A Characterization of Workflow Management Systems for Extreme-Scale Applications

Rafael Ferreira da Silva[a,*], Rosa Filgueira[b,c], Ilia Pietri[d], Ming Jiang[e], Rizos Sakellariou[f], Ewa Deelman[a]

[a] *Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA*
[b] *British Geological Survey, Lyell Centre, Edinburgh, UK*
[c] *School of Informatics, University of Edinburgh, Edinburgh, UK*
[d] *Department of Informatics and Telecommunications, University of Athens, Athens, Greece*
[e] *Lawrence Livermore National Laboratory, Livermore, CA, USA*
[f] *School of Computer Science, University of Manchester, Manchester, UK*

Abstract

Automation of the execution of computational tasks is at the heart of improving scientific productivity. Over the last years, scientific workflows have been established as an important abstraction that captures data processing and computation of large and complex scientific applications. By allowing scientists to model and express entire data processing steps and their dependencies, workflow management systems relieve scientists from the details of an application and manage its execution on a computational infrastructure. As the resource requirements of today's computational and data science applications that process vast amounts of data keep increasing, there is a compelling case for a new generation of advances in high-performance computing, commonly termed as *extreme-scale computing*, which will bring forth multiple challenges for the design of workflow applications and management systems. This paper presents a novel characterization of workflow management systems using features commonly associated with extreme-scale computing applications. We classify 15 popular workflow management systems in terms of workflow execution models, heterogeneous computing environments, and data access methods. The paper also surveys workflow applications and identifies gaps for future research on the road to extreme-scale workflows and management systems.

Keywords: Scientific workflows, workflow management systems, extreme-scale computing, *in situ* processing

1. Introduction

Scientific workflows are an important abstraction for the composition of complex applications in a broad range of domains, such as astronomy, bioinformatics, climate science, and others [1]. Workflows provide automation that increases the productivity of scientists when conducting computation-based studies. Automation enables adaptation to the changing application needs and resource (compute, data, network) behavior. As workflows have been adopted by a number of scientific communities, they are becoming more complex and need more sophisticated workflow management capabilities. A workflow now can analyze terabyte-scale data sets, be composed of a million individual tasks, and can process data streams, files, and data placed in object stores. The computations can be single core workloads, loosely coupled computations (like MapReduce), or tightly coupled (like MPI-based parallel programs) all within a single workflow, and can run in dispersed cyberinfrastructures [1, 2].

In recent years, numerous *workflow management systems* (WMSs) have been developed to manage the execution of diverse workflows on heterogeneous computing resources [3, 4, 5, 6, 7, 8, 9]. As user communities adopt and evolve WMSs to fit their own needs, many of the features and capabilities that were once common to most WMSs have become too distinct to share across systems. For example, Taverna [8] and Galaxy [9] support advanced graphical user interfaces for workflow composition, making them suitable for bioinformatics researchers with little programming experience. Other systems, such as DAGMan [10] and Pegasus [3] offer scalability, robustness, and planning for heterogeneous high-throughput computation execution. For a new user, choosing the *right* WMS can be problematic simply because there are so many different WMSs and the selection criteria may not be obvious.

To address this problem, several recent surveys [11, 12, 13, 14, 15, 16, 17, 18] have been compiled to help users compare and contrast different WMSs based on certain key properties and capabilities of WMSs. These surveys focused mostly on the characterization of the following properties: support for conditional structures (e.g., if and switch statements, while loops, etc.) [12], workflow composition (e.g., graphical user interface, command line, or web portals) [13, 14, 15, 16], workflow design (DAG or Non-DAG) [16, 17], types of parallelism (e.g., task, data, pipeline, or hybrid parallelism) [14, 16, 17], computational infrastructure (e.g., cluster, grid, and clouds) [12, 14, 15, 16], workflow scheduling (e.g., status, job queue, adaptive) [14, 15, 16, 17, 18], workflow QoS constraints (e.g., time, cost, reliability, security, etc.) [17], and fault-tolerance and workflow optimizations (e.g., task-level, workflow-level,

*Corresponding address: USC Information Sciences Institute, 4676 Admiralty Way Suite 1001, Marina del Rey, CA, USA, 90292

Email addresses: rafsilva@isi.edu (Rafael Ferreira da Silva), rosa@bgs.ac.uk (Rosa Filgueira), ipietri@di.uoa.gr (Ilia Pietri), jiang4@llnl.gov (Ming Jiang), rizos@manchester.ac.uk (Rizos Sakellariou), deelman@isi.edu (Ewa Deelman)

etc.) [15, 16, 17].

Unfortunately, the above characterization properties do not sufficiently address the following question that is on the mind of many computational scientists: "Are WMSs ready to support extreme-scale applications?" We define *extreme-scale applications* as scientific applications that will utilize *extreme-scale computing* to solve vastly more accurate predictive models than before and enable the analysis of massive quantities of data [19, 20]. It is expected that the requirements of such applications will exceed the capabilities of current leading-edge high-performance computing (HPC) systems. Examples of extreme-scale applications include: first-principles understanding of the properties of fission and fusion reactions; adaptation to regional climate changes such as sea-level rise, drought and flooding, and severe weather patterns; and innovative designs for cost-effective renewable energy resources such as batteries, catalysts, and biofuels [19].

Extreme-scale computing that includes planned U.S. Department of Energy exascale systems [21] will bring forth multiple challenges for the design of workflow applications and management systems. The next-generation of HPC architectures is shifting away from traditional homogeneous systems to much more heterogeneous ones. Due to the severe energy constraints, data movement will be constrained, both internode and on/off the system, and users will be required to manage deep memory hierarchies and multi-stage storage systems [22, 20]. There will be an increased reliance on *in situ* data management, analysis and visualization, occurring in parallel with the simulation [23, 24]. These *in situ* processing steps need to be captured to provide context and increase reproducibility.

In addition, as the scientific community prepares for extreme-scale computing, big data analytics is becoming an essential part of the scientific process for insights and discoveries [25]. As big data applications became mainstream in recent years, new systems have been developed to handle the data processing. These include Hadoop [26], a MapReduce-based system for parallel data processing, Apache Spark [27], a system for concurrent processing of heterogeneous data streams, and Apache Storm [28] for real-time streaming data processing. Integrating big data analytics with HPC simulations is a major challenge that requires new workflow management capabilities at the extreme-scale.

In this paper we present a novel characterization of WMSs focused specifically on extreme-scale workflows, using the following properties: (1) workflow execution models, (2) heterogeneous computing environments, and (3) data access methods. Associated with each property is a set of features that can be used to classify a WMS. To evaluate these properties, we select 15 state-of-the-art WMSs based on their broad and active usage in the scientific community, as well as the fact that they have been part of previous surveys. Through a detailed analysis using available publications and other documents, such as project webpages and code manuals, we derive the classification of these WMSs using our features for extreme-scale applications. Our primary contribution in this work is the distillation of all the available information into an easy-to-use lookup table that contains a feature checklist for each WMS. This table represents a snapshot of the state-of-the-art, and we envision it to evolve and grow based on future research in WMSs.

The remainder of this paper is structured as follows. Section 2 presents a background overview of WMSs in general and previous work on characterizing workflows and WMSs. Section 3 describes the two types of extreme-scale workflows that motivate this work. Section 4 presents the three properties for characterizing WMSs for extreme-scale applications, along with their associated features. Section 5 classifies 15 popular WMSs based on these features, and Section 6 describes their current usage in the scientific community. Section 7 identifies gaps and directions for future research on the road to extreme-scale workflows. Finally, Section 8 concludes the paper.

2. Background and Related Work

2.1. Scientific Workflows

The term *workflow* refers to the automation of a process, during which data is processed by different tasks. A WMS aids in the automation of these processes, by managing data and the execution of the application on a computational infrastructure. *Scientific workflows* allow scientists to easily model and express all the data processing tasks and their dependencies, typically as a directed acyclic graph (DAG), whose nodes represent workflow tasks that are linked via dataflow edges, thus prescribing serial or parallel execution of nodes. In general, there are four key properties of scientific workflows, which are handled differently by each WMS:

- *Design:* Most modern WMS provide a graphical user interface to make it easier to create and compose workflows. Alternatively, command line interfaces have the ability to capture more complex structures and scale better to larger problems, but they require programming skills.
- *Execution and Monitoring:* There is a plethora of computing infrastructures, where different optimization methods could be applied to reduce the workflow turnaround time [29]. Robust workflow executions require effective monitoring of a variety of resources including CPU cores, memory, disk, and network traffic [30].
- *Reusability:* WMSs have to make it easier for the workflow designer to reuse previously developed workflows. Many workflows provide mechanisms for tracing provenance and methodologies that foster reproducible science [31].
- *Collaboration:* Due to the collaborative nature of scientific research projects, there is a need for sharing and collaboration mechanisms to foster collaborative efforts among workflow scientists. Some projects, such as myExperiment [32] and Wf4Ever [33], have devoted substantial efforts toward this approach.

In this paper, we are particularly interested in the *Execution and Monitoring* property. We aim to identify, describe, and analyze the different workflow execution models provided by current WMSs to support extreme-scale workflows.

2.2. Characterizations of Workflows and WMSs

Workflow application profiling and classification can be useful to improve WMSs. It can be used to provide insight on the application needs and identify the different requirements in order to optimize the deployment of different workflows. Several attempts have been made to characterize scientific applications and workflows [34, 35, 36, 37, 38, 39, 40, 41]. The work in [37] describes the characterization of a range of scientific workflows to describe both their composition and the data and computational requirements of the tasks. Approaches that provide more fine-grained profiling data can be found in [38, 39, 40]. Tools such as Kickstart [30] monitor workflow execution and collect provenance information about the runtime behavior. Paratrac [38], a profiler for data-intensive workflows, enables the monitoring of low level I/O profiles of the workflow tasks. The approach described in [39, 40] enables more detailed profiling of the workflow execution, providing information about the CPU, memory and I/O utilization of workflow tasks. Finally, in [41], grid workloads are analyzed in terms of user population, overall system usage, general application characteristics, such as CPU and memory utilization, and characteristics of grid-specific application types, such as workload structure.

Workflow management systems have been characterized as well [12, 13, 14, 15, 16, 17, 18]. Most of the studies attempt to provide a comparison between existing tools. For instance, the work in [12] surveys the techniques used in automatic management of applications on grid systems with respect to application composition, scheduling, coordination, and execution monitoring deployed by these systems. Existing WMSs on grids are classified according to these key features. The survey in [13] also studies the capabilities of commonly used WMSs; however, the focus is on the support for conditional structures, such as `if` statements and `while` loops. Classification of WMSs based on their parallelization and scheduling techniques can be found in [14, 15]. The work in [14] covers existing solutions on parallel computing infrastructures. The survey in [15] focuses on data-intensive scientific workflows and the analysis is based on four functional levels: (1) workflow structure, (2) user interface, (3) type of data parallelism, and (4) scheduling mechanism used. In [17], a classification of grid workflow systems for scientific applications is presented and the architectural styles of a subset of existing tools is compared in terms of workflow design and information retrieval, scheduling, fault management, and data movement. In [18], an overview of the workflow technologies in both the business and the scientific domain is provided; however, such taxonomies may potentially suffer from the continuously evolving workflow solutions and computing paradigms. The work in [16] tries to cover this limitation by identifying the subset of features of WMSs that may be important for the workflow community as a whole.

In contrast to related work, this paper presents a novel characterization of WMSs using the following properties: (1) workflow execution models, (2) heterogeneous computing environments, and (3) data access methods. These properties and their associated features are motivated by extreme-scale workflow applications that are being actively developed.

3. Extreme-Scale Workflows

As mentioned earlier, extreme-scale computing is on the horizon and will bring forth multiple challenges for the design of workflow applications and management systems. To illustrate these challenges, we present two types of extreme-scale workflows that are being actively developed in the computational science community to meet the needs.

3.1. In Situ Workflows

Visualization and data analysis (VDA) will play a crucial role in gaining insight from extreme-scale simulations. Given the sheer volume of data that will be generated by such simulations, application scientists will be overwhelmed with raw data unless there is a radical improvement in the frequency and fidelity with which simulation data can be analyzed. However, it is becoming clear that data movement through the system has significant power and performance costs [20], and at extreme-scale, it will not be feasible to move the entire volume of raw data to global storage.

The current VDA approach of storing the full data set for post-processing becomes increasingly prohibitive due to the mismatch in bandwidth between compute and storage [24]. An effective approach to address this issue at extreme-scale is *in situ* VDA, which implies performing VDA processing at simulation time to minimize data storage and transfer cost. One approach is to integrate simulation and VDA routines so that they can operate on in-memory simulation data. Examples of this approach can be found in VisIt Libsim [42], Paraview coprocessing [43], and other visualization [44] and analysis [45] systems.

Another approach is to perform *in situ* VDA processing using dedicated processor cores [46] or dedicated compute nodes for both shared memory [47] and distributed memory [48] environments. Approaches that decouple I/O from the simulation by staging data to a secondary memory location are also known as *in transit* VDA. They primarily focus on fast and asynchronous data movement off simulation nodes to lessen the impact of expensive I/O operations. Examples of this approach include ADIOS [49], DataSpaces [50], and Glean [51].

The shift to *in situ* VDA that is more tightly coupled with simulations represents a new kind of workflow for scientists, creating both challenges and opportunities [52]. Existing workflow engines coordinate tasks within the HPC environment through the scheduler and remote connections to the HPC machine. They do little to help orchestrate tasks that must run concurrently and share memory-based data objects. Research is needed to design *in situ* workflow management methods to support VDA in residence with simulations at scale, as well as providing for the capture of sufficient data provenance to support subsequent re-analysis and/or validation of results. There is also a need to explore tradeoffs between data re-computation and data retrieval, taking into account time to solution and energy consumption [20].

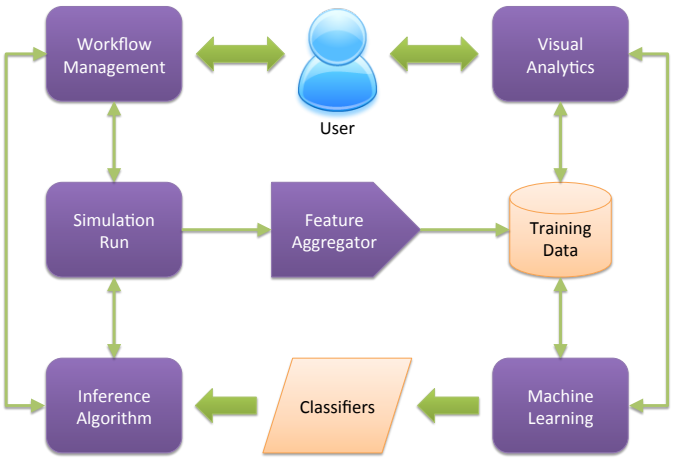

Figure 1: A high-level overview of an infrastructure for integrating machine learning into HPC simulations. At the top level is the user interface, where the Workflow Management interacts with the Simulation Run and the Visual Analytics interacts with the Machine Learning. At the middle level is the data collection, where the key component is the Feature Aggregator, which aggregates the massive simulation data into learning features for training data. At the bottom level is the predictive analytics, where the Machine Learning generates Classifiers that are then used for Inference Algorithm.

3.2. Hybrid Workflows

In general, HPC simulations and big data analytics are on a converging path. As the scientific community is preparing for the next-generation extreme-scale computing, big data analytics is becoming an essential part of the scientific process for insights and discoveries [25]. Effectively integrating big data analytics into HPC simulations still remains a daunting challenge, and it requires new workflow technologies to reduce user burden and improve efficiency.

Integrating big data analytics into HPC simulations can produce hybrid workflows that incorporate: (1) *in situ* workflows, which optimize data flows for coupled applications within a single HPC system; (2) big data workflows, which optimize data flows for machine learning and analytics; and (3) distributed workflows, which optimize computations and data transfers across the wide area. Although all these types of workflows address the execution of a set of tasks, there are several key differences between them. For example, distributed workflows typically consist of a set of serial, loosely-coupled tasks, while *in situ* workflows are typically concurrent, or combine both sequential and concurrent steps in a more tightly-coupled fashion, and big data workflows involve efficient parallel processing of data using technologies like MapReduce [53].

New workflow technologies are needed to bridge across the heterogeneous computing infrastructures used for these hybrid workflows. They need to enable applications to span HPC systems, grids, and cloud platforms by interfacing with different schedulers and storage systems across wide area networks. Each phase of the workflow can then be executed on the most appropriate infrastructure for the task, and the workflow system can plan, schedule, provision, deploy, and optimize the entire analysis for performance, cost, and other metrics. For example, simulations can run on an HPC system with *in situ* analysis and visualization, while post-processing of reduced simulation data can be performed in the cloud using Hadoop [26] and Spark [27], and final results can be uploaded to a community data repository.

Current research in exploiting big data analytics, such as machine learning, to semi-automate the development process for HPC simulation workflows has demonstrated tremendous promise [54, 55]. Integrating big data analytics into HPC simulations will require the creation of hybrid workflows that involve both distributed and *in situ* computations. For a new problem, deriving the initial predictive models will involve running machine learning on many small simulation outputs via big data and distributed workflows. For the predictive analytics to be effective, it must run along with the simulation to dynamically adjust the workflow accordingly using *in situ* workflows. Figure 1 provides an overview of the proposed infrastructure for integrating machine learning into HPC simulations that require a human in the loop for the WMS.

4. Characterization of Workflow Management Systems

As described in Section 2, there are a number of different ways to characterize WMSs, from the interfaces they present to the users, down to the types of provenance records they provide. With the growing complexity of extreme-scale applications and the ever-increasing heterogeneity of computing capabilities, the list of WMS features for computational science is getting longer and requires a fresh perspective to help users choose. In this paper, we focus on the features of WMSs that are most relevant to managing emerging extreme-scale workflows. We organize these features into three properties that, to the best of our knowledge, have not been exhaustively studied in previous surveys on WMSs. In this section, we present these properties, along with their associated features.

4.1. Workflow Execution Model

Let us consider the execution of two workflow tasks managed by a central WMS. The tasks can be of any type: single core codes, code fragments, parallel applications, MapReduce computations, etc. The data flowing through the tasks can be shared via files, memory, object store, databases, etc.

The possible models of interactions between the tasks are illustrated in Figure 2. We distinguish between two groups: (1) the *acyclic* group corresponds to models that are available in state-of-the-art WMSs; and (2) the *cyclic* group refers to the existence of cycles in the model, either in the interaction of the two tasks or, more importantly, in the interaction with the WMS. The cyclic group includes execution models that will be important on the road to extreme-scale, but are not often found in today's WMSs. We distinguish between the following execution models.

- *Sequential*: The WMS starts `T1`, which processes its data, completes, and then the WMS starts `T2`. This is a typical workflow model where a simulation phase is followed by a post-processing phase.

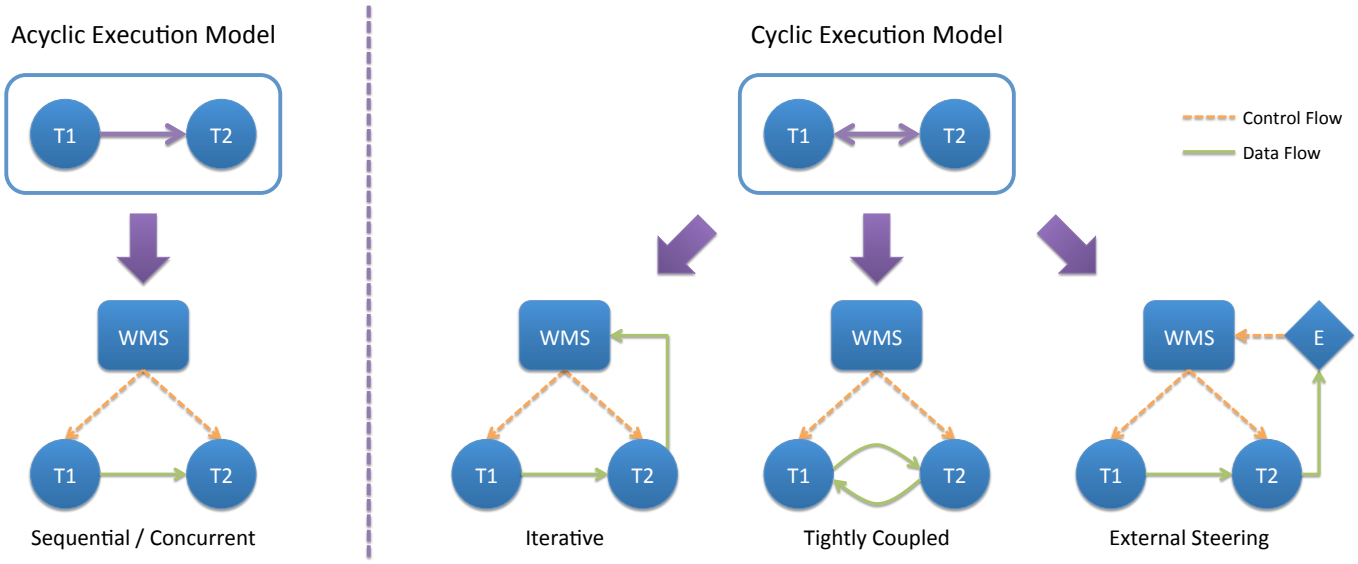

Figure 2: Five different workflow execution models, divided into two groups: *acyclic* is what is mostly provided by today's WMSs; and *cyclic* execution models will be needed to support extreme-scale workflows.

- *Concurrent:* `T1` and `T2` execute at the same time, `T1` produces data consumed by `T2`. This model supports workflows that process streaming data through multiple stages, for example applying different feature detection algorithms. The term *concurrent*, which is commonly used in the literature, is preferred instead of *pipeline* in order to describe the generalized form of parallelism where two tasks are executed during overlapping time periods.
- *Iterative*: The WMS starts `T1`, which processes its data, completes, and then the WMS starts `T2`. The results of `T2` decide whether there is another iteration of the sequence. This model supports Uncertainty Quantification (UQ) workflows, which explore a parameter space in an iterative fashion.
- *Tightly coupled:* `T1` sends partial results to `T2`, and vice versa. This model supports workflows where two simulations are tightly coupled and progress together in time, for example, a cross correlation analysis or a multi-physics simulation that couples together radiation transport with hydrodynamics.
- *External steering:* The underlying semantics is the same as with the Concurrent case, but the results of `T2` are inspected by a user or system, who may decide to affect the execution of `T1` or take some other action. In this case, `T1` could be a simulation that is sending data to the `T2` data analytics task. Upon viewing the results from the analysis, the user may decide to change the behavior of `T1`, or more broadly, change something else in the workflow, for example, how the data is collected at an instrument.

4.2. Heterogeneous Computing Environments

At extreme-scale, the ability to manage a workflow within a particular computational resource (HPC system) as well as across systems is critical. Often HPC workflows need the input data to be cleaned or calibrated, or retrieved from an instrument, or may involve post-processing, for example running data analytics, on another resource. As a result, we need the WMS to be able to orchestrate a workflow across different resources.

When going to extreme-scales, the WMS, or some portion of it, may need to be co-located with the computations it is managing within an HPC system. It may need to coordinate the scheduling of simulation and visualization for tightly coupled workflows. It may also need to decide how to do the data management and its flow through the memory layers, etc. This type of coordination cannot be done externally to the system (via the HPC scheduler for example), but rather within the system. The features associated with this property are:

- *Co-location with the execution system:* A workflow can be "compiled" to run in one environment, but cannot have two different environments within the same workflow.
- *External location to the execution system:* A workflow can be "compiled" and managed to run across different execution environments.
- *In situ:* The WMS is orchestrating the execution of tasks from within the HPC system.

4.3. Data Access Methods

Managing computations in heterogeneous environments also implies the necessity of managing access to data via a variety of mechanisms provided by the infrastructure. Tasks may need access to data files that are stored in memory, on a disk local to a previous task, an object store, or some other external storage. We assume that these methods will place data in a way that is needed by a task – for example, if a task assumes local access to a file that is in an object store, the WMS will retrieve that file and place it in a local disk or shared filesystem accessible to the task. The features associated with this property are:

- *Memory:* Data resides in a computer's RAM, and can be accessed by a single or multiple processes (on shared memory systems).
- *Messages:* Data is passed between processes through a communication channel. It is frequently used in shared memory systems, data streams, or tightly coupled applications (e.g., MPI).
- *Local disk:* Data resides in a physical disk attached to the computing resource and is often accessible via POSIX interfaces.
- *Shared file system:* Data resides in a remote server, but can be accessed locally using a network protocol (e.g., NFS).
- *Object store:* Data is represented as objects as opposed to, for example, file systems, which manage data as a file hierarchy (e.g., Amazon S3).
- *Other remote storage:* Data resides in external distributed data management systems such as iRODS, which may require access to further data transfer protocols (e.g., GridFTP, XRootD).

5. Classification of Workflow Management Systems

In the past two decades, a number of WMSs have been developed to automate the computational and data management tasks, the provisioning of the needed resources, as well as different execution models required by today's scientific applications. Based on the characterization of WMS presented in the previous section, Table 1 summarizes the features supported by the most popular WMSs described below in alphabetical order. We chose these WMSs for classification because they are widely and actively used by the scientific community, and they provide state-of-the-art capabilities. The selected WMSs are frequently referenced and studied in previous surveys, which characterized conditional structures [12], workflow composition [13, 14, 15, 16], workflow design [17, 16], types of parallelism [14, 17, 16], computational infrastructure [12, 14, 15, 16], workflow scheduling [14, 15, 17, 18, 16], workflow QoS constraints [17], and fault-tolerance and workflow optimizations [15, 17, 16].

- *ADIOS* [56] (Adaptable IO System) is an open-source middleware that aims to give a level of adaptability such that the scientist can change how the I/O in their code works simply by changing a single entry in a XML file and restarting the code. ADIOS creates abstract I/O for workflows, with focus on *in situ* analytics—run workflow components (analysis, visualization, data management) with computational simulation.
- *Apache Airavata* [57] is an open source, open community *software framework* used to compose, manage, execute, and monitor distributed applications and workflows on computational resources ranging from local resources to computational grids and clouds. Airavata builds on general concepts of service-oriented computing, distributed messaging, and workflow composition and orchestration through the Airavata Xbaya workflow system.
- *Askalon* [7] provides a suite of middleware services that support the execution of scientific workflows on the grid. Workflow specifications are converted into executable forms, mapped to grid resources, and executed. Workflow executions consist of coordinating control flow constructs and resolving data flow dependencies specified by the application developer. Askalon is mainly used by the computer science community to support workflow research.
- *Bobolang* [58] is a relatively new workflow system based on data streaming. It provides linguistic forms based on C++ and focuses on automatic parallelization. It also supports multiple inputs and outputs, meaning that a single node can have as many inputs or outputs as a user requires. Currently, it does not support automatic mapping to different Distributed Computing Infrastructures (DCIs).
- *dispel4py* [4] implements many of the original Dispel concepts, but presents them as Python constructs. It describes abstract workflows for data-intensive applications, which are later automatically translated to the selected enactment platforms (e.g., Apache Storm, MPI, Multiprocessing, etc.) at run time. dispel4py has been widely used by the geoscience community.
- *Fireworks* [6] is a workflow software for running high-throughput calculation workflows at supercomputing centers. A key functionality of the WMS is the ability to define dynamic workflows, i.e., the workflow graph can be modified at runtime.
- *Galaxy* [9] aims to bring computational data analysis capabilities to non-expert users in the biological sciences domain. The main goals of the Galaxy framework include accessibility to biological computational capabilities and reproducibility of the analysis result by tracking the information related to every step on the process.
- *Kepler* [59] is a GUI-based scientific WMS. It provides an assortment of built-in components with a major focus on statistical analysis. Kepler workflows are written in MoML (an XML format) or KAR files, which are an aggregation of files into a single JAR file. Kepler is built on top of the Ptolemy II Java library, from which it inherits the concepts of Directors and Actors. The former control the execution of the workflow, while the Actors execute actions when specified by Directors.
- *Makeflow* [5] is a command line workflow engine used to execute data-intensive scientific applications on a variety of distributed execution systems including campus clusters, clouds, and grids. The end user expresses a workflow using a syntax similar to Make. Makeflow has been used by the high energy physics community.
- *Moteur* [60] has the ability to handle complex workflows by supporting control structures. Moteur is the only system that provides native steering—it can be achieved in other systems via manual hacks. Moteur supports iterativity via control loops and beanshell (lightweight scripting for Java executed at runtime), where users can define

Workflow Properties	ADIOS	Airavata	Askalon	Bobolang	dispel4py	Fireworks	Galaxy	Kepler	Makeflow	Moteur	Nextflow	Pegasus	Swift	Taverna	Triana
Workflow Execution Models															
Sequential	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔
Concurrent	✘	✘	✘	✔	✔	✘	✘	✘	✘	✘	✔	✘	✔	✘	✘
Iterative	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔	✘	✔	✔	✘
Tightly coupled	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
External steering	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✘	✘	✘
Heterogeneous Computing Environments															
Co-location	✘	✘	✘	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	✘	✘
External location	✘	✔	✔	✘	✘	✘	✔	✔	✔	✔	✘	✔	✔	✔	✔
In situ	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
Data Access Methods															
Memory	✔	✘	✘	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	✘	✘
Messages	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Local disk	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Shared file system	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Object store	✘	✘	✘	✘	✘	✘	✔	✘	✘	✘	✘	✔	✘	✔	✘
Other remote storage	✔	✘	✘	✘	✘	✘	✔	✘	✘	✘	✘	✔	✔	✔	✘

Table 1: Characterization of state-of-the-art WMSs. The classification highlights relevant characteristics to attain *extreme-scale*.

stop conditions for the workflow. Moteur is mainly used to compute medical imaging workflows.

- *Nextflow* [61] is a domain science language (DSL) based WMS modeled around the UNIX pipe concept, that simplifies writing parallel and scalable pipelines in a portable manner. It promotes a programming approach, based on functional composition, to enable data streams processing. In Nextflow, iterativity is achieved through the evaluation of conditions defined as functions (in the applications code).
- *Pegasus* [3] supports workflow executions in distributed environments, such as campus clusters, grids, clouds, and supercomputers. Pegasus maps an application onto available resources, optimizing the execution in terms of performance and data management. Pegasus also focuses on scalability (number of workflow tasks and size of data) and reliability. Pegasus workflows have been used in a variety of science domains including astronomy, earth sciences, bioinformatics, climate modeling among others.
- *Swift* [62] is a dataflow language for scientific computing designed to enable easy composition of independent software tools and procedures into large-scale, throughput-oriented parallel workflows that can be executed in cluster, cloud, and grid environments. Concurrency can be achieved with Swift/T, the MPI version of Swift. Swift supports iterative execution through the evaluation of conditions defined as functions. The system has been used in several science domain, with emphasis on the bioinformatics community.
- *Taverna* [8] is a suite of tools that provides an easy to use environment to build, execute and share workflows of web services. It has been developed for the enactment of bioinformatics workflows. It emphasizes usability, providing a graphical user interface for workflow modeling and monitoring as well as a comprehensive collection of predefined services. In Taverna, iterativity is supported via control loops. Taverna is popular in the bioinformatics community.
- *Triana* [63] is an open source graphical problem-solving environment that allows users to assemble and run a workflow through a graphical user interface while minimizing the burden of programming. Triana is mainly used by the computer science community to support workflow research.

6. Applications of Workflow Management Systems

Given that many of the anticipated extreme-scale applications will have their roots in today's applications, it is instructive to note how WMSs are currently being used. After a thorough examination of the literature, we identified six categories of applications with notable usage of WMSs. In this section, we describe these application categories, along with specific applications from various scientific domains and the WMSs that are used to support them. We also point out the issues that may arise at extreme-scale.

6.1. Computational and Data Analysis Pipelines

Computational and data analysis pipelines normally consist of sequential and concurrent tasks that process streams of data [11]. Tasks with data dependencies may run simultaneously operating on different data sets, following the concurrent execution model. A task can start its execution as soon as the first data set required as input has been generated. Examples of

computational and data analysis pipelines include several simulations followed by data analysis and visualization. Telescope data processing [64] is an application that is normally processed as a pipeline as data arrive continuously like a stream. In telescope data processing applications, several tasks may need to be run multiple times or may contain several steps that need to be executed.

Systems like Askalon, Kepler, and VisTrails [65] integrate mechanisms to support pipeline dataflows. Communication between activities (tasks) in Askalon and Kepler is made by connecting input and output ports between them. Each task can read from its ports when a sequence of data becomes available to it. VisTrails is a framework for visualization through workflows, using an action-based provenance model to capture the evolution of dataflow pipelines. Other WMSs for pipelined parallelism include Pegasus, Swift, and Makeflow. Overall, in order to support the execution of dataflow pipelines, WMSs need to support different types of parallelism in an integrated manner to manage the concurrent execution of dependent tasks on different data sets, the concurrent processing of data sets by task replicas, and the concurrent execution of independent tasks [14]. For extreme-scale applications where data becomes an essential source of optimization, more data-driven solutions that exploit parallelism may be important.

6.2. Semi-Automated Workflows

Several exploratory workflows in domains like biology may include steps that are partially automated and depend on human inputs or actions. For example, user intervention may be required to provide tuning or convergence checks in order to adjust instrument settings and steer the analysis. In tomography workflows [66], 2D electron microscopic projection images are transformed to generate 3D volumes. During the feature tracking step in a particular workflow geared towards refining the correspondences [67], fine tuning can be performed using semi-automated methods that require user intervention.

The main requirement for WMSs to support semi-automated workflows is to provide an interface for user interaction and external steering so that the system can notify the user for the execution progress, present intermediate results, and receive user input to steer the analysis. Adjusting the workflow during the simulation based on user input is becoming paramount on the road to extreme-scale. Experiences from the business workflow community, in particularly BPEL (Business Process Execution Language) [68], which orchestrates human interactions and web services, may be applicable.

6.3. Service Integration

Several database repositories, computational programs and tools are increasingly made available on the Web as services. This allows researchers to build workflows of web services to retrieve and integrate a range of resources into a single analysis [69]. Processing may be performed remotely at service providers, without requiring the installation of tools and storing of data on local infrastructures.

Life sciences commonly adopt web services to compose workflows that perform different analyses, such as genome annotation and sequence analysis [70]. Investigating the genetic analysis of Graves' disease is an example where genome annotation workflows are used to retrieve information about genes that may be involved in the disease from biological databases [69].

Web service adoption requires the maintenance of a service catalog to register the available services, as well as data type checking and conversion to ensure compatibility between services. WMSs that support the designing and execution of workflows composed by web services include Taverna, Triana, and Galaxy. Taverna focuses on the web service level to allow researchers to conduct in silico experiments that involve the use of local and remote resources in biology [63]. Triana supports integration of multiple services and interfaces incorporating runtime type checking and data type conversion to support complex data types. Finally, the Galaxy framework allows researchers to develop a set of analysis steps in order to rerun workflows using different data. At the extreme-scale, the number of service requests may downgrade the performance of such systems, therefore scalable solutions that coordinate a large number of concurrent requests may be important.

6.4. Parameter Sweeps

Scientific simulations often require executing variations of the same workflow by varying the input parameter values in order to investigate the behavior of the model and the relation with model properties. This is typically modeled as a parameter sweep problem to explore the space of different parameter values. Executing workflow tasks in parallel may significantly speed up the execution [71]. The technique used to automate the scientific method and run simulations many times using HPC is known as data farming [72]. As data farming may involve lots of iterations, data provenance capturing for the parameter range is a key feature for the support of parameter sweep workflows, and particularly for extreme-scale.

The Spallation Neutron Source (SNS) [73] is a materials research facility that generates beams of pulsed neutrons for researchers to use. The data processing, analysis, and simulation for SNS experiments are performed by workflows managed by tools such as Pegasus. Researchers use these workflows to compute molecular dynamics and neutron scattering simulations using parameter sweeps to optimize model parameters, such as temperature and hydrogen charge parameters [74].

Data farming tools often operate over WMS. Nimrod [75] is a family of tools for exhaustive parameter sweep in workflows where a single computation is performed many times. The support for workflows is available via Nimrod/K [76], which incorporates Kepler's capabilities on building complex workflows, and Nimrod's ability to execute sweeps over distributed resources. Scalarm [77], which stands for Massively Scalable Platform for Data Farming, is another platform that supports parameter sweep applications on heterogeneous computing infrastructures, such as grids and clouds. Scalarm has been developed in order to support data farming, from the experiment phase to the simulation execution and analysis of results, allowing users to manage computational resources regardless of

the computing infrastructures used. Workflow support has been enabled in Scalarm via its integration with Pegasus [78].

6.5. Workflow Ensembles

In many scientific domains, large computational problems are frequently modeled using multiple inter-related workflows grouped into ensembles [79]. Typically, the workflows in an ensemble have similar structure, but may differ in their input data and parameters, such as the input data or the number of workflow tasks. Also, several workflows may be executed first to analyze the results and determine the next set of workflows to run. Scientific workflow ensembles are normally executed using the master/worker pattern with the master orchestrating the workflow based on some application specific logic and/or user-specified policies.

Computational engineering experiments are often modeled using workflow ensembles. Engineering problems may require reliable numerical simulations, which may fail due to the presence of uncertainty in the model data, such as constitutive equation parameters [80]. To cope with uncertainty, reliability methods such as uncertainty quantification (UQ) are used. UQ experiments typically rely on Monte Carlo simulations to compute response statistics, such as response mean and standard deviation. The simulation is performed using different input combinations, with each run comprising a different workflow.

Work Queue [81] is a master-worker framework that supports the execution of scientific workflow ensembles, while allowing the user to scale the number of workers as needed for the execution of the ensemble. DAKOTA [82] is another solution that provides stochastic tools and an extensible interface for the integration of UQ methods to existing numerical codes. Nimrod/O [83] enables the execution of concurrent evaluations, while providing a range of search algorithms. In order to facilitate the execution of workflow ensembles, WMSs need to support parameter tuning so that researchers can run multiple optimizations with different starting points, controlling the parameter selection to minimize the prediction error. With the large volume of data generated from the different iterations, support for *in situ* analysis and visualization within workflow ensembles may become crucial for extreme-scale.

6.6. Instrument Data Processing

Most scientific communities in the environmental domain (e.g., climate and earth science, etc.) have computer-based scientific experiments, which need to handle massive volumes of data that are generated by different sensors/instruments or observatories. In most cases, they have to handle primary data streams as well as data from institutional and global archives. Often, they employ the two-stage handling of data: establish initial collection with quality monitoring, then explore the data and simulation models, where researchers are responsible for the design of methods and the interpretation of results.

Data streaming is essential to enable scientists to move methods between live and archived data applications, and to address long-term performance goals. The growing volumes of scientific data, the increased focus on data-driven science, and the areal storage density doubling annually (Kryder's Law), severely stress the available disk I/O. This is driving increased adoption of data-streaming interconnections between workflow stages, as these avoid a write out to disk followed by reading in, or double that I/O load if files have to be moved. Therefore, data-streaming workflows are gaining more and more attention in the scientific communities.

Examples of such applications include (1) the analysis of seismic noise cross correlations [84], which aims to detect the relative seismic-wave velocity variations during the time of recording; and (2) the analysis of internal extinction galaxies, which uses astronomical data from the Virtual Observatory to compute the optical luminosity of a galaxy [85]. Both applications were implemented as dispel4py applications. As scientific communities move towards extreme-scale, data-aware workflow scheduling that enables co-location of *in situ* analysis and simulation will be required to support such data streaming workflows.

7. Future Research Challenges

Based on the characterization and classification of the state-of-the-art WMSs presented in this paper, we identify several challenges that must be addressed in order to meet the needs of extreme-scale applications. These research challenges are:

- *Data sharing for in situ workflows:* An important aspect of *in situ* integration is the exchange of data among the simulation and analytics. In distributed workflows, data is communicated primarily via named files. *In situ* data is often communicated via memory. In order to support the multiscale data sharing, we need to explore data naming strategies, potentially taking into account the fact that data is replicated. Research is needed to explore how to leverage existing *in situ* solutions in next-generation WMSs and how to leverage new HPC architectures for in-situ processing.
- *Data-aware work scheduling:* As data set sizes grow and applications exert mounting pressure on the energy consumption of HPC systems, we foresee the need to develop hybrid workflows that consider data placement not only in the wide area, but also within an HPC system. The better co-located analytics and visualization are with the simulation, the better overall time to solution one can achieve.
- *Data-driven workflow management:* Today, workflows are primarily task-driven: when a task completes, the dependent tasks are ready to run. However, in more data-aware hybrid workflows, data becomes the primary entity. This has implications for how task execution is triggered, how parallelism is exploited, and how failures are managed.
- *Robustness to faults:* As we scale up to the extreme, failures will increase in number and in complexity. New techniques and software need to be developed to detect, identify, and mitigate failures at different levels of the system. The WMS will have an important role to play here as it has a global view of the workload and can make decisions

about replication of computations or data, re-scheduling tasks to different resources, etc. However, the WMS cannot do it alone, it needs well-defined interfaces to other system components and needs to be able to rely on their error handling capabilities.

- *Dynamic provenance capture:* Although one would like to save all provenance as the workflow is executing, this is not a feasible approach today and will be even less so in the future. The WMS needs to be able to determine what is the critical information that needs to be captured and what information may be discarded as the workflow successfully progresses — basically performing an online triage of provenance information. One could explore an alternate approach, which would collect provenance at a coarse level initially, but capture more detailed information when problems are observed.
- *Human in the loop:* Manual intervention, or human in the loop, is an important aspect of HPC simulations. Examples include parameter tuning to complete the simulation or computational steering to change the outcome. Unfortunately, incorporating the human in the loop as part of the simulation workflow is still a major challenge. Research is needed to address issues related to this capability, such as how to handle user inputs, determine workflow adjustments, and execute changes in computation.

8. Conclusion

In this paper, we presented a novel characterization of state-of-the-art workflow management systems (WMSs) and the applications they support designed specifically for extreme-scale workflows. As scientific applications scale up in complexity and size, and as HPC architectures become more heterogeneous and energy constrained, WMSs need to evolve and become more sophisticated. To understand this better, we surveyed and classified workflow properties and management systems in terms of workflow execution models, heterogeneous computing environments, and data access methods. This paper has identified a number of properties that future WMSs need to support in order to meet extreme-scale requirements, as well as the research gaps in the state-of-the-art. As many WMSs have been built in a monolithic way, this may require a significant evolution effort. In this effort, one could argue that as the workflows they support demand different execution environments with different capabilities, it may be beneficial to develop more modular WMSs, where environment-tailored execution engines can be invoked as needed.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy (DOE) by Lawrence Livermore National Laboratory (LLNL) under Contract DE-AC52-07NA27344 (LLNL-JRNL-706700). This work was partially funded by the Laboratory Directed Research and Development Program at LLNL under project 16-ERD-036; by the Scottish Informatics and Computer Science Alliance (SICSA) with the Postdoctoral and Early Career Researcher Exchanges (PECE) fellowship; and by DOE under Contract #DESC0012636, "Panorama–Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows".

References

[1] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields, Workflows for e-Science: scientific workflows for grids, Springer Publishing Company, Incorporated, 2007.

[2] Pegasus applications, `https://pegasus.isi.edu/application-showcase/`.

[3] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, K. Wenger, Pegasus, a workflow management system for science automation, Future Generation Computer Systems 46 (2015) 17–35. `doi:10.1016/j.future.2014.10.008`.

[4] R. Filgueira, A. Krause, M. Atkinson, I. Klampanos, A. Moreno, dispel4py: A python framework for data-intensive scientific computing, International Journal of High Performance Computing Applications (IJHPCA) to appear.

[5] M. Albrecht, P. Donnelly, P. Bui, D. Thain, Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids, in: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, ACM, 2012, p. 1. `doi:10.1145/2443416.2443417`.

[6] A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, et al., Fireworks: a dynamic workflow system designed for high-throughput applications, Concurrency and Computation: Practice and Experience 27 (17) (2015) 5037–5059. `doi:10.1002/cpe.3505`.

[7] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, et al., Askalon: A development and grid computing environment for scientific workflows, in: Workflows for e-Science, Springer, 2007, pp. 450–471. `doi:10.1007/978-1-84628-757-2_27`.

[8] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, et al., The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud, Nucleic acids research (2013) W557–W561`doi:10.1093/nar/gkt328`.

[9] D. Blankenberg, G. V. Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, J. Taylor, Galaxy: a web-based genome analysis tool for experimentalists, Current protocols in molecular biology (2010) 19–10`doi:10.1002/0471142727.mb1910s89`.

[10] J. Frey, Condor dagman: Handling inter-job dependencies, University of Wisconsin, Dept. of Computer Science, Tech. Rep.

[11] C. S. Liew, M. P. Atkinson, M. Galea, T. F. Ang, P. Martin, J. I. V. Hemert, Scientific workflows: Moving across paradigms, ACM Computing Surveys (CSUR) 49 (4) (2016) 66. `doi:10.1145/3012429`.

[12] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, N. R. Tallent, HPCToolkit: Tools for performance analysis of optimized parallel programs, Concurrency and Computation: Practice and Experience 22 (6) (2010) 685–701.

[13] E. M. Bahsi, E. Ceyhan, T. Kosar, Conditional workflow management: A survey and analysis, Scientific Programming 15 (4) (2007) 283–297.

[14] M. Bux, U. Leser, Parallelization in scientific workflow management systems, arXiv preprint arXiv:1303.7195.

[15] J. Liu, E. Pacitti, P. Valduriez, M. Mattoso, A survey of data-intensive scientific workflow management, Journal of Grid Computing 13 (4) (2015) 457–493.

[16] E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-science: An overview of workflow system features and capabilities, Future Generation Computer Systems 25 (5) (2009) 528–540.

[17] J. Yu, R. Buyya, A taxonomy of workflow management systems for grid computing, Journal of Grid Computing 3 (3-4) (2005) 171–200.

[18] A. Barker, J. Van Hemert, Scientific workflow: a survey and research directions, in: Parallel Processing and Applied Mathematics, Springer, 2007, pp. 746–753.

[19] The Opportunities and Challenges of Exascale Computing, ASCAC Subcommittee Report, http://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf (2010).

[20] J. Dongarra, With extreme scale computing the rules have changed, in: International Congress on Mathematical Software, Springer, 2016, pp. 3–6. doi:10.1007/978-3-319-42432-3_1.

[21] Report on the ASCR Workshop on Architectures I: Exascale and Beyond: Gaps in Research, Gaps in our Thinking, http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/ArchitecturesIWorkshopReport.pdf (2011).

[22] Report out from the Exascale Research Planning Workshop Working Session on Data Management, Visualization, IO and Storage, http://exascaleresearch.labworks.org/apr2012planningworkshop/application/layouts/exascale-planning-workshop//public/docs/PRES_WorkingSession-DataIO_120420.pdf (2012).

[23] Scientific Discovery at the Exascale: Report from the DOE ASCR 2011 Work shop on Exascale Data Management, Analysis and Visualization, http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Exascale-ASCR-Analysis.pdf (2011).

[24] K.-L. Ma, In-Situ Visualization at Extreme Scale: Challenges and Opportunities, IEEE Computer Graphics and Applications 29 (6) (2009) 14–19.

[25] D. A. Reed, J. Dongarra, Exascale computing and big data, Communications of ACM 58 (7) (2015) 56–68.

[26] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, IEEE, 2010, pp. 1–10.

[27] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets., HotCloud 10 (2010) 10–10.

[28] Apache storm, https://storm.incubator.apache.org.

[29] A. Spinuso, R. Filgueira, M. Atkinson, A. Gemuend, Visualisation methods for large provenance collections in data-intensive collaborative platforms, in: EGU General Assembly 2016, Information in earth sciences: visualization techniques and communication of uncertainty, 2016.

[30] G. Juve, B. Tovar, R. Ferreira da Silva, D. Król, D. Thain, E. Deelman, W. Allcock, M. Livny, Practical resource monitoring for robust high throughput computing, in: Cluster Computing (CLUSTER), 2015 IEEE International Conference on, IEEE, 2015, pp. 650–657. doi:10.1109/CLUSTER.2015.115.

[31] I. Santana-Perez, M. S. Pérez-Hernández, Towards reproducibility in scientific workflows: An infrastructure-based approach, Scientific Programming 2015. doi:10.1155/2015/243180.

[32] D. D. Roure, C. Goble, R. Stevens, The design and realisation of the myexperiment virtual research environment for social sharing of workflows, Future Generation Computer Systems 25 (2008) 7. doi:10.1016/j.future.2008.06.010.

[33] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gómez-Pérez, S. Bechhofer, et al., Using a suite of ontologies for preserving workflow-centric research objects, Web Semantics: Science, Services and Agents on the World Wide Web 32 (2015) 16–42. doi:10.1016/j.websem.2015.01.003.

[34] M. W. Berry, Scientific workload characterization by loop-based analyses, ACM SIGMETRICS Performance Evaluation Review 19 (3) (1992) 17–29.

[35] L. Ramakrishnan, D. Gannon, A survey of distributed workflow characteristics and resource requirements, Tech. Rep. TR671, Indiana University (2008).

[36] S. Ostermann, R. Prodan, T. Fahringer, A. Iosup, D. Epema, in: T. Priol, M. Vanneschi (Eds.), From Grids to Service and Pervasive Computing, Springer, 2008, Ch. On the characteristics of grid workflows, pp. 191–203.

[37] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, K. Vahi, Characterization of scientific workflows, in: Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on, IEEE, 2008, pp. 1–10.

[38] N. Dun, K. Taura, A. Yonezawa, Paratrac: a fine-grained profiler for data-intensive workflows, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, 2010, pp. 37–48.

[39] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Generation Computer Systems 29 (3) (2013) 682–692.

[40] R. Ferreira da Silva, G. Juve, M. Rynge, E. Deelman, M. Livny, Online task resource consumption prediction for scientific workflows, Parallel Processing Letters 25 (03) (2015) 1541003. doi:10.1142/S0129626415410030.

[41] A. Iosup, D. Epema, Grid computing workloads, IEEE Internet Computing 15 (2) (2011) 19–26.

[42] B. Whitlock, J. M. Favre, J. S. Meredith, Parallel in situ coupling of simulation with a fully featured visualization system, in: EGPGV, 2011, pp. 101–109.

[43] N. Fabian, K. Moreland, D. Thompson, A. Bauer, P. Marion, B. Geveci, M. Rasquin, K. Jansen, The paraview coprocessing library: A scalable, general purpose *In Situ* visualization library, in: IEEE Symposium on Large-Scale Data Analysis and Visualization, 2011, pp. 97–104.

[44] H. Yu, C. Wang, R. Grout, J. Chen, K.-L. Ma, In situ visualization for large-scale combustion simulations, Computer Graphics and Applications, IEEE 30 (3) (2010) 45 –57.

[45] S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S.-H. Ku, S. Ethier, J. Chen, C. S. Chang, S. Klasky, R. Latham, R. Ross, N. F. Samatova, Isabela-qa: Query-driven analytics with isabela-compressed extreme-scale scientific data, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 31:1–31:11.

[46] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, H. Abbasi, Enabling in-situ execution of coupled scientific workflow on multi-core platform, in: Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, 2012, pp. 1352–1363.

[47] D. Ellsworth, B. Green, C. Henze, P. Moran, T. Sandstrom, Concurrent visualization in a production supercomputing environment, Visualization and Computer Graphics, IEEE Transactions on 12 (5) (2006) 997 –1004.

[48] A. Esnard, N. Richart, O. Coulaud, A steering environment for online parallel visualization of legacy parallel simulations, in: Distributed Simulation and Real-Time Applications, 2006. DS-RT'06. Tenth IEEE International Symposium on, 2006, pp. 7 –14.

[49] J. Lofstead, F. Zheng, S. Klasky, K. Schwan, Adaptable, metadata rich io methods for portable high performance io, in: Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, 2009, pp. 1 –10.

[50] C. Docan, M. Parashar, S. Klasky, Dataspaces: An interaction and coordination framework for coupled simulation workflows, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, 2010, pp. 25–36.

[51] V. Vishwanath, M. Hereld, M. Papka, Toward simulation-time data analysis and i/o acceleration on leadership-class systems, in: Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on, 2011, pp. 9–14.

[52] The Future of Scientific Workflows. Report of the DOE NGNS/CS Scientific Workflows Workshop, https://science.energy.gov/~/media/ascr/pdf/programdocuments/docs/workflows_final_report.pdf (2015).

[53] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Communications of ACM 51 (1) (2008) 107–113.

[54] W. Mitchell, J. Kallman, A. Toreja, B. Gallagher, M. Jiang, D. Laney, Developing a Learning Algorithm-Generated Empirical Relaxer, Tech. Rep. LLNL-TR-687141, Lawrence Livermore National Laboratory (2016).

[55] M. Jiang, B. Gallagher, J. Kallman, D. Laney, A Supervised Learning Framework for Arbitrary Lagrangian-Eulerian Simulations, in: IEEE International Conference on Machine Learning and Applications, 2016.

[56] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, C. Jin, Flexible io and integration for scientific codes through the adaptable io system (adios), in: Proceedings of the 6th international workshop on Challenges of large applications in distributed environments, ACM, 2008, pp. 15–24.

[57] S. Marru, L. Gunathilake, C. Herath, P. Tangchaisin, M. Pierce, C. Mattmann, R. Singh, T. Gunarathne, E. Chinthaka, R. Gardler, et al., Apache airavata: a framework for distributed applications and computational workflows, in: 2011 ACM workshop on Gateway computing environments, ACM, 2011, pp. 21–28. doi:10.1145/2110486.2110490.

[58] Z. Falt, D. Bednárek, M. Kruliš, J. Yaghob, F. Zavoral, Bobolang: A language for parallel streaming applications, in: Proceedings of the 23rd international symposium on High-performance parallel and dis-

tributed computing, ACM, 2014, pp. 311–314. doi:10.1145/2600212.2600711.

[59] D. Barseghian, I. Altintas, M. B. Jones, D. Crawl, N. Potter, J. Gallagher, P. Cornillon, M. Schildhauer, E. T. Borer, E. W. Seabloom, et al., Workflows and extensions to the kepler scientific workflow system to support environmental sensor data access and analysis, Ecological Informatics 5 (1) (2010) 42–50. doi:10.1016/j.ecoinf.2009.08.008.

[60] T. Glatard, J. Montagnat, D. Lingrand, X. Pennec, Flexible and efficient workflow deployment of data-intensive applications on grids with moteur, International Journal of High Performance Computing Applications 22 (3) (2008) 347–360. doi:10.1177/1094342008096067.

[61] Nextflow, http://www.nextflow.io/index.html.

[62] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, I. Foster, Swift: A language for distributed parallel scripting, Parallel Computing 37 (9) (2011) 633–652. doi:10.1016/j.parco.2011.05.005.

[63] I. Taylor, M. Shields, I. Wang, A. Harrison, The triana workflow environment: Architecture and applications, in: Workflows for e-Science, Springer, 2007, pp. 320–339.

[64] M. Wang, Z. Du, Z. Cheng, S. Zhu, A pipeline virtual service prescheduling pattern and its application in astronomy data processing, Simulation 83 (1) (2007) 123–132.

[65] C. T. Silva, J. Freire, S. P. Callahan, Provenance for visualizations: Reproducibility and beyond, Computing in Science & Engineering 9 (5) (2007) 82–89.

[66] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny, Pegasus: Mapping scientific workflows onto the grid, in: Grid Computing, Springer, 2004, pp. 11–20.

[67] A. Lathers, M.-H. Su, A. Kulungowski, A. W. Lin, G. Mehta, S. T. Peltier, E. Deelman, M. H. Ellisman, Enabling parallel scientific applications with workflow tools, in: Challenges of Large Applications in Distributed Environments, 2006 IEEE, IEEE, 2006, pp. 55–60.

[68] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, et al., Business process execution language for web services (2003).

[69] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, et al., Taverna: a tool for the composition and enactment of bioinformatics workflows, Bioinformatics 20 (17) (2004) 3045–3054.

[70] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, T. Oinn, Taverna: a tool for building and running workflows of services, Nucleic acids research 34 (suppl 2) (2006) W729–W732.

[71] D. De Oliveira, E. Ogasawara, F. Baião, M. Mattoso, Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows, in: Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD), IEEE, 2010, pp. 378–385.

[72] G. E. Horne, T. E. Meyer, Data farming: Discovering surprise, in: Proceedings of the 36th conference on Winter simulation, Winter Simulation Conference, 2004, pp. 807–813.

[73] T. Mason, D. Abernathy, I. Anderson, J. Ankner, T. Egami, G. Ehlers, A. Ekkebus, G. Granroth, M. Hagen, K. Herwig, et al., The spallation neutron source in oak ridge: A powerful tool for materials research, Physica B: Condensed Matter 385 (2006) 955–960.

[74] E. Deelman, C. Carothers, A. Mandal, B. Tierney, J. S. Vetter, I. Baldin, C. Castillo, G. Juve, D. Krol, V. Lynch, B. Mayer, J. Meredith, T. Proffen, P. Ruth, R. Ferreira da Silva, PANORAMA: An approach to performance modeling and diagnosis of extreme scale workflows, International Journal of High Performance Computing Applications 31 (1) (2017) 4–18. doi:10.1177/1094342015594515.

[75] D. Abramson, C. Enticott, I. Altinas, Nimrod/k: towards massively parallel dynamic grid workflows, in: Proceedings of the ACM/IEEE conference on Supercomputing, IEEE Press, 2008.

[76] D. Abramson, B. Bethwaite, C. Enticott, S. Garic, T. Peachey, Parameter space exploration using scientific workflows, in: Computational Science (ICCS), Springer, 2009, pp. 104–113.

[77] D. Król, J. Kitowski, Self-scalable services in service oriented software for cost-effective data farming, Future Generation Computer Systems 54 (2016) 1–15. doi:10.1016/j.future.2015.07.003.

[78] D. Krol, J. Kitowski, R. Ferreira da Silva, G. Juve, K. Vahi, M. Rynge, E. Deelman, Science automation in practice: Performance data farming in workflows, in: 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2016. doi:10.1109/ETFA.2016.7733677.

[79] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds, Future Generation Computer Systems 48 (2015) 1–18. doi:10.1016/j.future.2015.01.004.

[80] G. Guerra, F. A. Rochinha, R. Elias, D. De Oliveira, E. Ogasawara, J. F. Dias, M. Mattoso, A. L. Coutinho, Uncertainty quantification in computational predictive models for fluid dynamics using a workflow management engine, International Journal for Uncertainty Quantification 2 (1) (2012) 53–71.

[81] P. Bui, D. Rajan, B. Abdul-Wahid, J. Izaguirre, D. Thain, Work queue+ python: A framework for scalable scientific ensemble applications, in: Workshop on Python for High Performance and Scientific Computing at SC11, 2011.

[82] B. M. Adams, W. Bohnhoff, K. Dalbey, J. Eddy, M. Eldred, D. Gay, K. Haskell, P. D. Hough, L. Swiler, Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.0 user's manual, Sandia National Laboratories, Tech. Rep. SAND2010-2183.

[83] D. Abramson, T. Peachey, A. Lewis, Model optimization and parameter estimation with nimrod/o, in: Computational Science (ICCS), Springer, 2006, pp. 720–727.

[84] R. Filgueira, R. F. da Silva, A. Krause, E. Deelman, M. Atkinson, Asterism: Pegasus and dispel4py hybrid workflows for data-intensive science, in: Proceedings of the 7th International Workshop on Data-Intensive Computing in the Cloud, IEEE Press, 2016, pp. 1–8. doi:10.1109/DataCloud.2016.4.

[85] R. Filgueira, A. Krause, M. P. Atkinson, I. A. Klampanos, A. Spinuso, S. Sanchez-Exposito, dispel4py: An agile framework for data-intensive escience, in: 11th IEEE International Conference on e-Science, e-Science 2015, Munich, Germany, August 31 - September 4, 2015, 2015, pp. 454–464. doi:10.1109/eScience.2015.40.