Medical Big Data Analysis System to Discover Associations between Genetic Variants and Diseases

Daehee Kim, Stephanie Gamboa, Vanessa Hernandez, Marlen Martinez-Lopez
California State University-Stanislaus, Turlock, CA
{dkim10,sgamboa,vhernandez27, mmartinezlopez}@csustan.edu

Scott J Hebbring, John Mayer
Center for Precision Medicine Research, Marshfield Clinic Research Institute, Marshfield, WI
{hebbring.scott,Mayer.John}@marshfieldresearch.org

Jamie Fox
Prevention Genetics, Marshfield, WI
jamie.fox@preventiongenetics.com

***Abstract*—Electronic Health Records (EHRs) and genomic data are fruitful resources to find associations between genetic variants and human diseases. Medical researchers have used Phenome-Wide Association Study (PheWAS) to find diseases relevant to genomic variants, and Genome-Wide Association Study (GWAS) to discover genomic variants related to specific diseases. In the era of Big data, the quantity of disease and genomic data have significantly increased, resulting in long processing and research time for Big data analysis. We designed and implemented a prototype of Medical Big Data Analysis System that finds the links between genetic variants and human disease with PheWAS and GWAS dataset of a population-based biobank in Marshfield Clinic Institute. The system consists of front-end Web-based graphic user interface using Shiny application, and back-end Big data analysis servers using Apache Spark and Mongo database. We evaluated the associations with selected variants and diseases and present the performance and CPU/memory overhead of the system while varying the number of executors in Spark cluster, which provides readers with multiple factors for deploying medical Big data analysis system.**

***Index Terms*—Phenome Wide Association Study, PheWAS, Genome Wide Association Study, GWAS, Medical Big Data Analysis, Apache Spark, SparkR, R, Shiny, Mongo DB**

I. Introduction

Big data has been used in health care for clinical decision support, efficiency of health care professionals, and patient care. In 2015, the White House launched the Precision Medicine Initiative (PMI) [1] where National Institutes of Health (NIH) awarded $55 million dollars to seven institutions for building a diverse research cohort of 1 million individuals including their health information, blood sample for DNA sequencing, and access to electronic health records (EHRs). The goal of the initiative was to effectively prevent and treat illness based on an understanding of individual differences.

Electronic Health Records (EHRs) and DNA genotypes are the main resources used to discover individual differences. Regarding EHRs, doctors and health care providers acquire patient information from test and results, medical procedures, and disease diagnoses. The disease diagnoses are converted to codes, International Classification of Diseases (ICD)-9 [2] (e.g. diabetes: 250). Patient samples with DNA genotypes can be used to explore individual traits and diseases. The Human Genome Project [3] [4] discovered genome sequences containing 3 billion base pairs of nucleotides. The differences in sequence lead to genetic variants (hair color, eye color, height) and human disease.

Genome-Wide Association Studies (GWASs) [5] were performed to find genetic variants for certain diseases, which is a phenotype-to-genotype approach. Phenome-Wide Association Studies (PheWASs) were conducted to explore multiple diseases relevant to a genetic variant, which is a genotype-to-phenotype approach [6] [7] [8] [9] [10] [11].

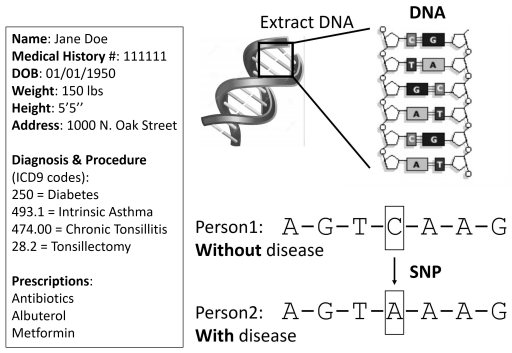
Marshfield Clinic Research Institute (MCRI)[1] located in Marshfield, WI, is a medical research institute with a population-based biobank, which is a human DNA repository linked to an EHR. When genetic data has been collected, these biobanks allow researchers to associate any one of millions of genetic variants to any one of thousands of phenotypes. To conduct these analysis, researchers often delegated data to highly trained experts in statistics and/or informatics. For example, researchers send a list of variants and/or phenotypes to experts who often run programs manually using data warehouse appliances, Cloud-based analysis, data management applications, and statistical packages. The experts return relevant statistical outputs to the researcher for interpretation. However, the vast quantity and diversity of disease and genome data, in combination with manual processes, can result in long turnaround times on the order of weeks or months.

To dramatically improve efficacy, we designed and implemented Medical Big Data Analysis System that retrieves association results from a GWAS-by-PheWAS dataset derived from extensive genomic and phenomic data in MCRI's biobank. The main goal of this system is that end users (medical researchers) are able to quickly search association data without the time-consuming processes described previously. The system uses Shiny application [12] for front-end Web-based graphic user interface and back-end Big data analysis servers using Apache Spark and Mongo database.

Our contributions are as follows.

- To the best of our knowledge, this is the first bidirectional approach combining PheWAS and GWAS automatically,

[1] https://www.marshfieldresearch.org/



(a) EHR (b) DNA Sequence

Fig. 1. EHR and Genetic Data

which is discussed compared to unidirectional previous studies.

- We introduce a prototype of medical Big data analysis system to explore associations between genetic variants and human disease
- With evaluation results, we share multiple factors to be considered for medical Big data analysis system.

The rest of the paper is organized as follows. We provide background information for medical research with disease and genetic data in Section II. We discuss design and implementation in Section III. We then present the performance and overhead of the system in Section IV and related work in Section V. We conclude in Section VI along with future work.

II. Background

A. Health Record Data

Health care providers record data on patients. Those records include biological measurements (e.g., blood pressure, temperature, and weight), disease diagnoses, medical procedures (surgeries), prescriptions, family history, and allergies. This data is stored in the patient's electronic health record (EHR) as depicted in Fig. 1(a). International Classification of Diseases (ICD) codes [2] describe common medical conditions and procedures. The codes are organized in a hierarchal structure. For example, with ICD-9 coding, 714.3 (Juvenile Chronic Polyarthritis) is a disease under 714 (Rheumatoid arthritis) disease code.

B. Genetic Data

Human DNA is distributed between 22 pairs of somatic chromosome and 2 sex chromosomes. As shown in Figure 1(b), genetic data is retrieved from DNA in blood sample. DNA is made up of 4 repeating nucleotide bases (A, T, C, and G) which are responsible for all diversity of life. In the Fig. 1(b), a nucleotide (A) of person2 with disease is different from that (C) of person1. The mapping between the two nucleotides (C/A) is called Single Nucleotide Polymorphism (SNP) or variant that is denoted with a unique RS number, RS ID (Reference SNP cluster ID).

C. Association Studies

Genome-Wide Association Study (GWAS) is the phenotype to genotype approach whereas Phenome-Wide Association Study (PheWAS) is a genotype-to-phenotype approach. Both strategies attempt to find associations between SNPs (variants) and disease where cases are individuals with a disease and controls are individuals without the disease according to EHR data. Through any one of a variety of statistical methods (e.g., chi-square, Fishers exact, logistic regression), a genetic variant (SNP) can be associated with a disease. P-values are often used to screen thousands to millions of association results to identify meaningful results. Hundreds of published GWASs are curated by GWAS Catalog [5] [13] and some PheWAS results have been curated in the GWAS Catalogue.

III. Design & Implementation

A. Dataset

The dataset for this paper is the summary data of biobank in MCRI. The biobank has been used extensively for discovery research including the electronic MEdical Records and GEnomics (eMERGE) network [7] and many other studies [8]. The dataset consists of genotyped DNA and EHR of 20,000 patients. The ages range from 18 to 98.5 years old, and the median of ages is 48. 57.2% are female and 98% are white Caucasian.

PheWAS and GWAS dataset contains CSV files of 22 chromosomes respectively. Table I displays column names, examples, and description of columns in the datasets. The PheWAS and GWAS datasets contain 191 and 21 million rows respectively. PheWAS dataset is queryable according to RS ID or genomic position of SNP. To search GWAS data, user searches by ICD-9 [2] disease code and disease description.

B. System Architecture

Fig. 2 illustrates system architecture of medical Big data analysis system that consists of front-end user interface and back-end server. Front-end user interface is developed with Shiny [12] on R to build up interactive Web UI. Back-end server consists of GlusterFS [24] network file system, Spark cluster for large-scale analytic data processing, Mongo Database to save massive results returned from Spark cluster for narrow search, and asynchronous Java daemon for processing in the background. SparkR [25] delivers requests and data between Shiny (front-end user interface) and Spark (back-end server).

Each node runs on Dell PowerEdge R710, 2U rack server, with 144 GB memory and 2 Intel Xeon 5660 processors each of which has 2.80 GHz and 12 CPU cores. In total, Spark cluster has 96 CPU cores and 576 GB memory. Each node has two 1 TB SSDs, resulting in 8 TB storage for Spark cluster. Ubuntu 18.04 is the operating system of each node. Four nodes are connected through a gigabit network. We use standalone cluster manager without using Mesos and Hadoop YARN.

A user types requests to find diseases relevant to genomic data using RS id/end positions or to find genomic data relevant to diseases using ICD-9 disease codes or description in Shiny

TABLE I
PHEWAS AND GWAS DATASET

PheWAS	**Column names**	chrom, pos, ref_wt, alt_wt, n_informative, af, stat, direction, pvalue, dx_code, dx_desc, dx
	Example	22,29854579,G,A,8613,0.19234,0.0065506,-,0.935493,dx903,Type I (Juvenile Type) Diabetes Mellitus With Ketoacidosis Uncontrolled,250.13
	Description	chromosome number, ending location of the variant, base nucleotide, variant nucleotide, the number of individuals for Fisher's exact test, Allele frequency, statistics estimated from statistical model, forward direction or reverse direction, P-value calculated by Fisher exact test, converted disease code, description of disease code, and actual code
GWAS	**Column names**	chrom, start_pos, end_pos, ref_wt, alt_wt, func_refgene, gene_refgene, genedetail_refgene, exonicfunc_refgene, aachange_refgene, x1000g2014oct_all, x1000g2014oct_eur, snp138, sift_score, sift_pred, polyphen2_hdiv_score, polyphen2_hdiv_pred, polyphen2_hvar_score, polyphen2_hvar_pred, lrt_score, lrt_pred, mutationtaster_score and pred, mutationassessor_score and pred, clinvar_20150330, cadd, cadd_phred
	Example	22,17265124,17265124,A,C,exonic,XKR3,NA,nonsynonymous SNV,XKR3:NM_175878:exon4:c.T765G:p.F255L, 0.694489,0.6282,rs5748623,1,T,0.0,B,0.0,B,0.001,N,1.000,P,-1.1,N,NA,,
	Description	columns 1-5: genomic positions in BED format [14], columns 6-10: annotation of variants to genes based on NCBI reference sequence collection [15] columns 11-12: frequencies of genetic variants within world populations based on 1000 Genomics project [16] column 13: RS id, columns 14-15: protein function by amino acid substitution [17] columns 16-19: Polyphen2 [18], columns 20-21: Likelihood Ratio Test Query [19], columns 22-23: mutation taster [20], columns 24-25: mutation assessor [21], column 26: CLINVAR [22] columns 27-28: Combined Annotation Dependent Depletion (CADD) [23]

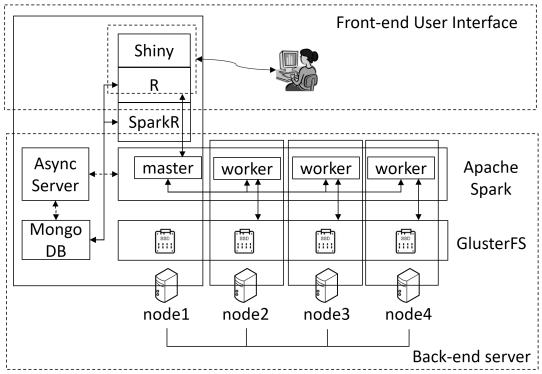


Fig. 2. System Architecture: worker in node1 is not shown for simplicity.

application. The requests are sent to master through SparkR with the type of Spark SQL [26]. Spark master distributes workloads for requests to executors (worker instances) that retrieve data from GlusterFS. Subsequently executors process the analysis, and results are sent back to Shiny application through master and SparkR. In case processing time is calculated to be long in Spark cluster, only partial results are returned to Shiny application, and the keys for further processing are temporarily saved in MongoDB. Asynchronous Java daemon server periodically checks and sends the outstanding requests to Spark cluster, save results into Mongo DB, and sends notification emails.

C. New Search: front-end

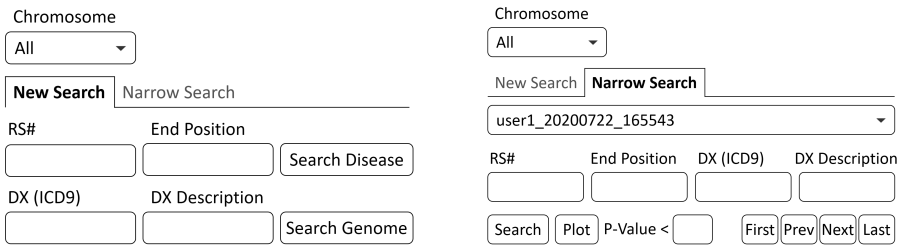
User interface consists of Shiny input widgets [27] (selectInput, textInput, and actionButton) and DataTable (DT) [28] output object (DT::dataTableOutput) as shown in Fig. 3(a). Inputs to find diseases are RS ids (e.g. rs116885116) or end positions (e.g. 20149361) along with a chromosome (or all chromosomes). Inputs to find genotypes are disease codes (e.g. 461) or description (e.g. Diabetes) along with a chromosome (or all chromosomes).

Algorithm 1 New Search: Disease

Input: chrom, {rs_id}, {end_pos}
Output: {disease} ▷ {disease}: data frame

```
1: phewas ← loadPheWAS(chrom)   ▷ load PheWAS data
2: gwas ← loadGWAS(chrom)   ▷ load GWAS data
3: listOfKeys{chrom_n,end_pos_n} ← getListOfKeys(chrom, {rs_id}, {end_pos})   ▷ using GWAS
4: if numberOfKeys > THRESHOLD then
5:                                   ▷ long processing time
6:     {disease} ← getDisease(...{chrom_t,end_pos_t})
7:                   ▷ using PheWAS, _t: threshold
8:
9:     Mongo.RequestDB ← ({chrom_t+1,end_pos_t+1}...)
10:    collectionName ← userId + timestamp
11:    Mongo.QueueDB ← (collectionName)
12:                          ▷ for back-end processing
13: else
14:    {disease} ← getDisease(...{chrom_n,end_pos_n})
15:  ▷ using PheWAS, _n: total # of {chrom,end_pos} pairs
16: end if
17: Mongo.DataDB ← {disease}   ▷ for narrow search later
18: return {disease}
19:        ▷ all column values of PheWAS shown in Table I
```

Algorithm 1 describes how system processes new search requests to find diseases. Given inputs (chromosome, list of RS ids and end positions), PheWAS and GWAS data for the chromosome are loaded from CSV files. Using GWAS, list



(a) New Search (b) Narrow Search

Fig. 3. Web UI: selectInput widget (dropdown button) is used for chromosome. textInput widget is used for RS ID, end position, dx, and dx description. actionButton widget is used for search disease and search genome buttons. DT::dataTableOutput object is used to show results in table format below input widgets. For narrow search, first, prev, next, and last buttons are used to load partial documents out of large number of documents.

of key pairs with a chromosome and an end position are retrieved. If the number of key pairs are less than threshold value set to 2000 which is configurable, we retrieve diseases (with all column values of PheWAS shown in Table I) with the type of data frame (similar to table format with rows and columns) by running Spark SQL [26] such as "select * from phewas where (chrom = 3 and pos in (18599078))". Having options to choose specific columns to be returned can reduce size of results, resulting in faster response time. This would be our future work. If the number of key pairs are larger than threshold value, it is expected to have long processing time which is longer than 3 to 100 seconds depending on the number of chromosomes. Thus, diseases using key pairs up to only threshold ($\{chrom_{2000}$,end_pos$_{2000}\}$) are processed. The remaining key pairs (starting from $\{chrom_{2001}$,end_pos$_{2001}\}$) are saved into request db in Mongo database while the collection name consisting of a user login id and timestamp is saved into queue db in Mongo database as a job key for back-end processing. The diseases are saved into data db in Mongo database for narrow search later. Mongolite R package [29] is used for transactions to Mongo database. Last, diseases are returned and shown through front-end UI with the table and plot formats.

The process of new search to find genotypes is similar except for using different inputs (disease codes or/and description along with chromosomes) and outputs (all column values of GWAS shown in Table I). Changes on line numbers are as follows: Line3: getListOfKeys(chrom,{dx},{dx_desc}), Line6,14: {genotypes} $\leftarrow$ getGenotypes({chrom,end_pos} pairs), Line17,18: Mongo.DataDB $\leftarrow$ {genotypes}, return {genotypes}. getListOfKyes() uses PheWAS, and getGenotypes() uses GWAS.

D. New Search: back-end

Outstanding requests to take long time are processed in the background after partial results are returned to front-end UI. A "Async server" daemon runs "spark-submit" command with an analysis application JAR file packaged by Maven and processes the outstanding requests periodically. The interval of running is configurable. In Spark cluster, the analysis application retrieves the list of key pairs of a chromosome and an end position from Mongo DB through Mongo DB Java driver, and processes the analysis in worker nodes. The data frame returned from workers are converted and saved into Mongo DB for narrow search later.

Analysis application sends an notification email every time each job is processed. We installed and configured Postfix as a send-only SMTP server. Codes sending emails run on JavaMail API and JavaBeans Activation Framwork (JAF).

E. Narrow Search

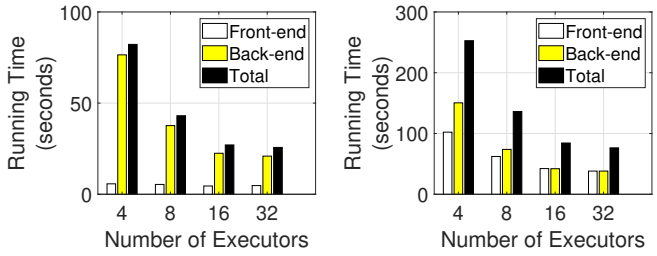
Narrow search shown in Fig.3(b) enables a user to find exact results by reusing search results saved in Mongo DB. Shiny application retrieves results from Mongo DB through mongolite R package [29]. The combination of user id and time stamp (e.g. user1_20200722_165543) is the collection name used as a key. A large number of result documents cannot be loaded at one time from MongoDB due to long waiting time and unwieldy user interface with a large number of pages (e.g. assume a collection with half million result documents). Thus, we retrieve partial documents using four buttons including first, prev, next, and last. The first and last buttons load the first block and the last block in a collection.

"plot" button is to visualize table format results using Manhattan plot [30]. Each point in the Manhattan plot represents a variant. With data of multiple chromosomes, the graph shows how much genomic data are strongly correlated with diseases using -$\log_{10}$(pvalue). We use manhattanly R package for the plot [31].To draw PheWAS data, we use scatter plot using plotly R graphing library which makes interactive graphs [32]

IV. Evaluation

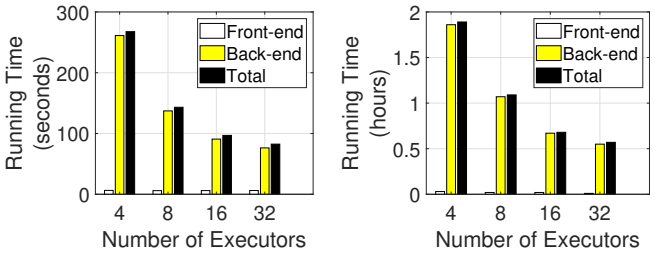
A. Performance

A new search consists of front-end operation that returns partial outputs through SparkR and back-end operation that processes the outstanding search in the background using spark-submit command. We measured the running time of front-end and back-end operations while varying the number of executors to 4, 8, 16, and 32. Each executor uses 2 CPU cores and 16 GB. Executors are equally distributed to four worker nodes. For example, for 32 executors, each worker node runs 8 executors with 16 cores and 128 GB, resulting in 64 CPU cores and 512 GB in total for processing a user request.



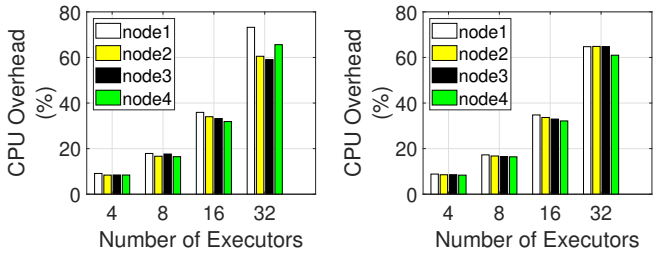
(a) chromosome 22 (b) All chromosomes

Fig. 4. Running time: search disease



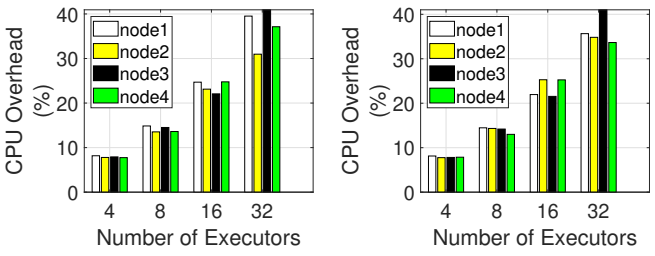
(a) chromosome 22 (b) All chromosomes

Fig. 5. Running time: search genome



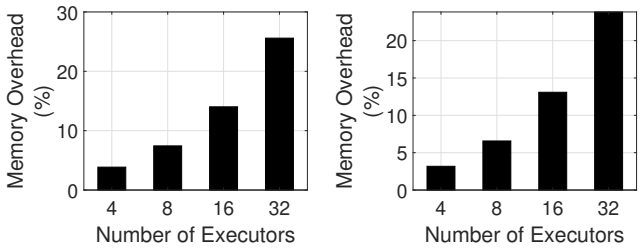
(a) chromosome 22 (b) All chromosomes

Fig. 6. CPU overhead: search disease



(a) chromosome 22 (b) All chromosomes

Fig. 7. CPU overhead: search genome



(a) search disease (b) search genome

Fig. 8. Memory overhead

Fig. 4 displays running time for searching disease with inputs of RS ids (rs116714698,rs116885116,rs17044614) and end positions (17040612,20149361,17044614) either in chromosome 22 or in all chromosomes. We averaged 5 times' running for the same request. 'Total' is the sum of front-end and back-end running time. Running time becomes faster with more executors due to the advantage of parallel processing. The running time with 32 executors is not decreased much compared to 16 executors. This indicates the existence of upper bounds of the number of executors depending on workloads, which we will explore deeply in the future work. The running time of front-end operation is much less than back-end processing considering fast-interactive response with small amount of results to user's request. For all chromosomes, it takes long time for front-end operation to return results. The percentage to separate workloads between front-end and back-end operations is configurable.

Fig. 5 show running time for searching genome data with inputs of DX codes (461,461.8) and DX description (Diabetes) either in chromosome 22 or in all chromosomes. Running time decreases with more executors. It takes much longer time to search genome data than searching disease as shown in Fig. 5(b). This is because the number of genome data associated with a disease is much more than the number of disease associated with a genome data. Narrow search is normal database operation on MongDB whose running time is not shown due to limited space.

B. Overhead

We measured CPU and memory overhead of four nodes in the Spark cluster using 'sar' command ('-u' and '-r' options for cpu and memory respectively) of sysstat package.

Figs. 6 and 7 shows additional CPU usage for searching disease and genome data. The same requests as those in performance section were used. The CPU usage per node increases with more executors, and the workload of each request is balanced to four nodes. The CPU overhead of node1 running a master additionally is a little bit larger than other nodes in general. Fig. 8 shows the memory overhead increases with more executors. We find CPU is underutilized for searching genome data as shown in Fig. 7: 66% (64 out of 96 cores) are allocated but 40% are used. For memory usage, 90% (512 out of 562 GB) are allocated but 25% are used. Considering that 32 executors are not much beneficial compared to 16 executors for the requests, dynamically changing configuration of the number of cores and memory size in an executor can increase performance while utilizing resources at maximum depending on the type of data set and the number of concurrent requests, which we will explore in the future work.

V. Related Work

[33] introduced using Spark cluster along with SparkR to increase better performance over message passing inter-

face (MPI) that is used by HPC clusters and packaged for most Linux distributions. Approaches for efficient dynamic resource allocation in Spark cluster were studied to reduce underutilization of resources and to maximize performance (e.g. decrease running time). Spark configuration is tuned dynamically using neural network-based search algorithm to find optimal configuration from prediction model [34]. [35] finds "diminishing return": considering Spark applications require many iterations, usage of computation resources diminishes as more iterations are completed. Thus, the number of allocated executors are reduced at underutilized worker nodes, and deallocated resources are used by other applications.

Shiny R package is used for developing interactive Web applications on R and integrating R functions for graphical and interactive analysis [36]. StructuRly [37] is a shiny application to produce interactive plots for population genetic analysis.

VI. Future Work and Conclusion

Medical big data analysis system is a prototype to check application design and system architecture with summary data of the biobank whose size amounts to 20PB. To handle full amount of data, large-scaled Spark cluster needs to be deployed with more worker nodes (presumably tens or hundreds of nodes) either in private or in Cloud. We run Spark on standalone cluster mode with fixed configuration. We are planning to exploit Spark configuration API in application, Apache Yarn, or Mesos for the dynamic resource allocation. Regarding the privacy issue of medical data, we envision a hybrid system where privacy-sensitive data are saved into private servers and summary data can be processed in large-scaled Cloud environment.

We introduced a medical Big data analysis system with front-end Web-based UI and back-end Spark/MongoDB servers, which can accelerate medical research on precision medicine by fast exploring correlation between diseases and genomic data. The design and system configuration would be helpful for researchers to deploy medical Big data analysis system.

VII. Acknowledgment

This research activity is funded by NIGMS grant R01GM097618 and in part by the Stanislaus State STEM Success program through a U.S. Department of Education Title III grant #P031C160070 and by the Stanislaus State McNair Scholars Program. We thank Deep Gill for helping system prepared.

References

[1] White House, "What is Precision Medicine," https://obamawhitehouse.archives.gov/precision-medicine, February 2016.
[2] National Center for Health Statistics (NCHS), "International Classification of Diseases, Ninth Revision (ICD-9)."
[3] Lander, E. S. et al, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, pp. 860–921, February 2001.
[4] Venter, J. C. et al, "The Sequence of the Human Genome," *Science*, vol. 291, pp. 1304–1351, February 2001.
[5] National Human Genome Research Institute, "NHGRI Catalog," https://www.ebi.ac.uk/gwas/, June 2020.
[6] J. C. Denny *et al.*, "PheWAS: demonstrating the feasibility of a phenome-wide scan to discover gene–disease associations," *Bioinformatics*, vol. 26, no. 9, pp. 1205–1210, March 2010.
[7] A. Verma *et al.*, "eMERGE Phenome-Wide Association Study(PheWAS) identifies clinical associationsand pleiotropy for stop-gain variants," *BMC Medical Genomics*, vol. 9(Suppl 1), no. 32, August 2016.
[8] Z. Ye *et al.*, "Phenome-wide association studies (PheWASs) for functional variants," *European journal of human genetics : EJHG*, vol. 23, July 2014.
[9] J. Denny, L. Bastarache, and D. Roden, "Phenome-Wide Association Studies as a Tool to Advance Precision Medicine," *Annual Review of Genomics and Human Genetics*, vol. 17, September 2016.
[10] W. S. Bush, M. Oetjens, and D. C. Crawford, "Unravelling the human genome–phenome relationship using phenome-wide association studies," *Nature Reviews Genetics*, vol. 17, February 2016.
[11] D. Joshua *et al.*, "Systematic comparison of phenome-wide association study of electronic medical record data and genome-wide association study data," *Nature biotechnology*, vol. 31, November 2013.
[12] RStudio, "Shiny," https://shiny.rstudio.com/.
[13] Annalisa Buniello et al, "The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019," *Nucleic Acids Research*, vol. 47, pp. D1005 – D1012.
[14] European Molecular Biology Laboratory's European Bioinformatics Institute (EMBI-EBI), "BED format."
[15] National Center for Biotechnology Information (NCBI), "NCBI reference sequence collection," https://www.ncbi.nlm.nih.gov/refseq/about/.
[16] European Molecular Biology Laboratory's European Bioinformatics Institute (EMBI-EBI), "The 1000 Genomes Project."
[17] P. Kumar, S. Henikoff, and P. C. Ng, "Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm," *Nature Protocols*, vol. 4, pp. 1073 – 1081.
[18] I. A. Adzhubei *et al.*, "A method and server for predicting damaging missense mutations," *Nature Methods*, vol. 7, pp. 248 – 249.
[19] S. Chun and J. C. Fay, "Identification of deleterious mutations within three human genomes," *Genome Research*, vol. 19, pp. 1553 – 1561.
[20] mutationstaster.org, "Mutation Taster," http://www.mutationtaster.org/info/documentation.html.
[21] mutationassessor.org, "Mutation Assessor," http://mutationassessor.org/r3/.
[22] NCBI, "ClinVar," https://www.ncbi.nlm.nih.gov/clinvar/.
[23] P. Rentzsch, D. Witten, G. M. Cooper, J. Shendure, and M. Kircher, "CADD: predicting the deleteriousness of variants throughout the human genome." *Nucleic Acids Research*, vol. 47.
[24] Red Hat Incorporation, "Gluster File System," https://www.gluster.org/.
[25] Apache Software Foundation, "SparkR (R on Spark)."
[26] Apache, "Spark SQL," https://spark.apache.org/docs/latest/sql-getting-started.html.
[27] RStudio, "Shiny Widget Gallery," https://shiny.rstudio.com/gallery/widget-gallery.html.
[28] RStudio , "DT: An R interface to the DataTables library," https://rstudio.github.io/DT/.
[29] Jeroen Ooms, " Mongolite," https://jeroen.github.io/mongolite/query-data.html.
[30] Stephen D. Turner, "qqman: an R package for visualizing GWAS results using Q-Q and manhattan plots," *biorXiv*, May 2014.
[31] Sahir Bhatnagar, "Interactive Q-Q and Manhattan Plots Using Plotly.js," http://sahirbhatnagar.com/manhattanly/, 2017.
[32] plotly.com, "Plotly R open source graphing library," https://plotly.com/r/.
[33] M. Sedlmayr *et al.*, "Optimizing R with SparkR on a commodity cluster for biomedical research," *Computer Methods and Programs in Biomedicine*, vol. 137, 2016.
[34] J. Gu, H. Tang, and Z. Wu, "Auto-Tuning Spark Configurations Based on Neural Network," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.
[35] D. Yang *et al.*, "Elastic executor provisioning for iterative workloads on apache spark," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 413–422.
[36] N. Hong, N. Prodduturi, C. Wang, and G. Jiang, "Shiny FHIR: An Integrated Framework Leveraging Shiny R and HL7 FHIR to Empower Standards-Based Clinical Data Applications," *Studies in health technology and informatics*, vol. 245, pp. 868–872, January 2017.
[37] N. G. Criscuolo and C. Angelini, "Structurly: A novel shiny app to produce comprehensive, detailed and interactive plots for population genetic analysis," *PLOS ONE*, vol. 15, 02 2020.