Modelling Genomics Datasets to Optimize Its Usage Patterns for Efficient Analytics

Zeeshan Ali Shah
College of Engineering and Technology
University of Derby, Derby, United Kingdom
King Faisal Specialist Hospital and Research Centre, Riyadh, KSA
z.shah1@unimail.derby.ac.uk

Prof. Ashiq Anjum
College of Engineering and Technology
University of Derby
Derby, UK
a.anjum@derby.ac.uk

Mohamed Abouelhoda
Faculty of Engineering, Cairo University, Giza, Egypt
King Faisal Specialist Hospital and Research Centre, Riyadh, KSA
mabouelhoda@yahoo.com

ABSTRACT
Data intensive computing requires fast access to the input data. Currently in an HPC environment, the computation part is managed by the job scheduler, whereas data access is unmanaged and reply completely on the underlying file system. The job scheduler tracks system parameters such as RAM, CPU and utilize this for efficient computation. However, it is the responsibility of the developer to take care of the data related attributes and their traffic over the network. To solve this issue , we suggest data models for genomics data which can be used to guide the computation engine from data management perspective in addition to the usual system and computation parameters – We will demonstrated that enabling the job scheduler with info about the data is an important steps towards efficient fully automated data analysis.

CCS Concepts
• **Information systems → Data management systems → Information integration → Mediators and data integration.**

Keywords
data scheduler, data modelling, scheduling , meta-data systems

1. INTRODUCTION

1.1 Next Generation Sequences
Genomics based medicine, which is, usually referred to as personalized or precision medicine, became an integral component in the healthcare system [21]. This is due to recent advancements in next generation sequencing (NGS) technology. This technology can reduced the cost and time of reading the genome. NGS is currently used in the clinic to find variants (mutations) related to the disease to improve the diagnosis, prognosis, or to find optimized treatment plans.

The wide use of NGS in the clinic has introduced new computational challenges. The Genomics grade data analysis requires optimized algorithms to reach reliable results. To reach a list of variants with the necessary information for the clinical practice, a sophisticated computational workflow of many software tools should be used. The number of these reads depends on the technology and the model of the NGS instrument. For Ion technology, one expects around 80 million reads per run. Processing such huge number of reads entails huge I/O operations, especially when a workflow of multiple independent programs is used.

ICAAI 2019, October 26-28, 2019, Istanbul, Turkey.
 ISBN 978-1-4503-7253-4
DOI: 10.1145/3369114.3369148

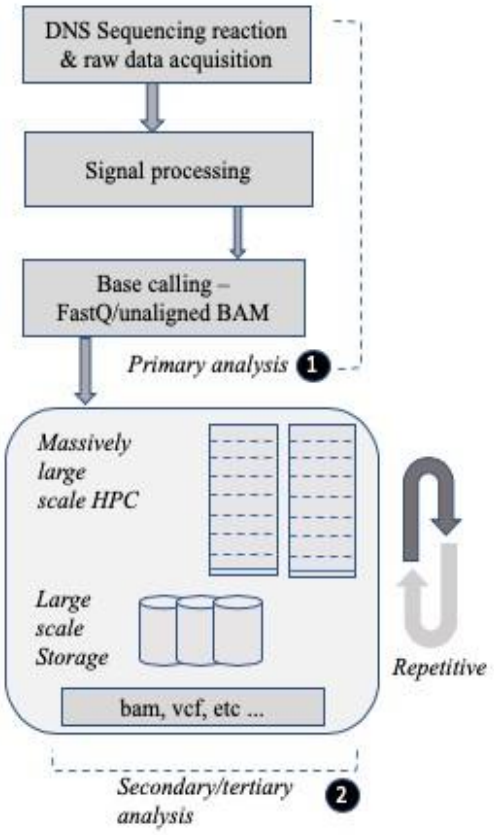

Figure 1: NGS Data analysis pipeline

Figure 1 describes the steps involved in NGS and consists on following steps:

1. The Primary analysis which mentioned in Step-1 consists on platform dependent procedures and occurred in the wet lab via

sequence instruments. The result produced in Step-1 is usually a FastQ file which process in subsequent step.

2. The Step-2 which is also called Secondary analysis requires high performance cluster and often accumulate parallel and distributed computing technologies, it also leverages massive large scale storage as show in (2) Figure 1 .

The secondary analysis is repetitive, which means various researchers and scientist use the same data-sets with different analytical[19] techniques and execute different workflows over it. For this reason large scale storage are required including some sort of cluster management system which should allow parallel execution of different tasks/jobs.

Our research problem deal with this secondary analysis step and tried to answer different challenges involved around it to optimize the performance and complexity.

1.2 Current HPC Cluster and How It Works

HPC stands for high performance computing it enables this via use of multiple physical compute nodes[20] inter-connected through shared network and access to share file system . Job scheduler as mentioned in figure 2 manages compute nodes with their parameters and attributes such as available RAM, storage, Network and Current CPU Load. These attributes taken into consideration with scheduler for selecting appropriate compute nodes for execution job.

Often these attributes are dynamic such as available RAM or CPU , which requires job schedule to continuously monitor compute nodes and store these values in database for further query and job schedule algorithm .

1. From Step-1 in Figure 2 users prepare the jobs as executable templates which than submitted to job scheduler

2. in step-2 . The job scheduler using the scheduling algorithm assign the suitable compute node for the job and further submit them in step-3 .

For data I/O the HPC Cluster [18] requires a shared network file-system which could be like NFS, Lustre, GPFS etc [19[]]. This file-system shared among all compute nodes and store the data require as input to job and data produced from job as output.

Few of the scheduler also benefit from local disk associated with the compute node but the job in this case do-not start until the data loaded into local disk completely , this delays the execution and overall performance .

Furthermore, as the job scheduler manages compute nodes and execution jobs, the data associated with it are adhoc and unmanaged. Following figure 3 depicts the data adhoc movement which is the pain area[1] for HPC cluster performance and which this paper tried to investigate.

As mentioned in figure 3 the data moved without preplanned routes or methods, often same data copied twice between different compute nodes and central storage since another job needs it, this cause network congestion and delays.

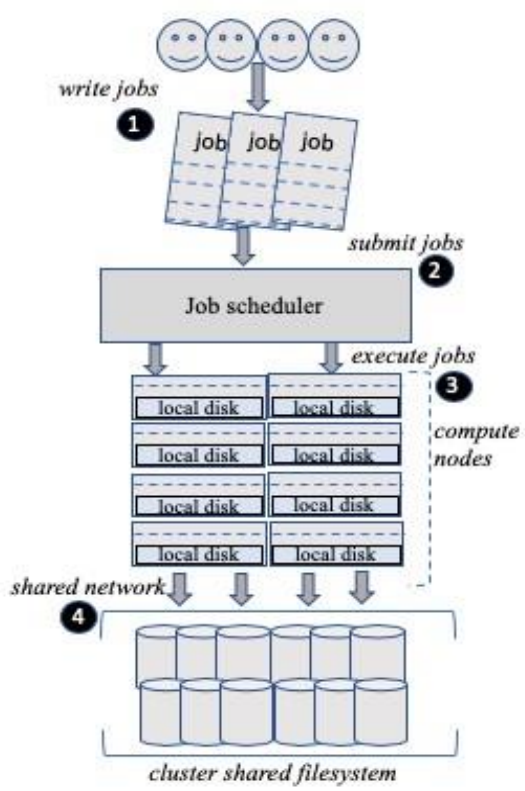

Figure 2: Traditional HPC scheduler

Question arises why the data movement cannot managed and preplanned ? The investigation of one aspect will be discussed in this paper.

2. PROBLEM

2.1 Data as black-box, The *blind-sight* for Scheduler

Computational jobs link the data with its location that is mostly contains folder/filenames . For job scheduler perspective the data appears as a black-box, It has no knowledge about its contents neither about any meta data associated with it. Below is the snippet of an execution job template:

```
1  #! /usr/bin/python
2  import sys, Common
3  flist=open(sys.argv[1]).readlines()
4  for file in flist:
5    file=file.replace("./","/.bam")
6          cmd="perl /filtrator/filter_generic
       .pl
7          -vcf %s.ext2.csv.hds.csv -kg 0.001
       -walsh 0.001 >
8          %s.ext2.csv.hds.csv_filteredM2.csv"
       %(file.strip(),file.strip())
9          print cmd
10         Common.run(cmd)
```

Listing 1: Sample job

In above code listing 1 at *line (4)* the data is mentioned as file location. The Job itself does not know the data attributes which appears as black box to the execution tasks.

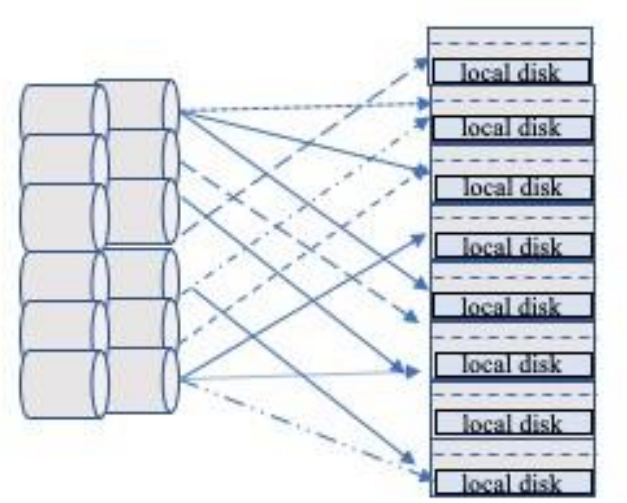

Figure 3: Adhoc Data movement between compute nodes and central storage

Furthermore, even the basic attributes of data such as its *size* is not considered although it is already stored in operating system. This is why we mentioned that data appears to the scheduler as black box[17], means it has no knowledge about it except location .

Like the saying goes , if you cannot measure , you cant improve , we goes one step backwards and say that if you don't model something you would not know it and in result could not measure and improve it.

In addition, if the same data mentioned in other piece of code the tasks are not able to co-relate them , for them it appear to be just another data to be processed . Neither the scheduler (see above section) knows that it is same data used earlier by another job. We named this *blind-sight* problem.

2.2 Random Data Movement

Chaos is the absence of management and this occurs because the data attributes are neither monitored , nor stored and not considered . Data moves haphazardly without planned routes , and even if same data mentioned X times by Job it will treated as fresh request X time, this cause network congestion and increase of IO load with latency to storage servers. Furthermore, in environments which are time critical and also data intensive, that data congestion cost performance delays.

3. PROPOSED MODEL

We proposed to model the data movement [2] but before that we have to adapt the mathematical notations of data itself. The data attributes are depend on the research domain for e.g. in Genomics next generation sequence domains we concluded the data model mentioned in Table: 1.

In terms of graphs it can be represented as in figure 4:

Some of the data attributes from Table : 1 are static such as *guid, name, PI and Location* and other are *dynamic like Temperature and State* . The static data attributes require to be filled before actual execution start and the dynamic ones should be updated on run time. Which means that data scheduler is the integral part of whole execution cycle[11]. As described in previous section that without storing the data attributes in its model the execution engine relies blindly on underlying file system . To improve this situation the data scheduler works side by side with computational scheduler and often governed it to select the best data route and to get the optimized computational platform [4].

The data scheduler works in two modes: Initially it works background for modelling existing data set with static attributes and in second mode which is more active and foreground is to model the data as it passes through the execution cycle.

For active mode it has to be integrate with existing compute scheduler and optimize its scheduling algorithms for best route to data and selection of optimum compute platform.

As we stated previously this optimum data movement all depends on data model and its characterization , without doing so would lead scheduling in an adhoc and random data patterns.

Table 1: Data model for NGS

Guid	unique id
Name	Free textual name
Location	Path to data
PI	Principal investigator name
Project	Projects it is belong to
Size	Size in Mega Byte (MB)
Temperature	1-60 fast-medium based on usage
State	In fast storage/hot medium
Created at	Time when it was created
Last accessed	Time when last used
Last edited	Time when last edited
Jobs list	List of Jobs it is used for
Related with	Any relation with other data

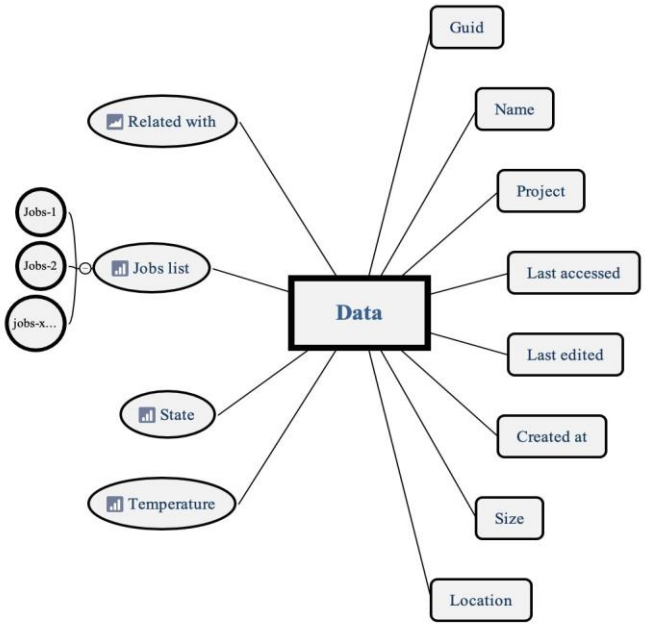

Figure 4: Data model in graph

4. GENERATION AND EXPERIMENT

From 1 some of these attributes are static and some are dynamic, The static attributes such as Guid , Location, PI , Project are set when the data is actually created , further more some of those attributes can be set automatically like created at, Last accessed, Last edited or storage system can be configured to do this task , rest of attributes are dynamic[12] such as Jobs List, Related with, access read, storage type and we need to investigate how to populate them on runtime [3][22].

Below tables depicted these types :

Table 2: Data model's attribute Types

Name	Type	How to generate
Guid	static	creation time by storage system
Name	static	set by LIMS
Location	static	storage system on creation time
PI	static	set by LIMS
Project	static	set by LIMS
Size	static	storage system on creation time
Temperature	dynamic	need to investigate
State	dynamic	need to investigate
Created	static	storage system on creation time
Last accessed	dynamic	storage system
Last edited	dynamic	storage system
Jobs list	dynamic	need to investigate
Related with	dynamic	need to investigate

Those mentioned as *italics* in *Table:2* are out of scope of this paper and will be addressed in future , we did the experiment to generate rest of model and following pseudocode used to generate that.

Data: folder/project/patient/data **Result:** How to generate attribute of data initialization;
while *files exist in Data folder* **do**
read current; **if** *file present* **then**
size = sizeof (current); name = filename; project= getproject (current); location = getpath (current); PI = principle-investigator (current); created = creation-data (current);
else
store attributes of current in database;
go to another folder; **end**
end

Algorithm 1: Pseudo code for generating model of data

4.1 Using iRods for Storing Data Model

iRods [7] and [8] stands for integrated rules oriented data systems. It is software middle-ware that manages a highly controlled collection of distributed[5] digital objects, while enforcing user-defined Management Policies across multiple storage locations. iRODS system is generic software infrastructure that can be tuned

to implement any desired data management application, ranging from a Data Grid for sharing data across collaborations, to a digital library for publishing data, to a preservation environment for long-term data retention, to a data processing pipeline, to a system for federating real-time sensor data streams.

In this paper we used iRods for storing data model generated from Algorithm 1 and store them as meta-data attributes. [21described that how iRODS[6] has been implemented and works as the production system for the sequencing pipeline of the Welcome Trust Sanger Institute. Other studies [Hedges et al. 2007] found that how iRODS can be used to preservation of research data. This is quite interesting to us as well since the modelling of data also enables and assist in proactive preservation for disaster recovery situations. Not only life sciences but NASA Center for Climate[13] Simulation are using the Integrated RuleOriented Data System (iRODS) to combine disparate data collections into a federated platform upon which various data services can be implemented [14].

Modelling of data[10] is also the pivot stone for Machine learning based Artificial intelligence which can further enhanced the applications for automated behaviour. [16] showed how this can be achieved with using meta-data.

Another life science domain such as Brain imaging also benefited use of data modelling with iRODS since the Brain image datasets pose a problem for data storage, access, and analysis due to their large and complex structure. To manage metadata associated with brain image collections iRODS proven to be stable and assisted in large scale analysis [Deitrich et al. 2018].

5. RESULTS

With above modeling parameters we were able to fix the black box and now know the data modelling. this is an initial step to improve the data movement pattern. Generating from above algorithm 1 and storing them in [15] enabled the removal of chaos in data movement. Following experiments are done on scheduler side to investigate if the meta-data inserted are visible from scheduler level.

Earlier the data model was unknown which caused the *blind-sight* problem to schedule and resulted a chaos and delay in data movement.

6. CONCLUSION AND FUTURE WORK

We have introduced a data model to be used by the computation engine for optimizing the analysis of genomics data. Our data model takes different data attributes into account, including size, location in the system, storage type and IO speed, among others.

This meta data enables the engine to perform extra optimization beyond usage of CPU and RAM[24]. We have shown that iRods can be used to maintain the meta data and facilitate access to the data, especially in a system with distributed and heterogeneous storage units.

Our data model avoid random access to the data, reduce network congestion, and lead to better execution time.

7. ACKNOWLEDGMENT

We are thankful specially to King Faisal Specialist hospital and Research centre, Riyadh with Saudi Genome program and King abdulaziz city of science and technology to provide enough resource in conducting this research work.

8. REFERENCES

[1] [Anjum et al. 2006] Ashiq Anjum, Richard McClatchey, Arshad Ali, and Ian Willers. 2006. Bulk scheduling with the DIANA scheduler. *IEEE Transactions on Nuclear Science* 53, 6 (2006), 3818–3829.

[2] Charlie Baker, Ashiq Anjum, Richard Hill, Nik Bessis, and Saad Liaquat Kiani. 2012. Improving cloud datacentre scalability, agility and performance using OpenFlow. In *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 20–27.

[3] Gen Tao Chiang, Peter Clapham, Guoying Qi, Kevin Sale, and Guy Coates. 2011. Implementing a genomic data management system using iRODS in the Wellcome Trust Sanger Institute. *BMC Bioinformatics* 12, July 2010 (2011). https://doi.org/10.1186/1471-2105-12-361

[4] Sean Deitrich, Jacob CzechAlexander Ropelewski, Arthur W. Wetzel, Greg Hood, Derek Simmel, Marcel Bruchez, Simon C. Watkins, and Alan M. Watson. 2018. Applying iRODS to the Brain Image Library. *Proceedings of the Practice and Experience on Advanced Research Computing PEARC '18* (2018), 1–4.

https://doi.org/10.1145/3219104.3229266

[5] Irfan Habib, Ashiq Anjum, Richard Mcclatchey, and Omer Rana. 2013. Adapting scientific workflow structures using multi-objective optimization strategies. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 8, 1 (2013), 4.

[6] Khawar Hasham, Antonio Delgado Peris, Ashiq Anjum, Dave Evans, Stephen Gowdy, Jos ́e M Hernandez, Eduardo Huedo, Dirk Hufnagel, Frank van Lingen, Richard McClatchey, et al. 2011. CMS workflow execution using intelligent job scheduling and data access strategies. *IEEE Transactions on Nuclear Science* 58, 3 (2011), 1221–1232.

[7] Mark Hedges, Adil Hasan, and Tobias Blanke. 2007. Management and preservation of research data with iRODS. *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience - CIMS '07* (2007), 17.

https://doi.org/10.1145/1317353.1317358

[8] Amanda Jones, Marzieh Bazrafshan, Fernando Delgado, Tania Lihatsh, and Tamara Schuyler. 2014. The Role of Metadata in Machine Learning for Technology Assisted Review. 2013 (2014), 1–12.

[9] Gaurav Kaul, Zeeshan Ali Shah, and Mohamed Abouelhoda. 2017. A High Performance Storage Appliance for Genomic Data. In *Bioinformatics and Biomedical Engineering*, Ignacio Rojas and Francisco Ortun˜o (Eds.). Springer International Publishing, Cham, 480–488.

[10] Saad Liaquat Kiani, Ashiq Anjum, Michael Knappmeyer, Nik Bessis, and Nikolaos Antonopoulos. 2013. Federated broker system for pervasive context provisioning. *Journal of Systems and Software* 86, 4 (2013), 1107–1123.

[11] Yang Liu, Hung Wei Tseng, Mark Gahagan, Jing Li, Yanqin Jin, and Steven Swanson. 2016. Hippogriff: Efficiently moving data in heterogeneous computing systems. *Proceedings of the 34th IEEE International Conference on Computer Design, ICCD 2016* (2016), 376–379. https://doi.org/10.1109/ICCD.2016.7753307

[12] Richard McClatchey, Andrew Branson, Ashiq Anjum, Peter Bloodsworth, Irfan Habib, Kamran Munir, Jetendr Shamdasani, Kamran Soomro, neuGRID Consortium, et al. 2013a. Providing traceability for neuroimaging analyses. *International journal of medical informatics* 82, 9 (2013), 882–894.

[13] Richard McClatchey, Irfan Habib, Ashiq Anjum, Kamran Munir, Andrew Branson, Peter Bloodsworth, Saad Liaquat Kiani, neuGRID Consortium, et al. 2013b. Intelligent grid enabled services for neuroimaging analysis. *Neurocomputing* 122 (2013), 88–99.

[14] Ioan Raicu, Yong Zhao, Ian T. Foster, and Alex Szalay. 2008. Accelerating large-scale data exploration through data diffusion. (2008), 9–18.

https://doi.org/10.1145/1383519.1383521

[15] Arcot Rajasekar, Jonathan

Crabtree, Tom Carsey, Hye-Chung Kum, Sharlini Sankaran, Gary King, Merce Crosas, Howard Lander, and Justin Zhan. 2013. The DataBridge. (2013), 1–14.

[16] Arcot Rajasekar, Reagan Moore, Chien-Yi Hou, Christopher A. Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, Paul Tooby, and Bing Zhu. 2010. *iRODS Primer: Integrated Rule-Oriented Data System*. Vol. 2. 1–143 pages. https:

//doi.org/10.2200/S00233ED1V01Y200912ICR012

[17] [n. d.]] Integrated Rule-oriented. [n. d.]. iRODS Primer 2. ([n. d.]).

[18] John L. Schnase, William P. Webster, Lynn A. Parnell, and Daniel Q. Duffy. 2011. The NASA center for climate simulation data management system. *IEEE Symposium on Mass Storage Systems and Technologies* (2011), 1–6.

https://doi.org/10.1109/MSST.2011.5937235

[19] Zeeshan Ali Shah, Mohamed El-Kalioby, Tariq Faquih, Moustafa Shokrof, Shazia Subhani, Yasser Alnakhli, Hussain Aljafar, Ashiq Anjum, and Mohamed Abouelhoda. 2018. Exploiting In-memory Systems for Genomic Data Analysis. In *Bioinformatics and Biomedical Engineering*, Ignacio Rojas and Francisco Ortun˜o (Eds.). Springer International Publishing, Cham, 405–414.

[20] Wenzhong Shi, Kawai Kwan, Geoffrey Shea, and Jiannong Cao. 2009. A dynamic data model for mobile GIS. *Computers and Geosciences* 35, 11 (2009), 2210–2221.

https://doi.org/10.1016/j.cageo.2009.03.002

[21] Frank van Lingen, Conrad Steenberg, Michael Thomas, Ashiq Anjum, Tahir Azim, Faisal Khan, Harvey Newman, Arshad Ali, Julian Bunn, and Iosif Legrand. 2005. The Clarens Web service framework for distributed scientific analysis in grid projects. In *2005 International Conference*

on Parallel Processing Workshops (ICPPW'05). IEEE, 45–52.

[22] Frank Van Lingen, M Thomas, T Azim, I Chitnis, A Anjum, D Bourilkov, M Kulkarni, C Steenberg, RJ Cavanaugh, J Bunn, et al. 2005. Grid enabled analysis: architecture, prototype and status. (2005).

[23] Muhammad Usman Yaseen, Ashiq Anjum, Omer Rana, and Richard Hill. 2018. Cloud-based scalable object detection and classification in video streams. *Future Generation Computer Systems* 80 (2018), 286–298.

[24] Ali Reza Zamani, Mengsong Zou, Javier Diaz-Montes, Ioan Petri, Omer Rana, Ashiq Anjum, and Manish Parashar. 2017. Deadline constrained video analysis via in-transit computational environments. *IEEE Transactions on Services Computing* (2017).