Why-Diff: Exploiting Execution Provenance to Answer the Difference Between Two Computational Results

Priyaa Thavasimani

I. Introduction

In the era of Big-Data, almost all scientific experiments are data-intensive and are rerun multiple times for verifying and validating the results. As data kept evolving, this promotes the chance of reproducibility by changing the input datasets or any other dependencies. Computational workflows consist of a series of steps in which data is generated, manipulated, analysed and transformed. Researchers use tools and techniques to capture the provenance associated with the data to aid reproducibility. The metadata collected not only helps in reproducing the computation but also aids in comparing the original and reproduced computations. Most computations that are reproduced do not guarantee the correctness of results because of the underlying fact that they use different artefacts. The execution provenance of a computation can be used to investigate why the results are different when compared to other. The researcher might want to understand the "cause" for the difference or "impact" that the computation achieved. To answer these questions, an algorithm "Why-Diff" is designed and implemented which intends to explain difference between outcomes of related computational experiments over time by exploring the execution provenance of computations. The Why-Diff algorithm aims to forward and backward traverse the provenance graph to answer "what is the impact of the computation?" and "what caused the different results?" respectively.

II. Background

In previous studies [1]–[8], the terms "Reproducibility" and "Replication" are widely discussed, of which most researchers emphasise the importance of reproducibility.

Researchers classify provenance captured by the system into two basic types [9] : Prospective and Retrospective. Prospective provenance captures the abstract specification of the computational tasks like library, data or any form of dependencies which make the Workflow to run whereas retrospective provenance captures more detailed information like execution environment, inputs, outputs and intermediate results cached.

Workflow Management System (WfMS) is used to execute "in-silico" experiments. It provides a collaborative environment for researchers to create, execute and share workflows. There are various types of Workflow Management Systems, some of which includes E-Science Central [10], Pegasus [11], Galaxy [12], VisTrails [13], Kepler [14], Taverna [15]. WfMS captures fine-grained provenance capturing Workflow evolutions and its associated datasets. Researchers take these provenance traces to ease reproducibility so that to avoid false leads by original experiments.

The comparison by existing tool Vistrail provenance detects files contents by hashes, allowing users to later check that provided data does indeed match that used in the original experiment [16]. The VisTrail team has also included persistence package feature that allows users to store input, output and intermediate data in a versioned history.

An issue with the VisTrail was the impact on reproducibility based on the updated database. To address this issue, a model was developed that integrated workflow and database provenance and enables reproducibility for workflows that interacts with relational databases [17]. Similar to database provenance, the researcher [18] tracked the dependencies of external web service and developed a Web Service Monitoring Framework. VFramework is a framework created to verify that a redeployed process performs according to expectations. This framework which was evaluated using Taverna, uses an ontology-based model [19] for description of processes and their dependencies. This framework was evaluated to check the verification and validation capabilities by re-executing the use cases in Linux and Windows based environments [20].

Galaxy is a web-based WfMS for data intensive biomedical research which supports provenance tracking through its "History" tool. Another provenance tracking system is Sumatra [21] which captures each execution in configuration management tool, allows users to tag and compares the collected provenance traces. Sumatra captures the difference between two workflows at the code-level i.e only the abstract provenance.

Almost all the above-mentioned WfMS keeps track of the evolution of experiments by version control. Further to monitoring tool that aid tracking changes in the external dependencies, it has become essential to analyse and monitor data transformations. Workflow is an orchestration of activities that would have used and generated files and activity could be database-related or web-service related, which acts up on data. Simply comparing files based on hash-value or code is not enough as there are numerous workflows run every single day. It is essential to link which piece of data that the particular workflow produced or consumed. To the best of our knowledge, there is no existing approach which traverse the execution traces of workflow to analyse the difference between two executions.

Research Context: As mentioned in background study, this research set in the context of reproducible research. The ability for one researcher to validate, verify, reproduce, and repurpose other researchers results has always been at the core of scientific discourse. Computational workows involves series of steps in which data is generated, manipulated, analysed and transformed. As modern experimental science becomes

increasingly data-driven and reliant upon big datasets and complex computations to generate new results, we have a chance of making science more reproducible, i.e., by appropriately managing the datasets, the experimental processes, and the associated metadata that describes both of them. Enabling reproducibility in science is therefore becoming the focus of new research within a vibrant research data management community.

Research Problem: My research addresses a specific problem in this setting. Suppose, in abstract, that you are given an experiment, described by a Workflow W with input X and additional dependencies D (these can be dependencies on external data resources, as well as on third party software libraries or system environment), which at a certain time t produced an output Y, our scientific result. A third party who wishes to reproduce P at a later time t would have to re-build, re-deploy, and re-execute P in a new environment. This is relatively easy to achieve using modern virtualisation technology (VM, Docker, etc.), but it is also not very interesting. A more challenging problem occurs when variations are introduced in the process and its dependencies: $\{W, X, D, Y\}$ —$\{W', X', D', Y'\}$, which lead to different outcomes Y' not equal to Y when the new version is executed. This may happen accidentally, i.e., when the external data sources are updated or not all requirements (some of which may not be explicit) have been met at the time P is deployed again, or intentionally, when the experimenter intends to explore variations of the original experiment. Intuitively, knowledge of the structure, execution trace and runtime dependencies on data and software are critical to answering these questions. These are all forms of metadata, which hopefully help explain the observed experimental outcomes. However there is no clear indication from the literature (please refer to background study) on just how much and exactly what type of metadata is needed, how it can be processed and analysed, and how much effort is required to collect it. These are the high level questions that my research will address.

III. Research Hypothesis:

It is possible to use retrospective provenance from computational experiments to explain differences in the outcomes of two repetitions of the same process, given limited changes to the process.

IV. Research Question

- Q1: Whether the computation achieved the desired impact?
- Q2: What would be the cause for unexpected results? Is it a single or multiple causes responsible for the difference?
- Q3: How the input difference affects the output data?
- Q4: What provenance must be collected and packed along with original experiment that enables the verification and validation of results of reproduced experiment with respective to results of original experiment?

V. Remaining Work

The algorithm "Why-Diff" is designed to forward and backward track the provenance of entire workflow for intentional and unintentional changes made to the reproduced Workflow blocks. The initial experiments are demonstrated with varied dependencies, input and workflow blocks assuming the two Workflow graphs are topologically similar. The remaining work is to compare the graphs which are topologically dissimilar. As of now, data nodes are compared based on labels/file names. Future work would be to compare the data nodes based on file content. Initial experiments are conducted on test sets comprising of 10 eScience workflow invocations. It is planned to extend the evaluation with respect to a more comprehensive set of case studies.

VI. Work done so far

Computational workflows consist of a series of steps in which data is generated, manipulated, analysed and transformed. By changing the computational artefacts, the initial investigations was done to confirm that the results of a reproduced computation may differ in accordance with original computation which is explained in Chapter I. The execution provenance of a computation can be used to investigate why the results are different when compared to other. The researcher might want to understand the "cause" for the difference or "impact" that the computation achieved. An algorithm named "Why-Diff" is designed and implemented which is explained in Chapter II.

VII. Time Plan

The project can perform its objective by executing the following scheduling time:

1: September, 2015 - April, 2016: Attending interdisciplinary training and coursework. Experiment with the provenance capturing tools to understand its provenance generation capability. Writing Literature Review.

2: May, 2016 to June 2016: Developing sample usecase that best describes the reproducible research. Experimented the usecase with YesWorklow and NoWorkflow, capturing retrospective and prospective provenance. Demonstrate the work to supervisory team in presentation.

3: July, 2016 to August, 2016: Finalising project plan, submitting, and presenting annual report.

4: September, 2016 to March, 2017: Begin working on algorithm and tool to analyse the divergence between results obtained using different versions of a process (with evolving dependencies).

5: April, 2017 to August, 2017: Implementation of the algorithm and initial evaluation of the research using the ReComp project as a testbed. Continued Aim to submit a paper to a highly ranked conference. Present findings to supervisory group in presentation.

6: September, 2017 to October, 2018: Augmenting features to the algorithm such as to support non-isomorphic graphs, file-content comparison. Extended evaluation with respect to a more comprehensive set of case studies, continued publication. Publication of results in top quartile conferences and journals.

7: November, 2018 to September, 2019: Writing up thesis and final examination.

VIII. Chapter I

The project aim to consider 4 possible cases for reproducing computations. We have used P, X, D, Y to refer to Program, Input, Dependency and Output of the original computation, respectively. Similarly, we denote the same quantities in the reproduced computation by P', X' , D' and Y', respectively. In "Why-Diff", we consider four possible cases for reproducing computations, as follows:

1) $\{P, X, D'\}$ - Changing Dependency keeping Program, input unchanged.
2) $\{ P,X', D\}$ - Changing Input keeping Program, Dependency unchanged
3) $\{P', X, D\}$ - Changing Program keeping Dependency, input unchanged
4) $\{ P', X', D' \}$ - All artefacts changed

Here, we tried to apply first case in which we changed the Dependency keeping Program and Input as constant while redoing computation.

A. Model usecase for Computational Reproducibility

Sentiment analysis or opinion mining has emerged as an active domain among the research fraternity because enormous amount of heterogeneous data is continuously increasing every day by the users via www, viz., e-commerce websites, social networks, discussion forums, blogs etc. Motivated by the Natural Language Processing techniques, we have developed a usecase Twitter Sentiment Analysis which is a python script which takes in static tweets and visualise the sentiment score percentage Positive, Negative and Neutral in a pie chart. Lot of Natural Language Processing tools are available to process the human language to classify whether it is subjective or objective and orient its polarity as positive, negative and neutral. TextBlob and NLTK are similar Natural Language Processing tools which provides API which is used to analyse the tweets and compute sentiment score for the text. It helps in converting an unstructured text to structured text which is easily manageable.

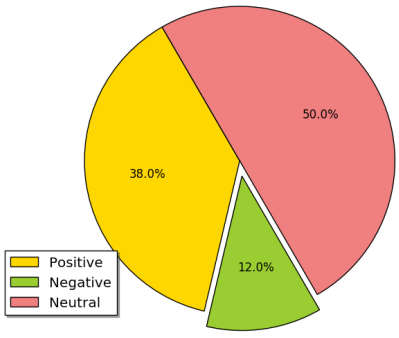


Fig. 2: Sentiment Scores using TextBLOB

While reproducing the usecase, we have intensionally used NLTK (Refer Fig 3) replacing TextBLOB (Refer Fig 2), for the same set of tweets. Positive, Negative and Neutral sentiment percentages using TextBLOB are *38.0%*, *12.0%* and *50.0%* respectively. Whereas NLTK computes *18.0%*, *17.0%* and *65.0%* as Positive, Negative and Neutral percentages respectively. In this case, Y != Y' when trying to mock the case $\{P, X, D'\}$ where D refers TextBLOB and D' refers NLTK respectively.

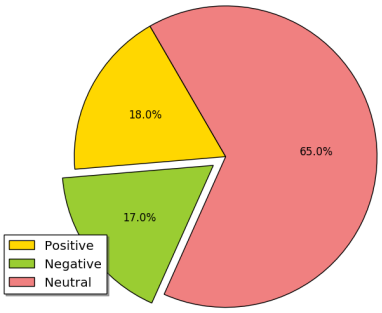


Fig. 3: Sentiment Scores using NLTK

We have observed a striking difference in the results when using different software dependencies keeping the Program and Input unchanged. We viewed TextBLOB and NLTK as a blackbox and recorded the provenance of the two scripts without bothering about the algorithm design of TextBLOB and NLTK. Motivated by the different results from related computations, we have analysed the provenance data collected by YesWorkflow and NoWorkflow. "YesWorkflow" and "noWorkflow" are the tools that captures the provenance of independent scripts. YesWorkflow [22] extracts the latent information from scripts, exports, visualize in graph form. Given a script, YesWorkflow generates the workflow based on user annotations which generally captures prospective provenance. "noWorkflow" [23] (not only Workflow) system uses Python runtime profiling functions to generate provenance traces that reflect the processing history of the script. noWorkflow captures three types of provenance - definition, deployment, and execution provenance.

B. YesWorkflow

YesWorkflow is one of the approach to capture prospective provenance from scripts. It helps to interpret the user annotations and visualise the graph that shows the computational steps and data flow in the script.We used YesWorkflow to visualize the graph to extract the API dependency (TextBLOB or NLTK) and retrieved two graphs showing the dependency. Furthermore, it is also possible to visualize the upstream and downstream of a specific data product. In addition to the visualization, YesWorkflow facilitates querying the workflow steps. The user annotations in the script play key role here. This provides very basic prospective provenance which is

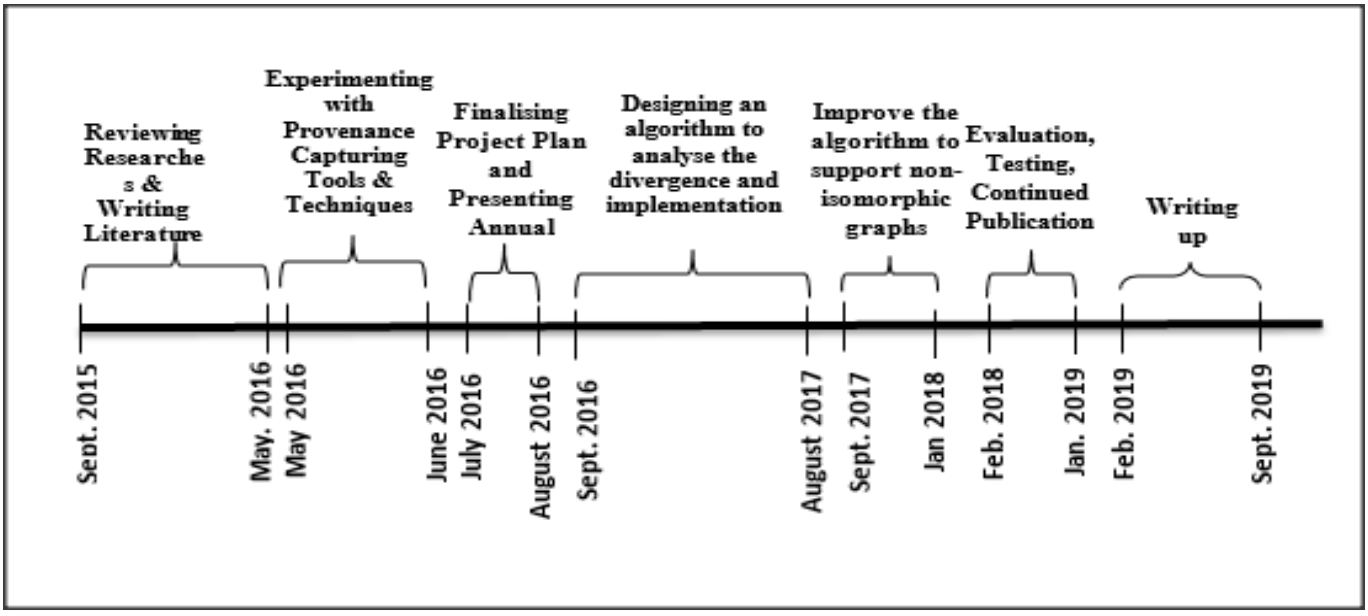


Fig. 1: Timeplan

inadequate for analysing difference between the two execution results.

C. NoWorkflow

Noworkflow is another tool that captures the retrospective provenance automatically. It does not require user annotations for recording provenance. NoWorkflow defines one execution of the experiment as Trial. Here, the execution of Sentiment analysis with TextBLOB is referred as Trial 1 whereas the execution with NLTK referred as Trial 2. "NoWorkflow" has a command called "Diff" which yields a difference graph between 2 executions.

For more details, please visit http://eprint.ncl.ac.uk/file_store/production/234967/D959C504-4859-434A-A59A-8B7DC13AC4BD.pdf

IX. Chapter II

The main contribution for the chapter II is the "Why-Diff" algorithm which is able to compare two workflows by "forward" and "backward" traversals.

1. Forward traversal is to find the divergent data nodes that resulted due to change in the artefacts that the researcher voluntarily made self. This aims to traverse the provenance graph of computational steps with experimental input as a starting point. In this case, we aim to provide evidence that the changes have achieved the observed impact.
2. Backward traversal is to find the divergent activity nodes that caused the change that the Researcher unaware of self.This aims to traverse the provenance graph backwards to trace the causes that explain the difference with experimental output as a starting point. By doing so, we aim to find the likely causes for the observed differences in the outputs.

A. Technologies and standards used

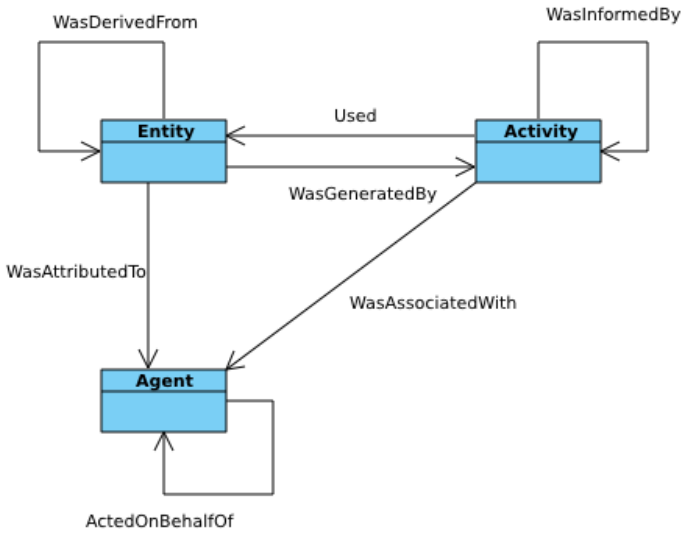


Fig. 4: PROV Core Structures

TABLE I: Provenance Terms Mapping

e-SC Format	PROV Format	Neo4j
Document	Entity	Node
Execution	Activity	Node
used	used	Relationship
wasGeneratedBy	wasGeneratedBy	Relationship

eScience Central, shortly called as "e-SC", is a Workflow Management System which provides Software and Platform as a Service to scientists to create, edit, run and share

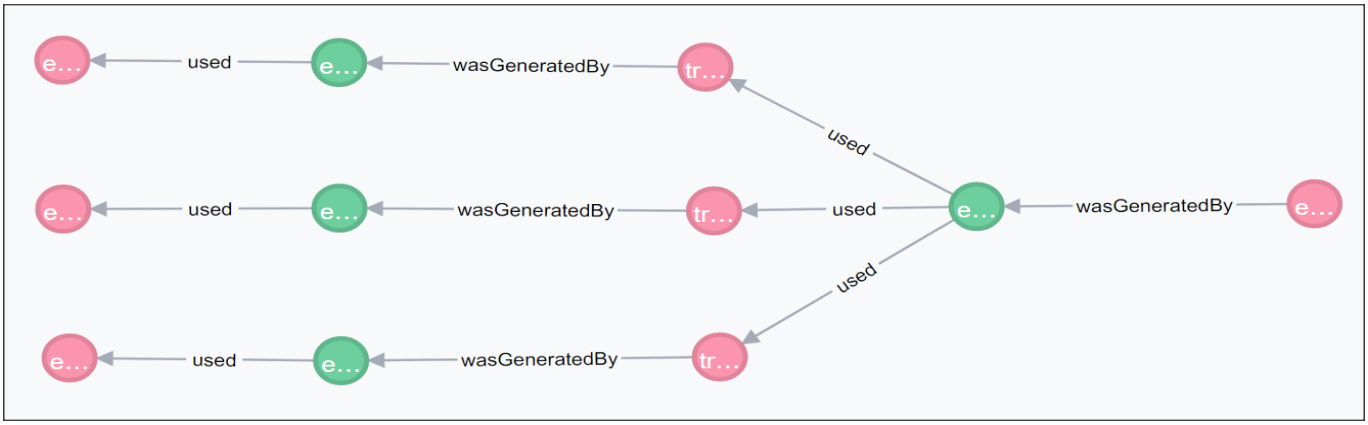


Fig. 5: Neo4j visualization of Sample Workflow

Workflows in the cloud [10]. It captures very rich provenance information including data access and Workflow executions. The usage of these provenance details varies from one to another. Some might be interested who created the workflow, who downloaded it, who the data has been shared with. In our perspective, we are looking for what are the workflow inputs, outputs, services and intermediate results. Each input and output files are given identities and file labels. Additionally each service in a particular workflow are given identities as well as service's configuration are updated. Intermediate files are given hash-values. Having all these piece of data together, we can able to answer why the results of 2 Workflows different and what is the result if a service version is changed say for example from $version1$ to $version2$. In order to verfiy the results of reproduced workflow in accordance with original workflow, an inbuilt differencing system like "Why-Diff" is mandatory. The Workflow execution provenance of e-SC stick to the PROV-DM format. We have collected the Workflow execution provenance of eScience Central and have visualised in form of Neo4j nodes and relationships.

B. Provenance Model

W3C PROV [24] defines a data model and a set of constraints. It represents provenance as a Directed Acyclic Graph. Vertices in the PROV model represents entities, activities or agents. As scientists use different Workflow Management Systems, the provenance generated by such systems are in different format. To address this problem, an integration model ProvONE [25] which takes in the provenance traces of heterogeneous workflow systems and generated integrated provenance database as Prolog facts. To have a common understanding of capabilities of different provenance systems, a data model was crafted by the authors and released as PROV-DM. PROV-DM is conceptual data model that has following nodes and relationships. Nodes are identified by prov:Activity, proc:Entity, prov:Agent, and relationship types are identified by prov:used, prov:wasInformedBy, prov:wasAssociatedWith, prov:wasGeneratedBy, prov:wasAttributedTo, prov:wasDerivedFrom, prov:actedOnBehalfOf. "Entity" refers to Physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary. An "Activity" is something that occurs over a period of time and acts upon or with entities. An "Agent" is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agents activity. "wasGeneratedBy" refers to relationship between entity and activity, saying the entity was generated by the activity. The relationship "used" infers that an entity is used by the activity and "wasDerivedFrom" refers to an entity being derived from another entity. As name indicates,"wasAssociatedWith" refers to an activity being associated to an Agent. "wasInformedBy" infers an activity being informed by another activity about an entity. "actedOnBehalfOf" refers to an agent acting on behalf of another agent. "wasAttributedTo" refers to an entity being attributed to an agent.

C. Why Neo4j?

In our project, we have taken advantage of the most popular graph database "Neo4j". The graph database Neo4j is used in majority of fields like Chemistry, social network, semantic web, etc as it enables querying in simple manner in comparison to typical relational database.

In e-SC, the number of available Workflows, services and datasets is constantly increasing. In order to accommodate large voluminous data, the optimal choice would be graph database. Relational database could not handle these datasets as that needs complex join queries. To traverse the high-interconnected Workflow graphs which is combination of entities and activities, Neo4j could be the advantageous. We have analysed Workflow execution provenance of e-SC by exploiting PROV-DM and Neo4j. The Table I represents the provenance terms in e-SC, PROV-DM and Neo4j formats. In the figure 5, all the pink nodes are "entities" whereas green nodes are activities. The flow of graph starts from left to right. Relationships are identified as "used" and "wasGeneratedBy".

ALGORITHM 1: Forward Traversal

Input: Invocation1_Id
Input: Invocation2_Id
Output: Δ Graph
$Compare\ the\ list\ of\ first\ Entities\ in\ both\ graphs$;
$first\_hop = true$;
do
  if $first\_hop == true$ **then**
    $List < Activity > Activity1List \leftarrow findFirstActivityList(Invocation1\_Id)$;
    $List < Activity > Activity2List \leftarrow findFirstActivityList(Invocation2\_Id)$;
  else
    $List < Activity > Activity1List \leftarrow firstActivityNewList$;
    $List < Activity > Activity2List \leftarrow secondActivityNewList$;
    $Clear firstActivityNewList$ & $secondActivityNewList$;
  end
  foreach $(A1, A2) \in zip(Activity1List, Activity2List)$ **do**
    $List < Entity > Entity1List \leftarrow getWasGeneratedBy(A1)$;
    $List < Entity > Entity2List \leftarrow getWasGeneratedBy(A2)$;
    foreach $(E1, E2) \in zip(Entity1List, Entity2List)$ **do**
      $\Delta Graph \leftarrow Compare(E1, E2)$;
      $List < Activity > returnedActivityList1 \leftarrow getUsed(E1)$;
      $List < Activity > returnedActivityList2 \leftarrow getUsed(E2)$;
      foreach $(A1, A2) \in zip(returnedActivityList1, returnedActivityList2)$ **do**
        $Add\ A1\ to\ firstActivityNewList$;
        $Add\ A2\ to\ secondActivityNewList$;
      end
    end
  end
while $firstActivityNewList \neq Empty$ && $secondActivityNewList \neq Empty$;
$Compare\ the\ list\ of\ last\ Entities\ in\ both\ graphs$;
Function *Compare (E1, E2)*
  if $E1.fileLabel \neq E2.fileLabel$ **then**
    $\Delta Graph \leftarrow (E1, E2)$;
  end
  return $\Delta Graph$
Function *findFirstActivityList (InvocationId)*
  $List\ all\ activityNodes\ WHERE\ activity - [USED] - entity\ WHERE\ NOT\ entity - [WASGENERATEDBY] - any\_activity\ AND\ activity.id\ CONTAINS\ InvocationId$;
  return activityNodes

D. WhyDiff

Why-Diff focuses on the execution provenance of the Workflows rather than the provenance of the Workflows itself. This relies on the execution provenance that are collected by E-Science Central WfMS. As mentioned in our previous work [26], the accuracy of the Why-Diff analysis is proportional to the amount of provenance details of the two Workflows.

The project Why-Diff consider 4 possible cases which resulted in output of reproduced computation different from output of original computation.

1. $\{W, X, D'\}$ - Changing Dependency keeping Workflow, input unchanged.
2. $\{W, X', D\}$ - Changing Input keeping Workflow, Dependency unchanged.
3. $\{W', X, D\}$ - Changing Workflow keeping Dependency, input unchanged.
4. $\{W', X', D'\}$ - All artefacts changed.

We consider the difference in the Workflow results may be intentional or unintentional. Intentional changes are something that the experimenter is making himself/herself whereas unintentional changes are the changes that is made without the knowledge of the experimenter (For example updated external dependencies or external data sources which is not actually in control of the experimenter). In our previous work [26], We have considered above cases and proved that there is possibility of the output difference when there is no change in Inputs X and Program P changing the Dependency D to D', mocking the scenario $\{P, X, D'\}$. The same applies to Workflows as well except the fact the Program P represents orchestration of Program statements where Workflow W represents orchestration of services. While tracking provenance of independent scripts, it is essential to keep track of each data product produced or consumed out of a particular program statement. In contrast to the scripts, workflows express the computations as service-level components. It considers each computational block as a service, which makes it feasible to compare two workflows based on the service version. It is more convenient to visually relate the Workflow steps which consumes data and produce data in the form of graph. At this early stage of our investigation, we make the simplifying assumption that the provenance graphs for the two executions are isomorphic, and only focus on detecting differences in properties of corresponding nodes. We use the retrospective provenance of the Workflow but not the abstract provenance of the Workflow. The Workflows to be compared are structurally similar (i.e. isomorphic). This is the simplest possible scenario where the topologies of the two graphs are identical, and only the properties on the nodes differ from one graph to another.

1) Forward Traversal: Algorithm I is designed to move forward the provenance graph to detect diverging entity nodes because of an intentional change introduced by the experimenter himself. If the changes are introduced intentionally, then the experimenter would like to know the effect (data difference that is resulted) of the change. Hence he/she forward traverse the graph to identify the data nodes that are not similar

in both the Workflow graphs. We consider this as "effect-analysis". In the algorithm, we have used symbols $E1$, $E2$, $A1$, $A2$ to refer entity node for first invocation , entity node for second invocation, activity node for first invocation, activity node for second invocation respectively. The start nodes of the traversal are picked by listing the activities which have used an entity which is not generated by any other activity. As the traversal proceeds, for each activity, the generated entities are compared. The traversal continues till generated entities have not been used by any further activity. The traversal compares the file label (or hash value) initially. In eScience central WfMS, the end nodes are attached with file label property which is used to find the physical file name. The intermediate (we refer as transient) files possess the "hash value" property which is the unique identity of the transient data. All the diverging entities are added to the Δ Graph. The zip function in the algorithm ensures that the nodes from $graph_1$ are compared with the corresponding nodes in $graph_2$.

2) Backward Traversal: Algorithm II is designed to move backward the provenance graph to detect diverging activity nodes because of an unintentional change which the experimenter does not know. If the changes are unintentional then the experimenter would like to know the cause (activity node that caused the difference). And so he/she backward traverse the graph to identify the activity nodes that are not similar in both the Workflow graphs. This can be termed as "cause-analysis". In the algorithm, we have used symbols $E1$, $E2$, $A1$, $A2$ to refer entity node for first invocation , entity node for second invocation, activity node for first invocation, activity node for second invocation respectively. The start nodes of the traversal are picked by listing the activities which have "generated" an entity which is further not "used" by any other activity. As the traversal proceeds, each activity configuration are compared and for each activity the "used" entities are identified. The traversal continues till "used" entities have not been generated by any further activity. In eScience central WfMS, each activity are associated with configuration property, which captures good provenance of which the interesting ones are the file labels it has used/ generated, library used, the conditions it has used, source file used, block name and version. These metadata will help us comparing one activity node to another and all the diverging activities are added to the Δ Graph. The zip function in the algorithm ensures that the nodes from $graph_1$ are compared with the corresponding nodes in $graph_2$.

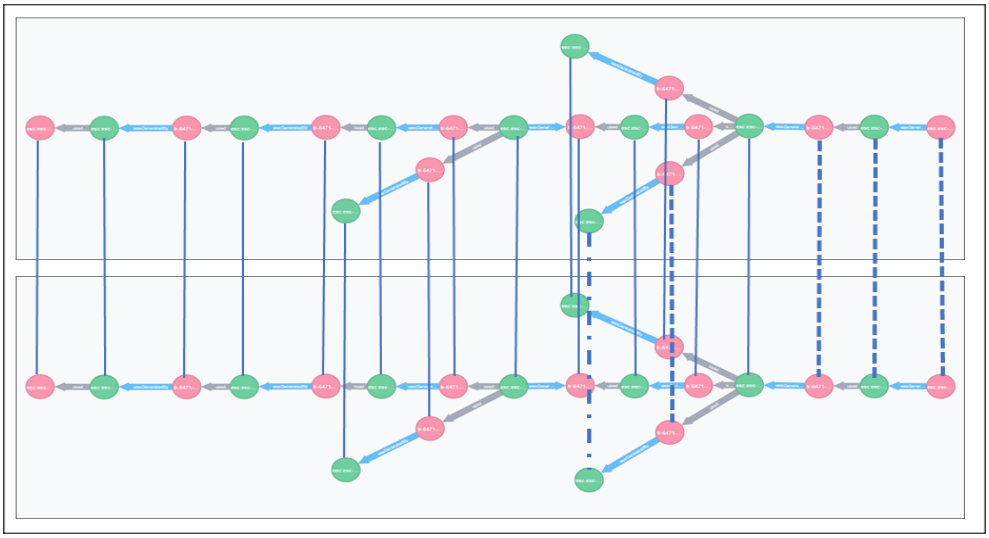
X. Research Outputs

A. Usecase - Freebayes Variant Calling

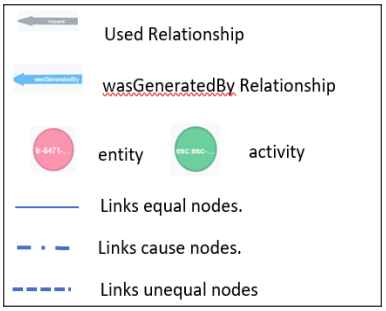
The use case we worked with was the Freebayes variant calling workflow. It processes aligned patient sequence files (in the BAM format) in the search for variants from the reference genome. The main source of change in this workflow is the version of Freebayes tool used (https://github.com/ekg/freebayes). Over the last three years the author has released about 15 versions of the program and so an important question is whether and to what extent these changes affect the output.

ALGORITHM 2: Backward Traversal

Input: Invocation1_Id
Input: Invocation2_Id
Output: Δ Graph
$first\_hop = true$;
do
  if $first\_hop == true$ **then**
    $List < Activity > Activity1List \leftarrow findLastActivityList(Invocation1\_Id)$;
    $List < Activity > Activity2List \leftarrow findLasttActivityList(Invocation2\_Id)$;
  else
    $List < Activity > Activity1List \leftarrow firstActivityNewList$;
    $List < Activity > Activity2List \leftarrow secondActivityNewList$;
    $ClearfirstActivityNewList$ & $secondActivityNewList$;
  end
  foreach $(A1, A2) \in zip(Activity1List, Activity2List)$ **do**
    $\Delta Graph \leftarrow Compare(A1, A2)$;
    $List < Entity > Entity1List \leftarrow getUsed(A1)$;
    $List < Entity > Entity2List \leftarrow getUsed(A2)$;
    foreach $(E1, E2) \in zip(Entity1List, Entity2List)$ **do**
      $List < Activity > returnedActivityList1 \leftarrow getWasGeneratedBy(E1)$;
      $List < Activity > returnedActivityList2 \leftarrow getWasGeneratedBy(E2)$;
      foreach $(A1, A2) \in zip(returnedActivityList1, returnedActivityList2)$ **do**
        $\Delta Graph \leftarrow Compare(A1, A2)$;
        $Add\ A1\ to\ firstActivityNewList$;
        $Add\ A2\ to\ secondActivityNewList$;
      end
    end
  end
while $firstActivityNewList \neq Empty$ && $secondActivityNewList \neq Empty$;
Function *Compare (A1, A2)*
  if $A1.configuration \neq A2.configuration$ **then**
    $\Delta Graph \leftarrow (A1, A2)$;
  end
  return $\Delta Graph$
Function *findLastActivityList(InvocationId)*
  $List\ all\ activityNodes\ WHERE\ entity - [WASGENERATEDBY] - activity\ WHERE\ NOT\ any\_activity - [USED] - entity\ AND\ activity.id\ CONTAINS\ InvocationId$;
  return activityNodes



(a) Difference Graph of Freebayes



(b) Notations

Fig. 6: Comparison graph of Freebayes and notations used

1) Evaluation of Freebayes Workflow: In fig 6, we have visualised the graphs of the two Freebayes invocations of which the top graph represents invovation 6471 whereas the bottom graph represents invocation 5463. The difference between the invocations are such that 6471 has used Freebayes library of version $Freebayes - 1.0.2$ whereas 5463 has used $Freebayes - 0.9.10$. To spot similar, dissimilar and cause nodes, we have used three types of lines. The straight line refers the two linked nodes are equal. "Dashed" lines link the nodes that are not equal. "Dashed dot" lines link the nodes that actually caused the difference.

We have tested the algorithm with another usecase "Simple Variant Interpretation (SVI)", which was implemented as part of Re-Comp project (recomp.org.uk). SVI is a workflow which takes advantage of two external data sources OMIM GeneMap and NCBI ClinVar to provide interpretation of human variants to facilitate clinical diagnosis of genetic diseases say for example, Alzheimer's disease. We have observed above stated Why-Diff scenario in SVI Workflow. For more results on SVI, please visit https://github.com/Priyaa140624191/Why-Diff/blob/master/WhyDiffAlgorithm/SVI%20Results.pdf

References

[1] M. Schwab, N. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science & Engineering*, vol. 2, no. 6, pp. 61–67, 2000.

[2] N. Barnes, "Publish your computer code: it is good enough," *Nature*, vol. 467, no. 7317, pp. 753–753, 2010.

[3] J. P. Mesirov, "Accessible Reproducible Research," vol. 327, no. January, pp. 415–417, 2010.

[4] A. Morin, J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, and P. Sliz, "Research priorities. Shining light into black boxes." *Science (New York, N.Y.)*, vol. 336, no. 6078, pp. 159–60, 2012.

[5] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson, "Best Practices for Scientific Computing," *PLoS Biology*, vol. 12, no. 1, 2014.

[6] J. Vitek and T. Kalibera, "Repeatability, Reproducibility, and Rigor in Systems Research," *Proceedings of the ninth ACM international conference on Embedded software - EMSOFT '11*, pp. 33–38, 2011.

[7] D. C. Drummond, "Replicability is not reproducibility: Nor is it good science," *Proceedings of the Evaluation Methods for Machine Learning Workshop 26th International Conference for Machine Learning*, no. 2005, pp. 1–4, 2009.

[8] D. G. Feitelson, "From repeatability to reproducibility and corroboration," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 3–11, 2015.

[9] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva, "Bridging workflow and data provenance using strong links," in *International Conference on Scientific and Statistical Database Management*. Springer, 2010, pp. 397–415.

[10] H. Hiden, S. Woodman, P. Watson, and J. Cala, "Developing cloud applications using the e-science central platform," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1983, 2012. [Online]. Available: http://rsta.royalsocietypublishing.org/content/371/1983/20120085

[11] J. Kim, E. Deelman, Y. Gil, G. Mehta, and V. Ratnakar, "Provenance trails in the wings/pegasus system," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 5, pp. 587–597, 2008.

[12] E. Afgan, D. Baker, M. van den Beek, D. Blankenberg, D. Bouvier, M. ech, J. Chilton, D. Clements, N. Coraor, C. Eberhard, B. Grning, A. Guerler, J. Hillman-Jackson, G. Von Kuster, E. Rasche, N. Soranzo, N. Turaga, J. Taylor, A. Nekrutenko, and J. Goecks, "The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update," *Nucleic Acids Research*, vol. 44, no. W1, p. W3, 2016. [Online]. Available: +http://dx.doi.org/10.1093/nar/gkw343

[13] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "Vistrails: Visualization meets data management," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '06. New York, NY, USA: ACM, 2006, pp. 745–747. [Online]. Available: http://doi.acm.org/10.1145/1142473.1142574

[14] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, no. 10, pp. 1039–1065, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1002/cpe.v18:10

[15] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: Lessons in creating a workflow environment for the life sciences: Research articles," *Concurr. Comput. : Pract. Exper.*,

vol. 18, no. 10, pp. 1067–1100, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1002/cpe.v18:10

[16] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva, "Bridging workflow and data provenance using strong links," in *International Conference on Scientific and Statistical Database Management*. Springer, 2010, pp. 397–415.

[17] T. U. P. Granularities, "Provenance and annotation of data and processes," *Lecture Notes in Computer Science*, vol. 7525, 2012.

[18] T. Miksa, R. Mayer, and A. Rauber, "Ensuring sustainability of web services dependent processes," *International Journal of Computational Science and Engineering*, vol. 10, no. 1-2, pp. 70–81, 2015.

[19] D. McGuinness, T. Lebo, and S. Sahoo, "PROV-o: The PROV ontology," W3C, W3C Recommendation, Apr. 2013. [Online]. Available: http://www.w3.org/TR/prov-o/

[20] T. Miksa, S. Pröll, R. Mayer, S. Strodl, R. Vieira, J. Barateiro, and A. Rauber, "Framework for verification of preserved and redeployed processes," 2013.

[21] A. Davison, "Automated capture of experiment context for easier reproducibility in computational research," *Computing in Science & Engineering*, vol. 14, no. 4, pp. 48–56, 2012.

[22] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. Dey, J. Freire, D. Huntzinger, C. Jones, D. Koop, P. Missier, M. Schildhauer, C. Schwalm, Y. Wei, J. Cheney, M. Bieda, B. Ludaescher, and B. Lud, "YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts," *International Journal of Digital Curation*, vol. 10, no. 1, pp. 1–15, 2015.

[23] J. Pimentel, V. Braganholo, L. Murta, and J. Freire, "Collecting and Analyzing Provenance on Interactive Notebooks: when IPython meets noWorkflow," *Usenix.Org*, pp. 3–8, 2015.

[24] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, and C. Tilmes, "Prov-dm: The prov data model," Tech. Rep., 2012. [Online]. Available: http://www.w3.org/TR/prov-dm/

[25] W. Oliveira, P. Missier, K. Ocaña, D. de Oliveira, and V. Braganholo, "Analyzing provenance across heterogeneous provenance graphs," in *International Provenance and Annotation Workshop*. Springer, 2016, pp. 57–70.

[26] P. Thavasimani and P. Missier, "Facilitating reproducible research by investigating computational metadata," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3045–3051.