of the combined features. Same we expect BEDTools to have similar bottlenecks and limitations as other single machine based software such as VCFTools. However, if such a merging is implemented to run on a cluster of nodes installed with Apache distributed systems, instead of a single location key in merging VCF files, the merging keys need to be represented as overlapping genomic intervals, and thus its data representation model is much more complicated. Another example is the merging of gVCF files [126] in which each line corresponds to a range of genomic locations and contains info about all samples. It is also an interval sorted merging problem and its implementation on distributed system is also a challenge and deserve further studies.

In the third project, using only summary level data (association p-values) from array-based EWAS together with external, precomputed static epigenomics profiling data as features, we develop a machine-learning model to predict the disease/trait-associated methylation level changes of CpG sites beyond the 450K array. For each trait, from existing EWASs conducted using arrays, we establish a training set that consist of trait-associated CpGs as the positive set and matching background CpGs (not trait-associated) as negative set. Since the discrimination power of supervised learning depends on the availability of sufficient informative features, we establish a comprehensive features set by thorough search for various data types from multiple sources. So

far, we have collected 1953 epigenomic/genomic features, mainly TF and histone modification ChIP-seq, FAIRE, DNase-seq, RNA-seq, ATAC-seq data from ENCODE and REMC consortia, and functional annotation scores from computational tools including CADD, GenoCanyon, Eigen/EigenPC and DANN. From these features, we develop an ensemble feature selection method which take classification model biases into consideration to select the best subset of features used in the following training and testing. For each selected feature, we also conduct a two-sample test comparing CpGs in positive and negative training sets. To achieve a good balance between overfitting and underfitting, we adopt the ensemble learning method in our study. It consists of five individual models which are logistic regressor, support vector classifier, random forest, XGBoost and deep neural network. The predicted class membership probability of a CpG site is the normalized sum of the predicted class membership probabilities of all individual models. In our study, we achieve a precision of 0.57 and a recall of 0.29 when testing our model on dataset of Alzheimer's disease, and a precision of 0.67 and a recall of 0.07 when testing on RICHS dataset. These results, although not completely convincing, still suggest that the static epigenomic/genomic annotations are associated with the methylation level changes at some CpG sites. The methylation status at other CpG sites might be regulated or affected by many other factors including both environmental and individual factors. For example, in one study, a random forest model is developed to predict methylation rates for bulk ensemble cells, which takes comprehensive DNA annotations into account, including individual-level genomic contexts, and tissue-specific regulatory annotations such as DNase-I hypersensitivity sites [31]. In another study, a recurrent deep neural network based model (DeepCpG) which leverages local DNA sequence windows and observed neighboring methylation states to predict binary CpG methylation states in single cells [127].

Bibliography

1. Merelli I, Pérez-Sánchez H, Gesing S, D'Agostino D: **Managing, analysing, and integrating big data in medical bioinformatics: open problems and future perspectives.** *BioMed research international* 2014, **2014**.
2. Genovese Y, Prentice S: **Pattern-based strategy: getting value from big data.** *Gartner Special Report G* 2011, **214032:**2011.
3. Johnson DS, Mortazavi A, Myers RM, Wold B: **Genome-wide mapping of in vivo protein-DNA interactions.** *Science* 2007, **316:**1497-1502.
4. Barski A, Cuddapah S, Cui K, Roh TY, Schones DE, Wang Z, Wei G, Chepelev I, Zhao K: **High-resolution profiling of histone methylations in the human genome.** *Cell* 2007, **129:**823-837.
5. Robertson G, Hirst M, Bainbridge M, Bilenky M, Zhao Y, Zeng T, Euskirchen G, Bernier B, Varhol R, Delaney A, et al: **Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing.** *Nat Methods* 2007, **4:**651-657.
6. Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B: **Mapping and quantifying mammalian transcriptomes by RNA-Seq.** *Nat Methods* 2008, **5:**621-628.
7. Visscher PM, Brown MA, McCarthy MI, Yang J: **Five years of GWAS discovery.** *The American Journal of Human Genetics* 2012, **90:**7-24.
8. Mughal S, Moghul I, Yu J, UKIRDC, Clark T, Gregory DS, Pontikos N: **Pheno4J: a gene to phenotype graph database.** *Bioinformatics* 2017, **33:**3317-3319.
9. Mathias RA, Taub MA, Gignoux CR, Fu W, Musharoff S, O'Connor TD, Vergara C, Torgerson DG, Pino-Yanes M, Shringarpure SS, et al: **A continuum of admixture in the Western Hemisphere revealed by the African Diaspora genome.** *Nat Commun* 2016, **7:**12522.
10. Siepel A, Bejerano G, Pedersen JS, Hinrichs AS, Hou M, Rosenbloom K, Clawson H, Spieth J, Hillier LW, Richards S, et al: **Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes.** *Genome Res* 2005, **15:**1034-1050.
11. Lek M, Karczewski KJ, Minikel EV, Samocha KE, Banks E, Fennell T, O'Donnell-Luria AH, Ware JS, Hill AJ, Cummings BB, et al: **Analysis of protein-coding genetic variation in 60,706 humans.** *Nature* 2016, **536:**285-291.
12. Margolis R, Derr L, Dunn M, Huerta M, Larkin J, Sheehan J, Guyer M, Green ED: **The National Institutes of Health's Big Data to Knowledge (BD2K) initiative: capitalizing on biomedical big data.** *J Am Med Inform Assoc* 2014, **21:**957-958.

13. Rose PW, Beran B, Bi C, Bluhm WF, Dimitropoulos D, Goodsell DS, Prlić A, Quesada M, Quinn GB, Westbrook JD: **The RCSB Protein Data Bank: redesigned web site and web services.** *Nucleic acids research* 2010, **39:**D392-D401.
14. Belleau F, Nolin M-A, Tourigny N, Rigault P, Morissette J: **Bio2RDF: towards a mashup to build bioinformatics knowledge systems.** *Journal of biomedical informatics* 2008, **41:**706-716.
15. Lu Q, Hu Y, Sun J, Cheng Y, Cheung K-H, Zhao H: **A statistical framework to predict functional non-coding regions in the human genome through integrated analysis of annotation data.** *Scientific reports* 2015, **5:**10576.
16. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M: **The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.** *Genome research* 2010, **20:**1297-1303.
17. Schatz MC: **CloudBurst: highly sensitive read mapping with MapReduce.** *Bioinformatics* 2009, **25:**1363-1369.
18. Pireddu L, Leo S, Zanetti G: **SEAL: a distributed short read mapping and duplicate removal tool.** *Bioinformatics* 2011, **27:**2159-2160.
19. Niemenmaa M, Kallio A, Schumacher A, Klemelä P, Korpelainen E, Heljanko K: **Hadoop-BAM: directly manipulating next generation sequencing data in the cloud.** *Bioinformatics* 2012, **28:**876-877.
20. Langmead B, Schatz MC, Lin J, Pop M, Salzberg SL: **Searching for SNPs with cloud computing.** *Genome Biol* 2009, **10:**R134.
21. Wang S, Mares MA, Guo YK: **CGDM: collaborative genomic data model for molecular profiling data using NoSQL.** *Bioinformatics* 2016, **32:**3654-3660.
22. Zhou W, Li R, Yuan S, Liu C, Yao S, Luo J, Niu B: **MetaSpark: a spark-based distributed processing tool to recruit metagenomic reads to reference genomes.** *Bioinformatics* 2017, **33:**1090-1092.
23. Niu B, Zhu Z, Fu L, Wu S, Li W: **FR-HIT, a very fast program to recruit metagenomic reads to homologous reference genomes.** *Bioinformatics* 2011, **27:**1704-1705.
24. **AWS Genomics Guide** [https://d0.awsstatic.com/Industries/HCLS/Resources/AWS_Genomics_WP.pdf]
25. Gruber K: **Google for genomes.** Nature Research; 2014.
26. Massie M, Nothaft F, Hartl C, Kozanitis C, Schumacher A, Joseph AD, Patterson DA: **Adam: Genomics formats and processing patterns for cloud scale computing.** *University of California, Berkeley Technical Report, No UCB/EECS-2013* 2013, **207**.
27. O'Brien AR, Saunders NF, Guo Y, Buske FA, Scott RJ, Bauer DC: **VariantSpark: population scale clustering of genotype information.** *BMC genomics* 2015, **16:**1052.
28. Wiewiórka MS, Messina A, Pacholewska A, Maffioletti S, Gawrysiak P, Okoniewski MJ: **SparkSeq: fast, scalable and cloud-ready tool for the interactive genomic data analysis with nucleotide precision.** *Bioinformatics* 2014, **30:**2652-2653.
29. Decap D, Reumers J, Herzeel C, Costanza P, Fostier J: **Halvade: scalable sequence analysis with MapReduce.** *Bioinformatics* 2015, **31:**2482-2488.

30. He M, Person TN, Hebbring SJ, Heinzen E, Ye Z, Schrodi SJ, McPherson EW, Lin SM, Peissig PL, Brilliant MH: **SeqHBase: a big data toolset for family based sequencing data analysis.** *Journal of medical genetics* 2015**:**jmedgenet-2014-102907.
31. Zhang W, Spector TD, Deloukas P, Bell JT, Engelhardt BE: **Predicting genome-wide DNA methylation using methylation marks, genomic position, and DNA regulatory elements.** *Genome biology* 2015, **16:**14.
32. Barrero MJ, Boué S, Belmonte JCI: **Epigenetic mechanisms that regulate cell identity.** *Cell stem cell* 2010, **7:**565-570.
33. Scarano MI, Strazzullo M, Matarazzo MR, D'Esposito M: **DNA methylation 40 years later: Its role in human health and disease.** *Journal of cellular physiology* 2005, **204:**21-35.
34. Robertson KD: **DNA methylation and human disease.** *Nature Reviews Genetics* 2005, **6:**597.
35. De Jager PL, Srivastava G, Lunnon K, Burgess J, Schalkwyk LC, Yu L, Eaton ML, Keenan BT, Ernst J, McCabe C: **Alzheimer's disease: early alterations in brain DNA methylation at ANK1, BIN1, RHBDF2 and other loci.** *Nature neuroscience* 2014, **17:**1156.
36. Boyle AP, Davis S, Shulha HP, Meltzer P, Margulies EH, Weng Z, Furey TS, Crawford GE: **High-resolution mapping and characterization of open chromatin across the genome.** *Cell* 2008, **132:**311-322.
37. Crawford GE, Holt IE, Whittle J, Webb BD, Tai D, Davis S, Margulies EH, Chen Y, Bernat JA, Ginsburg D: **Genome-wide mapping of DNase hypersensitive sites using massively parallel signature sequencing (MPSS).** *Genome research* 2006, **16:**123-131.
38. Giresi PG, Kim J, McDaniell RM, Iyer VR, Lieb JD: **FAIRE (Formaldehyde-Assisted Isolation of Regulatory Elements) isolates active regulatory elements from human chromatin.** *Genome research* 2007, **17:**877-885.
39. Consortium EP, Dunham I, Kundaje A, Aldred SF, Collins PJ, Davis CA, Doyle F, Epstein CB, Frietze S, Harrow J, et al: **An integrated encyclopedia of DNA elements in the human genome.** *Nature* 2012, **489:**57-74.
40. Kundaje A, Meuleman W, Ernst J, Bilenky M, Yen A, Heravi-Moussavi A, Kheradpour P, Zhang Z, Wang J, Ziller MJ: **Integrative analysis of 111 reference human epigenomes.** *Nature* 2015, **518:**317.
41. Barrett T, Wilhite SE, Ledoux P, Evangelista C, Kim IF, Tomashevsky M, Marshall KA, Phillippy KH, Sherman PM, Holko M, et al: **NCBI GEO: archive for functional genomics data sets--update.** *Nucleic Acids Res* 2013, **41:**D991-995.
42. Kodama Y, Shumway M, Leinonen R, International Nucleotide Sequence Database C: **The Sequence Read Archive: explosive growth of sequencing data.** *Nucleic Acids Res* 2012, **40:**D54-56.
43. Parkinson H, Kapushesky M, Shojatalab M, Abeygunawardena N, Coulson R, Farne A, Holloway E, Kolesnykov N, Lilja P, Lukk M, et al: **ArrayExpress--a public database of microarray experiments and gene expression profiles.** *Nucleic Acids Res* 2007, **35:**D747-750.
44. Consortium TGP: **An integrated map of genetic variation from 1,092 human genomes.** *Nature* 2012, **491:**56-65.

45. Network TCGAR: **Comprehensive genomic characterization defines human glioblastoma genes and core pathways.** *Nature* 2008, **455:**1061-1068.
46. Zhang J, Baran J, Cros A, Guberman JM, Haider S, Hsu J, Liang Y, Rivkin E, Wang J, Whitty B, et al: **International Cancer Genome Consortium Data Portal--a one-stop shop for cancer genomics data.** *Database (Oxford)* 2011, **2011:**bar026.
47. Gerstein MB, Lu ZJ, Van Nostrand EL, Cheng C, Arshinoff BI, Liu T, Yip KY, Robilotto R, Rechtsteiner A, Ikegami K, et al: **Integrative analysis of the Caenorhabditis elegans genome by the modENCODE project.** *Science* 2010, **330:**1775-1787.
48. mod EC, Roy S, Ernst J, Kharchenko PV, Kheradpour P, Negre N, Eaton ML, Landolin JM, Bristow CA, Ma L, et al: **Identification of functional elements and regulatory circuits by Drosophila modENCODE.** *Science* 2010, **330:**1787-1797.
49. Chadwick LH: **The NIH Roadmap Epigenomics Program data resource.** *Epigenomics* 2012, **4:**317-324.
50. Zhang Y, Cao X, Zhong S: **GeNemo: a search engine for web-based functional genomic data.** *Nucleic Acids Res* 2016, **44:**W122-127.
51. Brin S, Page L: **The anatomy of a large-scale hypertextual Web search engine.** *Computer Networks and ISDN Systems* 1998, **30:**107-117.
52. Leinonen R, Sugawara H, Shumway M, International Nucleotide Sequence Database C: **The sequence read archive.** *Nucleic Acids Res* 2011, **39:**D19-21.
53. Langmead B, Salzberg SL: **Fast gapped-read alignment with Bowtie 2.** *Nat Methods* 2012, **9:**357-359.
54. Zhu Y, Qiu P, Ji Y: **TCGA-assembler: open-source software for retrieving and processing TCGA data.** *Nat Methods* 2014, **11:**599-600.
55. Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform.** *Bioinformatics* 2009, **25:**1754-1760.
56. Li H, Durbin R: **Fast and accurate long-read alignment with Burrows-Wheeler transform.** *Bioinformatics* 2010, **26:**589-595.
57. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, Szczesniak MW, Gaffney DJ, Elo LL, Zhang X, Mortazavi A: **A survey of best practices for RNA-seq data analysis.** *Genome Biol* 2016, **17:**13.
58. Li B, Dewey CN: **RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome.** *BMC Bioinformatics* 2011, **12:**323.
59. Aryee MJ, Jaffe AE, Corrada-Bravo H, Ladd-Acosta C, Feinberg AP, Hansen KD, Irizarry RA: **Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays.** *Bioinformatics* 2014, **30:**1363-1369.
60. Core LJ, Waterfall JJ, Lis JT: **Nascent RNA sequencing reveals widespread pausing and divergent initiation at human promoters.** *Science* 2008, **322:**1845-1848.
61. Barretina J, Caponigro G, Stransky N, Venkatesan K, Margolin AA, Kim S, Wilson CJ, Lehar J, Kryukov GV, Sonkin D, et al: **The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity.** *Nature* 2012, **483:**603-607.
62. Lappalainen T, Sammeth M, Friedlander MR, t Hoen PA, Monlong J, Rivas MA, Gonzalez-Porta M, Kurbatova N, Griebel T, Ferreira PG, et al: **Transcriptome and genome sequencing uncovers functional variation in humans.** *Nature* 2013, **501:**506-511.

63. Hoffmann R: **A wiki for the life sciences where authorship matters.** *Nat Genet* 2008, **40:**1047-1051.
64. Bernstein BE, Stamatoyannopoulos JA, Costello JF, Ren B, Milosavljevic A, Meissner A, Kellis M, Marra MA, Beaudet AL, Ecker JR, et al: **The NIH Roadmap Epigenomics Mapping Consortium.** *Nat Biotechnol* 2010, **28:**1045-1048.
65. Prensner JR, Rubin MA, Wei JT, Chinnaiyan AM: **Beyond PSA: the next generation of prostate cancer biomarkers.** *Sci Transl Med* 2012, **4:**127rv123.
66. Lucila O-m, George A, Ian F, Maryann M, Susanna-Assunta S, Hua X: *bioCADDIE white paper - Data Discovery Index.* 2015.
67. Siretskiy A, Sundqvist T, Voznesenskiy M, Spjuth O: **A quantitative assessment of the hadoop framework for analyzing massively parallel dna sequencing data.** *Gigascience* 2015, **4:**26.
68. Reyes-Ortiz JL, Oneto L, Anguita D: **Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf.** *Procedia Computer Science* 2015, **53:**121-130.
69. Burren OS, Guo H, Wallace C: **VSEAMS: a pipeline for variant set enrichment analysis using summary GWAS data identifies IKZF3, BATF and ESRRA as key transcription factors in type 1 diabetes.** *Bioinformatics* 2014, **30:**3342-3348.
70. **Apache Hadoop** [http://hadoop.apache.org/]
71. Dean J, Ghemawat S: **Mapreduce: Simplified data processing on large clusters.** *Communications of the Acm* 2008, **51:**107-113.
72. Vora MN: **Hadoop-HBase for large-scale data.** In *Computer science and network technology (ICCSNT), 2011 international conference on*. IEEE; 2011: 601-605.
73. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I: **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*. USENIX Association; 2012: 2-2.
74. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, Maller J, Sklar P, de Bakker PI, Daly MJ, Sham PC: **PLINK: a tool set for whole-genome association and population-based linkage analyses.** *Am J Hum Genet* 2007, **81:**559-575.
75. Mohammed EA, Far BH, Naugler C: **Applications of the MapReduce programming framework to clinical big data analysis: current landscape and future trends.** *BioData Min* 2014, **7:**22.
76. Huang H, Tata S, Prill RJ: **BlueSNP: R package for highly scalable genome-wide association studies using Hadoop clusters.** *Bioinformatics* 2013, **29:**135-136.
77. White T: *Hadoop: The definitive guide.* " O'Reilly Media, Inc."; 2012.
78. Silberschatz A, Korth HF, Sudarshan S: **DatabaseSystem Concepts.** 2010.
79. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, et al: **The variant call format and VCFtools.** *Bioinformatics* 2011, **27:**2156-2158.
80. **Multiway-Merge Algorithm** [https://en.wikipedia.org/wiki/K-Way_Merge_Algorithms]
81. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE: **Bigtable: A distributed storage system for structured data.** *Acm Transactions on Computer Systems* 2008, **26**.

82. Genomes Project C, Abecasis GR, Altshuler D, Auton A, Brooks LD, Durbin RM, Gibbs RA, Hurles ME, McVean GA: **A map of human genome variation from population-scale sequencing.** *Nature* 2010, **467:**1061-1073.
83. Kwon Y, Balazinska M, Howe B, Rolia J: **A study of skew in mapreduce applications.** *Open Cirrus Summit* 2011, **11**.
84. O'Neil P, Cheng E, Gawlick D, O'Neil E: **The log-structured merge-tree (LSM-tree).** *Acta Informatica* 1996, **33:**351-385.
85. Li H: **Tabix: fast retrieval of sequence features from generic TAB-delimited files.** *Bioinformatics* 2011, **27:**718-719.
86. Amdahl GM: **Validity of the single processor approach to achieving large scale computing capabilities.** In *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM; 1967: 483-485.
87. Gustafson JL: **Reevaluating Amdahl's law.** *Communications of the ACM* 1988, **31:**532-533.
88. Sedgewick R, Flajolet P: *An introduction to the analysis of algorithms.* Addison-Wesley; 2013.
89. Özsu MT, Valduriez P: *Principles of distributed database systems.* Springer Science & Business Media; 2011.
90. Miner D, Shook A: *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems.* " O'Reilly Media, Inc."; 2012.
91. Chen L, Wang C, Qin ZS, Wu H: **A novel statistical method for quantitative comparison of multiple ChIP-seq datasets.** *Bioinformatics* 2015, **31:**1889-1896.
92. Li S, Hursting SD, Davis BJ, McLACHLAN J, Barrett J: **Environmental exposure, DNA methylation, and gene regulation.** *Annals of the New York Academy of Sciences* 2003, **983:**161-169.
93. Marsit CJ, Maccani MA, Padbury JF, Lester BM: **Placental 11-beta hydroxysteroid dehydrogenase methylation is associated with newborn growth and a measure of neurobehavioral outcome.** *PloS one* 2012, **7:**e33794.
94. Bollati V, Baccarelli A: **Environmental epigenetics.** *Heredity* 2010, **105:**105.
95. Lister R, Pelizzola M, Dowen RH, Hawkins RD, Hon G, Tonti-Filippini J, Nery JR, Lee L, Ye Z, Ngo QM, et al: **Human DNA methylomes at base resolution show widespread epigenomic differences.** *Nature* 2009, **462:**315-322.
96. Kingsley SL, Eliot MN, Whitsel EA, Huang YT, Kelsey KT, Marsit CJ, Wellenius GA: **Maternal residential proximity to major roadways, birth weight, and placental DNA methylation.** *Environ Int* 2016, **92-93:**43-49.
97. Maccani JZ, Koestler DC, Lester B, Houseman EA, Armstrong DA, Kelsey KT, Marsit CJ: **Placental DNA Methylation Related to Both Infant Toenail Mercury and Adverse Neurobehavioral Outcomes.** *Environ Health Perspect* 2015, **123:**723-729.
98. Green BB, Karagas MR, Punshon T, Jackson BP, Robbins DJ, Houseman EA, Marsit CJ: **Epigenome-Wide Assessment of DNA Methylation in the Placenta and Arsenic Exposure in the New Hampshire Birth Cohort Study (USA).** *Environ Health Perspect* 2016, **124:**1253-1260.
99. Everson TM, Punshon T, Jackson BP, Hao K, Lambertini L, Chen J, Karagas MR, Marsit CJ: **Cadmium-Associated Differential Methylation throughout the Placental Genome:**

Epigenome-Wide Association Study of Two U.S. Birth Cohorts. *Environ Health Perspect* 2018, **126:**017010.

100. Argos M, Chen L, Jasmine F, Tong L, Pierce BL, Roy S, Paul-Brutus R, Gamble MV, Harper KN, Parvez F, et al: **Gene-specific differential DNA methylation and chronic arsenic exposure in an epigenome-wide association study of adults in Bangladesh.** *Environ Health Perspect* 2015, **123:**64-71.
101. Salas LA, Bustamante M, Gonzalez JR, Gracia-Lavedan E, Moreno V, Kogevinas M, Villanueva CM: **DNA methylation levels and long-term trihalomethane exposure in drinking water: an epigenome-wide association study.** *Epigenetics* 2015, **10:**650-661.
102. Tarantini L, Bonzini M, Apostoli P, Pegoraro V, Bollati V, Marinelli B, Cantone L, Rizzo G, Hou L, Schwartz J, et al: **Effects of particulate matter on genomic DNA methylation content and iNOS promoter methylation.** *Environ Health Perspect* 2009, **117:**217-222.
103. Martin EM, Fry RC: **A cross-study analysis of prenatal exposures to environmental contaminants and the epigenome: support for stress-responsive transcription factor occupancy as a mediator of gene-specific CpG methylation patterning.** *Environmental epigenetics* 2016, **2**.
104. Flanagan JM: **Epigenome-wide association studies (EWAS): past, present, and future.** *Methods Mol Biol* 2015, **1238:**51-63.
105. Rakyan VK, Down TA, Balding DJ, Beck S: **Epigenome-wide association studies for common human diseases.** *Nat Rev Genet* 2011, **12:**529-541.
106. Cokus SJ, Feng S, Zhang X, Chen Z, Merriman B, Haudenschild CD, Pradhan S, Nelson SF, Pellegrini M, Jacobsen SE: **Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning.** *Nature* 2008, **452:**215-219.
107. Bibikova M, Barnes B, Tsan C, Ho V, Klotzle B, Le JM, Delano D, Zhang L, Schroth GP, Gunderson KL, et al: **High density DNA methylation array with single CpG site resolution.** *Genomics* 2011, **98:**288-295.
108. Bibikova M, Lin Z, Zhou L, Chudin E, Garcia EW, Wu B, Doucet D, Thomas NJ, Wang Y, Vollmer E, et al: **High-throughput DNA methylation profiling using universal bead arrays.** *Genome Res* 2006, **16:**383-393.
109. Bibikova M, Le J, Barnes B, Saedinia-Melnyk S, Zhou L, Shen R, Gunderson KL: **Genome-wide DNA methylation profiling using Infinium(R) assay.** *Epigenomics* 2009, **1:**177-200.
110. Busche S, Shao X, Caron M, Kwan T, Allum F, Cheung WA, Ge B, Westfall S, Simon MM, Multiple Tissue Human Expression R, et al: **Population whole-genome bisulfite sequencing across two tissues highlights the environment as the principal source of human methylome variation.** *Genome Biol* 2015, **16:**290.
111. Chen L, Jin P, Qin ZS: **DIVAN: accurate identification of non-coding disease-specific risk variants using multi-omics profiles.** *Genome biology* 2016, **17:**252.
112. Ritchie GR, Dunham I, Zeggini E, Flicek P: **Functional annotation of noncoding sequence variants.** *Nature methods* 2014, **11:**294.
113. Kircher M, Witten DM, Jain P, O'roak BJ, Cooper GM, Shendure J: **A general framework for estimating the relative pathogenicity of human genetic variants.** *Nature genetics* 2014, **46:**310.
114. Quang D, Chen Y, Xie X: **DANN: a deep learning approach for annotating the pathogenicity of genetic variants.** *Bioinformatics* 2014, **31:**761-763.

115. Ionita-Laza I, McCallum K, Xu B, Buxbaum JD: **A spectral approach integrating functional genomic annotations for coding and noncoding variants.** *Nature genetics* 2016, **48:**214.
116. Chen T, Guestrin C: **Xgboost: A scalable tree boosting system.** In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM; 2016: 785-794.
117. Le QV: **Building high-level features using large scale unsupervised learning.** In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE; 2013: 8595-8598.
118. Coates A, Ng A, Lee H: **An analysis of single-layer networks in unsupervised feature learning.** In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011: 215-223.
119. Bergstra J, Yamins D, Cox DD: **Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures.** 2013.
120. He K, Zhang X, Ren S, Sun J: **Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.** In *Proceedings of the IEEE international conference on computer vision*. 2015: 1026-1034.
121. Maaten Lvd, Hinton G: **Visualizing data using t-SNE.** *Journal of machine learning research* 2008, **9:**2579-2605.
122. Platt J: **Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods.** *Advances in large margin classifiers* 1999, **10:**61-74.
123. Gjoneska E, Pfenning AR, Mathys H, Quon G, Kundaje A, Tsai L-H, Kellis M: **Conserved epigenomic signals in mice and humans reveal immune basis of Alzheimer's disease.** *Nature* 2015, **518:**365.
124. Liu C, Xu J, Chen Y, Guo X, Zheng Y, Wang Q, Chen Y, Ni Y, Zhu Y, Joyce BT: **Characterization of genome-wide H3K27ac profiles reveals a distinct PM 2.5-associated histone modification signature.** *Environmental health* 2015, **14:**65.
125. Quinlan AR, Hall IM: **BEDTools: a flexible suite of utilities for comparing genomic features.** *Bioinformatics* 2010, **26:**841-842.
126. Cicotti P, Pfeiffer W, Gujral M, Sinkovits R, Strande S, Hawkins R: **Performance Characterization and Optimization Assessment of Bioinformatics Applications.** In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. ACM; 2017: 67.
127. Angermueller C, Lee HJ, Reik W, Stegle O: **DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning.** *Genome biology* 2017, **18:**67.