

The Evolution of the Pegasus Workflow Management Software

Ewa Deelman, *Fellow, IEEE*, Karan Vahi, *Member, IEEE*, Mats Rynge, Rajiv Mayani, Rafael Ferreira da Silva, *Member, IEEE*, George Papadimitriou, Miron Livny

Abstract—Since 2001 the Pegasus Workflow Management System has evolved into a robust and scalable system that automates the execution of a number of complex applications running on a variety of heterogeneous, distributed high-throughput, and high-performance computing environments. Pegasus was built on the principle of separation between the workflow description and workflow execution, providing the ability to port and adapt the workflow based on the target execution environment. Through its user-driven research and development it has adapted to the needs of a number of scientific communities, utilizing and developing novel algorithms and software engineering solutions. This paper describes the evolution of Pegasus over time and provides motivations behind the design decisions. The paper concludes with selected lessons learned.

Index Terms—workflow management systems, sustainability, large-scale distributed computing

1 INTRODUCTION

TODAY'S computational and data science applications process vast amounts of data (from remote sensors, instruments, etc.) and conduct large-scale simulations of underlying science phenomena. These applications comprise thousands of computational tasks and process large datasets, which are often distributed and stored on heterogeneous resources. Scientific workflows have emerged as a flexible representation that declaratively express the complexity of such applications with data and control dependencies. They have become mainstream in domains such as astronomy, physics, climate science, earthquake science, biology, and others [1]–[4]. Workflows can be cyclic or acyclic, hierarchical (a workflow within a workflow), and form workflow ensembles (sets of interrelated workflows). Different semantics can be associated with the workflow graph, resulting in different types of execution [4]: in some cases, the nodes of the graph are standalone executables while in other cases, the nodes are long-lived services.

Workflows enable scientists to think about a sequence of analysis that needs to be performed on the data they collected. Workflows also enable scientists to analyze a series of simulations that can model our physical world or predict new systems behavior. There are a number of community codes that have been developed within various science domains, including but not limited to astronomy, bioinformatics, ecology, and material science. Scientific workflows allow scientists to chain these codes together to solve problems of greater complexity and scale. Individual codes can be developed by experts in a particular domain, resulting in

workflows that can be used as a multi-disciplinary research instruments. Workflows do not examine the internals of these codes but rather treat them as black boxes with specific inputs, parameters, and outputs. The challenge in building workflows is to connect the right components together based on their capabilities and the data they require and produce. Sometimes, additional components (called shims [5], [6]) need to be added to execute the workflows correctly. Because of the high level of abstraction they provide, workflows give access to sophisticated analysis and simulations to non-developers. Since they explicitly state the “recipe” for the computation, they can be used to evaluate the quality of the scientific result and can foster reproducibility.

In this paper, we describe the Pegasus workflow management system [7], [8], which is being used in a number of scientific domains. Since 2001, Pegasus has been created and enhanced in response to the needs of scientists to conduct complex computations on heterogeneous and distributed cyberinfrastructures. We describe how Pegasus was conceptualized, how it benefited from advances in Computer Science and how the professional software development approach has contributed to the adoption and sustainability of the software. Pegasus is grounded in the challenging and ever-increasing needs of a multitude of scientific applications and thus continuously innovates and enhances its capabilities. Pegasus delivers robust automation capabilities to researchers studying seismic phenomena [9], to astronomers seeking to understand the structure of the universe [10], to material scientists developing new drug delivery methods [11], and to students seeking to understand human population migration [12]. This paper describes the challenges of developing cyberinfrastructure capabilities that have an impact on scientific discovery and innovate in the changing cyberinfrastructure landscape.

- E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. Ferreira da Silva, and G. Papadimitriou are with the Information Sciences Institute, University of Southern California, Marina Del Rey, CA, 90292, USA.
E-mail: {deelman,vahi,rynge,mayani,rafsilva,georgpap}@isi.edu
- M. Livny is with University of Wisconsin, Madison.
E-mail: miron@cs.wisc.edu

Manuscript received XXXX; revised XXXX.

2 FROM THE VIRTUAL DATA CONCEPTS TO SCIENTIFIC WORKFLOWS

In 2000, the National Science Foundation funded the Grids Physics Network (GriPhyN) project [13], which was aimed at developing solutions to support a number of physics applications executing across a distributed platform, the Grid [14]. Among the target physics were two high-energy projects: CMS [15] and ATLAS [16] experiments at the Large Hadron Collider, an astrophysics project called the Laser Interferometer Gravitational-wave Observatory (LIGO) [17], [18], and the astronomy effort, the Sloan Digital Sky Survey (SDSS) [19]. The main idea of the GriPhyN project was to extend the database concept of view materialization to distributed environments. GriPhyN aimed to develop a “Virtual Data Grid,” which would deliver data products in response to a user’s request. The requested data could already exist somewhere in the distributed environment, or it would have to be derived.

Science, as exemplified by the above applications, is often distributed in nature. Collaborators work at different institutions and have access to a variety of heterogeneous resources on campus and to resources that are part of Department of Energy (DOE) laboratories. Scientists also utilize resources from national cyberinfrastructures such as the Open Science Grid (OSG) [20], XSEDE [21], and commercial and academic clouds [22]–[25]. Workflows, as designed by scientists, can also be very heterogeneous and require a varied set of computational resources to execute (e.g., simple clusters, high-performance computing resources, visualization clusters). Datasets are also distributed in the environment among different community archives, project storage systems, and local disks. Scientific workflows need to operate across all these resources to access data and leverage the available resources. The Grid aimed at bridging the heterogeneous and distributed resources by providing standardized authentication, job scheduling, data transfer, and information services [26], [27]. However, the workflow management system needed to build on these capabilities to coordinate job submission, data transfers, and the like.

Initially, we pursued the use of artificial intelligence (AI) planning technologies to create a plan based on the user’s request (goal). Using such technologies, we assisted LIGO by developing planner rules for the LIGO pulsar search [28]. We designed a web-based interface to help the user specify the request in terms of application-specific parameters, such as the time frame in which the search should be conducted. Based on the request, the system would determine whether the data was already instantiated and if not, the system would then determine how to compute the data. Next, the system would plan the data movements and computations required to obtain the results and would then execute that plan. The plan was based on the workflow described by the scientists, shown below in Fig. 1. The user would fill out a web form with metadata characterizing the desired result and the system would develop the plan and provide the results on the fly. Unfortunately, the LIGO scientists did not like our approach. The request representation (metadata entered via a web interface) was too abstract and the users preferred to manipulate the science workflows (like the one in Fig. 1) directly. Additionally, we ran into scalability issues

with the underlying Prodigy AI planner [29]. The LIGO workflows were expected to have at least tens of thousands of tasks with hundreds of thousands of files. AI planning techniques could not effectively reason about and arrive at a viable plan at that scale. As a result, we focused on facilitating explicit workflow design and efficient and scalable workflow management in distributed environments.

Working with the GriPhyN applications, we found that they had common workflow challenges. These challenges included the need to describe complex workflows in a simple way, the ability to access distributed, heterogeneous data, and the ability to compute resources that may change over time (in terms of software and hardware). Consequently, we identified the separation between workflow description and workflow execution and the resulting mapping of the workflow description onto the available distributed resources as a key aspect of our work. In Pegasus, the workflow description is abstract in that it uses logical names for transformations (tasks) and for files (task input and output data). The workflow management system then maps the abstract workflow to the resources based on their availability, performance, and the availability of input data at the resource, among others. The abstract workflow description itself is described in a simple XML-based format called the DAX (Directed Acyclic Graph in XML) [7]. To construct and run DAXes, scientists interact with Pegasus through the command line and API interfaces (in Java, Perl, Python, and R), through Jupyter Notebook [30], through portals and infrastructure hubs such as CyVerse [31] and HUBzero [32], through higher-level workflow composition tools such as Wings [6], or through application-specific composition tools (e.g., OpenSees [33]). Although creating a Pegasus workflow is easy (can be done in less than one day), the challenge of the workflow creation lies in the conceptual workflow design. The workflow components also have to be codes that are portable across different environments. For example, the workflow components cannot contain hard-coded paths. In cases where users want to submit the workflows to remote resources, these resources need to be able to accept incoming jobs, which may require interactions with the resource providers.

By focusing on the separation of the abstract and executable workflows, Pegasus can map and execute a workflow across heterogeneous platforms (such as campus and high-performance clusters and clouds). The user’s workflow can also be migrated between platforms as the platform hardware and software change over time. This paradigm is especially powerful as it has allowed users to migrate and adapt to the changing computing landscape over the past 20 years without many changes to their workflow description. Users have been able to take advantage of improvements both in data management and computing infrastructure to scale the performance and throughput of their workflows, thus improving their scientific productivity.

3 BUILDING ON PROVEN ABSTRACTIONS AND TECHNOLOGIES

From the beginning, Pegasus’ philosophy was to rely on existing research in graph theory, databases (virtual data), and compilers (data re-use) and to augment and adapt

Fig. 1. A sequence of data processing steps and resulting products for the LIGO pulsar search. Raw instrumental data is stored in an archive. Short duration frames are cleaned/calibrated, combined into longer duration frames, and transposed to the time-frequency spectrum. Gravitational-wave templates are compared to the time frequency image and potential gravitational-wave events are stored in a database.

that research to the concept of workflows and the target cyberinfrastructure. Pegasus builds on the foundation of abstractions of directed acyclic graphs (DAGs), fundamental constructs (recursion), and scalable algorithms (graph traversals, graph node clustering).

Recognizing the importance of the separation of concerns and the benefits of re-use of robust software solutions, Pegasus built on top of the HTCondor task management system [34] and its DAGMan workflow engine, described below. The collaboration with the HTCondor team has lasted throughout the years, bringing mutual benefits to the two projects by addressing common challenges from different points of view (planning and scheduling) and delivering solutions that work well together for the scientists.

Initially, Pegasus had 3 main components: the mapper, the workflow engine, and the job scheduler. With time, other capabilities have been added. Fig. 2 shows the development of key Pegasus capabilities over time and indicates the main applications that have driven the development of these Pegasus capabilities (LIGO and SCEC—a major earthquake science project described in Section 7. In 2005, we started developing the first remote workflow execution engine to support task clustering and in 2009, we started developing web-based monitoring capabilities, augmenting our command-line interfaces. This brought our main system components to five. They are illustrated in Fig 3 and described in greater detail in Section 4.

The main Pegasus components are:

- 1) **Mapper:** Generates an executable workflow based on an abstract workflow. It finds the appropriate software, data, and computational resources required for workflow execution. The Mapper may restructure the workflow to improve performance and adds data management and provenance capture jobs to the workflow. To support the mapping, Pegasus uses two types of catalogs: a site catalog to discover resources and their properties: and a

transformation catalog to discover the location and resource needs of codes and a replica catalog to discover data location. These catalogs can be provided by the user or the cyberinfrastructure.

- 2) **Local Workflow Execution Engine** (provided by DAGMan [35]): Submits and tracks the execution of the jobs defined in the executable workflow according to their dependencies and constraints on number of queued jobs.
- 3) **Job Scheduler** (provided by HTCondor sched[36]): Manages individual jobs, supervises their execution on local and remote resources, and provides task-level reliability.
- 4) **Remote Workflow Execution Engine** (M) Manages the execution of tasks, which can be structured as a sub-workflow, on remote resources. The Remote Workflow Execution Engine is scheduled with the sub-workflow to the remote resource.
- 5) **Monitoring Component** Monitors the progress of the workflow, parses jobs and task logs, and populates the jobs and task logs into a database. The database stores both performance and provenance information. It also sends notifications back to the user about events such as failure, success, and completion of tasks, jobs, and workflows, as well as user-defined events. The database provides information to the dashboard, which displays real-time monitoring information and helps with debugging.

3.1 Mapper and Data Reuse

To support the concept of virtual data, one of the earlier capabilities that we developed was data reduction and reuse that the Mapper applies when generating an executable workflow from the abstract workflow. The information about the input and output files used and generated by tasks in the abstract workflow is coupled with the location information of existing datasets discovered in data catalogs.

Fig. 2. Pegasus development and releases over time. The bottom rows indicate the key developments driven by the LIGO and SCEC applications. Selected capabilities that were released at particular times are highlighted and their importance and relevance to the two key applications are described below.

Fig. 3. Pegasus with its components in context of user interfaces and cyberinfrastructure.

Based on the existence of the files named in the workflow, the Mapper reuses the existing data and reduces (prunes) the workflow to compute only the necessary results. The underlying algorithm [8] is similar to “make” functionality while building software. It works bottom up for each task whose output locations exist in the data catalog. If the outputs exist, the task is removed and the operation gets applied to task’s immediate parents, marking them for deletion (unless they are required to derive some other data product in that workflow). In a degenerate case where the final outputs of a workflow already exist, all the compute jobs in the workflow will be deleted. In that case, the Mapper will only add the data movement nodes that copy the final outputs from their existing locations recorded in the data catalog to the location the user requested.

This is a particularly powerful concept and has proved useful for collaborating groups and for workflow-level checkpointing. When intermediate results are saved and a fatal failure occurs during execution, the user can re-submit the abstract workflow and the system will reduce the workflow to include only the remaining tasks and map only those. In addition to abstract workflow-level checkpointing,

which provides fault tolerance at the mapper level, building on top of HTCondor allowed us to leverage its resilience capabilities such as checkpointing at the level of the executable workflow and job retries at the scheduler level.

4 PEGASUS THROUGH THE YEARS: USER-DRIVEN DESIGN

Since its inception, Pegasus was designed based on the user’s needs. We worked closely with our collaborators to understand their workflow needs and tried to abstract their needs to general concepts and challenges that could be solved using new algorithms and software solutions. Some of the main concepts implemented over the years are described below.

Since many of our users have access to a variety of heterogeneous and distributed environments, Pegasus allows scientists to submit locally and run globally. Scientists can deploy Pegasus in their local environment, requiring no support from a site administrator and eliminating any impact on the remote cyberinfrastructure. A collaboration can deploy Pegasus on a shared submit host to send jobs

Fig. 4. A typical environment available to today's scientists. Local resources include a work ow description, a local compute resources, data storage, and potentially a data collection capability. Pegasus resides on a "submit host" managed by a user or collaboration. The tasks and data movement are managed from that host across the heterogeneous, distributed resources.

to the distributed resources (a local resource is shown in Fig 4). Because of the various types of data and compute resources available to scientists, we have developed specialized work ow execution engines and data transfer tools that can operate seamlessly in varied infrastructures as described below. Pegasus also capitalizes on remote job submission interfaces that are available today, primarily provided by HTCondor.

Over the years we have developed and refined a number of capabilities to meet the needs of our users. In this section, we describe the developments in a chronological order (see Fig. 3) and explain the motivation behind them.

4.1 Support for Replica Catalog: Automated Data Selection and Management

LIGO and other projects store and potentially replicate their data across a number of storage systems and archives. As a result, the work ow management system needs to locate and select the data to be accessed. In some cases, the collaborations maintain their own catalogs and in other cases, they rely on Pegasus' built-in capabilities. Based on the catalog information, Pegasus selects and stages the data to the computations, moves the results to the user-specified location, registers data into user-indicated repositories, cleans up data from the execution sites when it is no longer needed downstream, and provides performance and provenance information for the results.

Because the data flows through the work ow execution, Pegasus needs to deliver these data files to the work ow jobs. The work ow jobs can run on shared or non-shared filesystem setups. When running on a shared filesystem setup, Pegasus uses the filesystem for communication between tasks running on that resource. When running on non-shared filesystem setups, Pegasus needs to pull the data to local storage and push the results back out to a remote location. This is done using a specialized remote execution engine called PegasusLite. In non-shared filesystem deployments, Pegasus also enforces integrity checking for

datasets that are staged to a compute node before a user task is launched by PegasusLite. For raw input data, users can specify the sha256 checksums in the replica catalog, or let Pegasus compute them as part of the data transfer tasks that place data to the data staging server. When a user task completes, sha256 checksums are automatically computed for the output files created by the task, recorded in the provenance record for task, and then populated in the Pegasus monitoring database on the submit host.

4.2 Data Cleanup, Data Footprint Management

Around 2003, LIGO wanted to extend its work ow execution platform. Thus, LIGO targeted Grid2003, the precursor to the Open Science Grid. The amount of shared space available to the user on the compute sites making up Grid2003 was on the order of 100's of MB, whereas the size of the data sets analyzed and generated by LIGO work ow was on the order of a TB. This resulted in a mismatch between the capacity of the execution system to store the needed data and the work ow data sizes. To minimize the work ow data footprint within an execution site, we developed algorithms that schedule the work ow tasks based on the storage capacity of execution sites [37] and complimentary algorithms that reduce the data footprint on an execution site based on the usage patterns of data within a work ow. The latter analyzed the entire work ow to determine when the data was used for computations and was no longer needed by the downstream tasks. When it was determined that it was safe to delete the data from the execution site, "clean up" nodes were automatically added to the work ow. As a result, some applications were able to reduce their data footprint by as much as 50

4.3 Task Clustering

In some cases, the work ows designed by scientists consist of a large number of short running tasks [38]. These tasks incur large scheduling overheads as compared to their runtime as each task needs to wait in the execution queue at the

remote site. When work flows have thousands of tasks, these overheads can overwhelm the computations. Clustering “small” tasks into “larger”, more compute-intensive jobs can reduce the overhead incurred by the overall work flow. We have explored various task clustering algorithms that can take into account the availability of resources, the potential parallelism in the work flow, and the overheads incurred when scheduling tasks onto distributed resources. In the case of an astronomy application, the performance of the work flow was improved by over 90% compared to the default configuration when appropriate task clustering was used [38]. Failures in distributed environments are also a common issue faced by applications. Thus, we developed fault-tolerant task clustering algorithms that explored the tradeoff between the size of the clusters and the need of task cluster rescheduling in case of failures. As the size of the task cluster grew, so did the cost of fault recovery. We also developed algorithms that dynamically adjusted cluster size based on the observed failures in the environment [39].

4.4 Work flow Partitioning, Just-in-time Planning

As the users started to push the scale of the work flows they ran through Pegasus, we started exploring the benefits of scheduling portions of a work flow at a time. Additionally, resource availability in a distributed environment could change while the work flow was executing. Task clustering could help reduce the size of the work flow that was mapped onto the resources. However, to address the dynamic nature of the resources, it was best to do the mapping of the abstract work flow to the resources “just-in-time”. In order to support this model, we developed three different approaches: 1) work flow partitioning, 2) hierarchical work flow description, and 3) the specialized PegasusLite engine. Work flow partitioning algorithms divide the work flow graph into smaller portions so that Pegasus can operate on the work flow a partition at a time [8], [40]. Dependencies between the partitions reflect the dependencies in the original work flow. This allows Pegasus to map and execute the partitions in order of their dependencies. Hierarchical work flows allow for a definition of a work flow within a work flow. As a result, they can help with scalability and just-in-time planning by having the work flow management system work on one subwork flow at a time. However, hierarchical work flows also support dynamic execution since they can be used to represent loops and conditional execution. Because LIGO was executing on HTCondor pools, the WMS did not know exactly which resource the jobs would be scheduled to. As a result, Pegasus could not stage data ahead of the job execution. Thus, we developed the PegasusLite remote work flow engine, which is staged with the job and has the functionality to discover the appropriate directory to run the job in, locate the input data, and to stage the input data dynamically to the computing resource. Then, PegasusLite remote work flow engine runs the job, stages the data out to a repository, and cleans up the directory on the remote node. This work flow engine also turned out to be useful in 2008, when we started running on clouds, and more recently to setup the application container if required for a user’s task. Pegasus currently supports both Docker [41] and Singularity [42].

4.5 Pegasus MPI Cluster: A Remote Work flow Execution Engine

In some cases, work flows, such as SCEC, have large numbers of data-intensive single core tasks that should be colocated with large-scale parallel computations. As a result, these work flows need to be executed on HPC systems. It is impossible to schedule hundreds of thousands of single core tasks to an HPC scheduler. Thus, we developed the Pegasus MPI Cluster (PMC) [43] work flow engine. The PMC uses MPI [44], a high-performance communication messaging library, and the master-worker paradigm to execute large, fine-grained work flows. In case of failure, PMC generates a rescue log which can be used to recover the state of a failed work flow when it is resumed.

4.6 Online Monitoring

Our initial work with applications highlighted an important but often overlooked aspect of work flow execution and management: comprehensive monitoring. Because the work flows can be large and contain millions of tasks, scalability and reliability are critical, implying the need for multi-level monitoring, fault detection, and debugging tools. With this goal in mind, we developed and hardened our monitoring infrastructure, which takes logs of a running work flow and populates them into a relational datastore [45]. The monitoring infrastructure then uses this information for various visualizations. We also developed debugging and performance statistics tools that pull information from this datastore and allow users to debug their work flows and gain insights into their performance in terms of resources used [46].

5 SOFTWARE DEVELOPMENT PROCESSES

To sustain Pegasus software over time, make it reusable, and sustain the scientific methods that depend on Pegasus, it is necessary to make the software easy to compile, install, configure, and modify. Over the years, we adopted specific engineering processes to support the code development:

- Employment of professional developers to develop code and support users; these developers have been on the project since 2001. Students also participate in the project by developing new algorithms and evaluating the algorithms’ performance on real world applications. Promising algorithms are integrated into Pegasus.

- Use of GitHub for code version control and as an inviting entryway for user and collaborators to contribute to the project: <https://github.com/pegasus-isi/pegasus>.

- Use of Atlassian JIRA Bug Tracking system for Bug Tracking.

- Use of the Atlassian Bamboo system for full code and documentation built at each code check-in. We also automatically exercise the code via Atlassian Bamboo, which runs an extensive test suite nightly and at developers’ requests. Our test suite consists of both unit tests and end-to-end work flow integration tests.

Providing access to online documentation, VMs that include software, tutorials, and example work flows, and documentation of APIs used for work flow construction and pluggable components (job schedulers, data source selectors, etc).

Updates to Pegasus software are released regularly. We have a major release approximately every 9 months and regular minor releases in the interim. An exhaustive list of features and fixed bugs is maintained both in JIRA and the release notes. We also release detailed documentation, tutorials, and webinars on the project website: <http://pegasus.isi.edu>.

An important component of our software is the support that we provide to our users. The code development is performed by research software developers that are committed to the project and the users. We provide email and mailing-list support, one-on-one assistance, conduct hands-on tutorials, and hold online office hours. Over the years, we have helped provide support to scientists executing work flows in their own environments as well as in national cyberinfrastructures such as NSF and commercial clouds, Open Science Grid (and its previous embodiment iVDGL), and XSEDE (and the initial TeraGrid) among others. As new systems come online, we request access to them and test work flow execution with a suite of realistic test work flows.

6 CROSS-DISCIPLINE POLLINATION

An important contribution of Pegasus research and software is cross-pollination of work flow management ideas between science fields. Grounding our work in challenges faced by domain scientists make our algorithms and software relevant.

A good illustration of such cross-pollination are the developments conducted to address the needs of LIGO and the Southern California Earthquake Center (SCEC) [47]. Traditionally, LIGO work flows were designed as single core workloads that targeted high-throughput infrastructures, such as the ones provided by HTCondor. On the other hand, SCEC's CyberShake work flows contained a mix of highly parallel work flow tasks (simulations) that require high-performance computing resources and MPI for communications followed by a large number of single core post-processing tasks. CyberShake work flows calculate the possible future seismic shaking that can be expected at different geographic locations. This information is needed, for example, by civil engineers to develop earthquake safe building codes. A run of Cybershake can be composed of 336 work flows that can run for 38 days and generate 0.5 PB of data. Because the data that needs to flow between MPI and single core tasks is large (on the order of hundreds of TBs), it is often beneficial to co-locate the MPI and single core computation within the HPC system. Both LIGO and SCEC had needs in terms of scalability (large work flows), reliability (the need to automatically recover from failures), and detailed performance monitoring. Because of their different types of work flows and different types of target cyberinfrastructures (high-throughput vs high-performance), however, the types of work flow capabilities they needed were different. Fig. 3 (above) illustrates the Pegasus developments driven by LIGO and SCEC over time.

LIGO's needs were focused particularly on data management: work flows accessing distributed data and large amounts of input data that needed to be processed on potentially storage-constrained systems that supported different data transfer protocols. As a result, we assumed a single logical file name space for a set of related work flows (also an assumption needed for the Virtual Data paradigm), included data file discovery through the use of replica catalogs (which maps logical files to potentially multiple physical file locations), developed algorithms for efficient "data cleanup", and developed a data transfer tool (pegasus-transfer) that supports a variety of data protocols (GridFTP [48], iRods [49], S3 [50], GS [23], SCP, and HTTP).

On the other hand, SCEC's work flows focused on efficient simulations, and a mix of parallel and single core codes drove requirements of efficient task execution. As a result, we developed algorithms for task clustering and work flow partitioning and designed a work flow engine (Pegasus MPI-cluster) that could efficiently manage the execution of single-core tasks on HPC platforms.

The power of this Pegasus-enabled capability to submit locally and run globally combined with the user-driven research and development process was on clear display in 2015 and 2016, when data streaming from the advanced LIGO detectors indicated the possibility of a gravitational wave detection. Time was of the essence, and the collaboration was looking to maximize their high-throughput computing (HTC) capacity. In addition to the traditional resources used by LIGO, the collaboration decided to use OSG and XSEDE. Because Pegasus supported an abstract work flow description (devoid of resources information) and because we developed the specialized MPI-based engine (for SCEC), LIGO scientists only needed to provide information about the OSG and XSEDE resources. Pegasus then mapped LIGO's work flows onto these resources and managed the flow of data and computations [51]. Additionally, Pegasus automatically generated the job submit files for the given infrastructures. It discovered the data locations, moved data to the computational resources, and moved the results back to LIGO archives, all without LIGO having to change its work flows. For the gravitational wave detection, LIGO used 23,405 work flows. Each of these work flows contained 60,000 computational tasks accessing 5,000 files (10GB total) and generating 60,000 output files (60GB total). Pegasus' ability to represent the work flow in an abstract way and its ability to truthfully translate the abstract work flow to an executable work flow were critical to scientific productivity and reproducibility. Since work flows developed by scientists are carefully validated, sometimes over a period of months (as is the case in LIGO), scientists cannot afford to change these work flows as the infrastructure and middleware are changing. Modifications would result in the need to re-validate the analysis.

Not all applications are as demanding as LIGO's, SCEC's, or other large-scale computing projects. However, all of these applications face at least some of the challenges described above. Thus, they directly benefit from the algorithms and software solutions we have designed for these cutting-edge work flows.

Fig. 5. Data Footprint of a single SoyKB work ow on Wrangler. Bottom curve shows the data footprint when executing the work ow with Cleanup. The top curve shows the data footprint when executing the work ow with No Cleanup.

7 EXAMPLES OF PEGASUS APPLICATIONS

Since January 2013, when we first started optionally collecting anonymous metrics of Pegasus usage, we have recorded approximately 1.5 million work ows with 1.2 billion jobs. This usage is primarily spread over 770 unique submit host domains, where each host supports a collaboration. These hosts are spread across 89 countries.

In addition to the LIGO and SCEC work ows described earlier in the paper, Pegasus is extensively used in production to achieve scientific results in a variety of domains such as astronomy, bioinformatics, civil engineering, climate modeling, earthquake science, and molecular dynamics. We describe a few of the applications here. One of the earliest adopters of Pegasus was the Montage project [52]. Montage delivers science-grade mosaics of the sky on demand. Montage work ows greatly benefited from our task clustering algorithms and ability to run on a variety of infrastructures, eventually supplanting the in-house MPI-based code [53]. Montage work ows served as a basis for a project to create a 16-wavelength infrared Atlas of the Galactic Plane [10] at a common spatial sampling of 1 arcsec, processed so that they appear to have been measured with a single instrument. For this effort, the astronomers used the concept of hierarchical work ows to create 900 image tiles, with each tile being a resultant output of the execution of a 5 degree Montage work ow. Overall, 18 million input files were processed, and all the execution was done on cloud resources provided by Amazon.

The ability to effectively work and distribute large datasets makes Pegasus an appealing work ow system for bioinformatics applications. An example is the OSG-GEM [54] work ow that processes DNA sequencing files to produce a Gene Expression Matrix (GEM), which contains quantified gene expression values across tens to thousands of samples. Due to storage and memory constraints on available compute nodes, the work ow splits raw input files into small pieces to process in parallel and merges intermediate output files. Demonstrating the portability of Pegasus work ows, OSG-GEM is configured to run on both the OSG and Jetstream [55], an academic cloud infrastructure. Other bioinformatics applications, such as those that

run a variety of plant sequencing work ows in Cyverse [31] and SoyKB [56], use Pegasus for their work. The SoyKB work ow, which sequences large sets of crop germplasm and generates whole genome scale structural variations and genotypic data, usually executes on WRANGLER, an HPC cluster with Pegasus managing the data transfers between Cyverse data stores and Wrangler's flash based scratch filesystem. This work ow benefits from Pegasus data cleanup capabilities that minimize the work ow data footprint on the scratch filesystems. For SoyKB pipelines, this reduces the peak work ow footprint from 3TB to 1.8TB as illustrated in Fig. 5.

Pegasus is also used in the field of human population genomics. For example, Pegasus was used to infer the demographic history of Ashkenazi Jews [12]. This work ow was a large set of independent simulations (100,000s to millions of tasks), followed by a set of merge and data summarization tasks. Overall, Pegasus managed approximately 3 million tasks, using more than 7 million CPU hours on the OSG. The PAGE project (Population Architecture using Genomics and Epidemiology) [57] has used Pegasus for imputation of 50,000 subjects genotyped in 20 different genome-wide association studies. Modelling the imputation pipeline as a Pegasus work ow enabled PAGE scientists to process data coming from new studies with minimal user intervention.

Over the years, Pegasus has been incorporated in a number of web portals targeting different scientific communities. One of the earliest uses of Pegasus in a portal was in astronomy, where Pegasus was used to launch montage work ows [58]. In recent years, Pegasus has been used in the bioinformatics domain to power phenotypic data curation systems for genomic consortiums [57] and for NIH data repositories such as the NIMH Repository and Genomics Resource [59]. The repository plays a key role in facilitating psychiatric genetic research by providing a collection of over 150,000 well characterized, high-quality patient and control samples for a wide-range of mental disorders. These funded studies are required to submit their phenotypic data to the NIHM repository via a web-based quality control system powered by Pegasus. Upon submission of new datasets, automated quality control work ows are launched by Pegasus that ensure data submitted is in accordance with study-specific requirements.

Pegasus is also used extensively in the modelling environment ranging from ecological models to seismic ones. As part of the MINT [60] project, Pegasus powers work ows that integrate heterogeneous models from separate disciplines, including geosciences, agriculture, economics, and social sciences. As part of the Simcenter CENTER project [61], Pegasus is used to manage large scale pipelines that execute on XSEDE and simulate the impact of natural hazards, such as earthquakes, on building structures. Pegasus also powers the Titan2D Hazard Map Work ow tool available on VHUB [62], which allows for simulation of volcanic ows. Scientists at ORNL have developed a SNS work ow [63] to characterize nanodiamonds that enhance the dynamics of tRNA in the presence of water. The work ow, enacted by Pegasus, calculates the model parameters that best match experimental data. These work ows use almost 400,000 CPU hours of time on DOE leadership class systems.

8 CONCLUSIONS

In this paper, we described the origins of the Pegasus work ow management system that is used in a number of scientific domains. Since its inception, Pegasus was driven by the needs of scientific applications executing in heterogeneous and often distributed computational environments. However, the needs specific to a group of scientists had to be balanced against the needs of other applications. Although there is a tendency to sometimes add new software components, providing more functionality can also lead to a more complex system, so it is important to carefully consider decisions about new features. Additionally, it is also important to abstract users' needs to general concepts applicable across domains to maximize return on investment.

Developing production quality software that the scientists can rely on takes time: not only to develop and harden the software but also to build the users' trust in the software and in the technical support they rely on. Releasing software open source with a permissive license also helps foster that trust. Although students have greatly contributed to the research agenda by developing new algorithms and testing their effectiveness and performance, the core software has been developed and supported by professional developers. Paying attention to good documentation, release notes, and nightly testing with realistic workloads and providing backward compatibility have also proven very useful for working with existing and new users.

In our work, we collaborated with domain scientists but also with computer scientists to look at the problems from different perspectives and experiences. In particular, we collaborated with experts in AI planning [28], semantic technologies [64], compilers [65], machine learning [46], cybersecurity, and networking [66] in addition to other distributed and high-performance computing researchers [39], [67]. The broad collaborations also helped us sustain funding for work ow management related research and software development (from NSF, DOE, NIH, and DARPA).

Looking into the future, we see a growing demand for automation and a convergence between the computing cyberinfrastructure. The HPC systems are becoming more complex, heterogeneous (CPUs, GPUs, FPGAs), and faulty and include specialized, multi-level data storage. Distributed systems have software defined capabilities (to set up dedicated network paths, for example) and specialized data storage attached to high-performance networks. Additionally, their performance is continuously increasing. Finally, clouds, with their virtualization technologies, provide a new platform for science. However, they can be very heterogeneous, providing a multitude of various virtual machine types. Not to mention, commercial clouds can be costly. In order to perform cutting edge science, researchers need to navigate this computing landscape with more sophisticated work ow systems and other automation technologies that allow one to provision and manage these resources on behalf of the users.

ACKNOWLEDGEMENTS

This work is funded by NSF contract #1664162, "SI2-SSI: Pegasus: Automating Compute and Data Intensive Science", and contract #1148515, "SI2-SSI: Distributed Work-

ow Management Research and Software in Support of Science"; and partly funded by DOE contract #DESC0012636, "Panorama—Predictive Modeling and Diagnostic Monitoring of Extreme Science Work ows", and NSF contract #1642053, "CICI: Secure and Resilient Architecture: Scientific Work ow Integrity with Pegasus".

REFERENCES

- [1] A. C. Jones, "Work ow and biodiversity e-science," in *Work ows for e-Science: Scientific Work ows for Grids*. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Eds. London: Springer London, 2007, pp. 80–90.
- [2] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Work ows for e-Science: scientific work ows for grids*. Springer Publishing Company, Incorporated, 2014.
- [3] D. Brown, P. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, "A case study on the use of work ow technologies for scientific analysis: Gravitational wave data analysis," in *Work ows for e-Science*. I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds. Springer London, 2007, pp. 39–59.
- [4] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific work ows," in *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004*. 2004, pp. 423–424.
- [5] P. Missier, K. Belhajjame, J. Zhao, M. Roos, and C. Goble, "Data lineage model for taverna work ows with lightweight annotation requirements," in *Provenance and Annotation of Data and Processes*. Springer Berlin Heidelberg, 2008, pp. 17–30.
- [6] Y. Gil, V. Ratnakar, J. Kim, P. A. González-Calero, P. Groth, J. Moody, and E. Deelman, "Wings: Intelligent work ow-based design of computational experiments," *IEEE Intell. Syst.*, vol. 26, no. 1, pp. 62–72, 2011.
- [7] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus: a work ow management system for science automation," *Future Gener. Comput. Systvol.* 46, pp. 17–35, 2015.
- [8] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: a framework for mapping complex scientific work ows onto distributed systems," *Scientific Programming Journal*, vol. 13, no. 3, pp. 219–237, 2005.
- [9] R. Graves, T. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, D. Okaya, P. Small, and K. Vahi, "CyberShake: A Physics-Based seismic hazard model for southern california," *Pure Appl. Geophys.*vol. 168, no. 3-4, pp. 367–381, 2011.
- [10] M. Rynge, G. Juve, J. Kinney, J. Good, B. Berriman, A. Merrihew, and E. Deelman, "Producing an infrared multiwavelength galactic plane atlas using montage, pegasus and amazon web services," in *23rd Annual Astronomical Data Analysis Software and Systems (ADASS) Conference 2013*.
- [11] V. Lynch, J. B. Calvo, E. Deelman, R. Ferreira da Silva, M. Goswami, Y. Hui, E. Lingerfelt, and J. Vetter, "Distributed work ows for modeling experimental data," in *2017 IEEE High Performance Extreme Computing Conference (HPEC'2017)*.
- [12] A. L. Gladstein and M. F. Hammer, "Substructured population growth in the ashkenazi jews inferred with approximate bayesian computation," *Molecular biology and evolution*vol. 36, no. 6, pp. 1162–1171, 2019.
- [13] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda, "GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. HPDC '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 225–.
- [14] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
- [15] "CMS collaboration: CMS experiment," <http://cms.web.cern.ch/content/cms-collaboration>.
- [16] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, and others, "Search for high-mass resonances decaying to dilepton nal states in pp collisions at TeV with the ATLAS detector," *J. High Energy Phys.* 2012.

- [17] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya, and Others, "All-sky search for periodic gravitational waves in the O1 LIGO data," *Phys. Rev. D Part. Fields* vol. 96, no. 6, p. 062002, 2017.
- [18] B. P. Abbott, S. Jawahar, N. A. Lockerbie, and K. V. Tokmakov, "LIGO scienti c collaboration and virgo collaboration (2016) GW150914: rst results from the search for binary black hole coalescence with advanced LIGO. physical review d, 93 (12). ISSN 1550-2368, <http://dx.doi.org/10.1103/PhysRevD.93.122003>," *PHYSICAL REVIEW D Phys Rev D* vol. 93, p. 122003, 2016.
- [19] D. G. York, J. Adelman, J. E. Anderson, Jr, and others, "The sloan digital sky survey: Technical summary," *Astron. J.*, 2000.
- [20] M. Altunay, P. Avery, K. Blackburn, B. Bockelman, M. Ernst, D. Fraser, R. Quick, R. Gardner, S. Goasguen, T. Levshina, M. Livny, J. Mcgee, D. Olson, R. Pordes, M. Potekhin, A. Rana, A. Roy, C. Sehgal, I. S. Igoi, and F. Wuertwein, "A science driven production cyberinfrastructure—the open science grid," *J. Grid Comput.*, vol. 9, no. 2, pp. 201–218, 2011.
- [21] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, and Others, "XSEDE: accelerating scienti c discovery," *Comput. Sci. Eng.*, vol. 16, no. 5, pp. 62–74, 2014.
- [22] Amazon.com, Inc., "Amazon Web Services (AWS)," <http://aws.amazon.com>.
- [23] "Google cloud platform," <https://cloud.google.com/>.
- [24] "Chameleon cloud," <http://www.chameleoncloud.org/>.
- [25] R. Ricci, E. Eide, and C. Team, "Introducing CloudLab: Scienti c infrastructure for advancing cloud architectures and applications," *the magazine of USENIX & SAGE* vol. 39, no. 6, pp. 36–38, 2014.
- [26] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience," *Concurr. Comput.*, vol. 17, no. 2-4, pp. 323–356, Feb. 2005.
- [27] I. Foster and C. Kesselman, "The globus project: a status report," *Future Gener. Comput. Syst.* vol. 15, no. 5, pp. 607–621, 1999.
- [28] J. Blythe, E. Deelman, Y. Gil, C. Kesselman, A. Agarwal, G. Mehta, and K. Vahi, "The role of planning in grid computing," in *ICAPS*, 2003.
- [29] J. A. Hendler, A. Tate, and M. Drummond, "AI planning: Systems and techniques," *AI magazine* vol. 11, no. 2, p. 61, 1990.
- [30] "Project jupyter," <http://www.jupyter.org>.
- [31] "CyVerse," <http://www.cyverse.org/>.
- [32] M. McLennan and R. Kennell, "HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering," *Comput. Sci. Eng.* vol. 12, no. 2, pp. 48–53, 2010.
- [33] F. McKenna, "OpenSees: A framework for earthquake engineering simulation," *Comput. Sci. Eng.* vol. 13, no. 4, pp. 58–66, Jul. 2011.
- [34] C. Hollowell, J. Barnett, C. Caramarcu, W. Strecker-Kellogg, A. Wong, and A. Zaytsev, "Mixing HTC and HPC workloads with HTCondor and slurm," *J. Phys. Conf. Ser.* vol. 898, no. 8, p. 082014, Nov. 2017.
- [35] J. Frey, "Condor DAGMan: Handling inter-job dependencies," University of Wisconsin, Dept. of Computer Science, Tech. Rep. 2002.
- [36] D. Thain and M. Livny, "Building reliable clients and services," *The Grid: Blueprint for a New Computing Infrastructure* vol. 2, 2003.
- [37] G. Singh, K. Vahi, A. Ramakrishnan, G. Mehta, E. Deelman, H. Zhao, R. Sakellariou, K. Blackburn, D. Brown, S. Fairhurst, and Others, "Optimizing work ow data footprint," *Sci. Program.*, vol. 15, no. 4, pp. 249–268, 2007.
- [38] G. Singh, M.-H. Su, K. Vahi, E. Deelman, B. Berriman, J. Good, D. S. Katz, and G. Mehta, "Work ow task clustering for best effort systems with pegasus," in *15th ACM Mardi Gras Conference*, 2008.
- [39] W. Chen, R. Ferreira da Silva, E. Deelman, and T. Fahringer, "Dynamic and fault-tolerant clustering for scienti c work ows," *IEEE Transactions on Cloud Computing*, 2015.
- [40] W. Chen and E. Deelman, "Partitioning and scheduling work ows across multiple sites with storage constraints," in *Parallel Processing and Applied Mathematics Springer Berlin Heidelberg*, 2012, pp. 11–20.
- [41] "Docker," <https://www.docker.com/>.
- [42] E. Le and D. Paz, "Performance analysis of applications using singularity container on SDSC comet," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact ACM*, Jul. 2017, p. 66.
- [43] M. Rynge, S. Callaghan, E. Deelman, G. Juve, G. Mehta, K. Vahi, and P. J. Maechling, "Enabling large-scale scienti c work ows on petascale resources using MPI Master/Worker," in *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond ser. XSEDE '12*. New York, NY, USA: ACM, 2012, pp. 49:1–49:8.
- [44] W. D. Gropp, W. Gropp, E. Lusk, A. Skjellum, and Argonne Distinguished Fellow Emeritus, *Using MPI: Portable Parallel Programming with the Message-passing Interface* MIT Press, 1999.
- [45] K. Vahi, I. Harvey, T. Samak, D. Gunter, K. Evans, D. Rogers, I. Taylor, M. Goode, F. Silva, E. Al-Shakarchi, G. Mehta, E. Deelman, and A. Jones, "A case study into using common Real-Time work ow monitoring infrastructure for scienti c work ows," *Int. J. Grid Util. Comput.*, vol. 11, no. 3, pp. 381–406, 2013.
- [46] T. Samak, D. Gunter, M. Goode, E. Deelman, G. Juve, F. Silva, and K. Vahi, "Failure analysis of distributed scienti c work ows executing in the cloud," in *Proceedings of the 8th International Conference on Network and Service Management International Federation for Information Processing*, Oct. 2012, pp. 46–54.
- [47] "SCEC Project, "Southern California Earthquake Center" ," <http://www.scec.org/>.
- [48] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The globus striped GridFTP framework and server," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing* Washington, DC, USA: IEEE Computer Society, 2005, pp. 54–.
- [49] A. Rajasekar, R. Moore, C.-Y. Hou, C. A. Lee, R. Marciano, A. de Torcy, M. Wan, W. Schroeder, S.-Y. Chen, L. Gilbert, P. Tooby, and B. Zhu, "iRODS primer: Integrated Rule-Oriented data system," *Synthesis Lectures on Information Concepts, Retrieval, and Services* vol. 2, no. 1, pp. 1–143, Jan. 2010.
- [50] "Amazon simple storage service (s3)," <http://aws.amazon.com/s3>.
- [51] D. Weitzel, B. Bockelman, D. A. Brown, P. Couvares, F. Würthwein, and E. F. Hernandez, "Data access for LIGO on the OSG," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact ACM*, Jul. 2017, p. 24.
- [52] G. B. Berriman, A. C. Laity, J. C. Good, J. C. Jacob, D. S. Katz, E. Deelman, G. Singh, M. H. Su, and T. A. Prince, "Montage: The architecture and scienti c applications of a national virtual observatory service for computing astronomical image mosaics," in *Proceedings of Earth Sciences Technology Conference*, 2006.
- [53] D. S. Katz, J. C. Jacob, E. Deelman, C. Kesselman, S. Gurmeet, S. Mei-Hui, G. B. Berriman, J. Good, A. C. Laity, and T. A. Prince, "A comparison of two methods for building astronomical image mosaics on a grid," in *International Conference on Parallel Processing (ICPP'05)*, 2005, pp. 85–94.
- [54] F. Feltus, W. Poehlman, M. Rynge, C. Branton, and B. Desinghu, "OSG-GEM: Gene expression matrix construction using the open science grid," *Bioinform. Biol. Insights* p. 133, 2016.
- [55] "Jetstream," <https://www.tacc.utexas.edu/systems/jetstream>.
- [56] Y. Liu, S. M. Khan, J. Wang, M. Rynge, Y. Zhang, S. Zeng, S. Chen, J. V. Maldonado dos Santos, B. Valliyodan, P. P. Calyam, N. Merchant, H. T. Nguyen, D. Xu, and T. Joshi, "PGen: large-scale genomic variations analysis work ow and browser in SoyKB," *BMC Bioinformatics* vol. 17, no. 13, p. 337, 2016.
- [57] T. C. Matisse, J. L. Ambite, S. Buyske, C. S. Carlsson, S. A. Cole, D. C. Crawford, C. A. Haiman, G. Heiss, C. Kooperberg, L. L. Marchand, T. A. Manolio, K. E. North, U. Peters, M. D. Ritchie, L. A. Hindorf, J. L. Haines, and PAGE Study, "The next PAGE in understanding complex traits: design for the analysis of population architecture using genetics and epidemiology (PAGE) study," *Am. J. Epidemiol.*, vol. 174, no. 7, pp. 849–859, Oct. 2011.
- [58] G. Singh, E. Deelman, G. Mehta, K. Vahi, M.-H. Su, G. B. Berriman, J. Good, J. C. Jacob, D. S. Katz, A. Lazzarini, K. Blackburn, and S. Koranda, "The pegasus portal: Web based grid computing," in *The 20th Annual ACM Symposium on Applied Computing*, 2005.
- [59] "NIMH repository and genomics resource," <https://www.nimhgenetics.org/>, accessed: 2019-NA-NA.
- [60] R. F. da Silva, D. Garijo, S. Peckham, Y. Gil, E. Deelman, and V. Ratnakar, "Towards model integration via abductive work ow composition and Multi-Method scalable model execution," in *9th International Congress on Environmental Modelling and Software* 2018.
- [61] "Simcenter: Computational modeling and simulation center," <https://simcenter.designsafe-ci.org/>, accessed: 2019-NA-NA.
- [62] J. L. Palma, L. Courtland, S. Charbonnier, R. Tortini, and G. A. Valentine, "Vhub: a knowledge management system to facilitate online collaborative volcano modeling and research," *Journal of Applied Volcanology* vol. 3, no. 1, p. 2, Feb. 2014.

