Workflow Provenance for Big Data: From Modelling to Reporting

Rayhan Ferdous, Banani Roy, Chanchal K. Roy, Kevin A. Schneider

Department of Computer Science, University of Saskatchewan
{rayhan.ferdous, banani.roy, chanchal.roy, kevin.schneider}@usask.ca

Abstract. Scientific Workflow Management System (SWFMS) is one of the inherent parts of Big Data analytics systems. Analyses in such data-intensive research using workflows are very costly. SWFMSs or workflows, keep track of every bit of executions through logs, which later could be used on demand. For example, in the case of errors, security breaches or even any conditions, we may need to trace back to the previous steps or look at the intermediate data elements. Such fashion of logging is known as workflow provenance. However, prominent workflows being domain specific and developed following different programming paradigms, their architectures, logging mechanisms, information in the logs, provenance queries and so on differ significantly. So, provenance technology of one workflow from a certain domain is not easily applicable in another domain. Facing the lack of a general workflow provenance standard, we propose a programming model for automated workflow logging. The programming model is easy to implement and easily configurable by domain experts independent of workflow users. We implement our workflow programming model on Bioinformatics research- for evaluation and collect workflow logs from various scientific pipelines' executions. Then we focus on some fundamental provenance questions inspired by recent literature that can derive many other complex provenance questions. Finally, the end users are provided with discovered insights from the workflow provenance through online data visualization as a separate web service.

Keywords: Scientific Workflow Management Systems, Workflow Provenance, Logs, Modular programming, Provenance Programming Model

1 Introduction

In Big Data analytics, the whole analysis process always go through Big Data analytics lifecycle [12], [9]. In each step of the lifecycle, a specific fashion of analysis is projected on the data which is/are derived from the previous step/s. Such fashion of analysis creates the necessity of using common data analysis tools over the whole process. Thus re-usability of data analysis tools becomes important. On the other hand, distributed and high performance computing technologies are necessary in data intensive research areas. They are provided through web services enabling collaboration. Better scientific workflow systems

or simply workflows leverage all these features and provide data scientists with cutting edge and most updated data analytic services- to implement data intensive pipelines [6], [18], [25], [19].

Big Data analytics deals with unstructured data and has to produce comprehensive information about different datasets from different sources. Such comprehensive information is more valuable than information of any single dataset [9]. Data provenance is the way of associating subject data with related log data during an analysis process. Such logs can be used to regenerate data lineage. Lineage of data is necessary to answer any question associated with the data source, configuration related to its analysis, any error or anomaly, its changes over time, process based information related to the data and so on. Data provenance itself has many different categories [7], [14], [8], [23], [11]. Any workflow itself, processes data through simple to complex pipelines. Such workflow process logs are called workflow provenance. In workflow provenance, the pipeline modules are considered as black boxes and internal operations are not taken into account. Data elements are both input and output of any module [2]. In case of any errors, unwanted behaviors of the analysis/security problems, or even to investigate particular data/process condition- the workflow cannot be just shut down/restarted. Data intensive task management does not allow such operations because only a single analysis run may need significant amount of time, memory and processing power to finish. Thus, tracing back the problem or condition using the logs becomes necessary. Backtracking dataflow events and situations also become important for further deeper investigation [2], [13]. A workflow system offering workflow provenance enables such investigations. For example, the provenance information could be used to answer the questions- ***'What particular parameter setting is responsible for a target error?', 'Which type of data element is most used for a particular module?', 'What is the source dataset of an intermediate data element?'*** and so on. So clearly, provenance analysis is the further analysis of data product that is derived from previous analyses and workflow runs. Big Data sources generating such provenance data, create variety (unstructured data) in those provenance data products. Thus workflow provenance itself becomes another Big Data problem.

Workflow systems are provided with a highly/loosely coupled provenance services [11]. Their logging mechanisms also vary significantly because of domain differences [11], [23]. In this era of Big Data analytics, we have already entered such a period where cross domain research and collaboration is taking place among researchers. Consequently, scientific systems like workflows also need to feature cross domain facilities and collaboration. The problem is, one workflow system which is best for a certain domain is not best for another. As already mentioned, different workflow systems' architectures differ significantly on different aspects. So, fusing any two workflows into one with their provenance features is not very easy. We focus only on workflow provenance and face the lack of a general and standard model for answering provenance questions. Working with any workflow also requires minimum amount of domain expertise. So, it is also necessary to separate all the concerns as much as possible while developing such

a standard workflow model. For example, provenance logging configuration can be handled by data scientists and only workflow implementation can be handled by developers.

In order to overcome the above problems, in this paper, we propose a programming model for workflow provenance. We build the programming model solely based on Object Oriented Programming model and inspired by existing novel workflow models [2], [1]. Thus it is easy to implement with less learning curve by any developer with minimum programming experience. We define different workflow components (e.g. Data, Module, Condition and Dataflow) to offer various workflow features such as, concise way of building pipelines, incorporating conditionals and so on. The model is designed in a manner that enables automated logging of workflow provenance data [5]. The logging mechanism is not only fused with the programming model, but the log structure is also easily configurable by a domain expert without modifying the model. So, while implementing any pipeline with the model, there is no burden of log management. This is how we separate the concerns of logging and workflow development in our approach. Besides, necessary data analysis tools can be easily and independently implemented with our model by any developer of a certain domain. They can later be used just as a tool. In summary, our model offers and makes everyone related to a workflow system to work in a systematic procedure that makes use of the SoC (Separation of Concerns) design principle [17]. Furthermore, it can be extended to scalability for Big Data analytics with prominent technologies (e.g. Hadoop [22], Apache Spark [26] and so on). Any Domain Specific Language (DSL) can be used on top of our programming model layer to facilitate further domain specific features. Then all the features of our programming model will be carried to that DSL automatically. The whole idea is illustrated as a layer based architecture in Figure 1. Here, the OOP layer is the base layer of the whole system that can be further extended with any technology from the Extension. The programming model is built on the OOP model. Logging Configuration is in the same layer of the modelling but also independent from the proposed model. Any DSL could be built on the proposed model. Various tools could be presented at the top Tool layer. The workflow user directly uses the tools and model developer directly uses the OOP model to develop/extend the proposed programming model. Domain expert can independently configure logging configuration independent of any model developer or user.

Finally, to evaluate our provenance model, we will develop tools and pipelines from Bioinformatics. We gather a bunch of log data through simulations to use our proposed model- to implement workflow tools and pipelines. We also focus on the logs to discover important insights related to provenance. So, we target a number of provenance questions [3], [20] and analyze how much we can answer those questions from the logs that are further described in Section 2.3. A future work is to find out all the elementary provenance questions. By elementary provenance questions we indicate such provenance questions that are atomic/fundamental and can be used to derive any other provenance questions including advanced ones. Finally, all these findings can be provided to the end

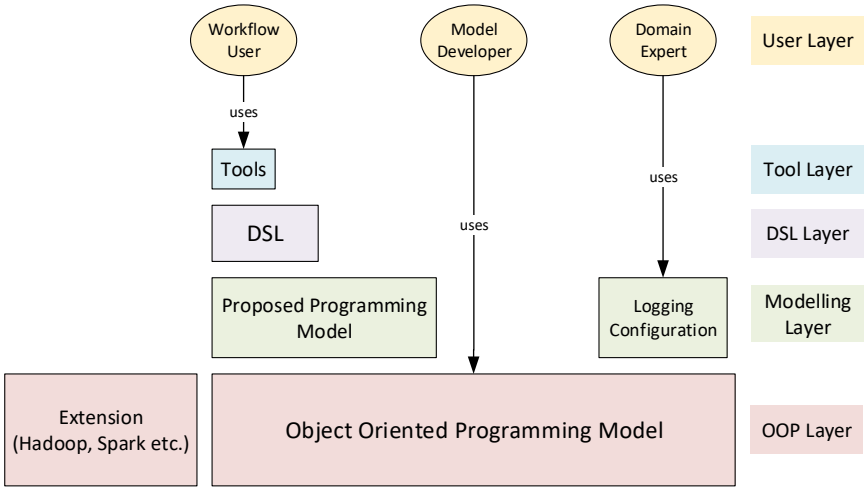

Fig. 1: A layer based architecture for our proposed programming model

users in a meaningful way such as data visualization or reporting [21], [9]. We are developing a data provenance visualization service that will be offered as a web service. It will connect itself with the workflow system to facilitate online streaming log data analysis. SoC is again followed here and even only this visualization service can be further developed independently. Prominent machine learning tools and deep learning methods can be integrated with this visualization system to conduct big log data analytics. Consequently, all the components associated with the whole system will provide with distinct online services. The distinct services and their internal communication architecture is illustrated in Figure 2. In this prototype of Figure 2 the user directly uses the workflow system with all enabled features inside it and the online visualization service as well. The workflow process logs are saved in a database that are parsed through a parser. The logs can only be written in the database and the parser can only read from the log database. The visualization or reporting service reports the users using the online parser data based on their needs.

2 Research Methodology

Our work contains several phases. Each of the phases is divided into different sub-phases and covers versatile topics of research and development. Those phases are briefly described below.

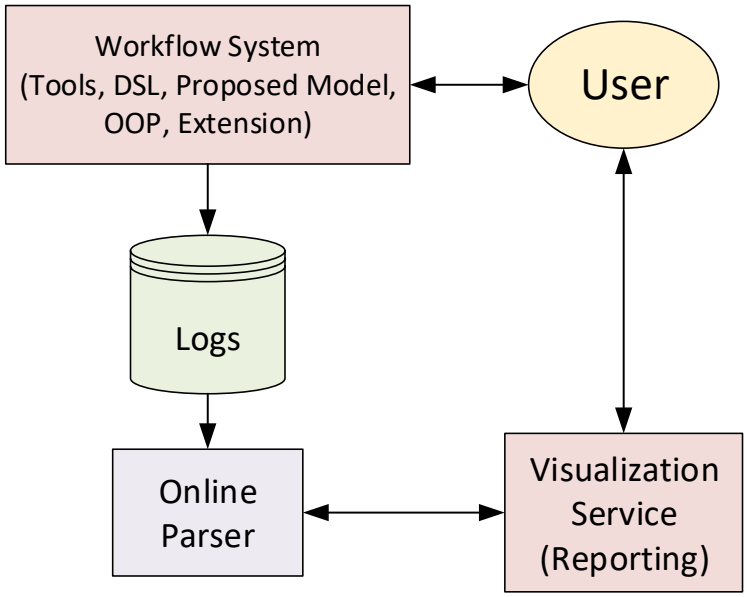

Fig. 2: System components services architecture

2.1 Modelling Phase

This phase covers the design and implementation of the programming model itself.

Designing Log Structure The structure of log is designed. This design can also vary. We primarily focus on plain text based logs. For certain reasons, other file formats can be useful. We will conduct a comparative study on which way is the best in a certain scenario.

Designing Programming Model We design the programming model and define certain workflow components. Such as, ***Data*** (that are classified into sub-categories), ***Module***, ***Dataflow***, ***User*** and ***Conditions*** at the moment and can be extended further.

Featuring Automated Logging The programming model is then integrated with the logging mechanism with latest logging technologies. We fully engineer the mechanism to make it automated.

Facilitating Data Management The logs are primarily saved as flat files and they can also be saved into database. We use NoSQL Database technology in this context. To enable online data streaming for any component, choose to use Apache Spark [26] and any related frameworks/technologies.

2.2 Implementation Phase

Implementation phase relates to make the developers use the programming model for building workflows. Also a number of workflow tools are developed and pipelines from different angles are implemented.

Configuring Logs The logs are easily configured by any domain experts. Developers do not need to touch the configurations.

Implementing Analysis Tools Analysis tools are implemented by a group of tool developers (e.g. image processing tools, statistical tools, data cleansing tools and so on). While converting their tools following our model, a migration of development happens. We design our model in a manner that the migration hassle is reduced and the task follows a similar pattern.

Layering with DSL A DSL can be layered upon our programming model. This way, all our model features are brought to the DSL layer. On top of it, the DSL provides more flexibility to develop workflows.

Designing Pipelines Pipelines and consequently complex workflows are designed and implemented using the DSL at this phase. This will generate log data for analysis.

2.3 Analysis Phase

This phase is about the generated logs and their analysis. The task of this phase is to discover as much provenance insights as we can and provide them to the end users in a standard way.

Parsing Logs Logs are parsed with our simple parser. These logs are semi-structured data because they are structured and also hold unstructured data.

Extracting Pipelines We leverage the power of Graph Database technology to re-extract the executed workflows and pipelines from previous phases. This way, all the features of Graph Database are integrated in our system.

Querying Provenance We focus on a number of provenance questions from different angles. They cover questions about data lineage, user-data, data-module or user-module patterns, errors and anomalies, information retrieval and recommendation.

Workflow Provenance Questions:

- What are the inputs of a module in an execution?
- What are the parameter settings of a module in an execution?
- What are the outputs of a module in an execution?
- What is the type of a data element in a workflow?
- What is the DAG (Directed Acyclic Graph) representation of a workflow?
- Who is the user of a workflow component (module/data)?
- What are all the properties of a workflow component (module/data/user)?
- What is the lineage DAG representation of a data element from root source to data product?
- What is the user defined condition of a module for its true execution?
- What is the time series data for any workflow component (module/data/user) with respect to a certain property?
- What is the DAG representation of a workflow for an error?
- What are all the properties of a workflow component (module/data/user) for an error?
- What is the classification of related modules in a workflow system?
- What is the classification of related data elements in a workflow system?
- What are the module-module, data-data, user-user, module-data, module-user, and data-user usage patterns in a workflow system?
- What is the best visualization approach for a particular provenance question?

Reporting or Data Visualization Building a web service that can analyze online streaming log data from the log database of a workflow system is possible. The reporting can be done with data visualization techniques using modern technologies.

```
#import libraries
from ProvModel import Object, Module

#A module to double the input data

#Inherit Module
class Double(Module):
    #Define body
    def body(self):
        #P is a list of parameters
        #Get value of 1st parameter
        a = self.P[0].ref
        c = a + a
        #Return output as model object
        res = Object(c)
        return res


#A workflow to double a data value
#d1 -> Double -> d2

#Create a data object holding value 111
d1 = Object(111)
#Define a pre-built module
double = Double(d1)
#Run the module
#Output data in d2
d2 = double.run()
```

Listing 1.1: A pipeline implementation with our model

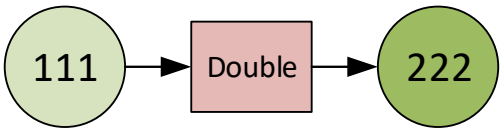

Fig. 3: A pipeline that doubles the input to output

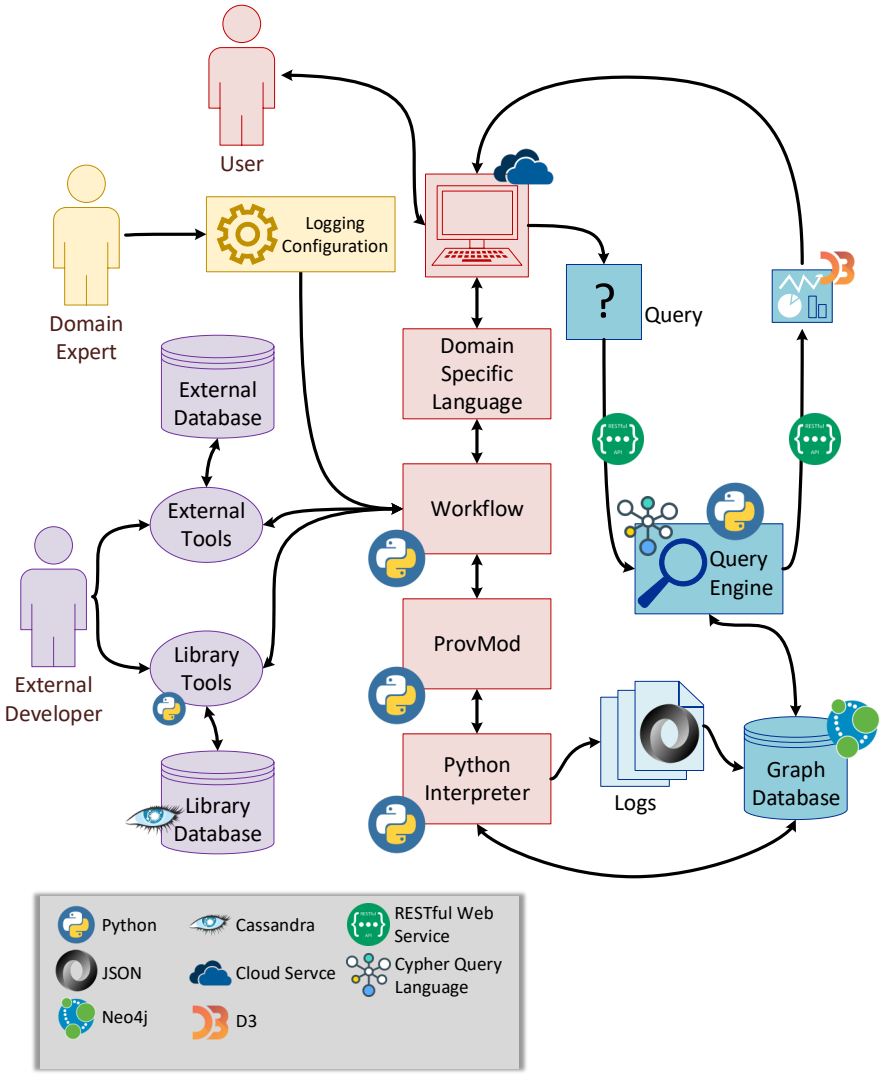

Fig. 4: Implementation of ProvMod and relation between other components of the whole workflow system that we propose

3 Implementation Details

The implementation of ProvMod is provided in Figure 4. The user uses a workflow through the user interface. Through the user interface, they may use a DSL to implement their workflows that stands on the ProvMod model. ProvMod stands on Python Interpreter at the core. The logging configuration can be cus-

tomized by a different user who is an expert in the domain. External tools can be integrated with ProvMod that can even be developed by other users. ProvMod also offers a number of tools as Library Tools. External tools and Library Tools may have their own database facilities. The ProvMod, leveraging the power of Python Programming Language and Logging Configuration, saves the logs in a Graph Database that we implemented with Neo4j [24]. Through the user interface, the user may later submit a query through a query engine. The query engine uses Cypher Query Language to parse logs from the Graph Database. The query result is provided with D3 in a web browser. All the communication between each pair if components is done through RESTful Web Services [10]. We also emphasize using NoSQL database such as Cassandra [16] for Library Tools. An example of the pipeline in Figure 3 is provided in Listing 1 code snippet.

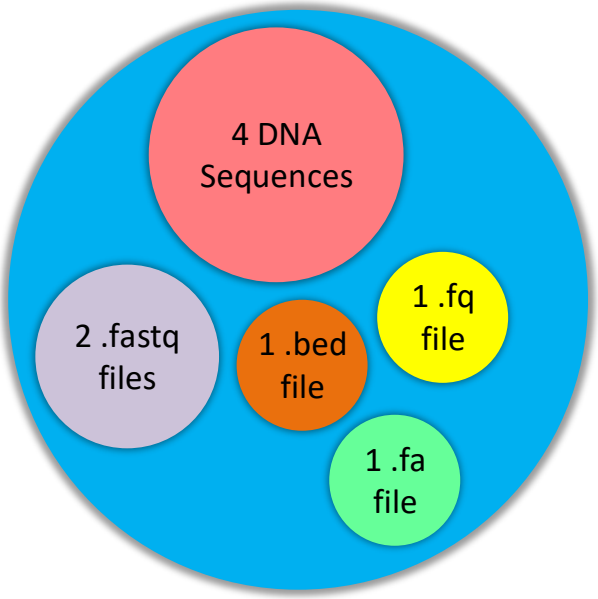

Fig. 5: Experimental dataset overview

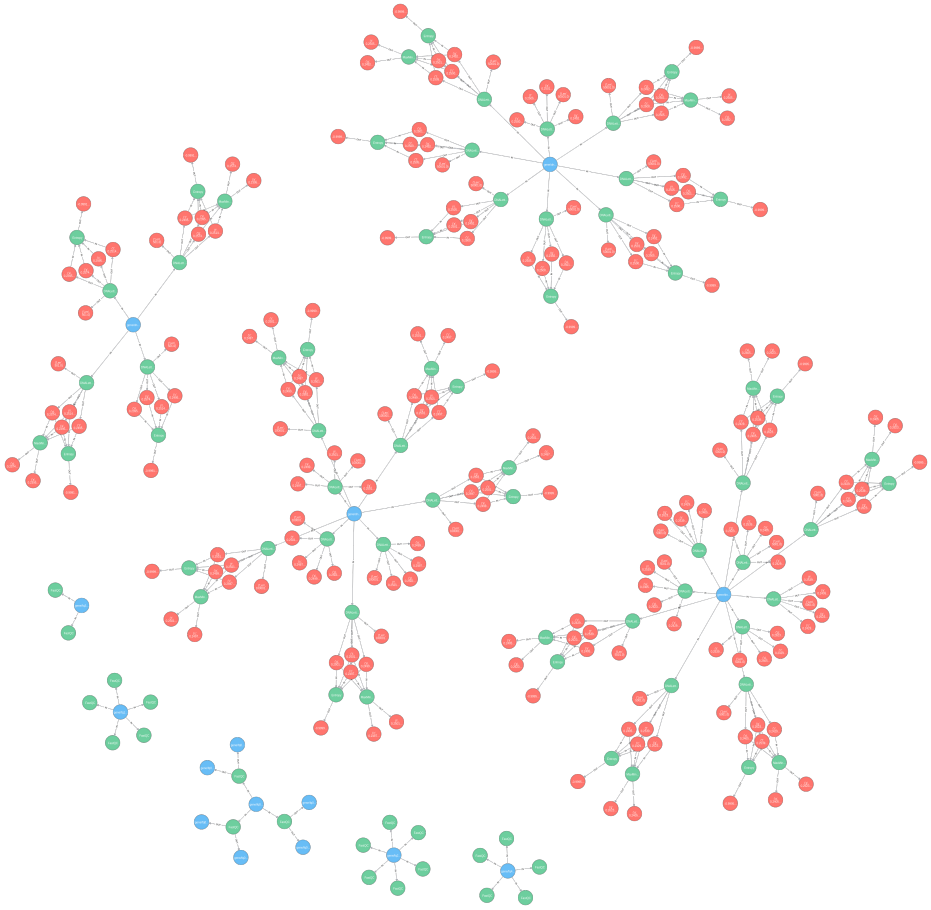

Fig. 6:
A portion of the provenance graph from the simulation. This snapshot is taken from the the big graph view of Kibana implementation. Different node types are represented with different colors with zoom in and out feature. The overall view is showing how big graphs can be used to capture provenance graph patterns.

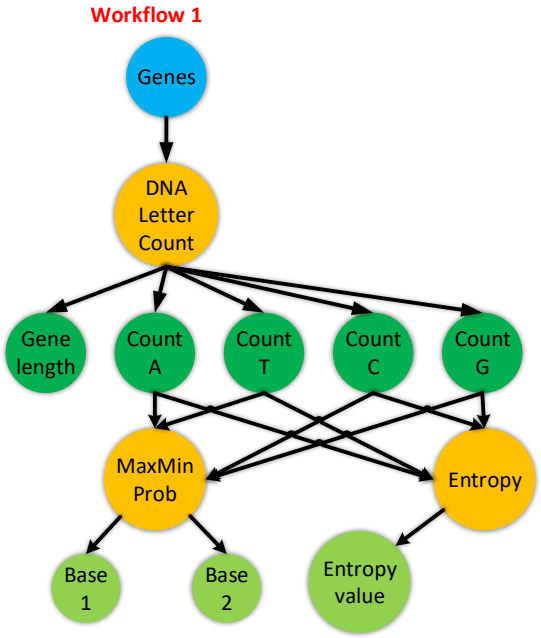

Fig. 7: First workflow for simulation

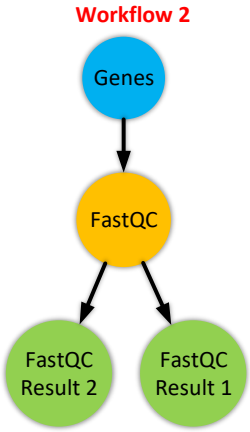

Fig. 8: Second workflow for simulation

4 Experiments

We implement two different workflows from Bioinformatics described in Figure 7 and 8. The first workflow is about counting DNA letters from a genetic dataset. We also count the length of the gene. Based on the nucleotide base counts, we can calculate the Entropy of the gene sequence. We also find out, which base is having the most and least probability of occurrence over the full sequence to get an understanding of the full sequence along with it's entropy. In the second workflow, we run FastQC [4] over the datasets to generate FastQC results. Note, in the simulation, a collection of genetic data is used that is described in Figure 5. The .fastq files are only valid inputs for FastQC and generates error otherwise. In the simulation, to generate user oriented usage scenario, we choose a data randomly from the dataset to input through the workflows. Also, from the first workflow, only DNALetterCount is executed or DNALetterCount with Entropy is executed or DNALetterCount with Entropy and MaxMinProv tools are executed. Otherwise, FastQC is executed with random inputs. They are selected randomly.Between the executions, there is a random gap of 1 to 7 seconds. From the simulation, we create a provenance graph of around 20,000 nodes that contains logs about FastQC errors too. A portion with 300 nodes from the provenance graph is shown in Figure 6.

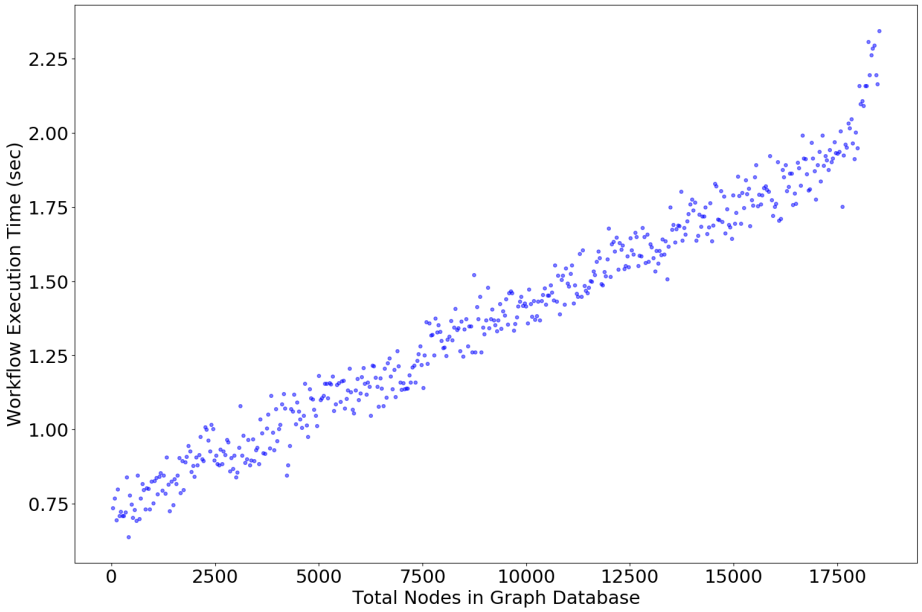

Fig. 9: Execution overhead of ProvMod

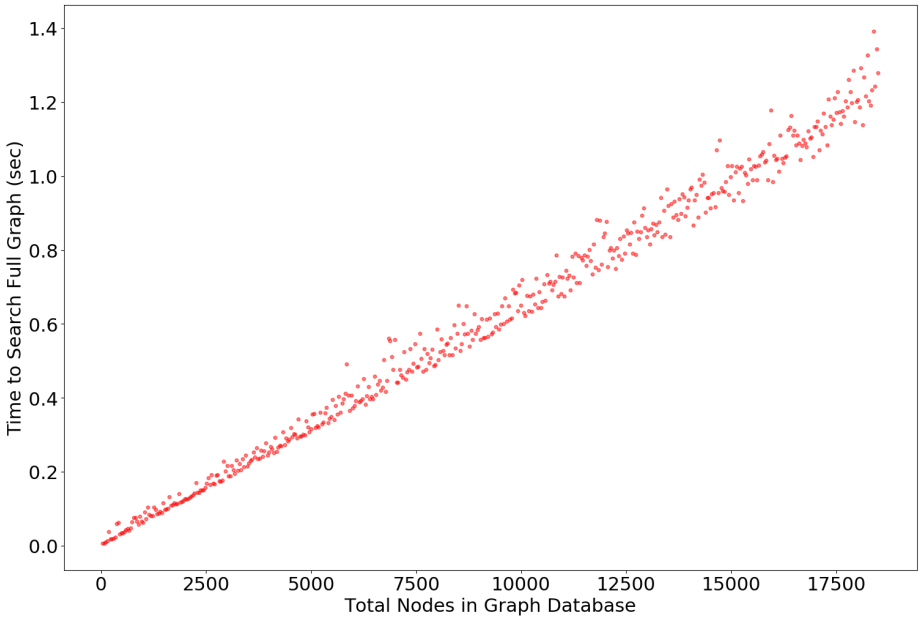

Fig. 10: Query overhead of ProvMod

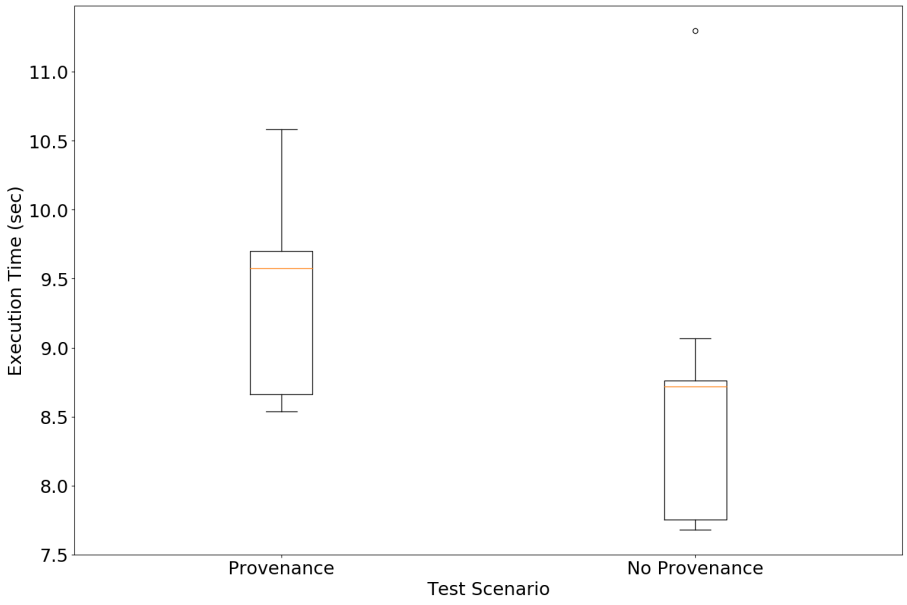

Fig. 11: Comparison of ProvMod execution time with and without provenance

5 Performance Analysis

We analyze several things about our ProvMod model. The analyses are described below with the found insights:

1. As our model is fused with the Graph Database, querying and adding nodes is always happening throughout the execution. It can be easily turned off, but we are eager to see how much performance overhead is occurring for that. In Figure 9, we can see, when there are around 20,000 nodes in the simulated provenance graph, the tool execution time is around 2 seconds. In real world workflows, a workflow may contain around 10 nodes or so, but never thousands of nodes for a single user. So, our ProvMod model shows good performance.
2. The query time will also increase as the graph is expanded. We search the full graph during the simulation at the end of every single simulation step and capture the time to collect the full graph. We find that the full graph search is returned withing 1.5 seconds when there are around 20,000 nodes in the provenance graph in Figure 10. This clarifies that, our ProvMod model query is not time consuming and fast.
3. Finally, we compare the whole simulation with and without provenance and measure the performance overhead the ProvMod logging is actually creating that we present in Figure 11. It becomes clear that, for such a huge graph with 20,000 nodes the provenance creates an overhead of around 0.5 seconds in average.

6 Related Works

Significant amount of research works were conducted on Big Data and that is still going on. Big Data being one of the most demanding technology now, flourished significantly. Dietrich et al. describes all the phases of Big Data analytics lifecycle in their book in details [12]. The survey of Chen et al. describes the challenges, open questions, analytics lifecycle, related important technologies and impact of Big Data in their work [9]. Online data streaming as well as analysis is a crucial part in Big Data. Hadoop [22] and Spark [26] are two well known technologies to achieve such requirement. The practice of Software Engineering is also very important for building a perfect system, specially when it involves different groups of developers and experts. Separation of Concerns (SoC) is one of the design principles of Software Engineering [17]. While designing different services in our work, we tried to follow this principle.
Scientific workflow systems or workflows have created various research directions. Each of them focuses on different research problems. A good survey of this topic and future direction could be found in the work of Barker et al. [6]. Besides, all workflows are not same and workflows in Big Data analytics require integration of further data intensive tools and technologies. Liu et al. in their work surveys on such data intensive workflows [18]. A scientific classification of workflow systems

is presented by Yu et al. [25]. Kepler [19] and Galaxy [15] are two such examples of scientific workflows for general purpose to domain specific research works.
Workflow provenance, that is the central topic of our work is a big issue. The problems, research angles, solutions and implementations vary from domain to domain. A well classification for all these provenance related topics in workflow systems is described by Cruz et al. [11]. They demonstrate how provenance in workflows differs in different aspects. They also take into account a number of existing workflows and evaluate them with their taxonomy. The importance of automated provenance in workflows is presented by Barga et al. [5]. How provenance information could be gathered from only manipulating logs is presented in the work of Ghoshal et al. [13]. Many programming models were emerged for solving problems in workflow provenance. One such work is the work of Amsterdamer et al. [2] that differentiates between data provenance and workflow provenance very clearly. They also propose a method that leverages the fashion of data provenance for capturing workflow provenance. Another work is the work of Acer et al. [1] that proposes a graph based workflow provenance model to describe lineage of dataflow.
Workflow provenance has been separated from the concept of data provenance itself. Workflow provenance is the provenance of such data that were originated from workflow systems. Still provenance of workflows are data elements. To be successful in analyzing provenance of any data, studying the topic of data provenance becomes necessary. A survey of data provenance in e-science was done by Simmhan et al. [23]. Buenman et al. [7] characterizes data provenance with respect to the question of 'Why' and 'Where'. Hartig et al. [14] presents an approach of data provenance that extends previous approaches and suitable for data involved in web technologies. Buenman et al. [8] overviews the concept of provenance in database technologies.
Querying is the final problem of provenance. A set of fundamental provenance questions may lead to an empirical research in this topic. Anand et al. [3] describes techniques for efficient querying of workflow provenance graphs. Missier et al. [20] presents an approach for fine-grained lineage querying that is also efficient. Analytic provenance is another topic that was emerged from workflow provenance and involves data visualization, which is also another research domain. The importance of data visualization in workflow provenance could be understood from the work of Ragan et al. [21].

7 Conclusion and Future Works

We try to bring every related component starting from workflow design to end user reporting into one place. Our proposed approach is for any kind of workflow system. Bioinformatics is primarily selected for our study and system evaluations because this research topic covers versatile ranges of scientific domains and also ranges to Big Data research problem. Our research can lead to a general recommendation and standard for workflow provenance research. Data provenance research can be merged with this topic and overall provenance research can flour-

ish. We also plan to find the elementary provenance questions that is applicable in all domains of Data Science.

References

1. Acar, U., Buneman, P., Cheney, J., Van Den Bussche, J., Kwasnikowska, N., and Vansummeren, S. A graph model of data and workflow provenance.
2. Amsterdamer, Y., Davidson, S. B., Deutch, D., Milo, T., Stoyanovich, J., and Tannen, V. Putting lipstick on pig: Enabling database-style workflow provenance. *Proceedings of the VLDB Endowment 5*, 4 (2011), 346–357.
3. Anand, M. K., Bowers, S., and Ludäscher, B. Techniques for efficiently querying scientific workflow provenance graphs. In *EDBT* (2010), vol. 10, pp. 287–298.
4. Andrews, S. Babraham bioinformaticsfastqc a quality control tool for high throughput sequence data, 2015.
5. Barga, R. S., and Digiampietri, L. A. Automatic generation of workflow provenance. In *International Provenance and Annotation Workshop* (2006), Springer, pp. 1–9.
6. Barker, A., and Van Hemert, J. Scientific workflow: a survey and research directions. In *International Conference on Parallel Processing and Applied Mathematics* (2007), Springer, pp. 746–753.
7. Buneman, P., Khanna, S., and Wang-Chiew, T. Why and where: A characterization of data provenance. In *International conference on database theory* (2001), Springer, pp. 316–330.
8. Buneman, P., and Tan, W.-C. Provenance in databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (2007), ACM, pp. 1171–1173.
9. Chen, M., Mao, S., and Liu, Y. Big data: A survey. *Mobile networks and applications 19*, 2 (2014), 171–209.
10. Christensen, J. H. Using restful web-services and cloud computing to create next generation mobile applications. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications* (2009), ACM, pp. 627–634.
11. da Cruz, S. M. S., Campos, M. L. M., and Mattoso, M. Towards a taxonomy of provenance in scientific workflow management systems. In *Services-I, 2009 World Conference on* (2009), IEEE, pp. 259–266.
12. Dietrich, D., Heller, B., and Yang, B. Data science & big data analytics: discovering, analyzing, visualizing and presenting data, 2015.
13. Ghoshal, D., and Plale, B. Provenance from log files: a bigdata problem. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (2013), ACM, pp. 290–297.
14. Hartig, O. Provenance information in the web of data. *LDOW 538* (2009).
15. Hillman-Jackson, J., Clements, D., Blankenberg, D., Taylor, J., Nekrutenko, A., and Team, G. Using galaxy to perform large-scale interactive data analyses. *Current protocols in bioinformatics* (2012), 10–5.
16. Lakshman, A., and Malik, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review 44*, 2 (2010), 35–40.
17. Laplante, P. A. *What every engineer should know about software engineering.* CRC Press, 2007.

18. Liu, J., Pacitti, E., Valduriez, P., and Mattoso, M. A survey of data-intensive scientific workflow management. *Journal of Grid Computing 13*, 4 (2015), 457–493.
19. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., and Zhao, Y. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience 18*, 10 (2006), 1039–1065.
20. Missier, P., Paton, N. W., and Belhajjame, K. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *Proceedings of the 13th International Conference on Extending Database Technology* (2010), ACM, pp. 299–310.
21. Ragan, E. D., Endert, A., Sanyal, J., and Chen, J. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE transactions on visualization and computer graphics 22*, 1 (2016), 31–40.
22. Shvachko, K., Kuang, H., Radia, S., and Chansler, R. The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on* (2010), Ieee, pp. 1–10.
23. Simmhan, Y. L., Plale, B., and Gannon, D. A survey of data provenance in e-science. *ACM Sigmod Record 34*, 3 (2005), 31–36.
24. Webber, J. A programmatic introduction to neo4j. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity* (2012), ACM, pp. 217–218.
25. Yu, J., and Buyya, R. A taxonomy of scientific workflow systems for grid computing. *ACM Sigmod Record 34*, 3 (2005), 44–49.
26. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. Apache spark: a unified engine for big data processing. *Communications of the ACM 59*, 11 (2016), 56–65.