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

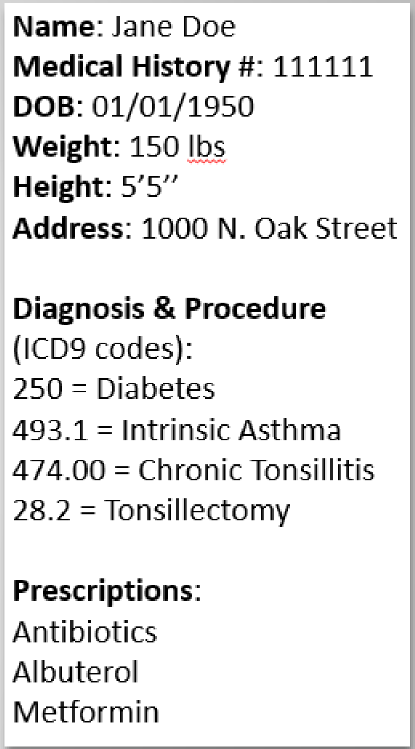
BY: DAEHEE KIM, STEPHANIE GAMBOA, VANESSA HERNANDEZ, **MARLEN MARTINEZ-LOPEZ**, SCOTT J. HEBBRING, JOHN MAYER & JAIME FOX

Table of Contents

- Background
- Motivations
- Dataset
- Architecture
    - System Architecture
    - Software Architecture
    - Control flow
- Development
    - New Search, Narrow Search
- Evaluation
    - Setup, Performance, Overhead
- Related Work
- Future work & Conclusion

Background

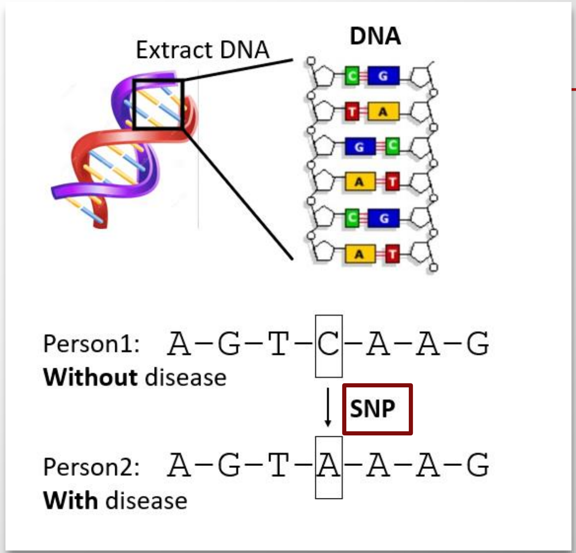
- Health Record Data
  - Used to record data on patients
  - Biological measurements
    - Disease Diagnoses
    - Medical procedures
- Genetic Data
  - Retrieved from DNA in blood samples



Name: Jane Doe
Medical History #: 111111
DOB: 01/01/1950
Weight: 150 lbs
Height: 5'5''
Address: 1000 N. Oak Street

Diagnosis & Procedure (ICD9 codes):
250 = Diabetes
493.1 = Intrinsic Asthma
474.00 = Chronic Tonsillitis
28.2 = Tonsillectomy

Prescriptions:
Antibiotics
Albuterol
Metformin

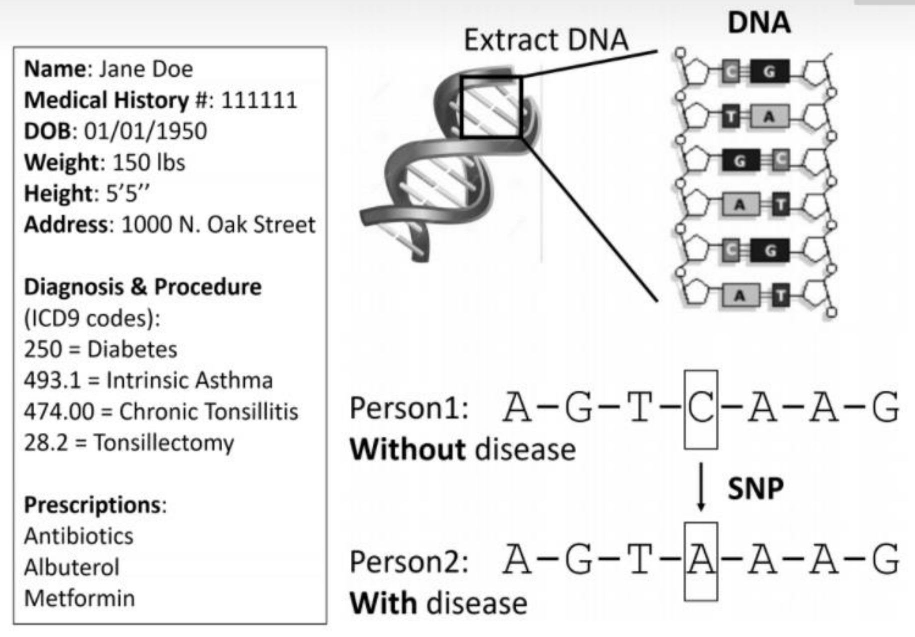


Background- cont'd

- Marshfield Clinic Research Institute (MCRI)
- Genome-Wide Association Studies (GWASs)
  - Find genetic variants for certain diseases
  - Phenotype-to-genotype approach
- Phenome-Wide Association Studies (PheWASs)
  - Explore multiple diseases relevant to genetic variant
  - Genotype-to-phenotype approach
- Electronic Health Records (EHRs) and DNA genotype are the main resources used to discover individual differences
- We designed and implemented a Medical Big Data analysis system that retrieves results from a GWAS-by-PheWAS dataset

Motivations

- Our motivation
  - Longer times to search results; We made this system
- Goal 1: Finds the link between genetic variants and human diseases
  - Aim to help medical professionals more with data analysis with their patients
- Goal 2: Implement a web query system that finds the links between human disease and genetic variants with PheWAS and GWAS
  - GWAS study scans markers across DNA or genomes of many people to find variations associated with a disease
  - PheWAS study explores multiple diseases relevant to a genetic variant



Name: Jane Doe
Medical History #: 111111
DOB: 01/01/1950
Weight: 150 lbs
Height: 5'5''
Address: 1000 N. Oak Street

Diagnosis & Procedure
(ICD9 codes):
250 = Diabetes
493.1 = Intrinsic Asthma
474.00 = Chronic Tonsillitis
28.2 = Tonsillectomy

Prescriptions:
Antibiotics
Albuterol
Metformin

(a) EHR (b) DNA Sequence

Fig. 1. EHR and Genetic Data

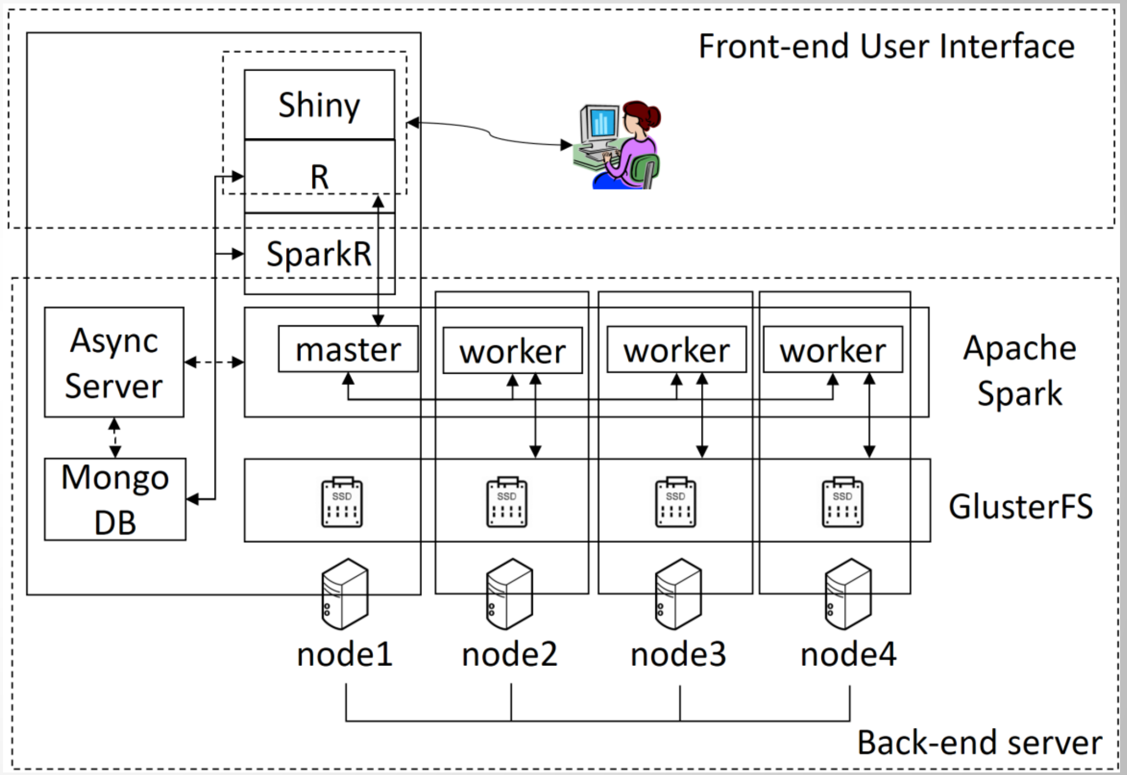
Dataset

- Data of biobank in Marshfield Clinic Research Institute (MCRI)
- Consists of genotype DNA and EHR of 20,000 patients
  - Age range 18 to 98.5
  - 57.2%
  - PheWAS dataset searchable by RS ID or genetic position of SNP
  - GWAS dataset searchable by ICD-9 disease code or description

PheWAS Example	**22,29854579**,G,A,8613,0.19234,0.0065506,-,0.935493,dx903, **Type 1 (Juvenile Type) Diabetes Mellitus With Ketoacidosis Uncontrolled**, **250.13**
GWAS 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,,

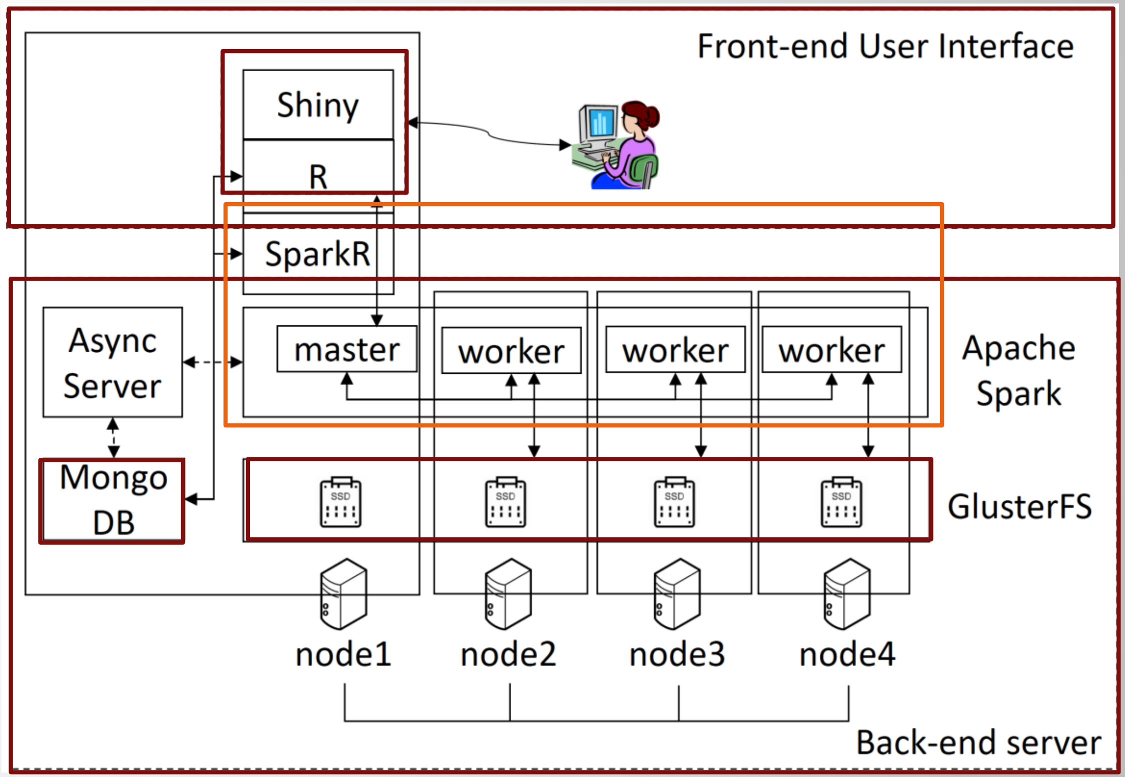
System Architecture

- Each node runs on
  - Dell PowerEdge R710
  - 2U rack sever (144GB)
  - 2 Intel Xeon 5660
- Each node has
  - 2 TB SSD
- 8 TB for Spark cluster
- Ubuntu 18.04
- Standalone cluster manager



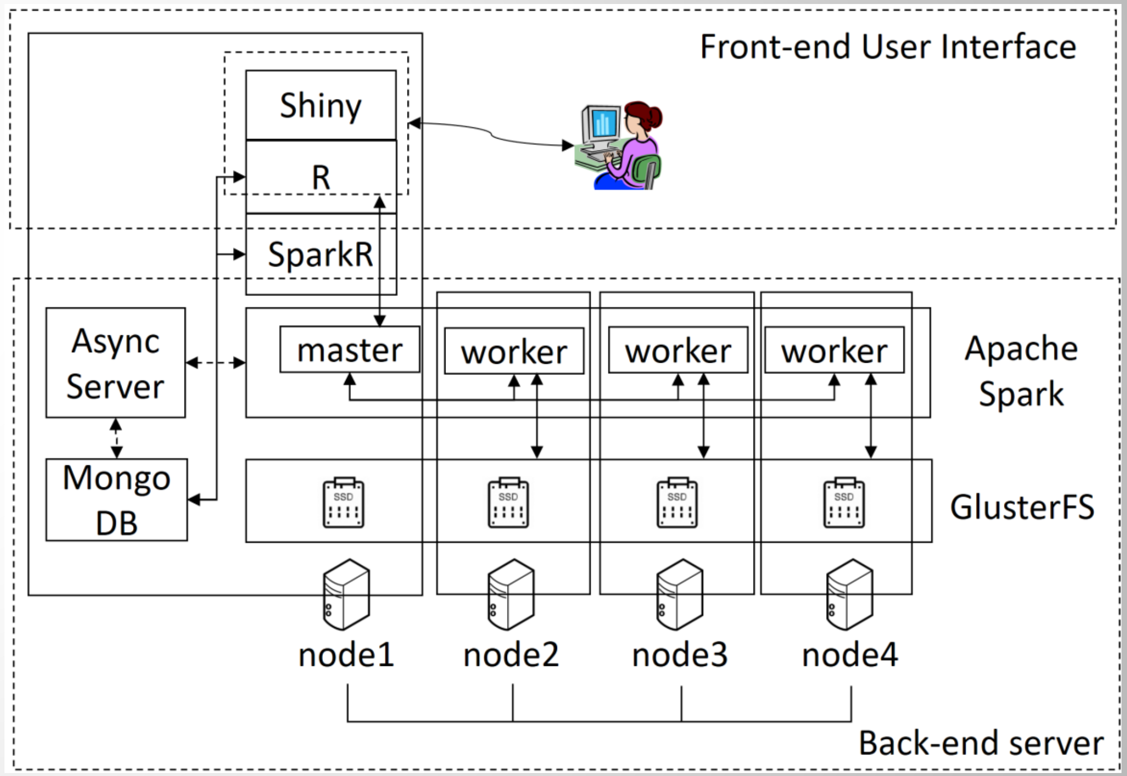
Software Architecture

- Web Query System architecture
  - Front-end user interface
    - R Shiny
  - Back-end server
    - GlusterFS
    - Spark
    - MongoDB
    - Java daemon



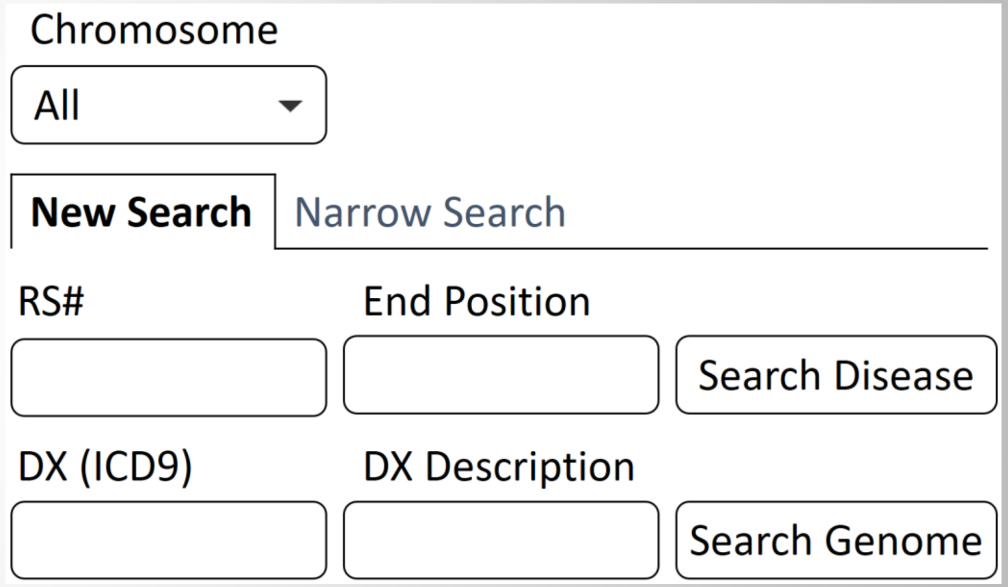
Control flow

- A user would type a request to find:
  - Diseases relevant to genomic data
  - Genomic data relevant to diseases
- Requests are sent to master through sparkR
- Partial results are sent to Shiny, the keys are saved temporarily in MongoDB and sends emails



New Search: Front-end

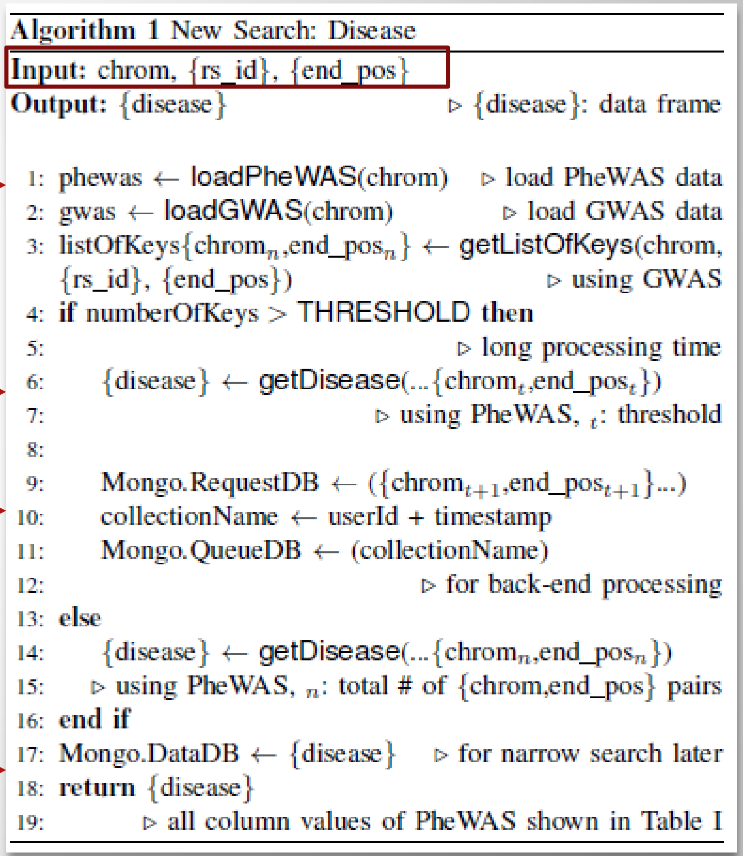
- New Search is used to find diseases
  - Inputs consist of RS# (location of the genome) and end positions
  - Inputs to find genotypes consist of disease codes or description
- UI consists of shiny widgets
  - selectInput (dropdown button)
  - textInput
  - actionButton
  - dataTable (showing results in table format)



New search user interface

New Search: Front-end - cont’d

- Algorithm 1
  - Processes new search requests to find diseases
- Given inputs (chromosomes, list of RS ids and end position) PheWAS and GWAS data for the chromosomes are loaded from CSV files



Algorithm 1 New Search: Disease

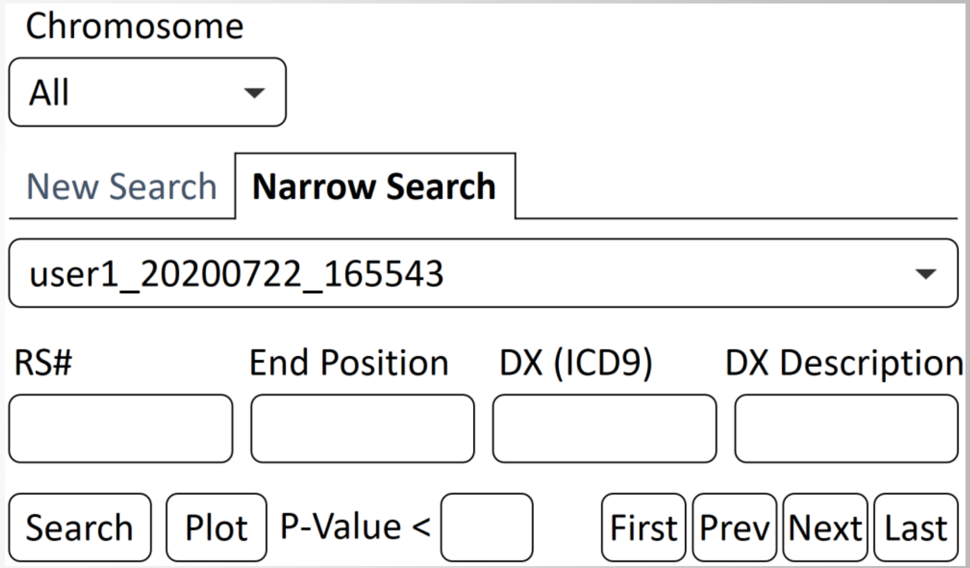
```
Input: chrom, {rs_id}, {end_pos}
Output: {disease}                         ▷ {disease}: data frame

phewas ← loadPheWAS(chrom)                ▷ load PheWAS data
gwas ← loadGWAS(chrom)                    ▷ load GWAS data
listOfKeys{chrom_n,end_pos_n} ← getListOfKeys(chrom,
{rs_id}, {end_pos})                       ▷ using GWAS
if numberOfKeys > THRESHOLD then
                                          ▷ long processing time
    {disease} ← getDisease(...{chrom_t,end_pos_t})
                                          ▷ using PheWAS, _t: threshold

    Mongo.RequestDB ← ({chrom_{t+1},end_pos_{t+1}}...)
    collectionName ← userId + timestamp
    Mongo.QueueDB ← (collectionName)
                                          ▷ for back-end processing
else
    {disease} ← getDisease(...{chrom_n,end_pos_n})
    ▷ using PheWAS, _n: total # of {chrom,end_pos} pairs
end if
Mongo.DataDB ← {disease}                  ▷ for narrow search later
return {disease}
                    ▷ all column values of PheWAS shown in Table I
```

New Search: Back-end

- Requests that take a long time are processed in the background after partial results are returned to front-end UI.
- Spark cluster
  - The analysis application retrieves the list of key pairs of a chromosome
  - An end position from Mongo DB through Mongo DB Java driver
    - Processes the analysis in worker nodes
- The data frame returned from workers are converted and saved into Mongo DB for narrow search later
- The analysis application sends a notification email every time each job is processed

Narrow Search

- 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
- Retrieve partial documents
  - First, prev, next and last
  - First and last buttons load the first and last block in a collection
- "plot" button used to visualize table formats using Manhattan plot
- To draw PheWAS data, we use scatter plot using plotly R graphing library which makes interactive graphs



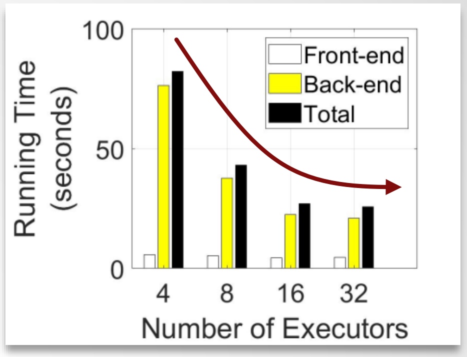
Narrow search user interface

Evaluation Set Up

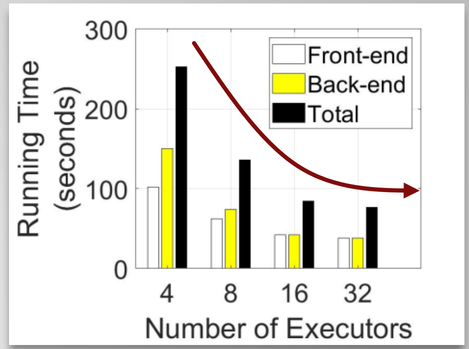
- SparkR (front-end), Spark-submit (back-end)
- Measured running time of:
  - Front-end & back-end operations
  - Averaged 5 times running the same request using ‘sar’ command
- Each executor: 2 CPU cores, 16 GB
- Varying the number of executors to 4, 8,16 and 32
- Equally distributed to four worker nodes
  - (e.g., 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)

Performance

- Running time for disease / genome data
  - Running time becomes faster with more executors on parallel processing
  - Running time of front-end is much less than back-end processing
  - Separating workloads between front-end and back-end is configurable
  - For all chromosomes, long time for front-end operation
  - Running time with 16 and 32 executors is similar, indicating the existence of upper bounds

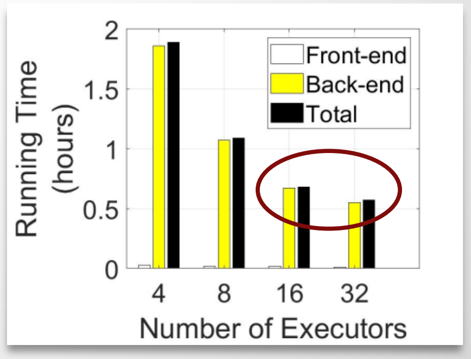


Chromosome 22

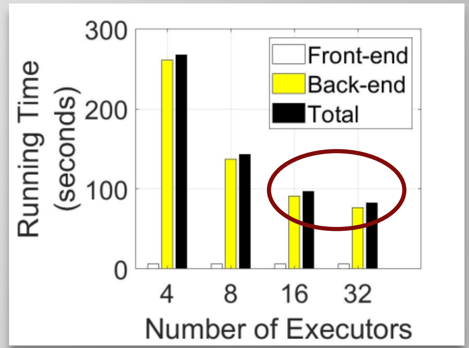


All chromosomes

Search disease



Chromosome 22

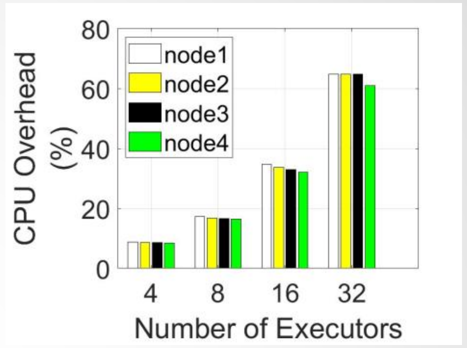


All chromosomes

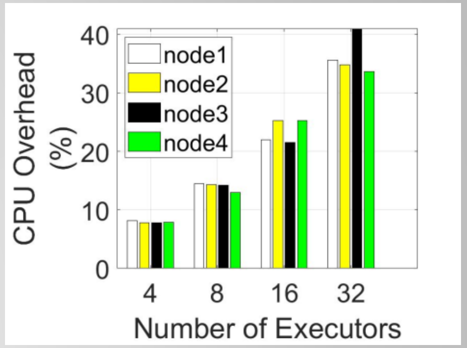
Search genome

Overhead

- Figures show additional CPU/Memory usage
- CPU overhead
  - CPU per node increases with more executors
  - Workload is balanced to four nodes
- Memory overhead
  - Memory usage increases with more executors
- Underutilized CPU/Memory
  - For genome, CPU used 40 % out of 66% allocated
  - Memory used 25 % out of 90% allocated
  - Dynamically changing # of cores and memory size in an executor can increase performance while utilizing resources at maximum

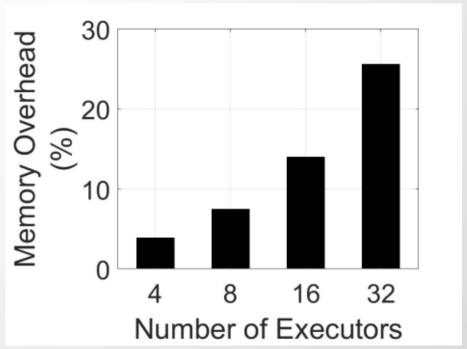


Search disease

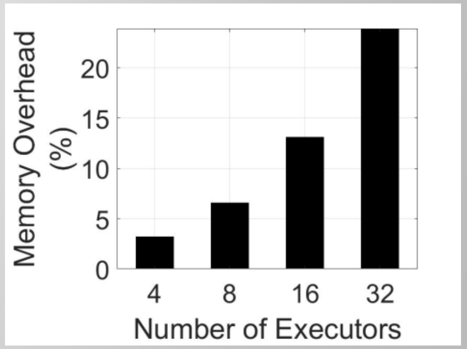


Search genome

CPU usage



Search disease



Search genome

Memory usage

Related Work

- Sedlmayr et al
  - Using Spark cluster along with SparkR increase better performance over message passing interface
- Hong et al
  - Shiny R package
  - Developing interactive Web applications in R
  - Graphical and interactive analysis
- Criscuolo & Angelini
  - StructuRly a shiny app: to produce interactive plots for population genetic analysis

Future Work & Conclusion

- Medical big data analysis system is a prototype
  - Check the application design and system architecture
- To handle more data
  - Large scale Spark cluster
  - More worker nodes
  - MCRI biobank: 20 petabytes
- Future
  - Dynamic resource allocation
  - A hybrid system

Thank you

Daehee Kim	dkim10@csustan.edu
Stephanie Gamboa	sgamoa@csustan.edu
Vanessa Hernandez	vhernandez27@csustan.edu
Marlen Martinez-Lopez	mmartinezlopez@csustan.edu
Scott J Hebbring	hebbring.scott@marshfieldresearch.org
John Mayer	mayer.john@marshfieldresearch.org
Jaime Fox	jaime.fox@preventiongenetics.com