Climate Analytics Workflow Recommendation as a Service – Provenance-driven Automatic Workflow Mashup

[1]Jia Zhang, [1]Wei Wang, [1]Chris Lee, [1]Xing Wei, [2]Seungwon Lee, [2]Lei Pan, [3]Tsengdar J. Lee
[1]Carnegie Mellon University -Silicon Valley, USA
[2]Jet Propulsion Laboratory, California Institute of Technology, USA
[3]Science Mission Directorate, NASA Headquarters, USA
jia.zhang@sv.cmu.edu, seungwon.lee@jpl.nasa.gov, lei.pan@jpl.nasa.gov, tsengdar.j.lee@nasa.gov

***Abstract*—Existing scientific workflow tools, created by computer scientists, require domain scientists to meticulously design their multi-step experiments before analyzing data. However, this is oftentimes contradictory to a domain scientist's daily routine of conducting research and exploration. This paper presents a novel way to resolve this dispute. After studying how domain scientists conduct data analytics research in their daily work, a provenance model is developed to record their activities. From the provenance, a technology is developed to automatically generate workflows for scientists to review and revise, supported by a Petri nets-based workflow verification instrument. In addition, datasets are proposed to be treated as first-class citizens to proactively contribute to the knowledge sharing and recommendation. A data-centric repository infrastructure is established to catch richer provenance to further facilitate collaboration in the science community. In this way, we aim to revolutionize computer-supported science.**

Keywords—Service recommendation, workflow generation

I. Introduction

The big data challenge has posed a significant demand for community-driven collaborative data analysis. A number of scientific datasets have been published online to ask the entire society to join the efforts to help analyze the data. For example, NASA NEX has published a collection of Earth science data sets on Amazon AWS in 2014 (http://aws.amazon.com/public-data-sets), including climate change projections and satellite images of the Earth's surface. Annual global competitions are arranged to call for the community to analyze the datasets together. Such big data analysis requires significant domain expertise that may not be possessed by individuals.

The advancement in services computing in the recent years has enabled researchers to wrap up their data analytics algorithms as programmable web services and publish them on the Internet. Other scientists can thus leverage these published algorithm services to build more comprehensive data analytics procedures, called workflows [1]. To help people find interested algorithms, existing data projects and platforms typically build centralized repositories to store and publish services and workflows. For example, myExperiment is the largest repository of bioinformatics workflows (2,044 publically accessible workflows by November 12, 2014); bioCatalogue.org is a known repository of bio services (2,500 services by November 12, 2014).

Existing scientific workflow tools, created by computer scientists, require domain scientists to meticulously design their multi-step experiments before analyzing data. However, this is oftentimes contradictory to a domain scientist's daily routine of conducting research and exploration. We hope to resolve this dispute. Imagine this: An Earth scientist starts her day applying NASA Jet Propulsion Laboratory (JPL) published climate data processing algorithms over ARGO deep ocean temperature and AMSRE sea surface temperature datasets. Throughout the day, she tunes the algorithm parameters to study various aspects of the data. Suddenly, she notices some interesting results. She then turns to a computer scientist and asks, "can you reproduce my results?" By tracking and reverse engineering her activities, the computer scientist creates a workflow. The Earth scientist can now rerun the workflow to validate her findings, modify the workflow to discover further variations, or publish the workflow to share the knowledge. In this way, we aim to revolutionize computer-supported Earth science.

This paper reports our on-going efforts to realize the aforementioned vision. Our major contributions are three-fold. First, we have studied how Earth scientists conduct data analytics research in their daily work, developed a provenance model to record their activities, and developed a technology to automatically generate workflows for scientists from the provenance. Second, we have built a data-centric provenance repository, and established a PDSW (People, Data, Service, Workflow) knowledge network to support workflow recommendation. Third, we have established a Petri nets-based verification instrument for provenance-based automatic workflow generation and recommendation. It should be noted that although our work is focusing on the domain of Earth science, it is not limited to the domain. The techniques and tool are generally applicable to other domains.

The remainder of the paper is organized as follows. In Section II, a motivating example is presented. In Section III, we present a new data-service-provider model. In Section IV, we present a provenance-based workflow Petri net. In Section V, we introduce knowledge network construction. In Section VI, we present system development and experiments. In Section VII, we discuss related work. In Section VIII, we draw conclusions and discuss future work.

II. Motivating Example

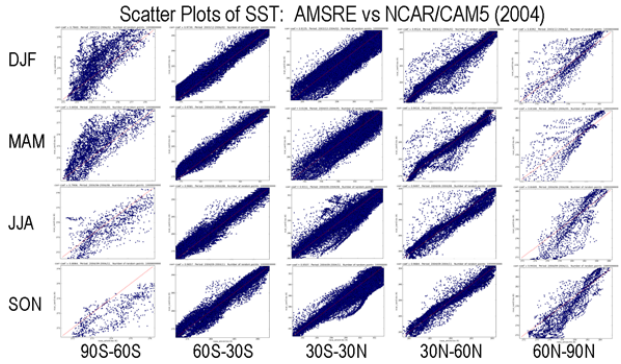

Fig. 1 Discrete manual service exploration.

NASA JPL has developed a collection of RESTful Climate Model Diagnostic Analyzer (CMDA) services. In order to help users find the right service and use service in a right way, we tried to apply our workflow recommendation engine [2] to the CMDA services for climate model comparison [3]. From our exploration, we have discovered three significant findings.

First, Earth scientists typically do not follow the procedure regulated by existing workflow tools, in which each iteration contains workflow design, execution, and revision. Instead, they experiment various possible data analytics over the datasets and seek for data products that may catch their attention. For example, one research challenge, over the datasets of ARGO deep ocean temperature and AMSRE sea surface temperature, in the summer school organized by JPL Center for Climate Sciences in 2014, is: "How is the seasonal cycle of the observations related to model output?" A team tuned the parameters of the web service "Scatter and Histogram Plot" on various latitude ranges: high (+-90-60), medium (+-60-30), and low (+-30-0). They also projected the algorithm over the datasets on different months and different years. From all resulting plots, valuable ones were selected and put together for a systematic comparison. Fig. 1 shows some of their comparison for year 2004 data. This motivating example reveals one significant finding:

The traditional way of "enforcing" scientists to design workflows before analyzing data is to some degree contradictory to their daily way of conducting scientific research and exploration.

The second issue is the granularity level of workflow recommendation. From our preliminary study, we found that scientists may need much finer grained level of service recommendation. For example, a CMDA service typically possesses over a dozen of parameters to tune before it can execute properly. As a result, it is not enough to merely recommend a service reuse. Rather, how to use a service (e.g., parameter tuning) together with previous data products provenance is critical to convince a scientist to reuse other people's work.

The third issue is the lack of workflows and services to train our recommendation tool, because the number of workflows and services available in Earth science remains too low to build a constructive knowledge network to feed into our recommendation engine. The reality is that, not only workflows are in infancy, but also web services are rather new in Earth science community. At the same time, it is difficult for scientists to reuse other's work without a powerful recommendation tool. This has resulted into a chicken-and-egg dilemma. What we do know is that we shall not just wait for the community to publish enough workflows and services before we can help scientists to find and leverage other peer's work.

These three findings directly lead us to explore new methods to help domain scientists design and construct their data analytics experiments, to help them focus more on science with more powerful computer support.

III. Workflow Provenance and Generation

In contrast to typical workflow orchestration starting from existing task components, our project aims to automatically generate and explore (possible) workflows for researchers based on their past activities. In other words, activity provenance will be converted into analytics workflows. Therefore, our first step is to develop a provenance model to record and track scientists' activities and behaviors.

A. Activity Provenance

When we built provenance model, granularity is an important criterion because it will directly impact the effectiveness of workflow generation. First, we constructed a web portal to host JPL CMDA services and asked domain scientists to use the services from our web portal. Second, we monitored how Earth scientists use the services and recorded their activities (e.g., analytics functions and parameters as well as datasets used). Third, we turned such recordings into provenance models. Fourth, we held a series of weekly workshops with domain scientists to verify, refine, as well as to brain storm the provenance models. We looped through Steps 2~4 until we reached an agreed-upon

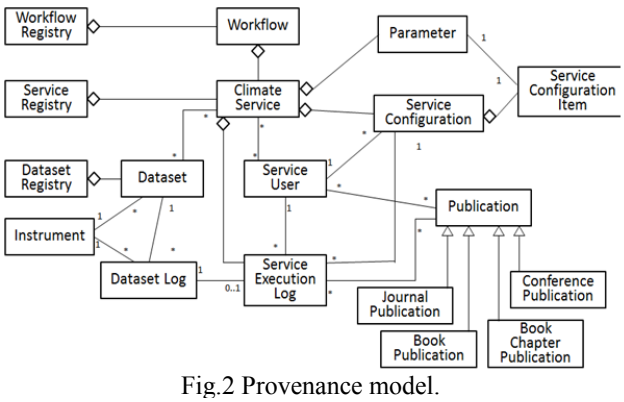

Fig.2 Provenance model.

provenance model as shown in Fig. 2.

The provenance model is centered on *Climate Service*. Each service carries a collection of parameters. When a service is invoked by a *service user*, the event is recorded into *Service Execution Log*, together with the *service configuration* that runs the *service* and the corresponding *dataset log*. A *service configuration* contains a set of <key,value> pairs, representing the parameter values set by the user for the specific service run. In other words, it records *how* aspect of data provenance. Note that intermediate data and data products are stored back to the *Dataset* in addition to original raw dataset. Thus, a *dataset log* records the data processing history associated with a service run, its input dataset, output dataset, as well as data product such as a resulting plot.

Toward the goal of *runnable paper*, as shown in Fig. 2, a specific *service execution log* may lead to a publication authored by its *service user*. Various types of publication are also classified. Such a linkage will not only help authors prepare their paper, but also help paper audience find their original experiments and repeat the experiments with the original settings.

B. Service Run Reproducibility and Adaptation

Based on the activity provenance model established, we can help researchers reproduce a service execution. Fig. 3 illustrates from the provenance recorded, a past service invocation is illustrated, together with its configuration of all comprising parameters and its resulting plot. A user can choose to rerun the service to verify a finding.

If desired, a user can tune the parameters with different values and run the service again with the new configuration. As shown in Fig. 3, the *execution purpose* field helps users annotate a specific service invocation. Particularly, such an execution dependency is captured in the activity provenance for later analysis. If a service execution is motivated by an earlier execution, the latter execution will remember its

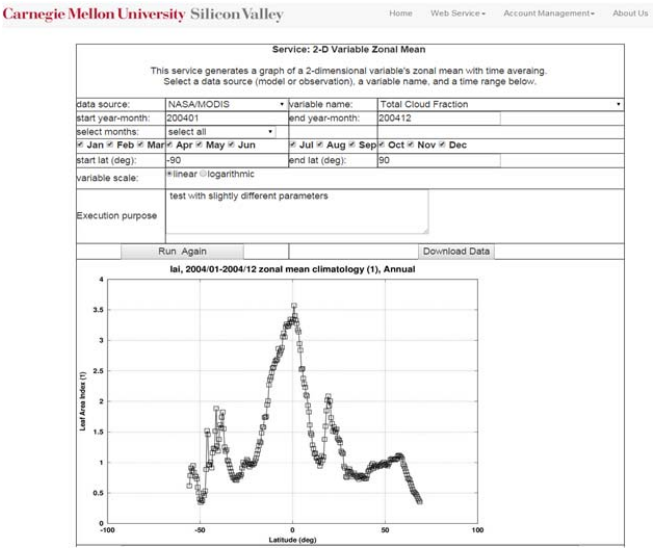

Fig. 3 Service run reproducibility and adaptation.

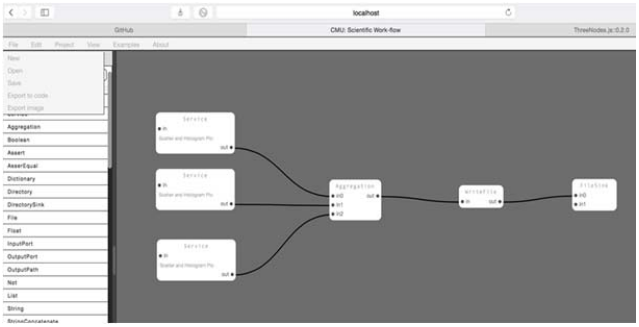

Fig. 4 Automatic workflow generation

former execution. Note that a chain of dependency can be captured. In this way, a collection of service executions with different parameter configurations can be easily grouped together for analysis, e.g., to compare their resulting plots. For example, Fig. 4 shows a scenario where a climate service is executed three times under different configurations, and their output plots can be aggregated for comparison. In summary, our activity provenance model will allow researchers to redo their data analytics to verify the reproduciability of their data product.

C. Workflow Generation

In addition to single service execution history analysis, we have developed an automatic workflow generation process as shown in Fig. 5. The procedure monitors and analyzes user activities and behaviors on climate data analytics services, and extract workflows. For example, if a user runs the service "Scatter and Histogram Plots of Two Variables Service" three times over the same dataset, each with different parameter settings. Our activity provenance will track the user activities and automatically generate a workflow, which aggregates three workflow runs so that the user can repeat the process, and compare the plots generated. Fig. 4 (in open-source workflow tool VisTrails [4]) shows the automatically generated workflow. The user can thus revise the workflow, e.g., run on different datasets with the same collection of parameter settings. Users can also decide to wrap up the generated workflow as publishable services if so desired. In summary, without users building workflows, reverse engineering [5] is conducted to create workflows for users, and incrementally generate reusable workflows.

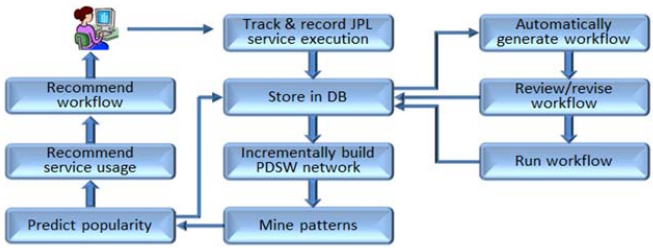

Fig. 5 Provenance-driven workflow generation, adaptation, and management.

As shown in Fig. 4, activity provenance and workflow generation and execution provenance will all be stored in database. Extended from our previous work [2, 5], a knowledge network People, Data, Workflow, Service (PDWS) will be incrementally established and evolved to support pattern recognition. By mining activity provenance, potential workflow may be extracted. For example as shown in Fig. 6, activity provenance shows two independent service calls on two web services. By examining the meta data of the input and output data of the two web service invocations, we may find the meta data of the output of web service 1 is compatible with the metadata of the input of web service 2. Thus, a potential workflow by chaining the two web services can be automatically generated for users to further investigate.

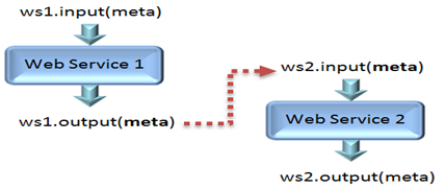

Fig. 6 Discrete manual service exploration.

IV. Petri Net-Based Workflow Generation and Verification

As Aalst [6] indicated, Petri nets are suitable to model and verify workflow systems due to three main reasons: its formal semantics despite the graphical nature, its state-based instead of event-based, and its abundance of analysis techniques. Our previous work leverages colored Petri nets to facilitate the verification and monitoring of web services integration [7]. However, the granularity level of the existing service-oriented workflow modeling methods stays at the service level. A Petri net model contains a set of transitions, a set of places, and a set of arcs. As explained in the previous sections, our work focuses on studying the user activity history to automatically generate workflows. Therefore, our granularity level has to be more finer-grained to service executions.

A. Provenance-based Workflow Net

We propose a unified model to construct Petri nets from service usage history. An activity, a *service run* is the unified building block. An activity records the running context of a service execution, that is, a piece of service execution log. As shown in the provenance model in Fig. 2, a service run records its configuration, input dataset and output dataset. Centered on the activities retrieved from service execution logs, we define a workflow net (W-net) as follows.

Definition 1 (W-Net): A W-Net is a 7-tuple $WN = (P, T, F, C, \sum, i, o)$ where:

- P is a finite set of places (data sets)
- T is a finite set of transitions presenting service APIs
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs representing flow relation
- C is a color function that satisfies $\forall p \in P, \exists \sigma \in \sum: C(p) = \sigma$ $\forall((p_i, t), (t, p_o) \in F | p_i, p_o \in P, p_i \neq p_o, t \in T), \exists c \in C$
- $\sum$ is a finite set of colors associated with a color function
- i is the input place with $\bullet i = \{x \in T | (x, i) \in F\} = \emptyset$
- o is the input place with $o\bullet = \{x \in T | (o, x) \in F\} = \emptyset$

In contrast with the definition of a standard colored petri net, a W-Net may have multiple color functions and color set pairs. The decision is driven by the merge functions that will be discussed in Section C.

Definition 2 (Atomic Activity): An atomic activity is the smallest building block in a W-net. An atomic activity represents one instance of a service run at a given time: a $= (p\_i, s, p\_o)$ where: $p\_i \in P$ is the input place; $s \in T$ is a service run on service s; $p\_o \in P$ is the output place. The configuration of the service is the guard function of the transition.

Definition 3 (Web Service): A web service is a container of metadata and execution history: s $= (metadata, A)$ where: $metadata$ of a service presents its service name, author names, URL, signature, etc. (i.e., what the service is and how to access it); $A = \{a\}$ represents the execution history of the service, i.e., a collection of atomic activities.

According to our definitions, a web service carries not only static metadata, but also its historical usage history. Such historical data will be able to help analyze how services have been used, in order to provide context-aware recommendation later on.

B. Petri Nets-guided Workflow Generation

In order to generate the W-Net, we need to define two mappings: one mapping from the space of service execution logs to an initial W-Net; the other mapping to merge and reduce the resulting W-Net into a stable Petri net.

As shown in Fig. 7(a), along the time axis, a query will retrieve a set of service execution provenance: a set of transitions. Let L be the set, then we have a mapping:

$$L \to A : L \to (P_I \xrightarrow{C_I} T \xrightarrow{C_O} P_O \times \Sigma)$$

Given a query, generation of the initial W-Net is conducted by an application of that mapping to each service execution log that is a member of the query. Leveraging the provenance model shown in Fig. 2, the set of transitions are expanded into a collection of atomic activities. Thus, the initialization phase yields an initial W-Net. Note that in our notation, each arc (flow relation) is associated with a color function. The color function is copied from an in-arc of a

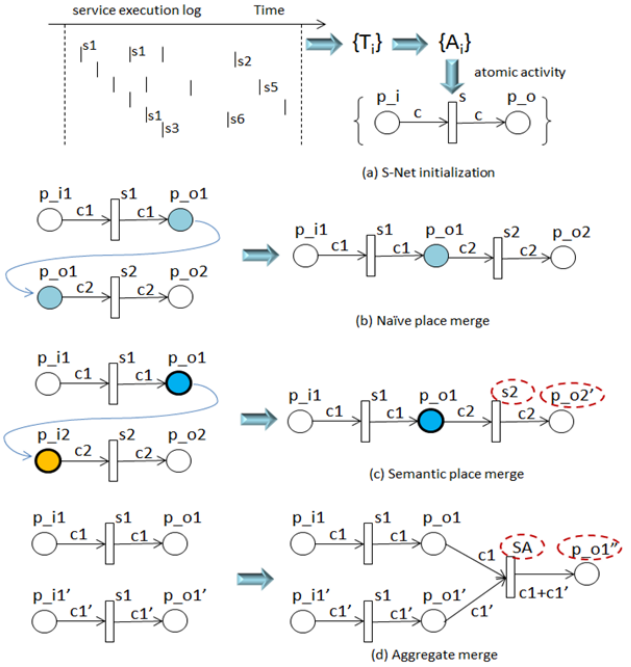

Fig. 7 Petri nets construction rules.

transition to corresponding out-arc for presentation purpose.

However, this structure will likely need to be reduced. For example, a single user may chain the output of one service directly into the input of another. This is what we call a simple merge. There are a plethora of different types of merges, a few of which we define here.

naïve merge: $M_N: P \times P \rightarrow P$
semantic merge: $M_S: P_i \times P_j \rightarrow P_i$
syntactic merge: $M_Y: P_i \times P_j \rightarrow P_i$
aggregate merge: $M_A: P^n \rightarrow T \times P$

Fig. 7(b) illustrates the algorithm of *naïve place merge*, or *naïve merge*. If the output dataset (p_o1) of a service execution (s1) is the same as the input dataset of another service execution (s2), the two service executions can be chained into a workflow by merging their identical places. Naïve place merge implies that the two services can be chained together.

Fig. 7(c) illustrates the algorithm of *semantic place merge*, or *semantic merge*. Although the output dataset (p_o1) of a service execution ($s1$) is different from the input dataset (p_i2) of another service execution ($s2$), the semantics of the two datasets match. For example, their data types are the same or compatible. Then a new workflow of service chain (s1→s2) can be recommended to users. Although users never chained the two services together before, the output dataset of the first service execution (s1) can be fed into the second service ($s2$). Such a workflow may create unprecedented data product (p_o2'). Thus, semantic place merge implies that the two services can be chained together.

Syntactic place merge, or syntactic merge, is a special case of semantic merge with stronger guard function. Take Fig. 7(c) again as an example, such a merge may happen if the data type of the two places (p_o1 and p_i2) match.

Fig. 7(d) illustrates the algorithm of *aggregate merge*. If two atomic activities comprise the same transitions with different color functions, a new *merge* transition (SA) will be created. Its output will be a set of individual datasets, resulting from the previous service execution, for comparison. As a matter of fact, our motivating example illustrated in Fig. 1 motivates the creation of the aggregate merge. Such a workflow will directly lead to the creation of the comparison of resulting plots as shown in Fig. 1. Aggregate merge thus suggests a new workflow, showing how scientists have experimented the same analytics service using different configuration (parameter) settings. Aggregate merge may have variants. For example, in Fig. 7(d), if two input datasets (p_i1 and p_i1') are identical to each other, they can be merged.

C. Merge Function and Analysis

As explained in Fig. 7(a), an initial W-Net is a collection of atomic activity units. A *merge* action, no matter being a naïve merge or a semantic merge, will destroy one place as shown in Fig. 7(b) and 7(c). The state of the W-Net will enter another state. If no further merge can apply, we call the M-Net reaches a stable state.

Each merge action actually determines how slowly a particular W-Net converges to a stable state. For these particular merge functions, convergence can be achieved by first applying M_N then M_A to each activity pair in the W-Net resulting from a query. If workflow recommendation is desired, then M_S is applied as well.

In order to enable and automate various types of merge over places, different color functions need to be defined. Note that aggregate merge is based on transition (service) instead of places (data).

Definition 4 (Equivalence relations): Three equivalence functions are defined over datasets:

- $\sim_N$ (naïve equivalence): two datasets $d_a \sim_N d_b$ iff $d_a = d_b$;
- $\sim_Y$ (syntactic equivalence): two datasets $d_a \sim_Y d_b$ iff $d_a.element.datatype = d_b.element.datatype$;
- $\sim_S$ (semantic equivalence): two datasets $d_a \sim_S d_b$ iff $d_a.element.datatype \,\tilde{s}\, d_b.element.datatype$.

A naïve equivalence means two datasets are identical. A syntactic equivalence means the data types of the two datasets are the same. A semantic equivalence means the data types of the two datasets are semantically compliant with each other.

Based on the three equivalence relations ($\sim_N, \sim_Y, \sim_S$), three color functions can be defined as follows.

Definition 5 (Color function C_x) $x \in \{N, Y, S\}$: In W-Net, color set is the quotient set of all datasets in the network by

an equivalence relation ($\sim_x$), $\Sigma = \{p.dataset, \forall p \in P\}/\sim_x$, $C_x = P \to \Sigma$, where C_x is injective based on the corresponding equivalence relation $\sim_x$.

Our hypothesis is that after a finite number of merges, the initial M-Net will reach a stable state.

Claim 1: For M_N, M_Y, and M_S a single all-pairs pass through the query set is sufficient to produce a W-Net with the minimum number of places. That is, subsequent applications of these merges produce no changes to the net.

Proof: We first examine naïve merge M_N. To see this, let $|W|_P$ be the number of places in the W-Net W and let $M \circ W$ be the all pairs application of merge M to W. Now assume that $W^* = M_N \circ W$ is not minimal with respect to $|\cdot|_P$. This means that at least one of the pairs in W^* was merged:

$$\begin{aligned}
&\exists W' && = M_N \circ W^* : |W'|_P < |W'|_P \\
&\exists A_i, A_j && \in W^*, A_i', A_j' \in W' : \\
&I_i && = I_i' \\
&O_j && = O_j' \\
&O_i' && = I_j' \\
&O_i && = O_i' \\
&I_j && = I_j'
\end{aligned}$$

where I, O are the inputs and outputs of each activity respectively. However, this means that $O_i = I_j$. By construction of W, this pair would have already been merged. Therefore, W^* is minimal.

Claim 2: A W-Net

The proof for the syntactic merge and semantic merge are similar. However, in those cases, equivalence relation must be replaced with syntactic equivalence relation and semantic equivalence relation, respectively. Note that this minimal set property does not hold for aggregate merges. Each aggregate merge generates a new place, which can potentially be used in other merges. For the interest of space, we will not discuss the cost of aggregate merge in this paper.

In practice, it may be desirable to limit the number of merges so that a recommendation process does not take an inordinate amount of time to complete.

V. People, Data, Workflow, Service (PDWS) Knowledge Network

As explained in the last section, our technology of workflow generation relies heavily on provenance mining. Thus, one core consideration is provenance organization and management. At present, data and programs are stored and managed separately like in procedural programming era, e.g., in GCMD [8] catalogue and model libraries respectively. Inspired by object-oriented programming, we propose to treat datasets as first-class citizen to track and exploit data analytics provenance.

A. Data Service Provider

We propose a concept of *"data service"* that encapsulates a dataset and related data processing services and workflows, as illustrated in Fig. 9. The idea is analogous to one most famous comparison in software engineering and programming language: procedural programming vs. object-oriented programming. At present, NASA data centers have accumulated many big datasets. Numerous scientists have been developing various algorithms and programs to process and analyze the datasets. This is like in the procedural language era, programs and data are separated, like verb+noun. What we are proposing here is to treat datasets as first-class citizen as "data-objects" (nouns). As shown in Fig. 9, related data processing services and workflows are arranged around the "data-objects." A data-resource may be associated with many procedures developed by different contributors (e.g., through crowd sourcing). Such procedures may exhibit different quality attributes under different context, which can be used to help users make decisions.

Meanwhile, we do not stop at modeling datasets as data-objects, since our ultimate goal is to provide better data services to help researchers build data analytics procedures faster than before. Moving one step further, we model datasets as "data-service-providers" – they will provide services to the outer world through their encapsulated tools and procedures. As shown in Fig. 9, such datasets also act as registries to accumulate related services and workflows that can process the datasets. From this setting, when a researcher intends to process some dataset, she can easily find out what other researchers have worked on the dataset and build a new experiment by leveraging existing services or workflows. Furthermore, we intend to find connections between datasets at different data centers. It may point us to how we can better organize the data and perhaps even toward data center consolidations and/or better internet connectivity between the centers.

Another aspect of treating datasets as first-class citizen implies that datasets keep evolving, as illustrated in Fig. 9. On one hand, instruments (e.g., satellite) keep on monitoring the world and send back data. On the other hand, intermediate data and data products from data processing services and workflows are stored to the datasets as well. Such a model leads to a new way of provenance

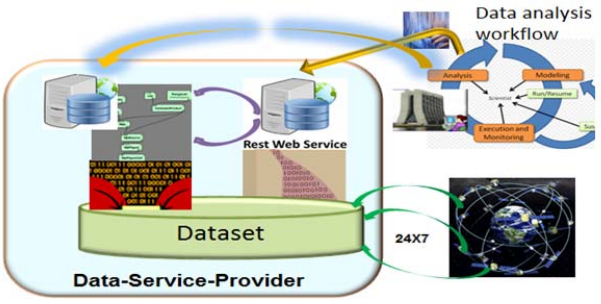

Fig. 9 Data service provider.

accumulation, storage, tracking, and management to support reusable and replicable data analysis workflows.

Note in Fig. 9, human is an integral element in the scenario. Researchers design, develop, conduct data analytics services and workflows. They also comment and drive the activities around the datasets.

Our previous work has developed knowledge networks based on People, Workflow, Service (PWS) [2, 5]. In this project, we have added data as a first-class element into the knowledge network and build a People, Data, Workflow, Service (PDWS) network.

B. Knowledge Network Construction

The debate between centralized and distributed service storage leads to the two categories of service publication methods: to a centralized repository or on distributed. Their central difference is whether web services are published to a centralized repository (e.g., UDDI) or at individual web servers (e.g., WSIL). In contrast to the aforementioned methods, we propose a data-centered distributed service publication method.

As shown in Fig. 10, the granularity of distribution is at dataset level. Each distributed dataset carries a proprietary service registry and a workflow registry. When a user executes a service or a workflow to process the dataset, the execution history is stored, in addition to intermediate and final data products.

Based on distributed data-centric repositories, a knowledge network PDWS can be established. The PDWS network can derive multiple views, such as a workflow repository, a service repository, and a dataset repository. Such views represent the traditional repositories currently existed and maintained.

Our dataset-oriented distribution model facilitates the management of PDWS network over the fine-grained distribution granularity of service-oriented infrastructure. As shown in Fig. 10, each dataset is a self-contained, autonomous entity. It manages the encapsulated service repository and workflow repository. Periodically, each dataset reports to the PDWS network its update. A common

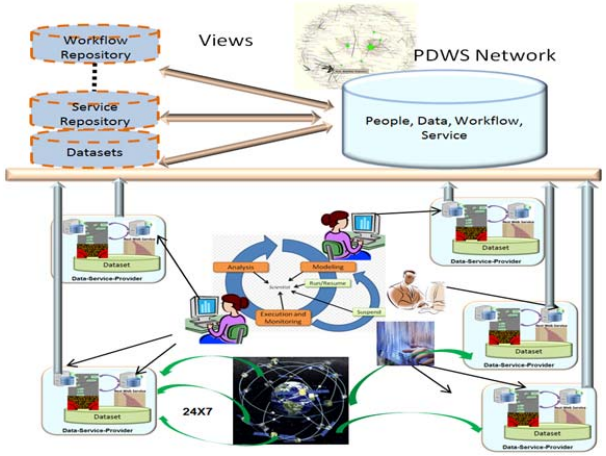

Fig. 10 Distributed PDWS Management.

publish/subscribe design pattern will fulfill the requirements. In this way, the PDWS network does not have to monitor every individual artifact. Instead, datasets manage associated artifacts and update the PDWS network periodically. In addition, compared to the number of services, the number of datasets is apparently more manageable.

C. System Implementation and Experiments

We have designed and developed a prototyping system to demonstrate the concepts and ideas presented in this paper: a climate data analytics service platform. Our platform is an SOA-based solution. We use Flask, a lightweight web application framework written in Python, to interact with JPL CMDA services that supported by JPL HPC center. This is also where we place our hooks to call our service execution log APIs, when a service call is sent to JPL. A number of libraries are required to establish the environment including NetCDF, Octave, ferret, numpy, and matplotlib.

Interfacing with JPL CMDA services, our system provides users an interface to select an interested climate service, configure its parameters and execute the service, monitor and manage resulting data products. In addition, a user can query and retrieve climate service execution history log, repeat some service execution with previous settings, and further tune some parameters to rerun the service for comparison. Fig. 3 is a screen shot of service execution; Fig. 4 is a screen shot of automatic workflow generation from aggregation merge described in Section IV. Ajax technique is used to enable an asynchronous web application.

We leveraged some widely adopted open-source frameworks as bases to build our platform, including Play!, Spring Data, Hibernate JPA, and Bootstrap. The Play! framework provides a backbone to support MVC-based web application development with powerful running environment support. We have adopted the Play! Framework to develop our backend system. The BootStrap is used for implementing responsive web pages, including .html, javascript and css. We used the Bootstrap to support the front-end of our platform. The Spring data and Hibernate JPA framework are combined together to provide platform persistence, by building the relationship between Java beans and database.

D. Related Work

In contrast to existing numerous data projects and data centers that treat datasets as “passive objects,” our project aims to make such datasets first-class citizens to proactively contribute to the knowledge sharing and recommendation. Our exploration may point us to how we can better organize the data and perhaps even toward data center consolidations and/or better internet connectivity between the centers.

Our investigation significantly differs from other workflow development efforts which require that scientists explicitly design workflow. Through reverse engineering scientist’ activities, we have built an intelligent service to mine and reproduce data analytics procedures for scientists.

In contrast to traditional software component reuse

methodologies [9-17] whose granularity level is functional component, our approach examines the historical usage relationships among data analytics activities. In contrast to workflow provenance mining techniques [18-22], e.g., the template-based workflow search facility [4], our approach focuses on service-oriented workflow mining and generation.

Our previous work applied colored Petri nets to support the verification of web services integration [7]. In contrast, our current modeling takes into consideration of service execution context, as well as data provenance to support automatic workflow generation.

Our previous work has developed knowledge networks based on People, Workflow, Service (PWS) [2, 5]. In this project, we have added data as a first-class element into the knowledge network and build a People, Data, Workflow, Service (PDWS) network.

E. Conclusions

In this paper, we have reported our design and development of a novel technique that will allow scientists to stay with their own research habits and styles, and more importantly, will allow them to focus on science. We have designed a provenance system that tracks and records scientists' climate service execution activities. By mining the activity provenance repository, workflows are automatically generated for scientists to repeat and evolve the experiments. A Petri nets-based instrument is also established to verify the generated workflows. A prototyping system has been developed as a proof of concept, collaborating with JPL domain scientists. Complementary with existing IDEs, our on-going research will provide a robust cyberinfrastructure to enable and facilitate transformative Earth science research.

We plan to continue our research along the following directions. First, we will open our system to JPL scientists to use and enrich the PDWS knowledge network by recording their daily scientific activities. Second, we will refine our Petri net-based modeling to further automatic verification over automatically generated workflows. Third, we plan to apply machine learning techniques to mine PDWS network and develop a service and workflow recommendation engine.

Acknowledgment

This work is partially supported by National Aeronautics and Space Administration, under grant NASA NNX13AB38G. We thank Chenran Gong, Ming Qi, Zhiyu Lin, Pinchao Wang, Kang Fang, and Minghan Chen for their support of software development.

References

[1]. C. Goble and D.D. Roure, *The Impact of Workflow Tools on Data-centric Research*, in *T. Hey, S. Tansley, and K. Tolle, eds., The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, Oct.* 2009. p. 137-145.

[2]. J. Zhang, C. Lee, S. Xiao, P. Votava, T.J. Lee, R. Nemani, and I. Foster, "A Community-Driven Workflow Recommendations and Reuse Infrastructure", in Proceedings of *The 8th IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, 2014, Oxford, UK, pp.

[3]. C. Mattmann, C. Lynnes, L. Cinquini, P. Ramirez, A. Hart, D. Williams, D. Waliser, and P. Rinsland, "Next Generation CyberInfrastructure to Support Comparison of Satellite Observations with Climate Models", in Proceedings of *European Space Agency 2014 Conference on Big Data from Space, ESA-ESRIN*, Nov. 12-14, 2014, Frascati (Rome), Italy, pp.

[4]. J. Freire, C.T. Silva, S.P. Callahan, E. Santos, and C.E. Scheidegger, "Managing Rapidly-Evolving Scientific Workflows ", *Lecture Notes in Computer Science*, May, 2006, 4145/2006: pp. 10–18.

[5]. J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Jul. 4-9, 2011, Washington DC, USA, pp. 48-55.

[6]. W.M.P.v.d. Aalst, *Three Good Reasons for Using A Petri-Net-Based Workflow Management System*, in *Information and Process Integration in Enterprises*. 1998, Springer. p. 161-182.

[7]. J. Zhang, C.K. Chang, J.-Y. Chung, and S.W. Kim, "S-Net: A Petri-net Based Specification Model for Web Services", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Jul. 6-9, 2004, San Diego, CA, USA, pp. 420-427.

[8]. NASA, "Global Change Master Directory (GCMD) - NASA", 2015, accessed on: Jan. 8, Available from: http://gcmd.nasa.gov/.

[9]. T. Xie, S. Thummalapenta, D. Lo, and C. Liu, "Data Mining for Software Engineering", *IEEE Computer*, 2009, 42(8): pp. 55-62.

[10]. R. Martin, W. Robert, and Z. Thomas, "Recommendation Systems for Software Engineering", *IEEE Software*, 2010, 27: pp. 80-86.

[11]. H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei, *MAPO: Mining and recommending API usage patterns*, in *European Conference on Object-Oriented Programming*. 2009, Springer. p. 318-343.

[12]. D. Mandelin, L. Xu, R. Bodík, and D. Kimelman, *Jungloid mining: helping to navigate the API jungle*, in *ACM SIGPLAN conference on Programming language design and implementation*. 2005, ACM. p. 48-61.

[13]. B. Ashok, J. Joy, H. Liang, S. Rajamani, G. Srinivasa, and V. Vangala, *Debugadvisor: A recommender system for debugging*, in *7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. 2009, ACM. p. 373-382.

[14]. A. Begel, K.Y. Phang, and T. Zimmermann, *Codebook: discovering and exploiting relationships in software repositories*, in *32nd ACM/IEEE International Conference on Software Engineering*. 2010, ACM: Cape Town, South Africa. p. 125-134.

[15]. T. Wolf, A. Damian, D. Panjer, L. Nguyen, and H. Thanh, "Mining task-based social networks to explore collaboration in software teams", *IEEE Software*, 2009, 26(1): pp. 58-66.

[16]. Y. Gil, P. Groth, and V. Ratnakar, *Social Task Networks: Personal and Collaborative Task Formulation and Management in Social Networking Sites*, in *AAAI Fall Symposium on Proactive Assistant Agents*. 2010.

[17]. O. Greenshpan, T. Milo, and N. Polyzotis, "Autocompletion for mashups", *Proc. VLDB Endow.*, 2009, 2(1): pp. 538-549.

[18]. 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*, 2008, 14: pp. 1691-1698.

[19]. C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva, "Querying and Creating Visualizations by Analogy", *IEEE Transactions on Visualization and Computer Graphics*, 2007, 13: pp. 1560-1567.

[20]. E. Chinthaka, J. Ekanayake, D. Leake, and B. Plale, *CBR Based Workflow Composition Assistant*, in *2009 Congress on Services -- I*. 2009, IEEE Computer Society. p. 352-355.

[21]. D. Leake and J. Kendall-Morwick, *Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance*, in *European Conferences on Case-Based Reasoning*. 2008. p. 269-283.

[22]. X. Xiang and G. Madey, *Improving the Reuse of ScientificWorkflows and Their By-products*, in *IEEE International Conference on Web Services*. 2007. p. 792-799.