Optimized Storing of Workflow Outputs through Mining Association Rules

Debasish Chakroborti, Manishankar Mondal, Banani Roy, Chanchal K. Roy, Kevin A. Schneider
Department of Computer Science, University of Saskatchewan
Email: {debasish.chakroborti, mshankar.mondal, banani.roy, chanchal.roy, kevin.schneider}@usask.ca

***Abstract*— Nowadays, workflows are being frequently built and used for systematically processing large datasets in workflow management systems (WMS). A workflow (i.e., a pipeline) is a sequential organization of a finite set of processing modules that are applied on a particular dataset for producing a desired output. In a workflow management system, the users generally create workflows manually for their own investigations. However, workflows can sometimes be lengthy and the constituent processing modules might often be computationally expensive. In this situation, it would be beneficial if a user could reuse intermediate stage results generated by previously executed workflows for executing his current workflow.**

In this paper, we propose a novel technique based on association rule mining for suggesting which intermediate stage results from a workflow that a user is going to execute should be stored for reusing in future. We call our proposed technique, RISP (Recommending Intermediate States from Pipelines) in this paper. According to our investigation on hundreds of workflows from two scientific workflow management systems, our proposed technique can efficiently suggest to store intermediate state results for reusing in future. The results that we can store have a high reuse frequency. Moreover, for creating around 51% of the entire pipelines, we can reuse results suggested by our technique. Finally, we can achieve a considerable gain (74% gain) in execution time through reusing intermediate results stored by the suggestions provided by our proposed technique. We believe that our technique (RISP) has the potential to have a significant positive impact on Big-Data systems, because it can considerably reduce execution time of the workflows through appropriate reuse of intermediate state results, and hence, can improve the performance of the systems.

***Index Terms*— Plant phenotyping, Association rules, Workflow, Intermediate states suggestion, Pipeline design**

I. Introduction

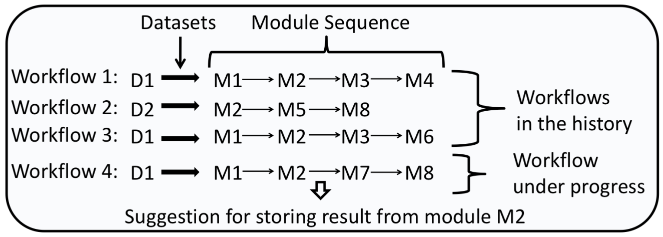
A scientific workflow management system (SWfMS) is a special type of WMS (workflow management system) that lets its users perform computationally expensive and time-consuming tasks by decomposing them into sequentially organized and inter-dependent modules. Our research in this paper deals with workflows (i.e., pipelines) in scientific workflow management systems. Currently, scientific workflow management systems are being commonly used in various scientific, engineering, and business organizations for conducting investigations, improving system efficiency, increasing productivity by reducing costs, and improving information exchange. In Big-data analytics, when a large volume of heterogeneous data needs to be processed with various mechanisms, scientific workflow management systems (SWfMS) should be considered for efficiency.

In a SWfMS, users can manually build their workflows by selecting and sequentially adding processing modules from a finite set of modules available in the SWfMS for performing their desired investigations. Each workflow or pipeline works on a particular dataset provided by a user. The processing modules in a workflow are ordered in such a way that the output produced by a particular intermediate module can be used as input by the next module in the workflow. The output that we obtain from the last module is considered the final output from the workflow. Users often create workflows for processing large datasets. The intermediate state results produced by the modules in such workflows can also be very large. Moreover, each of the processing modules might require a considerable amount of time for data processing. In such a situation, it would be beneficial for a user if he could reuse results produced by previous workflows that were executed on the same dataset that he is going to process.

In order to provide automatic support for reusing results from already executed workflows, we need to have a mechanism for determining which of the intermediate state results obtained from a particular workflow have a high possibility of being reused in future. In our research presented in this paper, we propose such a mechanism (i.e., technique) which we call RISP (Recommending Intermediate States from Pipelines). Our proposed mechanism provides suggestions for storing intermediate state results by analyzing association rules between data and processing modules from the pipelines in the history. To the best of our knowledge, our study is the first one to investigate providing suggestions for storing intermediate state results from pipelines.

Existing studies have stored all the intermediate state results from pipelines. While such a mechanism (storing all intermediate state results) is good for provenance, it is not suitable from the perspective of reusability. For storing all intermediate state results from all pipelines, we need a significant amount of storage space. As new pipelines will be created, the size of the stored results will continuously increase. Also, it might be seen that many of the stored intermediate state results are not being reused at all. On the other hand, if we do not store any of the intermediate state results, we might need to build and execute the same workflows again and again. This might have a significant negative impact on the efficiency when the processing modules in the workflows are time-consuming. In this situation, our proposed technique, RISP, can be useful. It suggests intermediate state results for storing by analyzing their reuse possibility through mining association rules. The following example will explain our idea.

Let us assume that a SWfMS without any mechanism for



Datasets
Module Sequence
Workflow 1: D1 M1 M2 M3 M4
Workflow 2: D2 M2 M5 M8
Workflow 3: D1 M1 M2 M3 M6
Workflow 4: D1 M1 M2 M7 M8
Workflows in the history
Workflow under progress
Suggestion for storing result from module M2

Fig. 1. Automatically suggesting intermediate state results for storing from a workflow under progress.

storing intermediate state results has been used three times for executing three workflows as shown in Fig. 1. A user is now going to execute the fourth workflow. In such a situation, if we integrate RISP with this SWfMS, then it will suggest for storing the result that will be obtained from module, M2, of the fourth workflow. The reason behind making this suggestion is that the dataset D1 that is going to be processed in the fourth workflow was also processed in the first and third workflows and the modules M1 and M2 were executed serially in both of these workflows. Thus, there is a high possibility that when a user will attempt to create a workflow in future using dataset D1, he will first apply the two modules M1 and M2 serially on D1. Moreover, during executing the fourth workflow, if the result from M2 is stored in the system, then in future when a user will attempt to create a workflow using dataset D1, RISP will notify him about the presence of the result that was stored from the fourth workflow. Section III describes our suggestion making technique.

We implement our proposed technique, RISP, as a prototype tool and apply it on hundreds of pipelines created and used by the researchers and users in two scientific workflow management systems. We have the following findings:

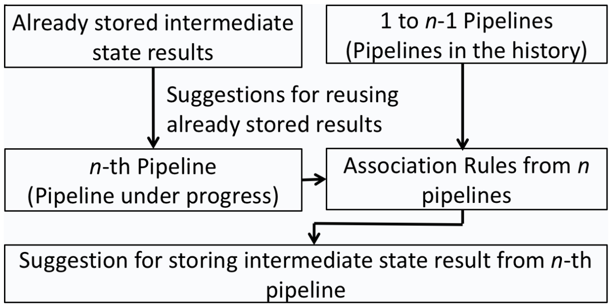
(1) By analyzing association rules from the previously executed pipelines, our proposed technique can automatically suggest which intermediate state results from a pipeline under progress should be stored for reusing in future.

(2) The intermediate state results that can be stored according to the suggestions from our proposed technique have a high frequency of being reused. The 508 pipelines that we investigated had 7165 possible intermediate state results. However, our technique suggests storing only 49 of these. Each of these 49 results can be reused 5 times on an average. The stored results can be reused for creating and executing around 51% of the entire set of pipelines.

(4) By storing intermediate state results according to the suggestions from our proposed recommendation technique we can have a considerable gain (around 74% gain) in execution time while executing pipelines.

Our findings indicate that our proposed technique (RISP) can significantly improve the performance of Big-Data systems through appropriate reuse of the intermediate state results from the workflows.

The rest of the paper is organized as follows. Section II defines and describes association rules, Section III presents our proposed technique for suggesting intermediate state results to store for reuse, Section IV describes our experiment setup, Section V compares our proposed technique with



Already stored intermediate state results
1 to *n*-1 Pipelines (Pipelines in the history)
Suggestions for reusing already stored results
n-th Pipeline (Pipeline under progress)
Association Rules from *n* pipelines
Suggestion for storing intermediate state result from *n*-th pipeline

Fig. 2. Our proposed technique for suggesting intermediate state results to store for reusing in future

three other techniques in suggesting intermediate state results to store, Section VI describes possible threats to validity, Section VII discusses the related work, and finally, Section VIII concludes the paper by mentioning future work.

II. Background

Association rules have been used in software engineering research and practice for performing impact analysis tasks. We define an association rule in the following way.

Association Rule. An association rule [1] is an expression of the form $X => Y$ where X is the antecedent, and Y is the consequent. Each of X and Y is a set of one or more program entities. The meaning of such a rule is that if X gets changed in a particular commit operation, Y also has the tendency of being changed in that commit.

Support and Confidence. As defined by Zimmermann et al. [2], *support is the number of commits in which an entity or a group of entities changed together*. The support of an association rule is determined in the following way.

$$\mathit{support}(X => Y) = \mathit{support}(X, Y) \quad (1)$$

Here, (X, Y) is the union of X and Y, and so $\mathit{support}(X => Y) = \mathit{support}(Y => X)$. *Confidence of an association rule, $X => Y$, determines the probability that Y will change in a commit operation provided that X changed in that commit operation*. We determine the confidence of $X => Y$ in the following way.

$$\mathit{confidence}(X => Y) = \mathit{support}(X, Y)/\mathit{support}(X) \quad (2)$$

In our research, we derive association rules between datasets and modules from pipelines and investigate those for providing suggestions regarding which intermediate state result from a pipeline under progress should be stored.

III. Recommending intermediate stage results to store for reusing

Let us assume that a user is going to create the n-th pipeline in a scientific workflow management system (SWfMS) as shown in Fig. 2. Let us further assume that during the creation of each of the previous $n-1$ pipelines, our recommendation system (RISP) recommended particular intermediate state results to store. Thus, we already have some stored results. The top left rectangle in Fig. 2 denotes these already stored results. When the user attempts to create the n-th pipeline, our recommendation system observes which dataset she is going to use and determines whether there are any stored intermediate state results using this dataset. These results are shown to the user so that she can decide whether she wants to reuse any of these already stored results. Let us assume that the user completes her pipeline with or without using the already stored intermediate state results. Just after she completes her pipeline (i.e., the n-th pipeline), our recommendation system determines all the distinct association rules from all the n pipelines (i.e., $n-1$ previous pipelines and the n-th pipeline that the user is going to execute), calculates their supports and confidences, and analyzes these support and confidence values for recommending which of the intermediate state results from the newly created pipeline (i.e., n-th pipeline) should be stored. On the basis of this recommendation, one intermediate state result from the n-th pipeline might be stored if that was not stored previously. In this way, when a user attempts to create a new pipeline in SWfMS, our recommendation technique performs the following two tasks:

- Recommending a number of already stored intermediate stage results to the user so that she can reuse it.
- Recommending an intermediate stage result for storing from the pipeline she created by analyzing association rules from the pipelines in the history.

A. Determining association rules from pipelines

We derive association rules among datasets and modules by analyzing existing pipelines in the following way. Let us consider the first pipeline in Fig. 1. It consists of four modules (M1, M2, M3, and M4) that sequentially work on the dataset D1. The results generated from each of these modules except M4 are intermediate state results. The result that we obtain from M4 is the final result from the pipeline. It is possible to store results from each of these four modules. If we store result from a particular module, then it might discard the necessity of executing any previous modules including that particular one whenever a user needs to execute a similar pipeline in future with the same dataset. For example, if we store the result obtained from module M3, then for executing the possible pipeline (D1− >M1− >M2− >M3− >M7) in future, a user does not need to execute the first three modules. He can just reuse the previously stored result from module M3 and can only execute module M7 on the stored result.

We determine association rules from a pipeline on the basis of how many results can be stored from it. For example, from the first pipeline in Fig. 1, we determine the following four association rules: **(1)** D1=>M1, **(2)** D1=>[M1, M2], **(3)** D1=>[M1, M2, M3], and **(4)** D1=>[M1, M2, M3, M4]. For the first rule, D1 is the **antecedent** and M1 is the **consequent**. For the second rule, the consequent is the module sequence (M1, M2). An association rule, for example the second one, means that if a user attempts to make a pipeline using the dataset D1 in future, then he has a possibility of applying the modules M1 and M2 sequentially on D1 as the first two modules. From all four pipelines in Fig. 1, we get ten distinct association rules: (1) D1=>M1, (2) D1=>[M1, M2], (3) D1=>[M1, M2, M3], (4) D1=>[M1, M2, M3, M4], (5) D2=>M2, (6) D2=>[M2, M5], (7) D2=>[M2, M5, M8], (8) D1=>[M1, M2, M6], (9) D1=>[M1, M2, M7], and (10) D1=>[M1, M2, M7, M8].

B. Determining the supports and confidences of the association rules obtained from the pipelines

After determining all the distinct association rules from all the pipelines, we determine the support and confidence of each of the rules. For example, we will now determine the support and confidence of the association rule D1=>M1.

Support of an association rule. Support of an association is the number of times it can be generated from the pipelines. The support of the association rule, D1=>M1, is 3 because, from all the four pipelines in Fig. 1, we can generate this association rule three times (i.e., from the first, third, and fourth pipelines). In the same way, the supports of the association rules, D1=>[M1, M2] and D1=>[M1, M2, M3], are 3 and 1 respectively. We express the support of an association rule formally in the following way.

$$support(D1 => M1) = 3 \quad (3)$$

Confidence of an association rule. We determine the confidence of the association rule D1=>M1 in the following way from its support value:

$$confidence(D1 => M1) = \frac{support(D1 => M1)}{support(D1)} \quad (4)$$

Here, support(D1) is the number of times D1 was used in the pipelines. Thus, support (D1) = 3, because D1 was used in three pipelines according to Fig. 1. Finally, confidence (D1=>M1) = 3/3 = 1. The highest confidence of an association rule is 1. Such a confidence for the association rule D1=>M1 implies that if some one attempts to make a pipeline in future using the dataset D1, then there is a high probability that he will choose M1 as the first module in the pipeline. The confidence of the association rule D1=>[M1, M2, M3] is 0.33 (confidence(D1=>[M1, M2, M3]) = support(D1=>[M1, M2, M3])/support(D1) = 1/3). Thus, if a user attempts to build a pipeline in future using the dataset D1, then there is a little probability (0.33) that he will use M1, M2, and M3 respectively as the first three modules in his pipeline.

C. Recommending an intermediate state result for storing

In Fig. 1, we see that the fourth pipeline is the one that is going to be executed (under progress). For recommending which intermediate state result should be stored from this pipeline, we sort the association rules made from this pipeline on the basis of their confidence values and determine the highest confidence rules. For example, from the fourth pipeline in Fig. 1 we get four association rules: D1=>M1, D1=>[M1, M2], D1=>[M1, M2, M7], and D1=>[M1, M2, M7, M8] with confidences 1, 1, 0.33, and 0.33 respectively. The highest confidence rules are D1=>M1, D1=>[M1, M2]. We select the longest of these highest confidence rules for making suggestion because it helps us skip the highest number of modules. Thus, from the fourth pipeline, we recommend to store the result obtained from module M2.

IV. Experiment Setup

For conducting our experiment, we downloaded 508 workflows from Galaxy public server [3] at *usegalaxy.org*. These workflows were created and executed on the SWfMS of Galaxy during the time between February, 2010 and August, 2018. The downloaded workflows were text-based files with a specific JSON format. We automatically retrieved the module execution sequence and dataset details of each workflow from its corresponding file using our implementation. The workflows were executed mostly for investigations related to bioinformatics with biological datasets as the inputs. We applied our recommendation technique (RISP) on these workflows to investigate how efficiently it can recommend intermediate state results for storing so that the results can be reused in future. Section V-C describes our investigation.

We also integrated our proposed technique with the SWfMS in the Plant Phenotyping and Imaging Research Center called P2IRC in USASK. This SWfMS runs on an OpenStack based Spark-Hadoop cluster having 6 nodes, 40 cores, and 40 GB RAM. It is frequently used for making pipelines to analyze and investigate large volumes of image data. The image data is stored in the Hadoop Distributed File System (HDFS) of the cluster. For investigating how much gain in execution time can be achieved by storing intermediate state results according to the suggestions provided by our proposed technique, we first integrated our technique (RISP) with the SWfMS, and then executed 32 image processing pipelines. The input dataset of each of these 32 pipelines consists of 4000 to 10000 images. The pipelines were created for different purposes such as image segmentation, image registration, counting the number of flowers, and image clustering. The Canola datasets (4KCanola, 10KCanola) of the P2IRC project of USASK were used as inputs for the pipelines. The findings of our investigation regarding execution time gain will be described in Section V-D.

V. Evaluating our proposed technique

A. *Candidate techniques for comparison*

We compared our proposed technique (RISP) with three other candidate recommendation techniques by doing investigation on 508 pipelines that we downloaded from Galaxy public server [3]. All four recommendation techniques (including our proposed one) are listed below.

- **PT** (Proposed Technique): Our proposed technique recommends storing intermediate state results indicated by the association rules with highest confidence.
- **TSAR** (Technique that recommends Storing All Results): This second technique recommends storing each of the intermediate state results of each of the pipelines.
- **TSPAR** (Technique that recommends Storing Previously Appeared Results): The third technique recommends storing those intermediate state results that were produced previously at least once.
- **TSFR** (Technique that recommends Storing the Final Result): This technique recommends storing the final result (output) from a pipeline.

The following paragraphs describe how we evaluate these techniques for making a comparison among them.

Procedure of investigation using PT (Proposed technique). For evaluating this technique, we analyze each of the pipelines in the history serially beginning from the first one. While examining the n-th pipeline, we analyze our technique's recommendation capability in the following way.

First, we apply our technique to determine which intermediate state results might have already been stored if our technique was integrated with the SWfMS from the very beginning of the history. For this purpose, our system analyzes association rules from the previous n-1 pipelines. If we see that any of the intermediate state results of the n-th pipeline have already been stored previously, then we can reuse that intermediate state result in this n-th pipeline, and also, we realize that our system previously took a correct decision regarding storing those intermediate state results.

After completing the first step, we make association rules from this n-th pipeline and determine their supports and confidences by dealing with the association rules from the previous $n-1$ pipelines. We select the highest confidence rule(s) from the n-th pipeline and assume that we have stored the intermediate state result denoted by the longest of these highest confidence rules. In this way, we examine all the pipelines in the history and determine the number of cases where we could reuse previously stored intermediate state results if we could use our proposed technique.

Procedure of investigation using TSAR (Technique that recommends Storing All Results). By applying this second technique we analyze the pipelines from the very beginning one serially. When analyzing the n-th pipeline, we first check the database to see if any of the stored intermediate state results can be useful for skipping some processing modules in the pipeline. If several results are available, then the result that helps us skip the highest number of processing modules is used. After executing the pipeline, we store each of its intermediate state results in the database. If a particular result is already in the database, we do not need to store it again.

Procedure of investigation using TSPAR (Technique that recommends Storing Previously Appeared Results). The third candidate technique is a variant of our proposed technique. While our proposed technique depends on the confidence values of the association rules for deciding which of the intermediate state results should be stored from the pipeline under progress, the third technique solely depends on the support values of the association rules. When analyzing the n-th pipeline, this technique first checks which of the already stored intermediate state results can be the most appropriate one for reusing for this pipeline. Then, it determines the association rules from the pipeline. It identifies which of the rules have a support of at least one in the previous usage history (i.e., first $n-1$ pipelines). The intermediate state result indicated by the longest one of these association rules is considered for storing.

Procedure of investigation using TSFR (Technique that recommends Storing the Final Result). The fourth technique stores the final outcome of each of the pipelines. Thus, if the same pipeline is attempted to be re-executed in future, then we can just reuse the result from the first execution. We do not need to again execute any module in the pipeline. We consider such a technique for comparison because we wanted to understand how often the same pipelines get executed.

B. Investigated measures

We calculate four measures for our investigation for each of the candidate systems. The measures and their calculation mechanisms have been discussed below.

- **LR** (Likeliness of Reusing from previously stored results). This measure determines how often we can reuse intermediate state results that can be stored according to the suggestions from a candidate system.
- **PSRR** (Percentage of Stored Results that were Reused). The second measure determines what percentage of the intermediate state results that can be stored according to the suggestions from a candidate recommendation technique can be reused in future.
- **FRSR** (Frequency of Reusing Stored Results). This measure determines how many times a stored intermediate state result was reused on an average.
- **PISRS** (Percentage of Intermediate State Results that were Stored). This measure calculates what percentage of the intermediate state results were stored for reusing according to the suggestions from a candidate recommendation technique.

Calculation mechanism for LR. By sequentially analyzing all the pipelines from the history, we determine two quantities: (1) the total number of pipelines that we have analyzed, (2) the number of pipelines for which we could reuse previously stored intermediate state results. From these quantities we calculate LR using the following equation.

$$LR = \frac{\begin{array}{c}Number\ of\ pipelines\ for\ which\ we\\ could\ reuse\ previously\ stored\ results\end{array}}{Total\ number\ of\ pipelines} \times 100 \quad (5)$$

Calculation mechanism for PSRR. If we see that only a very small percentage of the previously stored results get reused (most of the stored results remain unused) during creating pipelines, then the corresponding recommendation mechanism should not be considered as an efficient one. Thus, when comparing the efficiencies of the candidate recommendation mechanisms, we should also focus on the second measure (PSRR). It determines what percentage of the stored results get reused during creating pipelines. By examining all the pipelines in the history we determine two quantities: (1) the total number of intermediate state results that were stored by a candidate mechanism and (2) the number of intermediate state results that could be reused. We then calculate PSRR by the following equation.

$$PSRR = \frac{No.\ of\ reused\ results}{No.\ of\ stored\ results} \times 100 \quad (6)$$

Calculation mechanism for FRSR. The measure FRSR (Frequency of Reusing Stored Results) focuses on how many

TABLE I

WORKFLOW INFORMATION FOR THE COMPARISON

Description of calculated measures	PT	TSAR	TSPAR	TSFR
Number of investigated pipelines considering each candidate technique	508	508	508	508
Number of pipelines for which we could reuse previously stored data	264	314	261	70
Number of intermediate state results that were saved according to the suggestion from a candidate technique	49	7165	159	457
PT = Proposed Technique **TSAR** = Technique that Recommends Storing All intermediate state Results **TSPAR** = Technique that Recommends Storing Previously Appeared Results **TSFR** = Technique that Recommends Storing the Final Result				

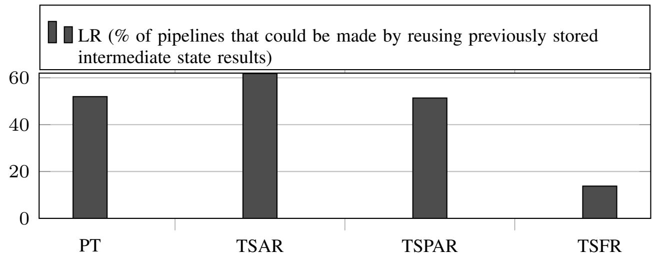


Fig. 3. Comparing the candidate techniques according to the percentage of pipelines that could be made by reusing previously stored results.

times a previously stored result was used during creating pipelines. If it is observed that the intermediate state results stored according to the recommendation from a candidate technique was used rarely during creating pipelines, then the technique should not be regarded as an efficient one. For calculating FRSR, we determine two quantities by examining the entire set of pipelines: (1) the total number of intermediate state results that were stored from the recommendations of a candidate technique and (2) the total number of times the stored results were used for making pipelines. We then calculate this measure according to the following equation.

$$FRSR = \frac{No.\ of\ times\ stored\ results\ were\ reused}{No.\ of\ stored\ results} \quad (7)$$

Calculation mechanism for PISRS. The measure PISRS (Percentage of Intermediate State Results that were Stored) calculates what percentage of the intermediate state results that were produced during the execution of the entire set of pipelines were stored according to the recommendation of a candidate technique. A candidate technique that stores comparatively less number of results than other techniques but ensures higher re-usability should be considered an efficient one. For measuring PISRS, we examine the whole set of pipelines and determine the following two quantities for each recommendation technique: (1) the total number of possible intermediate states (including the final states), (2) the total number of intermediate state results that were stored. We then calculate PISRS using the following equation.

$$PISRS = \frac{No.\ of\ stored\ results}{No.\ of\ intermediate\ states} \times 100 \quad (8)$$

C. Comparing the candidate recommendation techniques on the basis of the measures

The following paragraphs compare the four candidate techniques (PT, TSAR, TSPAR, and TSFR) on the basis of the four measures (LR, PSRR, FRSR, and PISRS).

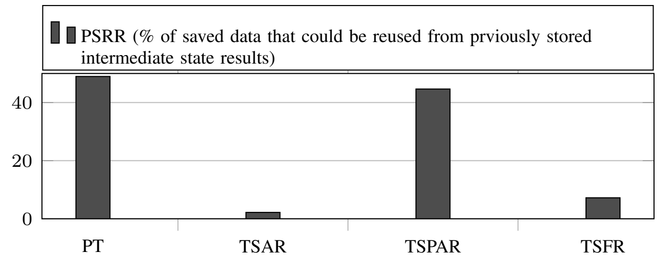


Fig. 4. Comparing the candidate techniques on the basis of the percentage of saved data that could be reused from previously stored results

Comparison regarding LR (Likeliness of Reusing from previously stored results) For each candidate technique, Fig. 3 shows the percentages of pipelines such that while building those pipelines we could reuse previously stored intermediate state results. We can see that for the second candidate technique (TSAR), the percentage of pipelines (61.81%) that could be built using previously stored results is the highest among the four techniques. However, to achieve this percentage, we need to store all 7165 intermediate state results from 508 workflows (as indicated in Table I), and a SWfMS can face a huge storage overhead for storing such a big amount of results. For our proposed technique (PT), the likeliness of reusing from stored results is around 51.97%. Although this percentage is smaller compared to TSAR, we can achieve this by storing only 49 intermediate state results. We also see that the percentage regarding our proposed technique is higher than the percentages regarding the third and fourth candidate techniques. From Table I we see that the third and fourth techniques suggest storing 159 and 457 intermediate state results respectively. Thus, even by storing a considerably smaller number of intermediate state results, our technique ensures a higher likeliness of reusing.

Comparison regarding PSRR (Percentage of Stored Results that were Reused) Fig. 4 makes a comparison among the candidate techniques by considering the PSRR measure. We see that the PSRR value regarding our proposed technique (PT) is the highest among all the techniques. The second candidate technique TSAR (that saves all the results) exhibits the lowest value (2.19%). Thus, only 2.19% of the intermediate state results stored according to the suggestion from TSAR can be reused. The remaining 97.81% of the stored results (i.e., around 7008 of the 7165 stored results) stay unused. Finally, our proposed technique ensures the highest reuse of stored results.

Comparison regarding FRSR (Frequency of Reusing Stored Results). Fig. 5 illustrates how frequently the intermediate state results stored according to the suggestion from a candidate technique get reused during making pipelines. From the figure we again realize that our proposed technique (PT) exhibits the highest reuse frequency (5.39) among all four techniques. Finally, our comparison in Fig. 5 establishes our proposed technique to be the best one.

Comparison regarding PISRS (Percentage of Intermediate State Results that were Stored) Fig. 6 compares the candidate techniques on the basis of the PISRS measure. According to our definition of PISRS, the technique with

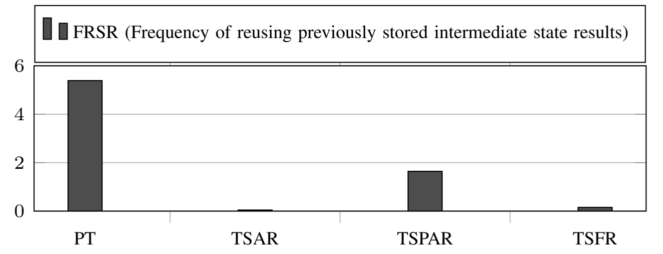


Fig. 5. Comparing the candidate techniques on the basis of the frequency of reusing previously stored intermediate state results

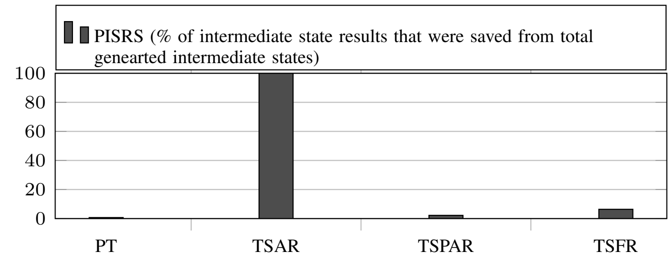


Fig. 6. Comparing the candidate techniques on the basis of the percentage of intermediate state results that were saved from all intermediate states

the lowest PISRS value should be regarded as the most efficient one. Fig. 6 shows that our proposed technique (PT) suggests storing the smallest percentage (0.68%) of all 7165 intermediate state results. According to Table I, this percentage (0.68%) indicates storing of 49 results only. The percentage regarding candidate technique TSAR is 100% because it recommends storing all 7165 results. The PISRS value for each of the other two techniques is higher compared to our proposed technique. Thus, even according to this last measure (PISRS) our proposed technique performs the best.

Investigating how often we can skip processing modules by reusing previously stored results. We also wanted to visually investigate how often the intermediate state results stored by the suggestions from our proposed technique can be reused to skip some of the processing modules in a pipeline under progress. For this purpose, we draw the bar graph in Fig. 7. The bars in the graph indicate how many processing modules could be skipped for building pipelines by reusing previously stored results. The graph indicates the followings:

(1) We can often reuse the intermediate state results that were stored previously according to the suggestions from our proposed recommendation technique. According to the figure, we could first reuse previously stored result for executing the ninth pipeline. From the thirty sixth pipeline, reusing became more frequent. The graph also implies that reuse frequency increases with the increase of pipelines. In general, as more pipelines get created, more intermediate state results get stored, and eventually, the frequency of reusing increases as well.

(2) We can often skip a good number of processing modules by reusing previously stored results. The highest number of modules that could be skipped is 15.

D. Considering module execution time for evaluating our proposed recommendation technique

Execution times of the processing modules in a pipeline should be considered important when making decisions about

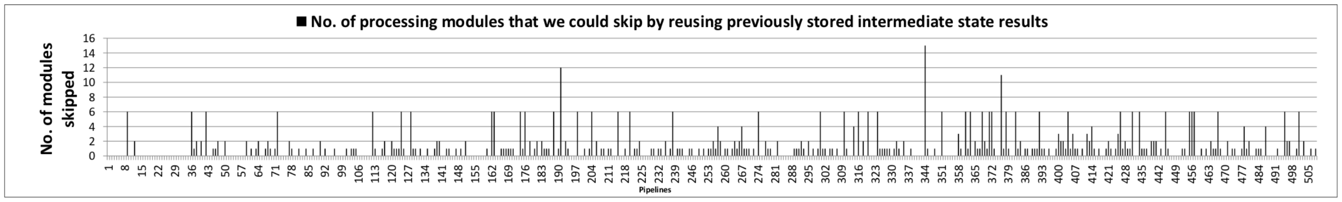


Fig. 7. No. of processing modules that could be skipped by reusing intermediate state results that could be stored according to our proposed technique

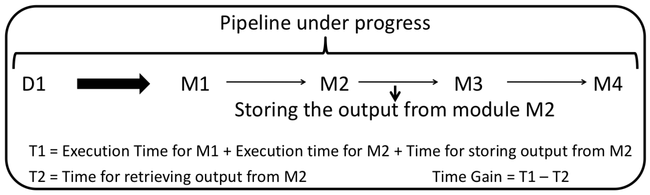

Pipeline under progress
D1
M1
M2
M3
M4
Storing the output from module M2
T1 = Execution Time for M1 + Execution time for M2 + Time for storing output from M2
T2 = Time for retrieving output from M2
Time Gain = T1 – T2

Fig. 8. Calculation of execution time gain

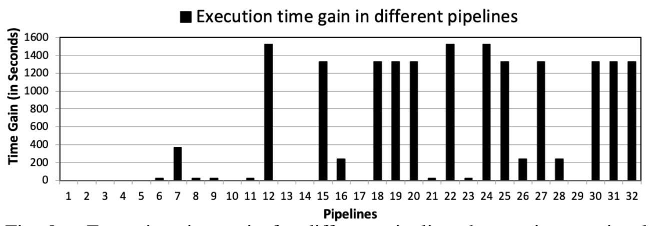


Fig. 9. Execution time gain for different pipelines by reusing previously stored intermediate state results according to our proposed technique

which intermediate state results we should store from a pipeline. We consider module execution time to evaluate our proposed technique in the following way.

Let us consider the pipeline in Fig. 8. A recommendation technique recommends to store the result obtained from module M2. Let us assume that T1 is the time which is required to execute M1 and M2 and store results from M2 to HDFS (Hadoop Distributed File System). T2 is the time to retrieve the result of M2 from HDFS. Now, only if T1 is greater than T2, then storing the result from M2 is beneficial. Eq. 9 calculates the execution time gain in this case.

$$Execution\ Time\ Gain = T1 - T2 \tag{9}$$

We consider this timing factor for determining whether storing an intermediate state result from a pipeline according to the suggestion from our technique is beneficial.

Our investigation regarding module execution time. For conducting this investigation, we executed 32 pipelines in the scientific workflow management system (SWfMS) of a plant phenotyping based image research center (P2IRC). This SWfMS runs on a parallel and distributed environment empowered by a SPARK-Hadoop cluster consisting of 6 nodes, 40 cores and 40GB RAM. While executing the 32 pipelines, we recorded the following three things from each pipeline: **(1)** The time that was needed to execute each of the modules in the pipeline, **(2)** The time that was needed to store the output from a particular module in the HDFS. **(3)** The time that was needed to retrieve the previously stored output of a particular module from the HDFS.

We apply our recommendation mechanism on the 32 pipelines that we executed on the SWfMS and determine which modules could be skipped from which pipelines if we stored the intermediate state results according to the suggestions from our recommendation technique. We finally determine the possible gain in execution time if we could reuse previously stored results.

Fig. 9 shows the execution time gain (in seconds) calculated using Eq. 9 for each of the 32 pipelines. For the first few pipelines, we did not have a gain in execution time because while creating these pipelines there were no stored results for reusing. However, for the remaining ones, we often got a considerable gain. After executing all 32 pipelines by reusing results stored according to our proposed technique's suggestions, we can have a total gain of 17720 seconds (4 hours 55 minutes and 20 seconds).

We also calculated time for executing these 32 pipelines without reusing stored results and found that 23865 seconds were required for executing all pipelines. However, if we reuse the results stored according to the suggestions from our proposed technique, we can execute all the pipelines in only 6145 seconds (i.e., we can save 17720 seconds). In other words, we can save around 74% of total execution time by reusing results. Thus, our proposed technique can help us achieve a considerable gain in execution time.

VI. Threats to Validity

We analyzed 540 (508 pipelines from Galaxy public server [3] and 32 pipelines from the SWfMS of P2IRC) pipelines in our experiment for analyzing our proposed technique's efficiency in making suggestions. While a higher number of pipelines could make our findings more generalized, we see that our technique exhibits efficient performance on pipelines from two different workflow management systems. Thus, we believe that our findings cannot be attributed to a chance. Our proposed technique can be considered an important contribution towards managing pipelines.

VII. Related Work

A great many studies [4] [5] [6] [7] [8] [9] [10] [11] have been conducted on analyzing, reproducing, linking, visualizing, and managing workflows in a scientific workflow management system. We discuss these studies in the following paragraphs.

Woodman et al. [12] proposed an algorithm for determining which subset of the intermediate state results from a pipeline can be stored with the lowest cost. While their study solely focuses on storage cost, our study is fundamentally different because we focus on the re-usability of stored results. We apply association rule mining technique for identifying which intermediate state results should be stored so that they can be reused for making pipelines in future.

Yuan et al. [13] proposed an algorithm for determining which set of intermediate state results can be stored with the minimum cost. Our study is different because we focus on the maximum reuse of the stored results. We propose a technique on the basis of association rule mining for determining which intermediate state results from workflows should be stored for reusing in future.

Koop et al. [14] proposed a technique for automatically completing a pipeline under progress by analyzing pipeline execution history. While their technique focuses on identifying which processing modules could be added after an anchor module in a partially completed pipeline, our study has a completely different focus. We propose a technique for identifying which of the intermediate state results from a pipeline should be stored for reusing.

Many studies [15] [16] [17] have investigated the possibility of discovering and reusing services and workflows in a SWfMS. Our study is fundamentally different than those studies because we do not aim to help users in building pipelines. We identify which of the intermediate state results from a pipeline under progress should be stored for reuse.

From our above discussion it is clear that none of the existing studies investigated recommending intermediate state results from a pipeline for storing so that the stored result can be reused in future. To the best of our knowledge, our study is the first attempt towards such a recommendation. Our in-depth investigation on a good number of workflows indicates that our proposed technique can efficiently suggest which intermediate state result from a workflow should be stored for reusing.

VIII. CONCLUSION

In this paper, we propose and investigate a novel technique (RISP) for suggesting which intermediate state result from a pipeline under progress should be considered for storing so that the result can be reused in future. For making suggestions, our technique mines association rules from the already executed pipelines in the history and analyzes their support and confidence measures. We analyze the efficiency of our technique by applying it on 508 workflows downloaded from Galaxy public server. According to our investigation, our proposed technique can efficiently suggest which intermediate state result from a pipeline under progress should be stored for reusing. The 508 workflows that we investigated had 7165 intermediate state results in total. However, our recommendation technique suggests storing only 49 of these results. We find that these 49 results can be reused for building around 51% of the entire pipelines. Moreover, the intermediate state results stored according to the suggestions from our technique exhibit a high reuse frequency.

We also apply our technique on the scientific workflow management system (SWfMS) in a plant phenotyping based image research center (P2IRC) for investigating how much gain in execution time can be achieved by reusing the stored results. From our investigation on 32 workflows that we executed on the SWfMS we find that by reusing previously stored results we can save around 74% of the execution time that would require if we did not reuse previously stored results. Findings from our research make us realize that our proposed recommendation technique (RISP) has the potential to significantly improve the performance of Big-Data systems by suggesting appropriate reuse of the intermediate state results from the scientific workflows.

ACKNOWLEDGMENT

This work is supported in part by the Canada First Research Excellence Fund (CFREF) under the Global Institute for Food Security (GIFS).

REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, pp. 207–216, 1993.

[2] T. Zimmermann, P. Weisgerber, S. Diehl, and A. Zeller, "Mining version histories to guide software changes," in *Proc. of ICSE*, 2004, pp. 563–572.

[3] E. Afgan, D. Baker, B. Batut, M. vandenBeek, D. Bouvier, M. ech, J. Chilton, D. Clements, N. Coraor, B. A. Grning, A. Guerler, J. Hillman-Jackson, S. Hiltemann, V. Jalili, H. Rasche, N. Soranzo, J. Goecks, J. Taylor, A. Nekrutenko, and D. Blankenberg, "The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update," *Nucleic Acids Research*, vol. 46, pp. W537–W544, 2018.

[4] P. Missier, S. Woodman, H. Hiden, and P. Watson, "Provenance and data differencing for workflow reproducibility analysis," *Concurr. Comput. : Pract. Exper.*, vol. 28, pp. 995–1015, 2016.

[5] J. Zhao, C. Goble, R. Stevens, and S. Bechhofer, "Semantically linking and browsing provenance logs for e-science," in *Semantics of a Networked World. Semantics for Grid Databases*, 2004, pp. 158–176.

[6] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *Future Gener. Comput. Syst.*, vol. 26, pp. 1200–1214, 2010.

[7] E. Deelman and A. Chervenak, "Data management challenges of data-intensive scientific workflows," in *Proc. of CCGRID*, 2008, pp. 687–692.

[8] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in *Proc. of SC*, 2008, pp. 1–12.

[9] S. Hampton, C. Strasser, J. Tewksbury, W. Gram, A. Budden, A. Batcheller, C. Duke, and J. H Porter, "Big data and the future of ecology," *Frontiers in Ecology and the Environment*, vol. 11, pp. 156–162, 2013.

[10] A. Goodman, A. Pepe, A. W. Blocker, C. L. Borgman, K. Cranmer, M. Crosas, R. Di Stefano, Y. Gil, P. Groth, M. Hedstrom, D. W. Hogg, V. Kashyap, A. Mahabal, A. Siemiginowska, and A. Slavkovic, "Ten simple rules for the care and feeding of scientific data," *PLOS Computational Biology*, vol. 10, pp. 1–5, 2014.

[11] E. P. White, E. Baldridge, Z. T. Brym, K. J. Locey, D. J. McGlinn, and S. R. Supp, "Nine simple ways to make it easier to (re)use your data," *PeerJ PrePrints*, vol. 1, p. e7v2, 2013.

[12] S. Woodman, H. Hiden, and P. Watson, "Workflow provenance: An analysis of long term storage costs," in *Proc. of WORKS*, 2015, pp. 1–9.

[13] D. Yuan, Y. Yang, X. Liu, and J. Chen, "On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems," *J. Parallel Distrib. Comput.*, vol. 71, pp. 316–332, 2011.

[14] D. Koop, C. E. Scheidegger, S. P. Callahan, J. Freire, and C. T. Silva, "Viscomplete: Automating suggestions for visualization pipelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1691–1698, 2008.

[15] E. Chinthaka, J. Ekanayake, D. Leake, and B. Plale, "Cbr based workflow composition assistant," in *Proc. of World Congress on Services*, 2009, pp. 352 – 355.

[16] J. Zhang, W. Tan, A. John, I. Foster, and R. Madduri, "Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse," in *Proc. of SCC*, 2011, pp. 48 – 55.

[17] O. Spjuth, E. Bongcam-Rudloff, G. C. Hernández, L. Forer, M. Giovacchini, R. V. Guimera, A. Kallio, E. Korpelainen, M. M. Kańduła, M. Krachunov, D. P. Kreil, O. Kulev, P. P. Łabaj, S. Lampa, L. Pireddu, S. Schönherr, A. Siretskiy, and D. Vassilev, "Experiences with workflows for automating data-intensive bioinformatics," *Biology Direct*, vol. 10, p. 43, 2015.