Community Resources for Enabling Research in Distributed Scientific Workflows

Rafael Ferreira da Silva, Weiwei Chen, Gideon Juve, Karan Vahi, Ewa Deelman
Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA
{rafsilva,wchen,gideon,vahi,deelman}@isi.edu

***Abstract*—A significant amount of recent research in scientific workflows aims to develop new techniques, algorithms and systems that can overcome the challenges of efficient and robust execution of ever larger workflows on increasingly complex distributed infrastructures. Since the infrastructures, systems and applications are complex, and their behavior is difficult to reproduce using physical experiments, much of this research is based on simulation. However, there exists a shortage of realistic datasets and tools that can be used for such studies. In this paper we describe a collection of tools and data that have enabled research in new techniques, algorithms, and systems for scientific workflows. These resources include: 1) execution traces of real workflow applications from which workflow and system characteristics such as resource usage and failure profiles can be extracted, 2) a synthetic workflow generator that can produce realistic synthetic workflows based on profiles extracted from execution traces, and 3) a simulator framework that can simulate the execution of synthetic workflows on realistic distributed infrastructures. This paper describes how we have used these resources to investigate new techniques for efficient and robust workflow execution, as well as to provide improvements to the Pegasus Workflow Management System or other workflow tools. Our goal in describing these resources is to share them with other researchers in the workflow research community. All of the tools and data are freely available online for the community at http://www.workflowarchive.org. These data have already been leveraged for a number of studies.**

Keywords*—*Scientific Workflows; Workload Profiling and Characterization; Workload Archive; Workflow Simulation

I. Introduction

In the last decade, scientific workflows have been used extensively by the scientific research community to exploit coarse-grained parallelism in applications running on distributed infrastructures such as clusters, grids, and clouds [1]. During that time, these infrastructures have become more complex, heterogeneous, and prone to failures. At the same time, scientific workflows have been growing in terms of the size and complexity of data and computations. Therefore, several techniques, heuristics, and mechanisms have been developed to address these challenges aiming to optimize the workflow execution.

Simulation enables experimentation under controlled conditions. It is often more efficient at exploring large sets of influential parameters and it can evaluate solutions in environments that do not yet exist [2]–[5]. When studying the execution of scientific workflows, simulation often requires models (e.g. tasks inter-arrival time, system overheads, and failure rates), and a set of workflow applications or benchmarks to validate assumptions and methods. However, there is a shortage of realistic datasets and tools that can be used for such simulations.

In this work, we describe a collection of collaborative tools and data that together have enabled research and the development of the **Pegasus Workflow Management System** (WMS) [6], [7]. Fig. 1 shows an overview of the research process that integrates these community resources. Workflow execution traces are collected and published in the **Workflow Gallery** [8]. From these traces, workflow execution profiles are built as the foundations for the development of workflow execution distributions. Execution traces, distributions, and application metadata are freely available online for the community. Execution distributions enable the development of the **Workflow Generator** toolkit [9]. The toolkit generates synthetic workflows, resembling those used by real world scientific applications, to facilitate the evaluation of workflow algorithms and systems under different configurations. *External Models* represent external characteristics not captured by synthetic workflows as for example, execution overheads, task and resource failures, or energy consumption. Models and synthetic workflows are used as input to the **WorkflowSim** [10] simulator framework. Finally, relevant research results are implemented in the Pegasus WMS or other workflow tools.

This paper describes the following community resources:

- a collection of workflow analysis and evaluation tools that can work independently or together;

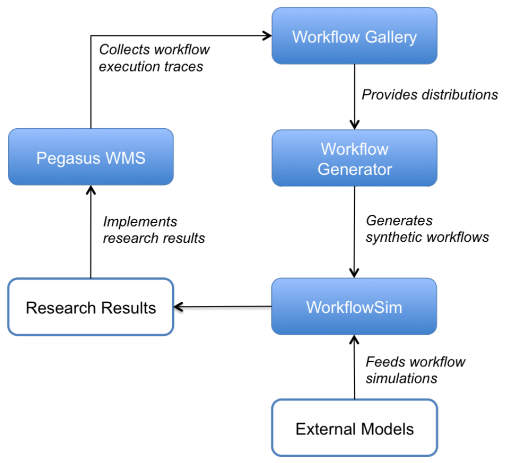
Workflow Gallery
Collects workflow execution traces
Provides distributions
Pegasus WMS
Workflow Generator
Implements research results
Generates synthetic workflows
Research Results
WorkflowSim
Feeds workflow simulations
External Models

Fig. 1: Overview of the community resources.

- a workflow *gallery* that collects and publishes execution traces of real scientific workflow applications;
- a synthetic workflow generator that can produce realistic synthetic workflows based on profiles extracted from execution traces;
- a research process to integrate these tools along a simulator framework that enables research in new methods and systems for scientific workflow management.

The paper is organized as follows. In Section II, we describe the Pegasus WMS and its main subsystems. Section III presents the Workflow Gallery and how workflow execution profiles and characterizations are built. In Section IV, we introduce the Workflow Generator toolkit, and in Section V we describe our simulator framework, and show use cases about how these tools and data have contributed to advance computer science research. Section VI presents the related work, and Section VII concludes the paper and presents possible future work.

II. The Pegasus Workflow Management System

Simulations and data-analysis of complex systems often require the execution of large sets of computational tasks, as well as the coordination of large-scale data movement on distributed computational resources such as campus clusters, grids, and clouds. Pegasus WMS can manage workflows comprised of millions of tasks, all the while recording information about the task execution and the intermediate result generation so that the provenance of the final result is clear. The Pegasus WMS approach is to bridge the scientific domain and the execution environment by mapping a scientist-provided high-level workflow description, an *abstract workflow* (does not contain resource information, or the physical locations of data and executables), to an *executable workflow* description of the computation. Workflows are described as directed acyclic graphs (DAGs), where nodes represent individual computational tasks and the edges represent data and control dependencies between tasks. In Pegasus, the abstract workflow description is represented as a DAX (DAG in XML), which captures all the computational tasks, the execution order of these tasks, and for each task the required inputs, expected outputs, and the arguments with which the task should be invoked.

A workflow is submitted to Pegasus WMS that resides on a user-facing machine named the *submit host*. The target execution environment can be a local machine, like the submit host, a remote physical cluster or grid, or a virtual system such as the cloud. In our model, it is the workflow management system's responsibility to not only translate tasks into jobs and execute them, but also to manage data, monitor the execution, and handle failures. A job is an atomic unit seen by the execution system. A job may contain multiple tasks to be executed in sequence or in parallel if applicable. Data management includes tracking, staging, and acting on workflow inputs, intermediate products (data sent between tasks in the workflow), and the output products requested by the scientist. These actions are performed by the following major Pegasus subsystems:

***Mapper*:** Generates an executable workflow based on an abstract workflow provided by the user or a workflow composition system. It finds the appropriate software, data, and computational resources required for the execution. The Mapper can also restructure the workflow in order to optimize performance, and adds transformations for data management and provenance information generation.

***Local Execution Engine*:** Submits the jobs defined by the workflow in order of their dependencies. It manages the jobs by tracking their state and determining when jobs are ready to run. It then submits jobs to the local scheduling queue.

***Job Scheduler*:** Manages individual jobs present in the local scheduling queue; supervises their execution on local and remote resources.

***Remote Execution Engine*:** Manages the execution of one or more tasks, possibly structured as a sub-workflow, on one or more remote compute nodes.

***Monitoring Component*:** A runtime monitoring daemon launched when the workflows start executing. It monitors the running workflow, parses the workflow job logs, and populates them into a workflow database. The database stores both performance and provenance information. It also notifies the user about events such as failures, success, and completion of jobs and workflows.

Fig. 2 shows an overview of the Pegasus WMS architecture illustrating how scientists interface with Pegasus and subsystems to execute workflows in distributed execution environments. Scientists can interact with Pegasus via command line and API interfaces, portals and infrastructure hubs [11], and high-level, or application-specific composition tools. The system maps the abstract workflow onto the execution environment, by transforming the abstract workflow into an executable workflow, which includes computation invocation on the target resources, the necessary data transfers, and data registration. The monitoring component captures job execution information that are used to build profiles and characterizations of workflow executions. Details on how the data is captured, and on how profiles and characterizations are built, are presented afterwards in Section III-A.

III. The Workflow Traces Archive

Workload traces have been widely used to profile and characterize workflow executions, and to build distributions of workflow execution behaviors, which are used to evaluate methods and techniques in simulation or in real conditions. In this section, we introduce the tool used to collect, summarize, and publish execution traces from Pegasus workflow executions. In addition, we present methods and techniques we use to profile and characterize these executions. Traces, profile data, and characterizations are freely available online for the community through the *Workflow Gallery* (http://www.workflowarchive.org) [8].

The execution profile data and characterizations are used to build distributions of workflow applications. Distributions are used to generate synthetic workflows (see Section IV) and may include task runtime and data sizes, and the number of tasks

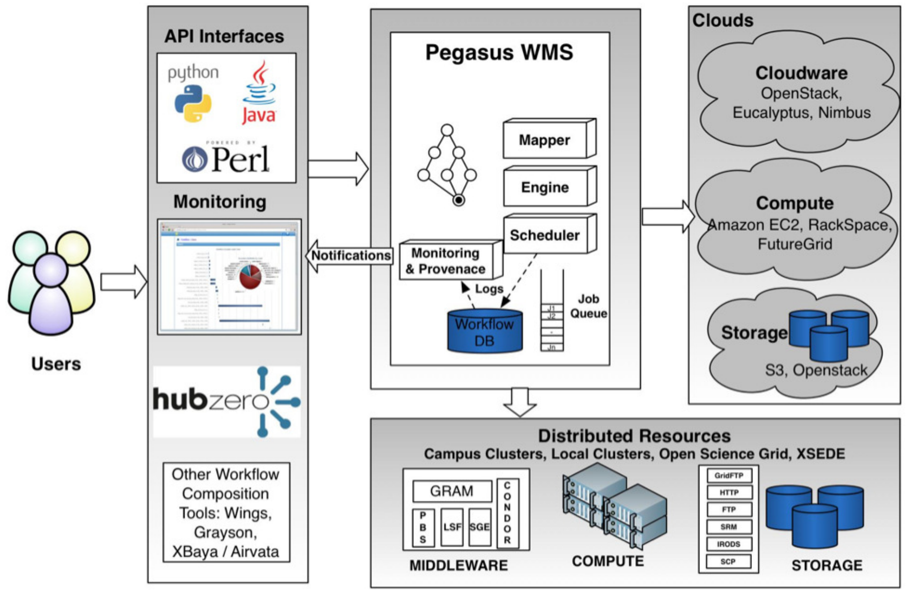

Fig. 2: Overview of the Pegasus WMS architecture.

in a workflow. Moreover, profile data and characterizations can be used by task runtime estimation methods to predict workload needs that will inform resource provisioning and task scheduling decisions. As estimates of task runtimes can vary over time (based on new data being collected), we have developed an online estimation process of task runtime and resource needs for scientific workflow applications.

A. *Workflow Profiling and Characterization*

We have developed the Kickstart [12] profiling tool to collect and summarize performance metrics for workflow applications. Kickstart wraps workflow tasks to monitor and record task execution information. It captures profiling data such as process I/O, runtime, memory usage, and CPU utilization (*utime* and *stime*) by using query mechanisms and low overhead notifications. Fine-grained information can also be gathered by using interpositions and events for detecting when processes start and stop; or by using full system calls or function interposition for detailed measurements of files accessed and I/O operations. However, this approach may add a significant overhead to the workflow execution.

Table I shows an example of the execution profile of an 8 degrees square Montage [13] workflow executed on Amazon EC2. For some task types, resource utilization is very low, which suggests that I/O operations dominate the runtime. Thus, these tasks are characterized as data-intensive tasks. This assumption is supported by the large amount of data consumed by these tasks. On the other hand, high CPU utilization suggests compute-intensive tasks. Similarly, this assumption is supported by the small amount of data consumed by these tasks. Profile data and detailed characterization of several real scientific workflow applications are available in [14], [15].

Capturing workflow task execution information allows us to automatically estimate the resource needs of future tasks. We assume that task needs such as runtime, I/O write, and memory peak, can be estimated based on the I/O read parameter. Thus, our characterization method [16] looks for correlations between the input data and selected parameters. If no correlation is detected, execution datasets are divided into sub-datasets by a density clustering technique. Smaller datasets may have a higher correlation coefficient, or a lower standard deviation from the mean.

Table II shows an example of the use of the automatic characterization method to determine correlation (ρ) and standard deviation values (σ) for the Montage workflow. Datasets with high correlation values are not clustered. For datasets with low correlation values, we cluster the datasets. The goal of clustering is to find sub-sets (clusters) of the datasets with a higher correlation, or smaller standard deviation values. In clusters, where the correlation is null and the standard deviation is negligible, the data is seen as a constant value.

B. *Estimation of Workflow Task Needs*

Task characteristics such as runtime, disk space, and memory consumption, are commonly used by scheduling algorithms and resource provisioning techniques to provide successful and efficient workflow executions. These methods often assume that accurate estimates are available, but in production systems it is hard to pre-compute such estimates with good accuracy. Therefore, we use an online estimation process [16] based on the MAPE-K loop (Monitoring, Analysis, Planning, Execution, and Knowledge), where task executions are constantly monitored. Upon task completion, estimated values for the task are updated with the real values, and based on these values a new prediction is done for subsequent tasks (tasks that are data-dependent of the current task). Parameter predictions (e.g. runtime) are based on regression trees. First, tasks are classified by application, then by task type. The next step decides

Task Type	Count	Runtime		I/O Read		I/O Write		Memory Peak		CPU Utilization	
		Mean (s)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (%)	Std. Dev.
mProjectPP	2102	1.73	0.09	2.05	0.07	8.09	0.31	11.81	0.32	86.96	0.03
mDiffFit	6172	0.66	0.56	16.56	0.53	0.64	0.46	5.76	0.67	28.39	0.16
mConcatFit	1	143.26	0.00	1.95	0.00	1.22	0.00	8.13	0.00	53.17	0.00
mBgModel	1	384.49	0.00	1.56	0.00	0.10	0.00	13.64	0.00	99.89	0.00
mBackground	2102	1.72	0.65	8.36	0.34	8.09	0.31	16.19	0.32	8.46	0.10
mImgtbl	17	2.78	1.37	1.55	0.38	0.12	0.03	8.06	0.34	3.48	0.03
mAdd	17	282.37	137.93	1102.57	302.84	775.45	196.44	16.04	1.75	8.48	0.11
mShrink	16	66.10	46.37	411.50	7.09	0.49	0.01	4.62	0.03	2.30	0.03
mJPEG	1	0.64	0.00	25.33	0.00	0.39	0.00	3.96	0.00	77.14	0.00

TABLE I: Execution profile of the Montage workflow execution for a 8 degrees square region of the sky.

Task	Runtime			I/O Write			Memory Peak		
	c	ρ	σ	c	ρ	σ	c	ρ	σ
mProjectPP	1	0.00	0.68	1	0.88	0.19	1	0.88	0.40
	2	0.00	0.54						
	3	0.00	0.28						
mDiffFit	1	0.04	1.08	1	0.01	1.17	1	0.01	1.03
	2	0.05	0.84	2	0.00	0.00	2	0.00	0.00
	3	0.07	0.61						
mConcatFit	1	0.00	5.27	1	0.00	0.01	1	0.00	0.01
mBgModel	1	-0.99	88.50	1	1.00	0.00	1	0.96	0.01
mBackground	1	-0.02	1.46	1	0.99	6.44	1	0.99	5.78
	2	0.00	0.00						
	3	-0.09	0.66						
mImgtbl	1	0.00	0.17	1	0.92	0.05	1	0.88	0.13
	2	0.28	1.85						
mAdd	1	0.84	14.03	1	0.98	383.86	1	0.97	3.40
mShrink	1	0.00	2.25	1	1.00	0.00	1	0.00	0.01
	2	0.00	1.58						
mJPEG	1	0.00	0.07	1	0.00	0.00	1	0.98	0.01

TABLE II: Characterization of the Montage workflow: cluster number (c), correlation (ρ), and standard deviation (σ) values.

whether runtime, I/O write, or memory parameters should be estimated based on the input data size. If the parameter is strongly correlated to the input data, values are estimated according to the ratio parameter/input data size. Otherwise, values are estimated as the mean. Fig. 3 summarizes the online estimation process. Experimental results show that our online estimation process provides more accurate estimates than an offline method, where the resource usage for all workflow tasks is estimated *a priori*.

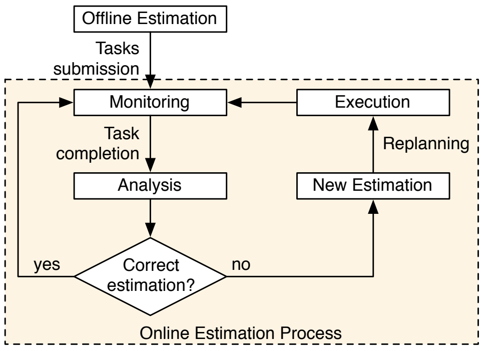

Fig. 3: Online task estimation process.

IV. The Workflow Generator Toolkit

Workflow execution traces are also commonly used to evaluate novel algorithms and systems. However, current trace archives do not have a collection of scientific workflow runs that include a large number of possible application configurations. On the other hand, synthetic workflows can produce a large number of workflow instances, resembling those used by real world scientific applications, based on parameters derived from real workflow traces and user-specified parameters.

We have developed a Workflow Generator toolkit [9], which can generate a set of synthetic workflows with a variety of characteristics. The toolkit uses randomization and parameter sweeps to create many different synthetic workflows using the same set of input parameters. The generation process includes the identification of individual tasks and their composition, followed by the annotation of the workflow with information about task's runtime and memory usage, and input and output data sizes. The resulting workflows are represented in the DAX format (used by the Pegasus WMS, but easily readable by other systems).

The structure of a synthetic workflow depends on the targeted workflow application. The generator uses distributions derived from execution profiles and characterizations provided by the Workflow Gallery, and user-specified parameters such as the number of inputs and the number of tasks, to build synthetic workflows that are proportional to the real workflow. Pipeline structures commonly reflect the same structure as that in the real workflow. Data distribution, aggregation, or redistribution structures may vary according to the number of input data file or jobs. For instance, synthetic Montage workflows can be generated based on the degree square of the final image, where the generator will estimate the number of input files according to patterns observed in execution profiles; or based on the number of input images.

Workflow task runtimes and data sizes are also generated according to patterns observed in execution profiles. For instance, a specific job type may have the same input data size regardless of the number of jobs or input parameters; or the data size may be a function of the scale of the output data (e.g. the size of the final mosaic for the Montage workflow). When these values cannot be defined deterministically, the toolkit generates variations in the estimated values using truncated normal distributions. Users can specify whether a synthetic workflow will be compute- or data-intensive by tuning the approximate size of the input data or the factor to scale runtimes parameters.

Currently, the Workflow Generator toolkit is able to generate 20 types of synthetic workflows from different domains including astronomy, earth sciences, bioinformatics, weather, and ocean modeling. These synthetic workflows have been used

Application	Jobs	Avg. Execution Time	Avg. Data Size
LIGO	166	75.60s	∼258 MB
Montage	158	11.60s	∼4 MB
CyberShake	1675940	460840.60s	∼132 MB
Epigenomics	590	84.01s	∼408 MB
SIPHT	28	27.05s	∼1 MB

TABLE III: Characteristics of the workflow applications.

in several research studies as presented in Section V-C. We also published a large collection of toolkit-generated synthetic workflow samples [9], which have been used as benchmarks. This collection includes 5 workflow applications and 2,840 workflow instances with different parameter sets derived from our trace analysis (see Section III-A). Table III summarizes the main characteristics of these workflow applications, and below we briefly introduce them:

LIGO: Laser Interferometer Gravitational Wave Observatory (LIGO) [17] workflows are used to search for gravitational wave signatures in data collected by large-scale interferometers. The observatories' mission is to detect and measure gravitational waves predicted by general relativity (Einstein's theory of gravity), in which gravity is described as due to the curvature of the fabric of time and space.

Montage: Montage [13] is an astronomy application that is used to construct large image mosaics of the sky. Input images are reprojected onto a sphere and overlap is calculated for each input image. The application re-projects input images to the correct orientation while keeping background emission level constant in all images. The images are added by rectifying them to a common flux scale and background level. Finally the reprojected images are co-added into a final mosaic.

Cybershake: CyberShake [18] is a seismology application that calculates Probabilistic Seismic Hazard curves for geographic sites in the Southern California region. It identifies all ruptures within 200km of the site of interest and converts rupture definition into multiple rupture variations with differing hypocenter locations and slip distributions. It then calculates synthetic seismograms extracting the peak intensity measures to produce probabilistic seismic hazard curves for the site.

Epigenomics: The Epigenomics workflow [19] is a CPU-intensive workflow application. Initial data is acquired from the Illumina-Solexa Genetic Analyzer in the form of DNA sequence lanes. Each Solexa machine can generate multiple lanes of DNA sequences. The mapping software maps short DNA reads from the sequence data onto a reference genome, which displays the number of times a certain sequence expresses itself on a particular location on the reference genome.

SIPHT: The SIPHT workflow [20] conducts a wide search for small untranslated RNAs (sRNAs) that regulates several processes such as secretion or virulence in bacteria. The kingdom-wide prediction and annotation of sRNA encoding genes involves a variety of individual programs that are executed in the proper order using Pegasus WMS. These involve the prediction of ρ-independent transcriptional terminators and comparison of the inter genetic regions of different replicons and the annotations of any sRNAs that are found.

V. The Workflow Simulator Framework

Simulation frameworks are often used to evaluate methods and assumptions in controlled environments, and to understand details and configurations that are hard to model analytically. *WorkflowSim* [10] is an open-source workflow simulator framework that extends the CloudSim [3] simulator framework for Clouds by adding support for workflow execution simulations. The simulator implements several of the most popular dynamic and static workflow scheduling algorithms (e.g. HEFT [21], Min-Min). In addition, the simulator provides an extended model of resource and task execution failures, and a model of execution overheads intrinsic to the execution of scientific workflows in distributed environments [22].

In this section, we introduce the main components and capabilitiesof the WorkflowSim framework, and describe how the simulator has been used to investigate new techniques for efficient and robust workflow execution.

A. Components Overview

WorkflowSim is built on the top of the task scheduling layer of CloudSim. It also provides application and resource failure models. The simulator design includes the main categories of a WMS for scientific applications: composition, mapping, execution, and provenance [23]. The main components and their interactions are summarized in Fig. 4, and are described as follows:

Workflow Mapper: Imports DAX (DAG in XML) files and workflow execution metadata (e.g. data size); and creates a set of tasks, and assigns them to an execution site. Commonly, DAXes are synthetic workflows generated by the Workflow Generator toolkit, or traces gathered from workflow executions available in the Workflow Gallery.

Workflow Engine: Manages task state and dependencies determining when tasks are ready to run. Tasks are released to the Clustering Engine once all parent tasks are successfully completed.

Clustering Engine: Merges tasks into jobs to reduce the scheduling overhead [24]. Note that a job may contain multiple tasks. Task re-clustering can also be performed in a faulty environment that has transient failures.

Workflow Scheduler: Matches jobs to worker nodes based on the user-specified criteria. WorkflowSim relies on CloudSim

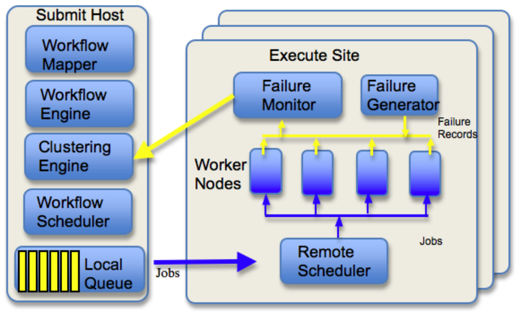

Fig. 4: Overview of the WorkflowSim architecture.

to provide an accurate and reliable job-level execution model, such as a time-shared model and a space-shared model. In addition, WorkflowSim has introduced different types of system overheads, and application and system failures [22] to improve the realism of the simulations.

Failure Generator**:** Injects task/job faults during the simulation. Failures are randomly generated following a given probability distribution and an average failure rate specified by the user.

Failure Monitor**:** Collects failure records (e.g. resource, job, and task identifiers). These can be used by dynamic scheduling strategies to improve the workflow execution.

B. Functionalities and External Models

WorkflowSim supports the common features exposed by workflow management systems, and includes the major optimization methods described in the literature. In addition, WorkflowSim is been constantly extended by its open-source research community with new methods and techniques. Below we summarize the main functionalities and methods available in the framework:

File Systems**:** In addition to the local filesystem provided by CloudSim, our simulator framework also provides shared and distributed filesystem models. The shared filesystem is a centralized filesystem in which data storage is shared among all the execution nodes within a data-center. The communication cost is thereby considered in the task execution time. As a result, scheduling algorithms do not consider the time elapsed during data movement operations. In a distributed filesystem, data storage is modeled as separate local data storages. In this context, data movement operations may have increased communication costs. Data-aware scheduling algorithms may have an opportunity to optimize data-locality, and therefore the overall application runtime. A replica catalog is used to keep track of the data locations.

Dynamic Scheduling**:** In static scheduling, jobs are assigned to an execution site at the workflow planning stage, preventing any further modifications to the schedule. In contrast, task assignment in dynamic scheduling is performed at runtime, i.e. whenever an execution node is available. We extended CloudSim to support dynamic scheduling algorithms to propose new options for researchers to evaluate dynamic methods.

Task Clustering**:** The poor performance of fine-grained tasks within a workflow is a common problem in distributed platforms, where the scheduling overheads and queuing times are high. As mentioned above, task clustering is a runtime optimization technique that merges multiple short tasks into a coarse-grained job to reduce these overheads [24]. WorkflowSim provides task granularity control through the tuning of the granularity size parameter specified by the user.

In addition to the aforementioned functionalities, WorkflowSim has incorporated models of overheads, failures, and energy consumption:

Overhead Modeling**:** The execution of scientific applications in distributed environments, are subject to significant system overheads that may adversely affect the application performance [22]. In our simulator, we represent these overheads as follows: *Workflow Engine Delay*, measures the time between the task completion and the release of the subsequent task; *Queue Delay*, defines the time between the job submission to a local queue and the time the local scheduler sees the job running; *Data Transfer Delay*, measures the data transfer time between execution nodes; and *Clustering Delay*, measures the elapsed time to extract tasks from clustered jobs.

Failure Modeling**:** WorkflowSim provides support for the simulation of task and job failures. Task failures are often caused by task execution errors such as unavailability of input data or application execution errors, while job failures arise during the preparation of a job. A job failure means that all tasks within the job also fail. Failure rates follow a probability distribution (e.g. Weibull, Uniform, Normal, or Gamma) specified by the user.

Energy Consumption Modeling**:** The need to manage energy consumption of compute, storage, and network systems has received attention in the last few years. In this context, we are adding energy consumption models [25] to WorkflowSim so that we can analyze the energy consumption of scientific workflows.

C. Use cases

We have extensively used WorkflowSim for our research in a number of areas. Below we summarize some research results obtained with WorkflowSim:

Balanced Task Clustering**:** Recently, we have used WorkflowSim to investigate the dependency and runtime balance problems when performing task clustering in scientific workflows [24], [26]. We extended the task clustering feature with imbalance metrics in order to evaluate the impact of these balance problems in the performance of workflow executions. We then proposed balanced task clustering algorithms to address these issues and thereby improve the overall performance of scientific workflows. This study used synthetic workflows generated by the Workflow Generator toolkit to perform the experiments.

Fault-Tolerant Task Clustering**:** Many existing clustering strategies ignore or underestimate the impact of the occurrence of failures on system behavior, despite the increasing impact of failures in large-scale distributed and high-performance systems. We have used WorkflowSim to evaluate and propose fault-tolerant task clustering algorithms that dynamically adjusts the task clustering strategy according to observed failures [27]. At runtime, one strategy estimates the failure distribution among all the resources, and dynamically re-clusters failed jobs to reduce the job failure probability. This study relied on the generation of transient failures and clustering techniques provided by the simulator framework. In addition, synthetic workflows were also used to evaluate the algorithms.

Energy-Efficiency**:** We have used WorkflowSim to simulate a controlled distributed environment for profiling and analyzing energy-efficiency in data-intensive scientific workflows [25]. The goal of this study was to develop an energy consumption model to address real large-scale infrastructure conditions (e.g. heterogeneity, resource unavailability, external loads), and a multi-objective optimization approach to explore tradeoffs among makespan, energy consumption, and reliability for multi-objective workflow scheduling. This study also used synthetic workflows to validate the assumptions and a post-scheduling task management strategy.

Since 2012, researchers outside our group have also used and extended WorkflowSim:

***Cloud Broker*:** Jrad et.al. [4] have developed a broker-based framework for running workflows in a multi-Cloud environment to exploit the economic benefits of clouds. They extended WorkflowSim with a Cloud Service Broker to allow an automatic selection of the target clouds with respect to user service level agreements. A Replica Catalog was used to keep a list of data replicas by mapping input/output filenames to their current site locations. Data transfers were initiated by tasks during their execution at the respective data-centers. The Replica Catalog was managed by the Data Manager.

***Scheduling Performance*:** WorkflowSim has also been used to evaluate the performance of scheduling algorithms under different scenarios. Kumari et. al. [28] used the framework to evaluate the cost and makespan of running scientific workflows on clouds. Prathibha et. al. [29] evaluated the cost benefit of using task clustering in WorkflowSim. PowerWorkflowSim [5] extends WorkflowSim with an energy simulation API to develop scheduling algorithms for energy savings in computational clouds.

VI. Related Work

Computing simulations have been widely used to derive new methods, conduct comparative studies, and understand and improve the behavior of workflow management systems. Typically, these studies rely on historical execution data to validate assumptions, to model computational activity, and to evaluate methods under experimental conditions. Available workload archives [30]–[32] mainly capture information about task executions and resource utilization, but lack fine-grained information about scientific workflows and their executions, such as dependencies among tasks, and artifacts introduced by application-level scheduling. Efforts in collecting and sharing workflow experiments [33]–[35] enable the reuse and repeatability of workflow executions by the scientific community. However, the validation of a novel method, often requires tuning of the execution parameters and workflow patterns.

Synthetic workflows enable scientists to evaluate their methods under a number of different configurations. Several studies have used synthetic workflows to explore different workflow structures varying the number of jobs, or jobs dependencies [36], [37]. However, the workflows are often randomly generated (i.e. not realistic). SDAG [38] is a toolkit that generates synthetic workflows that are reasonably close/far from the real workflow. The goal is to generate abstract workflows that can be converted to executable workflows to run on WMS. The toolkit provides several tunable parameters such as the number of jobs, the level of parallelism, the maximum job depth dependency-level, and the input data size. However, some of the generated workflows may not be realistic, since the toolkit allows arbitrary changes to the workflow structure. In contrast, our generator toolkit enforces that the generated synthetic workflows resemble those used by real world scientific applications.

Several workflow simulator frameworks [2], [39] have been developed to address a single characteristic of the WMS such as workflow scheduling. CloudSim [3] is a general-purpose framework for modeling and simulating cloud computing infrastructures and services, however it only supports the execution of single task workloads. Our framework extends CloudSim to support workflow execution and it introduces the concept of task/job execution where different clustering strategies can be employed. DynamicCloudSim [40] also extends CloudSim to support workflow executions, but the framework is focused on the analysis of dynamic changes at runtime, as well as on the failures occurring during execution (due to the sharing of common resources with other VMs and users).

VII. Conclusions

In this work, we described a collection of tools and data that have enabled research in new methods and systems for the management of scientific workflows. These resources include: execution traces of real workflow applications from which workflow and system characteristics such as resource usage and failure profiles can be extracted; a synthetic workflow generator that can produce realistic synthetic workflows based on profiles extracted from execution traces; and a simulator framework that can simulate the execution of synthetic workflows on realistic distributed infrastructures. Some of the research results have been implemented in the Pegasus WMS [7]. For instance, the balanced task clustering algorithms have been incorporated to address cases when the task runtime variance within the workflow is high and the level-based clustering fails.

Although the collection of tools and data proposed in this work are linked to Pegasus WMS, the research process used to integrate the resources can be extended to any WMS. In addition, these tools are not limited to the Pegasus community, since DAGs are used by many other WMS.

In the future we plan to extend the Workflow Generator toolkit to support resource characteristic annotations and task memory usage. We also plan to collect and publish failure traces from a large number of workflow executions.

Acknowledgements

This research was supported by the National Science Foundation under the SI^2–SSI program, award number 1148515, and by the DOE under the contract number ER26110, “dV/dt–Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science”. Traces are collected from experiments that were run on FutureGrid that is supported by NSF FutureGrid 0910812.

References

[1] I. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science*. Springer, 2007.

[2] A. Hirales-Carbajal, A. Tchernykh, T. Roblitz, and R. Yahyapour, “A grid simulation framework to study advance scheduling strategies for complex workflow applications,” in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–8.

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, 2011.

[4] F. Jrad, J. Tao, and A. Streit, "A broker-based framework for multi-cloud workflows," in *2013 international workshop on Multi-cloud applications and federated clouds*, 2013, pp. 61–68.

[5] E. Watanabe, P. P. V. Campos, K. Braghetto, and D. Batista, "Algoritmos para economia de energia no escalonamento de workflows em nuvens computacionais," in *32nd Brazilian Symposium on Computer Networks and Distributed Systems*, Florianopolis, Brazil, 2014.

[6] 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 workflows onto distributed systems," *Sci. Program.*, vol. 13, no. 3, pp. 219–237, 2005.

[7] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *submitted to Future Generation Computer Systems*, 2014.

[8] "Workflow gallery," http://pegasus.isi.edu/workflow_gallery.

[9] "Workflow generator," http://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

[10] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th International Conference on E-Science*, ser. eScience, 2012, pp. 1–8. [Online]. Available: https://github.com/WorkflowSim

[11] M. McLennan, S. Clark, F. McKenna, E. Deelman, M. Rynge, K. Vahi, D. Kearney, and C. Song, "Bringing scientific workflow to the masses via pegasus and hubzero," in *IWSG*, 2013.

[12] J. S. Vockler, G. Mehta, Y. Zhao, E. Deelman, and M. Wilde, "Kickstarting remote applications," in *2nd International Workshop on Grid Computing Environments*, 2006.

[13] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M. Su, "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," in *SPIE Conference on Astronomical Telescopes and Instrumentation*, vol. 5493, 2004, pp. 221–232.

[14] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *3rd Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2008.

[15] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013.

[16] R. Ferreira da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, and M. Livny, "Toward fine-grained online task characteristics estimation in scientific workflows," in *8th Workshop on Workflows in Support of Large-Scale Science*, ser. WORKS '13, 2013, pp. 58–67.

[17] "Laser Interferometer Gravitational Wave Observatory (LIGO)," http://www.ligo.caltech.edu.

[18] 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 and Applied Geophysics*, vol. 168, no. 3-4, pp. 367–381, 2011.

[19] "USC Epigenome Center," http://epigenome.usc.edu.

[20] "SIPHT," http://pegasus.isi.edu/applications/sipht.

[21] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[22] W. Chen and E. Deelman, "Workflow overhead analysis and optimizations," in *The 6th Workshop on Workflows in Support of Large-Scale Science*, Nov. 2011.

[23] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.

[24] W. Chen, R. Ferreira da Silva, E. Deelman, and R. Sakellariou, "Balanced task clustering in scientific workflows," in *2013 IEEE 9th International Conference on eScience (eScience)*, 2013, pp. 188–195.

[25] R. Ferreira da Silva, T. Fahringer, J. J. Durillo, and E. Deelman, "A unified approach for modeling and optimization of energy, makespan and reliability for scientific workflows on large-scale computing infrastructures," in *Workshop on Modeling & Simulation of Systems and Applications*, 2014, p. to appear.

[26] W. Chen, E. Deelman, and R. Sakellariou, "Imbalance optimization in scientific workflows," in *27th International ACM Conference on International Conference on Supercomputing*, ser. ICS '13, 2013, pp. 461–462.

[27] W. Chen and E. Deelman, "Fault tolerant clustering in scientific workflows," in *IEEE 8th World Congress on Services*, 2012, pp. 9–16.

[28] P. Kumari and N. Kumari, "Performance evaluation of cost-time based workflow scheduling algorithms in cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 9, 2013.

[29] S. Prathibha, B. Latha, and G. Sumathi, "Monitoring the performance analysis of executing workflow applications with different resource types in a cloud environment," in *1st International Symposium on Big Data and Cloud Computing Challenges*, 2014, pp. 27–28.

[30] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. Epema, "The grid workloads archive," *Future Generation Computer Systems*, vol. 24, no. 7, pp. 672–686, 2008.

[31] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 398–407.

[32] R. Ferreira da Silva and T. Glatard, "A science-gateway workload archive to study pilot jobs, user activity, bag of tasks, task sub-steps, and workflow executions," in *Euro-Par 2012: Parallel Processing Workshops*, 2013, vol. 7640, pp. 79–88.

[33] J. Stoyanovich, B. Taskar, and S. Davidson, "Exploring repositories of scientific workflows," in *1st International Workshop on Workflow Approaches to New Data-centric Science*, 2010, p. 7.

[34] V. Korkhov, D. Krefting, T. Kukla, G. Z. Terstyanszky, M. W. Caan, and S. D. Olabarriaga, "Exploring workflow interoperability for neuroimage analysis on the shiwa platform," *Journal of grid computing*, vol. 11, no. 3, pp. 505–522, 2013.

[35] S. Holl and et al., "On specifying and sharing scientific workflow optimization results using research objects," in *8th Workshop on Workflows in Support of Large-Scale Science*, 2013, pp. 28–37.

[36] M. Rahman, S. Venugopal, and R. Buyya, "A dynamic critical path algorithm for scheduling scientific workflow applications on global grids," in *IEEE International Conference on e-Science and Grid Computing*, 2007, pp. 35–42.

[37] M. Naseri and S. Ludwig, "Evaluating workflow trust using hidden markov modeling and provenance data," in *Data Provenance and Data Management in eScience*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2013, vol. 426, pp. 35–58.

[38] M. Amer and R. Lucas, "Evaluating workflow tools with sdag," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, 2012, pp. 54–63.

[39] S. Camarasu-Pop, T. Glatard, and H. Benoit-Cattin, "Simulating application workflows and services deployed on the european grid infrastructure," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, May 2013, pp. 18–25.

[40] M. Bux and U. Leser, "Dynamiccloudsim: Simulating heterogeneity in computational clouds," in *2Nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, ser. SWEET '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:12.