SUPERMAN: A Novel System for Storing and Retrieving Scientific-Simulation Provenance for Efficient Job Executions on Computing Clusters

Young-Kyoon Suh and Jin Ma
Convergence Research Platform Development Laboratory,
Center of Computational Science and Engineering,
National Institute of Supercomputing and Networking,
Korea Institute of Science and Technology Information (KISTI)
Daejeon, Korea 34141
{yksuh, majin}@kisti.re.kr

***Abstract*— Compute-intensive simulations typically charge substantial workloads on an online simulation platform backed by limited computing clusters and storage resources. Some (or most) of the simulations initiated by users may accompany input parameters/files that have been already provided by other (or same) users in the past. Unfortunately, these duplicate simulations may aggravate the performance of the platform by drastic consumption of the limited resources shared by a number of users on the platform. To minimize or avoid conducting repeated simulations, we present a novel system, called *SUPERMAN* (SimUlation ProvEnance Recycling MANager) that can record simulation provenances and recycle the results of past simulations. This system presents a great opportunity to not only reutilize existing results but also perform various analytics helpful for those who are not familiar with the platform. The system also offers interoperability across other systems by collecting the provenances in a standardized format. In our simulated experiments we found that over half of past computing jobs could be answered without actual executions by our system.**

Keywords—EDISON; Simulation; HPC; Computing Clusters; Provenance; PROV;

I. INTRODUCTION

EDISON [1, 2] is a well-known online scientific-computing simulation platform developed by KISTI. This platform has been designed and developed to educate students and assist researchers to conduct their research online with a variety of large-scale computing software tools from diverse computational science and engineering fields. The platform has been quite mature enough to support about cumulative 40K users over the past five years in Korea.

Challenge: An EDISON user selects a software tool on the platform and initiates a great number of computational jobs on the chosen running on our infrastructure. One challenge we are faced with is to have to serve a flooding of simulation requests with *no* change of input parameters. For instance, every student in a class utilizing EDISON simultaneously runs simulations with identical input data along with a teaching assistant's guidance, resulting in writing the same output files and thus drastically consuming our limited computing/storage resources.

To address this concern, we propose a novel system for storing and processing scientific simulation provenance with which a duplicate simulation request that has been seen before can be noticed and then quickly answered with existing results matching that request. This system is termed *SUPERMAN* (SimUlation ProvEnance Recycling MANager), since it recycles previously-executed simulations through provenances to handle future requests from a number of users, in order to save limited resources. SUPERMAN records simulation provenance in accordance with a de-facto standard, PROV specification [3], of which the use enables interoperability for further processing or verification of that provenance across other systems understanding the PROV specification.

This paper is organized in the following. In the following section we discuss design principles of our system. We in turn provide what information is collected for and how to model provenances using an example of W3C PROV. In Section IV, we present an overall architecture of our system. Section V conducts a literature survey. Finally, we conclude this paper by summarizing our contributions.

II. DESIGN PRINCIPLES

SUPERMAN holds the "SINGLE" properties satisfying the six goals in its design:

- Scalability: support for a flooding of simulations from numerous users
- Interoperability: understandable by any system following a standard form
- Nonintrusive: not interrupting the operation of the main platform while collecting provenance without notice
- Generality: support for an arbitrary tool in a computational science and engineering field
- Lightweightness: using minimal information for provenance modeling
- Effectiveness: support for not only reproducibility but several analytics tasks, mainly for enabling system efficiently in simulation.

Specifically, SUPERMAN should be scalable to manage provenances on simulation requests flooding from the EDISON platform. Also, the produced provenances should be interoperable [4] via a standardized form by another system wishing to request our provenances and perhaps perform further processing on the received provenances. Our system shouldn't interrupt the normal simulation execution performed by the EDISON platform. In addition, the system should be able to manage provenances from an arbitrary simulation tool from a specialized discipline. Moreover, the provenance modeling should avoid containing superfluous information while preserving the minimality in describing an executed simulation. Finally, the utilization of produced provenances should be effective so that the platform gets more efficient.

We now discuss what and how to model provenance for our simulations on EDISON.

III. SIMULATION PROVENANCE MODELING

A. *Collected Simulation Information*

Table I summarizes what information regarding our HPC simulation is collected from the EDISON platform. Most of the information is used for locating and preserving simulation output. Some other information is used for the accessory purpose of browsing past simulations. There are four categories: (1) user, (2) science app (or simulation tool), (3) simulation, and (4) (computing) job. For the user category, we keep what affiliation a user belongs to as well as identification information like ID and name. In the science app category we want to know which computational science and engineering field a science app on which a user's simulation gets executed comes from, let alone the name and version of the app.

TABLE I. COLLECTED INFORMATION FOR PROVENANCE ON EDISON

Item		**Description**
Category	**-**	
User	user ID	ID, name, and affiliation of a simulation user registered on platform
	user name	
	affiliation	
Science App	domain ID	a specific domain in computational science and engineering (e.g. CFD, Nano, and other areas.)
	science app name	Name and version of a science app of interest on provenance
	science app version	
Simulation	simulation ID	simulation ID
	title	simulation title
	creation date	when this simulation was created
	job count	how many jobs were produced from this simulation
Job	job seq. no.	sequence number of this job
	job status	status of this job: success or fail
	job submit date	job submission date (in queuing)
	job start date	job start date (out of queue)
	job end date	job end date
	job exec path	where input files are located, and output files are generated
	job data	specific parametric values provided to a science app

In the simulation category, we collect the ID, title, and creation date of that simulation, as well as how many jobs are created. When it comes to the job category, we collect a different kind of information about a job created by the simulation: sequence number, status (success or fail), submit date (in job queue), start date (out of job queue), end date, execution path, and input data with specific parametric values provided for that job. This kind of simulation information forms a specific provenance instance for further processing. In particular, each instance is standardized in PROV [5] as stated in the introduction. The following section discusses how to associate the collected information with what specific elements in PROV and then gives an example of how to physically represent the instance in JSON format [6].

B. *Detailed Provenance Modeling Using PROV*

In the SUPERMAN system any simulation provenance generated by the EDISON platform is captured along with PROV-DM and physically serialized to JSON format. Why JSON? There are two main reasons for that. First, there is a compatibility issue; many data objects are handled in JSON in the EDISON platform. Also, JSON offers a compact way of transmitting data over network, considering dissemination.

Table II describes core structures of PROV-DM. An entity indicates all the concepts that actually exist; an agent means the delegator of the entity; the activity concerns the functions and attributes of the agent and the entity.

TABLE II. CORE STRUCTURES OF PROV-DM [4]

Element	**Description**
Agent	An agent can be understood as an agent concept that can act on behalf of an entity. Agent is responsible for the entity.
Activity	An activity is an action, a function, an attribute, and a relation that an agent and an entity can perform.
Entity	Physical, digital, and conceptual types are called entities.

PROV-JSON [7] physically generates a file with a lightweight PROV model in the JSON format. PROV-JSON can be converted to other types (such as PROV-N, PROV-O, and PROV-XML). Therefore, a generated PROV-JSON file supports interoperability that can be shared by and disseminated to other platforms understanding PROV.

Fig. 1 depicts how a simulation instance performed on the EDISON platform can be represented by the elements in PROV-DM and their relations.

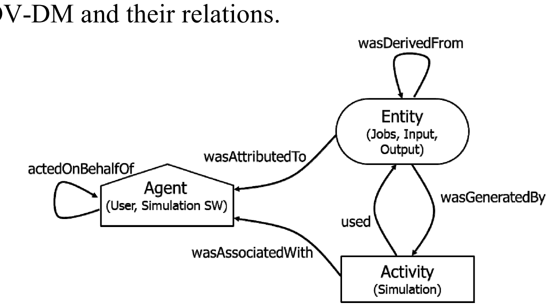

Fig. 1. Reconstruction of the PROV Model on the EDISON Simulation.

The following is the mapping of a specific item to a corresponding element:

- *Agent*: user and science app
- *Entity*: computing job, input data (file or parameter sets), and output (file or path) data, and
- *Activity*: simulation.

We also use the following relation types to make an association with the elements:

- *Relation*: *actedOnBehalfOf*, *wasAssociatedWith*, *wasAttributeTo*, *wasDerivedFrom*, *wasGeneratedBy*, and *used*.

The following is an description of each relation type:

- *actedOnBehalfOf*, being used to delegate authority and responsibility between Agents,
- *wasAssociatedWith*, meaning that Activity was associated with Agent,
- *wasAttributeTo*, meaning that Entity was attributed to Agent,
- *wasDerivedFrom*, meaning that (one) Entity was derived from (another) Entity,
- *wasGeneratedBy*, meaning that Entity was generated by Activity, and
- *Used*, meaning that Activity used Entity.

Let's consider a specific simulation provenance in the following. A simulation instance, identified by `394197c0-1134-41df-9a6b-2b49460aaec1`, was run by a science app (`2D_Comp_P`), chosen by a user (id: `zacwhee`). The simulation instance concerned two computing job instances, each identified by `677eb1d5-14c8-437a-a270-acb7aa8885c5` and `585a7cd7-0919-4824-af8e-fdf024b2d174`, with an input file named `NASAsc2-0714(2).msh`. The simulation instance produced the output directory: `/EDISON/LDAP/zacwhee/394197c0-1134-41df-9a6b-2b49460aaec1/result`. Then, this provenance is represented using the elements and the relations in the PROV model, as shown in Table III.

TABLE III. AN EXAMPLE OF PROV-JSON REPRESENTATION ON A SIMULATION PROVENANCE

PROV-JSON with a namespace prefix (ex. `prov:`) omitted


```
// Element definitions
"agent": {"simulation SW": {"(prov:)label": "2D_Comp_P"}},
"entity": { "Input": {"fileName":"NASAsc20714(2).msh"},
    "job1":{"677eb1d5-14c8-437a-a270acb7aa8885c5"},
    "job2":{"585a7cd7-0919-4824-af8efdf024b2d174"},
    "Output": {"/EDISON/LDAP/zacwhee/394197c0-1134-41df-
9a6b-2b49460aaec1/result"}},
"activity": {"simulation ID": { "label": "394197c0-1134-
41df-9a6b2b49460aaec1"},
// Relation definitions
"actedOnBehalfOf": {
  "zacwhee": {"delegate":"2D_Comp_P", "responsible":
"zacwhee"}},
"wasAssociatedWith": {"2D_Comp":
{"activity":"2D_Comp_P","agent":"zacwhee"}},
"wasAttributedTo": {"677eb1d5-14c8-437a-a270-acb7aa8885c5
": {"agent":2D_Comp_P","entity": "Job1"},
 {" 585a7cd7-0919-4824-af8e-fdf024b2d174": {"agent":
  "2D_Comp_P","entity": "Job2"},
 " NASAsc20714(2).msh": {" agent": " zacwhee ","entity":
"Input"}},
"wasGeneratedBy": {
  "activity": "394197c0-1134-41df-9a6b-2b49460aaec1",
   "role": {"WasGeneratedBy"},"entity": "Output",
      "activity": "394197c0-1134-41df-9a6b-2b49460aaec1",
      "role": {"WasGeneratedBy"}, "entity": "job1",
      "activity": "394197c0-1134-41df-9a6b-2b49460aaec1",
      "role": {"WasGeneratedBy"}, "entity": "job2",
   "activity": "394197c0-1134-41df-9a6b-2b49460aaec1"},
"used": {{"activity": "394197c0-1134-
41df9a6b2b49460aaec1", "entity":"input"}}
```


IV. SYSTEM ARCHITECTURE

In this section we present the system architecture of SUPERMAN, along with an altered simulation-running scenario applying provenance utilization.

A. Overall Architecture

Conventionally, HPC simulations on the EDISON platform are conducted in the following. A user logs into Application Portal Framework, based on Liferay Portal [8], which allows the user to access various science apps from several specialized disciplines. The user selects one of the apps and creates and submits to Job Execution Framework a computing job, which is then put in queue and sequentially taken out of the queue for execution on Infrastructure. By getting the SUPERMAN system to interact with EDISON, however, this simulation-running scenario can be altered as follows. If a user requests an HPC simulation on the platform, that request is forwarded to SUPERMAN to see if any duplicate request with the same input data has been formerly recorded as provenance. If not, the requested simulation is normally conducted, and completed simulation results are provided to the user. If that exists, the existing simulation results are immediately returned to the user without executing the requested simulation that would otherwise be long-running and repeated.

Fig. 2 shows what components constitute SUPERMAN and how the system collaborates with the EDISON platform. SUPERMAN provides eight main functionalities to enable simulation provenance service. (i) Simulation Provenance Collection indicates that all the HPC simulations executed on EDISON are captured as provenance records. A detailed piece of information collected for provenance is already provided in Table I. When a provenance record is created, SUPERMAN determines if it already exists in a repository, called Simulation Data Repository, to be discussed shortly. (ii) Simulation Provenance Validation checks the validity of a stored provenance record compared with existing records. For instance, with the past records we can determine whether a wrong value is entered into an input parameter of a chosen app. (iii) Simulation Provenance Modeling supports standardizing the collected provenance. As mentioned earlier, each provenance record is modelled along with PROV and physically stored as a PROV-JSON file. (iv) Simulation Data Querying/Search performs a search on stored provenance via a search engine, which will be discussed in the subsequent section. (v) Provenance-based Parameter Assistance and Visualization Service can recommend to a user which parametric values would be typed based on popularity when she runs a simulation. (vi) Data-driven Simulation Reproduction enables users to rerun existing simulations from chosen provenance records or estimating simulations results by the same or similar methodology taken by an earlier article [9].

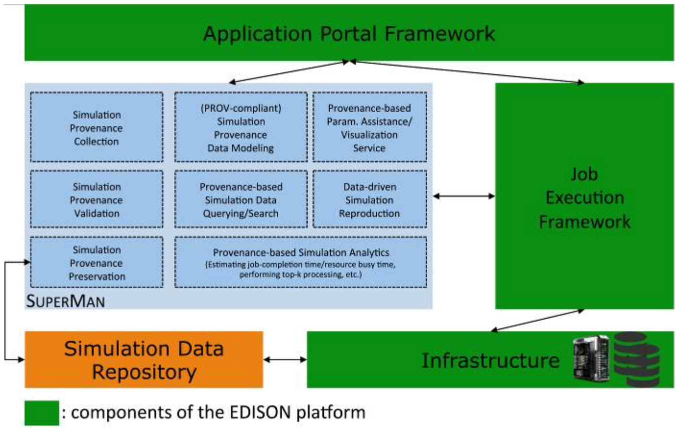

Fig. 2. SUPERMAN Architecture

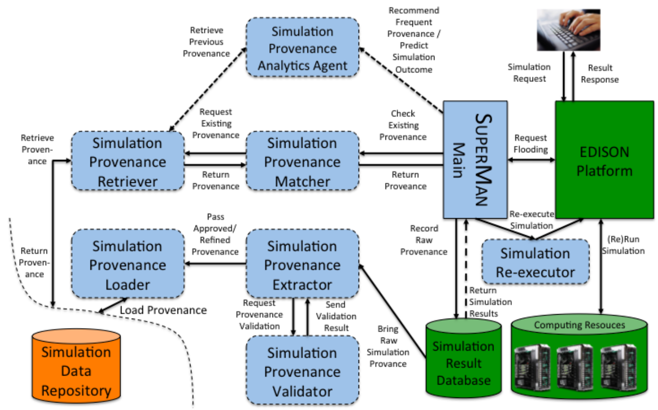

Fig. 3. A Functional Flow of SUPERMAN, interacting the EDISON platform

(vii) Provenance-based Simulation Analytics provides a service for users to get more informed about their ready-to-run simulations, specifically with estimated job completion time, resource busy time, top-k processing with regard to popularity, job fail rate, and error detection based on provenance records. Lastly, (viii) Simulation Provenance Preservation provides a service of permanently storing provenances with actual simulation results in a separate repository for future access.

Fig. 3 shows an overall architecture of the SUPERMAN system. It consists of seven components to perform their respective tasks to collect and manage provenances from the EDISON platform.

1) Simulation Provenance Extractor

This component extracts the provenance of a completed simulation from the result database. For instance, the provenance covers Table 1's information and some additional helpful information such as input file size and output data paths obtained via Job Execution Framework of the EDISON platform. We also collect the information of whether a simulation succeeded or failed (due to user cancellation or something else) for future analytics. But note that the extraction is not performed unless a user agrees with sharing her simulation results with other users. For those who do not wish to make their results in public, we preserve the right to reject the sharing.

2) Simulation Provenance Validator

A collected provenance record is validated by another component, called Simulation Provenance Validator. Even if that record is correctly constructed, it may have an out-of-range or wrong value for an input parameter of a chosen science app, compared with what values were provided for that parameter in the past. In this case we do not execute the requested simulation and throw a run-time error to the platform, so that the user can be informed with that error and perhaps retry her simulation. A validated provenance record is sent to the next component to be described below.

3) Simulation Provenance Loader

Simulation Provenance Loader loads provenance data into Simulation Data Repository to be described shortly. It first takes an approved record and checks if any previously-stored record exists via Provenance Hash Table, termed PHT, designed to perform the comparison quickly. PHT effectively eliminates duplicate provenance records, thus preventing any identical output data from being stored more than twice into Simulation Data Repository. PHT is a crucial component for efficiently managing provenance data.

Table IV details the following three steps taken to construct PHT. The key idea is to utilize input data (parameters and files if any) for hash key generation as described at Steps 1-3 in Table IV. For a new provenance record, the Loader stores its key and the output path into PHT as specified at Step 4.

TABLE IV. BUILDING PROCEDURE FOR PROVENANCE HASH TABLE

Step.	Description
1	If an input file is used in a given simulation, generate the hash value based on the file contents.
2	If the simulation does not use an input file, generate a hash value based on the input parameter and values entered into a computing job.
3	A combination of the hash values generated through Steps 1–2 is used as a parameter of a hash function to generate the final hash value to be used for a key for PHT.
4	Once the given simulation is completed, the value of PHT is the output path and gets associated with the generated key in PHT.

If none of the existing records is found, then the Loader component converts the approved record to the corresponding PROV-JSON file and stores the file with the corresponding output data into the Repository for future access.

4) Simulation Provenance Matcher

This component compares a new provenance record with existing records, returned from Simulation Provenance Retriever, to be discussed shortly, performing provenance retrieval along with matching conditions. The Matcher generates a hash key of the new simulation as described in Table IV. It then matches whether it is identical to that of any of the retrieved provenances. If so, then SUPERMAN returns existing simulation results back to the user.

5) Simulation Provenance Retriever

This component is responsible for retrieving the user-requested simulation provenance (parameter value) from the EDISON platform through Simulation Data Repository. Fig. 4 shows the retrieval process of the Simulation Provenance Retriever connected with the existing EDISON platform.

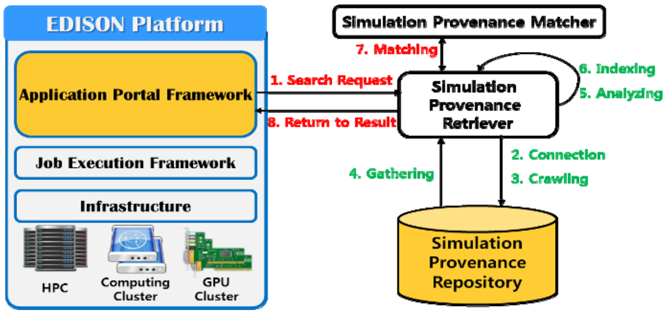

Fig. 4. Simulation Provenance Retriever's Process with EDISON

6) Simulation Provenance Analytics Agent

Simulation Provenance Analytics Agent can provide a variety of analytics service based on accumulated provenances. For instance, the Agent can recommend the most frequently accessed provenance records via top-k processing or estimate the completion time of a requested simulation based on past provenance records. We are also considering detecting a potential error on specified parameters. These analytics will be provided with users for assistance for their simulations.

7) Simulation Re-executor

The last component is Simulation Re-executor, which allows the platform to recycle existing simulation results or re-run the simulations with altered parametric values. The Re-executor passes a selected provenance record to the platform, which in turns automatically loads into simulation parameters previously entered values extracted from the record. The user can run the simulation with the loaded values on the platform as they are, or change the values. Once the simulation is initiated, then SUPERMAN figures out whether a provenance record of the requested simulation exists. If so, then the results matching the existing provenance are returned to the user. Otherwise, the requested simulation is normally executed on computing resources via the platform. Certainly, its provenance is stored as a new record.

Simulation re-execution (reproduction) can be triggered from provenance search (retrieval). Fig. 5 illustrates a flow diagram of how a user's provenance search can reach simulation reproduction. A user selects a science app on EDISON. The user queries the app's simulation provenances matching a given condition. Subsequently, SUPERMAN checks if there exists any matching provenance records from the retrieved. If no existing record is found then the user can keep making further inquiries on different conditions.

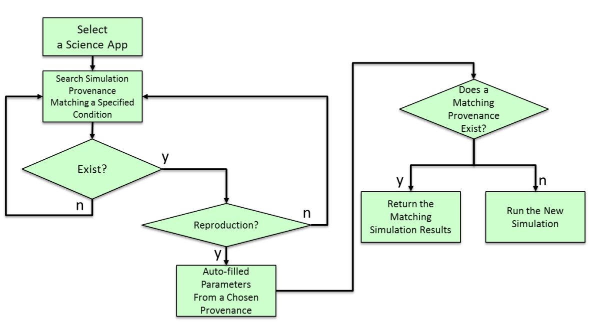

Fig. 5. A flow diagram of simulation-running utilizing past provenances

But if any matching provenance record(s) exists, the user can reproduce the same simulation from that provenance. That is, SUPERMAN passes the matching provenance record, and then the EDISON portal automatically fills up parametric values extracted from the passed for that user. It is totally acceptable for the user to run simulations with different input parameter values than the auto-filled values. In any case, once the user re-executes the simulation, then SUPERMAN snatches the request and inspects if a provenance record associated with that request exists. If so then, the associated simulation results are immediately reported back to that user. Otherwise, the new simulation is initiated and gets normally executed on the EDISON infrastructure. Again, upon the completion of the simulation, its provenance is captured for future use.

V. RELATED WORK

Several HPC service platforms have been proposed to provide online simulation services in many diverse fields in computational science and engineering. WebMo [10] focuses on running simulations in chemistry. NEESHub [11] is a simulation service platform on which simulations regarding earth science are performed. NanoHUB [12] is a well-known platform to allow users to run nano-electronics simulations. None of these platforms is general-purpose. Also, they do not have any capability of storing completed simulation results.

There is also a general-purpose platform, called HUBZero [13]. This platform supports running online simulation tools from a variety of computational science and engineering disciplines. But HUBZero do not consider utilizing provenances unlike our work.

We further examine how our system is distinguished from the two popular existing platforms such as HUBZero and nanoHUB, as shown in Table V. The support is expressed as O, X, and ∆. Especially, we choose these two because they function quite similarly compared to the EDISON platform on which SUPERMAN is applied. The EDISON that will be equipped with SUPERMAN is expected to support (i) rerunning simulations on the same parameters with no actual execution on computing resources, (ii) sharing input/output files, (iii) retrieving existing simulation data matching a given condition on input parameters of a selected on past provenances, and (v) grouping input parameters on a given output for a researcher to quickly grasp what conditions to be provided if possible.

TABLE V. COMPARISON WITH HUBZERO AND NANOHUB

	HUBZero	nanoHUB	SUPERMAN
Simulation Re-run	O	O	O
In/Output Data Share	∆	∆	O
Simulation Provenance Retrieval	X	X	O
Parameter Assistance	X	X	O
Prediction of Sim.-Completion Est.- time	X	X	O
Sim. –Exec. Pattern Analysis	X	X	O

A description of the function in table as item is as follows:

- Simulation Re-run: indicates whether to support re-execution of previously executed simulations. HUBZero and nanoHUB can share simulation sessions through user, group, and email.
- In/Output Data Share: indicates the sharing support for the input data provided for executing the simulation or the output data produced as the result. nanoHUB can share the user name, user ID and e-mail. HUBZero is an incomplete data sharing system, as it must be uploaded directly to the project page of the simulation.
- Simulation Provenance Retrieval: indicates a support for retrieving simulation provenances executed by another user and displaying retrieval result.
- Parameter Assistance: indicates a support for recommendation of appropriate parameters before running the simulation.
- Prediction of Simulation Completion Estimation Time: indicates a support for estimating completion time based on provenances.
- Simulation Execution Pattern Analysis: indicates a support for analyzing what group of input parameters is frequently given on a chosen simulation tool. In this way, a user can be assisted on a platform.

SUPERMAN will address all of these issues through the designed components as shown in Fig. 2.

Finally, there are some other simulation service systems, providing capability to record simulation results. DataSpaces [14] exposes an abstraction for sharing simulation data. SciDrive [15] provides an interface to publish data like DropBox. Scibox [16] also uses an interface similar to DropBox to support sharing and storing simulation data in the cloud. An earlier research [8] supports reusing existing simulation results with its own format. As far as we know, none of these systems utilize standardized provenances to treat future simulation requests unlike our proposed system. There is also a well-known website [17] where completely independent users can record and open their workflows. In our case, it is acceptable to share simulation provenances and their output data but almost not possible to make source code of the science apps in public, mainly because the respective domain developers are very afraid of infringement of intellectual property about the apps.

VI. CONCLUSION AND FUTURE WORK

In this paper we presented a new architecture, called SUPERMAN, to collect simulation provenances to make an online simulation platform more efficient and data-centric. The system supports recording and searching simulation provenances. For the same simulation requests that have been seen before, the system can quickly respond with existing results, without actual execution from scratch. Indeed, we simulated the performance of our system with existing computing jobs (or simulations) conducted on eight different science apps in Nano Physics from August 2016 to July 2017. As a result, about 57% (12,137 out of 21,301) of the submitted simulations were identical, thus avoiding repeated resource consumption. This result is very encouraging. We plan to extend the simulated experiments to other science apps from another domain and then evaluate the actual performance with the simulation data produced while EDISON is in service.

ACKNOWLEDGEMENTS

This research was supported by the EDISON Program through the National Research Foundation of Korea(NRF) (No. NRF-2011-0020576).

REFERENCES

[1] Young-Kyoon Suh, Hoon Ryu, Hanki Kim, and Kum Won Cho, "EDISON: A Web-based HPC Simulation Execution Framework for Large-scale Scientific Computing Software," in *Proceeding of IEEE/ACM 16th International Symposium on Cluster, Cloud and Grid Computing* (CCGrid 2016), May 2016, pp. 608–612.

[2] Jin Ma, Jongsuk Ruth Lee, Kumwon Cho, and Minjae Park, "Design and Implementation of Information Management Tools for the EDISON Open Platform," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 2, pp. 1089–1104, 2017.

[3] W3C, "PROV-Overview," https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/, viewed on April 28, 2017.

[4] Luc Moreau, Paul Groth, Jame Cheney, Timothy Lebo, and Simon Miles, "The Rationale of PROV," *Journal Web Semantics*, vol. 35, no. 4, pp. 235–257, 2015.

[5] W3C, "PROV-DM," https://www.w3.org/TR/prov-dm/, viewed on April 28, 2017.

[6] ECMA, "Standard ECMA-404: The JSON Data Interchange Format", https://www.ecma-international.org/publications/standards/Ecma-404.htm , viewed on May 8, 2017.

[7] W3C, "The PROV-JSON Serialization," https://www.w3.org/Submission/2013/SUBM-prov-json-20130424/, viewed on May 8, 2017.

[8] Liferay, "Liferay Portal 6.2," https://web.liferay.com/products/liferay-portal/liferay-portal-6.2, viewed on June 1, 2017.

[9] Ki Yong Lee, YoonJae Shin, YoonJae Shin, YeonJeong Choe, SeonJeong Kim, Young-Kyoon Suh, Jeong Hwan Sa, and Kum Won Cho, "Design and Implementation of a Data-driven Simulation Service System," in *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory* (EDB 2016), October 2016, pp. 77–80.

[10] J. Schmidt and W. Polik, "WebMO Portal (Chemistry)," http://www.webmo.net, viewed on June 3, 2017.

[11] T. J. Hacker *et al.*, "The NEEShub Cyberinfrastructure for Earthquake Engineering," *Computing in Science & Engineering*, vol. 13, no. 4, pp. 67–78, 2011.

[12] G. Klimeck *et al.*, "nanoHUB.org: Advancing Education and Research in Nanotechnology," *Computing in Science & Engineering*, vol. 10, no. 5, pp. 17– 23, 2008.

[13] M. McLennan and R. Kennell, "HUBzero: a Platform for Dissemination and Collaboration in Computational Science and Engineering," *Computing in Science & Engineering*, vol. 12, no. 2, pp. 48–53, 2010.

[14] Docan, C., Parashar, M., and Klasky, S., "DataSpaces: an interaction and coordination framework for coupled simulation workflows," *Cluster Computing*, Vol. 15, No. 2, pp.163-181.

[15] Mishin, D., Medvedev, D., Szalay, A. S., Plante, R., and Graham, M., "Data Sharing and Publication Using the SciDrive Service," in *Proceedings of Astronomical Data Analysis Software and Systems XXIII*, 465, 2014.

[16] Huang, J., Zhang, X., Eisenhauer, G., Schwan, K., Wolf, M., Ethier, S., Klasky, S. "Scibox: Online Sharing of Scientific Data via the Cloud," in *Proceedings of the 28th IEEE International Parallel & Distributed Processing Symposium*, pp. 145-154, 2014.

[17] Univ. of Manchester and Univ. of Southampton, "myExperiment," https://www.myexperiment.org/, viewed on July 20, 2017.