International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

The future of scientific workflows

The International Journal of High Performance Computing Applications
1–17
© The Author(s) 2017
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342017704893
journals.sagepub.com/home/hpc
SAGE

Ewa Deelman[1], Tom Peterka[2], Ilkay Altintas[3], Christopher D Carothers[4], Kerstin Kleese van Dam[5], Kenneth Moreland[6], Manish Parashar[7], Lavanya Ramakrishnan[8], Michela Taufer[9] and Jeffrey Vetter[10]

Abstract
Today's computational, experimental, and observational sciences rely on computations that involve many related tasks. The success of a scientific mission often hinges on the computer automation of these workflows. In April 2015, the US Department of Energy (DOE) invited a diverse group of domain and computer scientists from national laboratories supported by the Office of Science, the National Nuclear Security Administration, from industry, and from academia to review the workflow requirements of DOE's science and national security missions, to assess the current state of the art in science workflows, to understand the impact of emerging extreme-scale computing systems on those workflows, and to develop requirements for automated workflow management in future and existing environments. This article is a summary of the opinions of over 50 leading researchers attending this workshop. We highlight use cases, computing systems, workflow needs and conclude by summarizing the remaining challenges this community sees that inhibit large-scale scientific workflows from becoming a mainstream tool for extreme-scale science.

Keywords
Scientific workflows, extreme-scale computing, distributed computing, in situ computing, workflow models

1. Introduction

All science campaigns of sufficient complexity consist of numerous interconnected computational tasks. A *workflow* in this context is the composition of several such computing tasks. A workflow management system (WMS) aids in the automation of those operations, namely, managing the execution of constituent tasks and the information exchanged between them. We define an in situ workflow as one whose tasks are coupled by exchanging information over the memory/storage hierarchy and network of a high-performance computing (HPC) supercomputing system such as current leadership-class US Department of Energy (DOE) facilities and future extreme-scale machines. A *distributed* workflow is one whose tasks are more loosely coupled, for example, through files, and that execute on geographically distributed clusters, clouds, and grids, or multiple computational facilities and/or scientific instruments at user facilities.

The main drivers for workflows, whether in situ or distributed, are the application requirements of scientists and the computing systems on which they generate and/or process data. Examples are computational simulations and data analysis and visualization software (LANL et al., 2016). An instantiation of a workflow represents both the operations and the data products associated with a particular scientific problem. It is assumed that individual tasks and data products in a workflow are developed independently, potentially by different scientists or communities.

[1]University of Southern California, Information Sciences Institute, Marina del Rey, CA, USA
[2]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA
[3]UCSD, San Diego Supercomputing Center, San Diego, CA, USA
[4]Computer Science Department, Rensselaer Polytechnic Institute, New York, NY, USA
[5]Brookhaven National Laboratory, New York, NY, USA
[6]Sandia National Laboratories, USA
[7]Computer Science Department, Rutgers University, New Brunswick, NJ, USA
[8]Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[9]Computer Science Department, University of Delaware, Newark, DE, USA
[10]Oak Ridge National Laboratory, Oak Ridge, USA

Corresponding author:
Ewa Deelman, University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292, USA.
Email: deelman@isi.edu

Against the backdrop of application use cases and extreme-scale computing architectures, a forum of over 50 leading domain and computer scientists met under the auspices of the DOE Office of Science to identify future challenges facing the broad adoption of workflow systems in the DOE and the broader science community. The community identified four main research areas for future workflow development. The first is the design of *task coupling* and data movement between workflow tasks. The key factors studied were scalable and robust control and data flow and the need for efficient and portable migration of heterogeneous data models across tasks. The second area is *programming and usability*. The participants found that lack of support for workflows on HPC platforms impedes adoption of workflow technologies and that programming models, design patterns, the user interface, task communication, and portability are potential areas for improvement. Monitoring is important for scientists to understand how their workflows are progressing and for anomaly detection algorithms to detect any problems with workflow execution. The workflow management system needs to be efficient, scalable, and reliable. When errors occur, it needs to be able to gracefully recover from them, potentially re-trying tasks or rescheduling them to different resources. The fourth key is *validation* of results. The validation of a workflow execution enables being able to reproduce the workflow on the same or another computing environment and involves comparing the workflow execution against performance models, comparing with provenance captured during the execution, and comparing the science results with expectations. The WMS is the natural place to capture much provenance data. Relevant topics include the content and format of provenance data, capture mechanisms, communication of metadata across system software levels, short-term storage and long-term archival, and data mining of provenance information.

An investigation into the workflow research areas above resulted in the following high-level findings. The remainder of this article presents a detailed explanation of these findings, including state of the art, research challenges, and specific recommendations of research and development activities.

1. As the complexity and heterogeneity of scientific workflows increase, there is a need to characterize and study the processes surrounding simulations, instruments (experiments and observations), and collaborations in order to be able to design WMSs that facilitate those processes.
2. Research is needed to understand extreme-scale architectures and their impact on the design of WMSs. Research is also needed to characterize and predict the needs of future scientific workflows and how they will influence the design of future architectures.
3. Workflow systems interact with system software, and as the demand for data awareness in workflow and system software grows, the interactions between the workflow system and operating system (OS) will become more complex. Thus, research is needed to define the relationship between the WMS and the OS/runtime (OS/R) and how the WMS fits into the software ecosystem of HPC platforms. Resource management, scheduling, and provenance capture are potential areas where the WMS and other software systems share responsibilities.
4. The design of control and data flows, data models, and programming interfaces needs further research in the general area of WMS design.
5. During and after the workflow execution, the capture of provenance information and its use to validate performance and correctness and to support data reuse and reproducibility are areas where much research is needed.
6. Benchmarks and community data sets are needed to drive workflow research.

2. Background

The dynamics of complex workflows can range from a simple bag of tasks (e.g. MG-RAST and DOCK (Moustakas et al., 2006) in bioinformatics) and sets of distributed applications with intermediate key-value pairs (e.g. data histograms for high-energy physics (Ekanayake et al., 2008) and object ordering (Brin and Page, 1998)), to more sophisticated iterative chains of MapReduce jobs (e.g. graph mining (Malewicz et al., 2010)). Sets of distributed applications can have multiple stages using files for intermediate data (e.g. astronomy: Montage (Jacob et al., 2010), bioinformatics: BLAST (Mathog, 2003), and earthquake science: CyberShake (Maechling et al., 2006)), and iterative applications may have varying sets of tasks that must be run to completion in each iteration (e.g. Kalman filtering (Sorenson, 1985)).

Workflows may have both in situ HPC and distributed-area computing components. We use the term in situ to describe data processing, triage, filtering, analysis, or visualization that occurs while the simulation is running prior to moving data off the supercomputer for further post hoc analysis done in a distributed fashion. For example, Figure 1 shows a science workflow that integrates simulations with experiments through data analytics.[1] The entire process consists of three (sub)workflows: the measurement and reconstruction of experimental images, the modeling and in situ analysis of simulation data, and the comparison of the two.

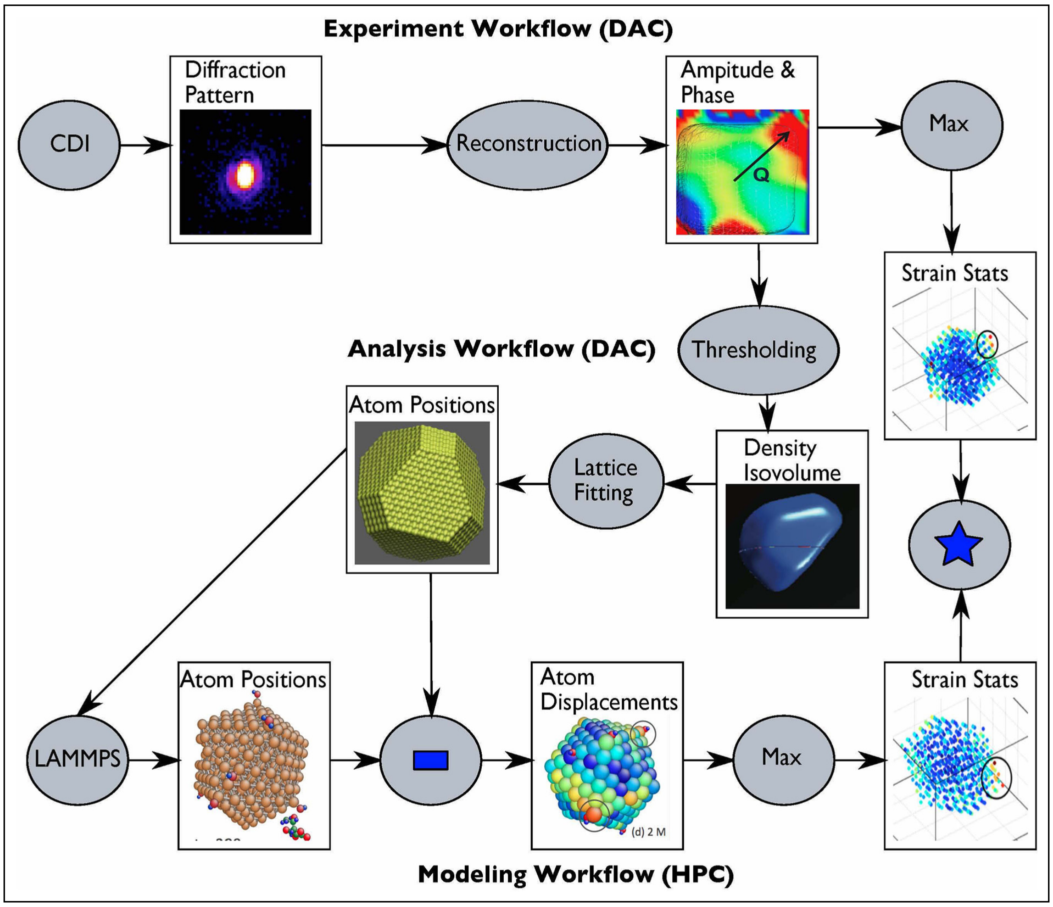

Figure 1. Science workflow for the comparison of a molecular dynamics simulation with a high-energy X-ray microscopy of the same material system includes three interrelated computational and experimental workflows.

No widely adopted general-purpose workflow tools are available that work seamlessly across both in situ and distributed settings. However, specific applications have been designed to work in both domains. CyberShake, a seismic hazard model from the Southern California Earthquake Center (Graves et al., 2011), combines components that use HPC and high-throughput computing. The advanced photon source coordinates HPC with detector hardware and other processing systems (Khan et al., 2013). Likewise, the National Synchrotron Light Source II has initial processing in situ, and the results are then sent to remote users. The Hardware/Hybrid Accelerated Cosmology Code cosmology simulation can interface its HPC with analysis codes on other systems through the CosmoTools analysis framework (Habib et al., 2014). KBase, the DOE systems biology knowledge base, contains in situ modeling and reconstruction tools as well as offloading to cloud-based distributed-area systems (Benedict et al., 2014).

One motivation for in situ workflows is to minimize the cost of data movement by exploiting data locality, operating on data in place. A second reason is to support human-in-the-loop interactive analysis. The third driver for in situ HPC workflows is the need to capture provenance for interactive data analysis and visualization, as well as any computational steering that results. In situ workflows, especially workflows designed for current- and next-generation leadership-scale HPC environments, face particular challenges. These challenges include power, performance, resilience, and productivity: Heterogeneous computing cores, increasingly complex hierarchical memory systems, and small or no growth in bandwidth to external storage systems are some of the main hurdles for HPC in situ workflows at scale.

Large-scale science facilities—supercomputing centers, high-energy light sources, particle accelerators, telescope installations—all need larger scale workflow capability than are currently available. Whether in situ or distributed, scalability of workflows is being driven to the extreme. Many other challenges obstruct the deployment of production workflows at extreme scale. One is performance: In the past, distributed WMSs were designed for reliability first and performance second and they are mostly unaware of optimizing execution on HPC systems. Research into in situ WMSs is still in its infancy, and performance is untested.

Another is usability. Today, scientists use Python or shell scripts to specify workflows, or they integrate the workflow directly into their simulation code. The overlap between "big data" programming models and scientific data programming models adds complexity to the scientific workflow space (BDEC Committee, 2016). Other challenges involve the interaction between the WMS and the OS/R, which should cooperate in the workflow execution. Still others arise from multiple intermediate representations of data, which may be augmented with provenance information, which is important for validation of results. In light of these and other challenges, the community found that the WMS must manage the following aspects as part of extreme-scale workflows.

2.1. Data movement between and within workflow components

The WMS needs to provide efficient, scalable, parallel, and resilient communication through flexible coupling of components with different data and resource needs and utilize extreme-scale architectural characteristics such as deep memory/storage hierarchy and high core count. Memory management systems that support scratch pad and nonvolatile memory (NVM) are needed for workflows. Increasing the number of computing tasks in the same memory will increase the likelihood that workflows will need to extend memory to NVM. Moreover, the potential for more complicated analyses that manage more state (e.g. several time steps of data) offered by workflows will further require the extended footprint offered by NVM.

2.2. Programming models for workflows and their components

Workflows must manage various and possibly heterogeneous software stacks, expressing tasks and their relationships productively and portably and defining data models and their semantics. The WMS needs to negotiate with the OS/R through a well-defined interface on behalf of the entire application workflow. The OS/R must provide the WMS with the interfaces to provision resources, schedule, and coordinate various tasks (such as the simulation and data analysis codes) and capture the provenance that scientists need to support validating and publishing scientific results, including capturing any changes from the initial workflow that result from human-in-the-loop interactive analysis and steering.

2.3. Resource selection and provisioning

Allocation of various types of resources for the generation, movement, analysis, and retention of data in a workflow, with particular attention to heterogeneity (nodes vs. cores, virtualization, memory hierarchy), power and time costs of moving data, and centralized and distributed systems for storage and staging data, is needed. The WMS needs the OS/R to move from its traditional role of managing single tasks to managing global (i.e. internode, distributed) services. A supercomputer usually runs more than one application at a time, and those jobs are managed at some level of global system software. Today, however, user-space jobs are isolated from the others by design, and even OS instances are isolated, with each job booting a new image of the OS (micro)kernel. Research is needed to develop services and expose them to higher levels of the software stack so that the WMS and users can access them. Such global management may be through a hierarchy of resource groupings (enclaves), with heterogeneous programming models/runtimes managing the resources within a given enclave or task.

2.4. Execution and scheduling

Task launching and data transfer over all of the above resources during the staging and execution of a workflow must be coordinated by the WMS. To address the challenges of experimental and observational workflows, the WMS must coordinate the end-to-end workflow life cycle, including real-time scheduling and execution of measurement instruments and analytic platforms, which may include supercomputers. Efficient low-overhead scheduling of multiple cooperative tasks, various forms of communication (messages, interrupts, publish–subscribe) between independent tasks, and provisioning of shared resources (e.g. shared storage) among tasks are needed from the OS/R to support the WMS. Reliability of the execution is also an important concern, not only in wide area but also increasingly in HPC environments.

2.5. Provenance tracking, monitoring, and validation

The WMS needs to capture the high volume, velocity, and variety of provenance data, enable querying, mining, and analyzing these data to validate accuracy of results, compare with expected performance, and ensure reproducibility. Monitoring the infrastructure and applications, understanding workflow behavior (modeling, anomaly detection, and diagnosis), detecting, isolating, recovering from hard and soft errors, and maintaining security are also needed.

3. Toward extreme-scale workflow systems

Extreme-scale workflow systems need to provide the following capabilities: programming and usability, the efficient coupling and execution of tasks, robust execution and monitoring, and validating outcomes.

3.1. Programming and usability

Programming and usability are key factors determining the extent of adoption of workflow methods at extreme scale. The relationship between programming models for the workflow and those used for individual tasks in the workflow is one aspect of programmability. The definition of workflow graphs can be aided by reusing workflow design patterns or templates. The nature of the WMS user interface, whether textual or graphical, also affects usability.

DOE applications have a diverse set of workflow needs. Research to identify common needs and expression patterns (akin to design patterns in software engineering) in workflows with respect to a number of properties including data management, error control, reproducibility, programmability, and mapping to physical resources are needed. Research is needed to determine appropriate levels of abstraction to define the user interface for workflow systems and their component modules, including an interface for the human-in-the-loop in interactive workflows.

3.1.1. Programming models. Although many HPC workflows are hand constructed through shell scripts, batch scheduling, and human intervention, there exist programmable tools such as Swift (Wilde et al., 2011), Tigres (Ramakrishnan et al.,2014), Kepler (Altintas, et al., 2004), Trident (Barga et al., 2008), Weaver (Bui et al.,2010), Triana (Churches et al., 2006), Pegasus (Deelman et al., 2015), Galaxy (Goecks et al.,2010), and Taverna (Oinn et al., 2006) to better manage complex workflows. Programming models for cloud, web service, and other big data applications are abundant. The programming model for MapReduce (Dean and Ghemawat, 2004) is probably the most well known, but many others exist (Fox et al., 2014; Jha et al., 2014; Qui et al., 2014). Many of these models may be leveraged for use in HPC.

Most workflow management tools use a scripting language to define tasks and dependencies and to manage execution (Altintas et al., 2004; Barga et al., 2008; Bui et al., 2010; Churches et al., 2006; Deelman et al., 2015; Goecks et al., 2010; Oinn et al., 2006; Ramakrishnan et al., 2014; Wilde et al., 2011). Within the scripting language is an application programming interface (API) that scripts use to define and execute a workflow. There also exist examples of workflow building tools that use a graphical interface (Bavoil et al., 2005; Parker and Johnson, 1995). Such interfaces provide a trade-off between simplicity and expressiveness. The appropriate level of abstraction for workflows is unclear. Having different levels of abstractions for new and advanced users could also be advantageous. Studying how business workflow models and techniques (Weerawarana et al., 2005) can be helpful in this regard.

As leadership-class machines and the workflows executing on them increase in complexity, the division between the workflow and the task programming model becomes blurred. The workflow system's representation of a mix of coarse data- and task-parallelism mirrors the finer-grained task-parallel programming models emerging for programming individual tasks. Because both workflows and programming models address high concurrency, dynamic application execution, dynamic resource availability, architectural diversity, and in-system storage, integration and coordination become fruitful and perhaps necessary.

3.1.2. Design patterns. Workflow management tools such as Swift (Wilde et al., 2011) and Tigres (Ramakrishnan et al., 2014) build Directed-Acyclic Graph (DAGs) and internally manage parallel execution and dependencies within them. Other tools such as AVS (Upson et al., 1989), SCIRun (Parker and Johson, 1995), and VisTrails (Bavoil et al., 2005) allow users to build and view workflow DAGs visually with a graphical representation and user interface. Wings (Bergmann and Gil, 2014) uses the concept of templates to represent the overall workflow structure and then automatically fills out the template based on user needs.

Many scientists build workflows by example; they iteratively construct one workflow using a previous one as a template. This iterative modification of existing workflows shortens the development time and can also be integrated in software design to accelerate workflow tool development (Maheshwari et al., 2013). Some recent tools use pattern-like structures as part of the creation and execution of workflows. For example, Tigres has a collection of templates that can be applied when building workflow structures (Balderrama et al., 2014). VisTrails can collect the provenance of many previously built workflows to find common patterns that can automatically assist users in other endeavors (Silva et al., 2007).

It is important to understand and classify various workflows and workflow needs through user studies. Identifying common patterns for next-generation in situ and distributed workflows is needed to address programmability and usability concerns. Workflows need to correctly and more formally handle task dependency loops. Part of this effort requires workflows to understand and manage time and data that change over time, as demonstrated in the similar Visualization Toolkit (VTK) dataflow network (Biddiscombe et al., 2007). Research is needed into ways to help users easily develop high-level templates and to instantiate them for concrete problems. Scientists often use ensemble workflows, and research in ensemble management is also needed to support the end-to-end computational methods.

3.1.3. Portability. Sometimes the same conceptual workflow is run in situ (within a single system) and other times as distributed processes where tasks are coordinated across multiple independent systems that may be physically distant from each other. Workflows that need to run in both modes should not be implemented twice. For example, the same code may need to point to data in memory, read data from a file, output data to memory or a file, run in serial or parallel, compute a small- or large-scale job, process data in core or out of core, and be built as a library or as an executable. All of this should be uniform so that data and control can seamlessly flow between environments.

Portability is difficult on complex, heterogeneous systems. Workflows can help match tasks to the architecture best suited to run them; this situation is more common in distributed workflows, but it is still an area of research for HPC. Containers are a possible solution for some distributed area and in situ workflow solutions. In general, WMSs that operate with good performance across heterogeneous platforms are needed. Understanding the role of containers, virtualization, and security—features found in distributed computing—is needed in HPC. Understanding the effect of disruptive HPC architectures such as deep memory hierarchies and NVM is needed as well.

Researchers are asking the following portability questions. How can we build WMSs and common application components that operate with good performance both in situ and distributed? What is the role of containers and other virtualization technologies in workflows? What are security requirements for in situ and distributed workflows? Can workflows manage deep memory hierarchies? As HPC and workflows become more complex, what should be the interaction between human and system? The human interaction can become even more complex as we mix in situ and distributed components in the same workflow.

3.2. System design and execution

The design of WMSs for extreme-scale distributed and in situ workflows presents multiple challenges; some are unique to each class of workflows, while others are common to both. The two types of WMSs also typically implement different design trade-offs. While concurrency, locality, system topology awareness, and minimizing data movement are common to both distributed and in situ workflows (Bennett et al., 2012; Zheng et al., 2011), distributed workflows additionally address issues related to security, crossing administrative domains, and so forth that do not exist in HPC. Monitoring the execution progress of in situ workflows and adapting the execution also tends to be harder, largely because of restrictions from the system. A related concern is dealing with failures. Although being able to detect and handle unreliable resources and failures has been an integral part of distributed WMSs, the in situ WMSs have only recently started to address failures (Cappello, 2009; Gamell et al., 2014).

WMSs may have different optimization considerations. For example, analytics may be scheduled in situ or on a different system depending on the relative costs of the latencies associated with data movement and the loss in performance due to the repurposing of some compute nodes for analytics. Several research efforts are already exploring such a convergence. For example, Wide-Area-Staging (Aktas et al., 2014) is extending data staging abstractions to distributed environments, and DataSpaces-as-a-Service is exploring persistent staging and service-oriented architecture (SOA) models on extreme-scale systems.

3.2.1. Resource provisioning and scheduling. Because of the increased complexity of the workflows and the execution environments, we need more sophisticated WMSs that can dynamically provision (potentially distributed) resources as the workflow (or set of workflows) are executing and that can manage the execution of workflow tasks on these resources in an efficient manner. Provisioning and management should also include storage and network resources.

3.2.2. Workflow resilience and fault detection. As system scales increase and the mean time between failures (MTBF) becomes smaller, process and node failures becomes important. Recovery from these failures often involves terminating the job and restarting from the last checkpoint available in stable storage. However, it is unclear whether this approach will work when the MTBF approaches the time needed to execute the checkpoint. Online, application-aware, and local recovery mechanisms are possible alternatives. In addition to addressing system and process failures, there is also an increasing need for data validation mechanisms. Data corruption, whether occurring as a result of bugs, attacks, or background radiation, will be more likely in workflows running on increasingly complex hardware and software systems than in past.

3.3. Coupling tasks

Coupling control and data flow between heterogeneous components requires research in dataflows that can buffer, prefetch, aggregate, and distribute data, and research is needed to solve challenges in the transport, layout, attributes, and provenance of these data. Workflow tasks that are developed independently invariably operate on different data models. The

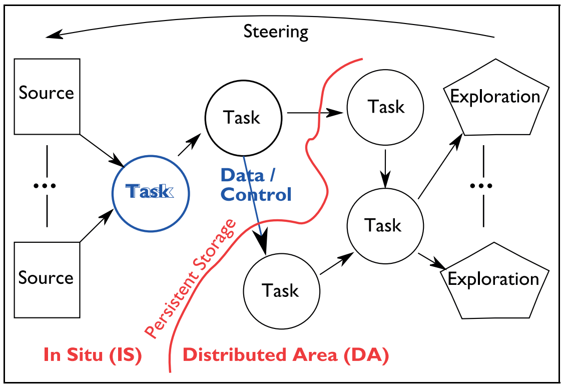

Figure 2. A typical workflow includes both in situ and distributed area subworkflows and may include cycles.

transfer of data models across tasks requires defining the data model by the programmer in a way that the WMS can efficiently transfer parts of the model between the tasks that need it, with a minimum of data copying, while retaining the semantic validity of the data. Complicating coupling is the fact that tasks have varying amounts of concurrency—processes or threads—and data must be redistributed between these tasks efficiently (in parallel) and safely (retaining semantics).

3.3.1. Task coupling. Tasks are coupled according to a workflow graph, which can be cyclic and which can include both in situ and distributed components. Figure 2 shows one such example graph.

There may be decoupled dataflows (Dorier et al., 2015) between pairs of producer–consumer tasks in the graph. These dataflows are a hybrid of tight and loose couplings. The dataflow resources can be disjoint from the producer and consumer tasks, or they may overlap either or both tasks, enabling time- and space-division coupling of tasks from the same logical graph topology (Figure 3).

Efficient design patterns for parallel data communication are needed based on either higher level messaging libraries such as Message Passing Interface (MPI) or lower level network protocols such as Transmission Control Protocol/Internet Protocol.

Such standard design patterns are needed in order to compose different workflow abstractions from heterogeneous WMSs. For example, tasks may be defined in different in situ and distributed WMSs, and a common coupling library may provide a standardized data model redistribution and a parallel message communication for the entire ensemble. Runtimes are message-driven, executing tasks when all messages have been received and enabling cyclic executions anywhere in the workflow graph.

3.3.2. Control and data flow. Distributed systems have largely relied on files for control and data flow (Altintas et al., 2004; Deelman et al., 2015), while in situ systems have explored in-memory staging and Remote direct memory access (Dayal et al., 2014; Docan et al., 2010). While distributed WMSs support more dynamic and service-oriented compositions, recent research in HPC is exploring how SOA-type compositions can be supported on extreme-scale systems (Abbasi et al., 2009).

Another important development is dedicating resources to the links that connect the tasks of the workflow. For example, link components can be used for buffering, prefetching, and aggregating data but may also be used for transforming and redistributing data so that it is more compatible with the needs of the downstream task. Staging-based approaches for specifying these links and their semantics have been explored by recent projects (Docan et al., 2011; Zheng et al., 2010). Lofstead et al. (2008) propose the use of "glue components" that encapsulate such link semantics and can be defined as part of a workflow. Because the link resource requirements can be dynamic and not known a priori, adaptive runtime resource allocation and management become critical. Furthermore, this resource management must effectively address technical issues such as failures and adaptivity (Bhat, 2008; Dayal et al., 2013).

As distributed and in situ workflows are integrated into end-to-end science applications, new abstractions and mechanisms can help support control and data flow requirements in a consistent and scalable manner.

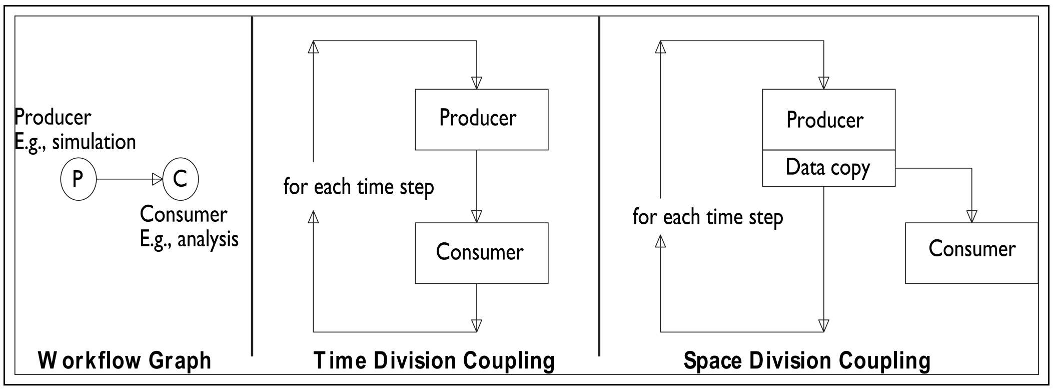

Figure 3. Time- and space-division coupling of a producer and consumer task.

Ways to augment the coupling between tasks with meaningful semantics and to effectively manage these components in an adaptive and autonomic runtime are needed. A catalog of data/control flow (link) semantics can meet the needs of emerging application workflows, and the efficient implementation of such components can enable workflow scalability. Further research into automatic placement and allocation of intermediate link resources and managing the flow control over them is also needed.

3.3.3. Data models. Traditionally, workflows used files to transmit data, requiring programs to understand each other's file format. Many HPC file formats are "self-describing" in that arrays are organized using names, attributes, and hierarchies (Lofstead et al., 2008; Prabhat and Koziol, 2014), prompting a storage API to be used even when data are coupled in memory or through the network. Similar organization is present in array databases (Stonebraker et al., 2011) and NoSQL databases (Cattell, 2011), where conventions (Eaton et al., 2011) and schemas (Clarke and Mark, 2007; Shasharina et al., 2010; Tchoua et al., 2010) are often applied. Much work has focused on providing a unified interface to a variety of storage implementations. For example, ADIOS can support numerous input/output (I/O) backends and switch between them at runtime (Lofstead et al., 2008). Tools such as Google Dremel (Melnik et al., 2010) and Apache Drill (Apache Drill—Schema-free SQL for Hadoop, NoSQL, and Cloud Storage) provide a unified interface to multiple data backends as well.

Nonetheless, today's workflow systems have limited support for data models not derived from files, databases, or *N*-dimensional regular arrays. As workflows integrate ever-varying tasks, it becomes more challenging to communicate data between software that uses heterogeneous and unstructured data models. One solution (Dreher and Peterka, 2016) is to annotate a data model with enough information to preserve the semantic integrity of the data during redistribution.

The community needs to continue developing appropriate infrastructure that allows seamless integration of various data sources including streaming, data management across the memory-storage hierarchy of next-generation systems, and data semantics in workflows. Greater understanding is needed in how data can be communicated among tasks that are developed independently and have different data models. The mechanism for connecting data sources and the ability to identify, convert, and verify data models must be well established.

3.4. Monitoring and provenance

WMSs offer a unique monitoring opportunity because they encapsulate the entire process of solving a computational problem. They are the managers of many different operations and, at the same time, interact closely with many other relevant components and resources related to the execution of the workflow. However, the number of components, the complexity of their connections, and the rate of execution complicate monitoring of workflows. As a result, the monitoring system needs to provide the right level of abstraction for the task at hand: evaluating workflow progress, understanding causes of failure, and understanding the results.

The real-time status of the workflow needs to be accessible to users. A human-in-the-loop is needed in many different types of workflows including exploration and failure recovery. For example, a human may be required to determine whether missing telescope data are a result of a cloudy night or failures in the hardware or software. Similarly, light source experiments can benefit from real-time feedback to improve the quality of experimental decisions. Today, only limited support is provided to integrate the human in workflows and to automatically track the provenance from such activities.

3.4.1. Provenance models. Monitoring the performance and correctness of workflows depends on capturing provenance data: system statistics such as timing and memory footprint as well as science results. Such provenance data allow validation of workflow performance, traceability of workflow execution, and workflow reproducibility and reuse. We need new provenance models suitable to support these usage models and that integrate provenance from multiple sources. Mining of provenance data has emerged as a key challenge in a number of applications. Today, provenance usually represents a simple directed graph, describing one level of abstraction of an environment or process (e.g. a single workflow representation (Kleese van Dam et al., 2015; Muniswamy-Reddy et al., 2009)). Some systems such as Pegasus provide details about the environment in which the workflow is executed. To enable greater interoperability between provenance models, a working group for the World Wide Web Consortium defined the core specification for an open provenance model (OPM) (Moreau et al., 2011) in 2011. Subsequently, a range of OPM-compliant workflow provenance models have been developed, such as OPMW (Garijo and Gil, 2012), D-OPM (Cuevas-Vicenttín et al., 2012), and the work of Lim et al. (2011). All these models focus exclusively on the description of the workflow, usually in graph form, describing its components and the data utilized (Giardine et al., 2005; Goecks et al., 2010; Scheidegger et al., 2008; Wolstencroft et al., 2013). All models treat workflow tasks as black boxes, with few details attached, and none of them captures details of the execution environment or human interactions.

Existing workflow provenance solutions are only effective for low-volume capture. Research is needed in high-velocity capture mechanisms in extreme-scale environments. None of the systems available today can communicate provenance across system layers. Indeed, the majority of the systems were implemented with post hoc forensic analysis in mind (storage in log files or databases); thus, they do not include provisions for real-time exchange, negotiation, and analysis. We need to study, characterize, and model the types of processes that are usefully supported by such an approach; what information needs to be captured, when, and to whom it needs to be communicated. Effective capture and communication mechanisms, in situ triage, and analysis are other areas of research and development.

3.5. Validation and reproducibility

The increasing complexity of both workflows and their computational environments makes it critical to provide the HPC community with the approaches, methods, and tools to ensure that workflows are executed with sufficient reproducibility (Stodden et al., 2016). Extreme-scale systems, with higher degrees of concurrency and in situ data analysis that is triggered by specific events in very large-scale modeling and simulation applications, will further accentuate the community's need for validation and reproducibility (Top Ten Exascale Research Challenges, 2014).

3.5.1. Performance validation and predictability. Although there is a rich portfolio of tools for performance analysis, modeling, and prediction for single applications in homogeneous computing environments (Thakur, 2015), relatively few workflow performance studies were conducted, usually focused on a specific workflow or WMS (Burtscher et al., 2010; De Oliveira et al., 2012; Juve et al., 2013; Samak et al., 2011; Talukder et al., 2009; Truong et al., 2007). Attempts to address the situation include skeletons for performance prediction (Logan et al., 2012; Meyer et al., 2013; Zhang and Katz, 2014) and targeted distributed computing middleware such as Pegasus (Deelman et al., 2015, 2004, 2005) and Swift (Wilde et al., 2011, 2009; Zhao et al., 2007). Validation goes hand in hand with predictability. For extreme-scale workflows, however, the meaning of performance prediction and result reproducibility needs to be reviewed. Communities other than HPC can provide hints for addressing these issues, for example, service-level agreements in cloud computing.

Performance goals can be expressed and measured either for the entire facility or for the specific workflow, and how to best express performance goals and assess their achievement for a facility versus for a specific workflow is an open challenge. As the complexity of applications increases, the workflow must play a major role in application performance prediction. The question is how can scientists keep track of aspects of the workflow execution at extreme scale, especially when competing for resources with other workflows and when frequent unexpected events such as silent errors or node failures occur. Intelligent schedulers should use the knowledge of events' occurrences to optimize the performance according to defined goals, despite the challenges introduced by new extreme-scale architectures, execution environments, geographical distribution, and scientific instruments. Runtime use of acquired knowledge can be used to continually improve performance and optimize resource use at the scheduler level—for example, by simultaneously running storage- and compute-intensive jobs.

The following steps are needed to validate performance. We need to identify and quantify sources of workflow performance variability as they relate to different performance goals, taking into account the workflow tasks, the WMS, the execution environments, and system architectures. The community needs to investigate the impact of system, scientific data, and user events on workflow performance. In order to have a systematic approach to the problem, workflow benchmarks, execution traces, and performance data are needed. Suitable performance analysis and modeling tools are lacking. Given models and benchmarks, the community can then be used to investigate runtime optimization strategies and methodologies, including suitable interactions between the WMS and the system software stack including resource provision, runtime system, storage and I/O, and software-defined networking.

3.5.2. Fault tolerance and recovery. The increasing scale and complexity of computing systems make fault tolerance and recovery key challenges for workflows. Many existing WMSs handle task and system failures and incorporate fault-tolerant mechanisms (e.g. task re-execution or rollback from checkpoints). Research efforts are addressing these issues by extending existing programming systems to support fault tolerance. Emerging task DAG-based programming models also include resilience features (Wilke et al., 2014).

In addition to addressing system and process failures, there is an increasing need to protect against data corruption. Data corruption, whether the result of bugs, attacks, or background radiation, will be more likely in workflows running on increasingly complex hardware and software systems than in past. For example, if the provenance archival system includes unreliable resources (e.g. interim storage in network devices or exascale memory), resilience measures, and their performance and energy costs need to be considered. Recent research is exploring online mechanisms for resilience (e.g. Gamell et al., 2014; Teranishi and Heroux, 2014).

Research is required into developing automatic, online, and possibly local recovery mechanisms that can handle the scales, complexities, and failure rates of emerging systems. Exploring application-aware mechanisms will be critical. Programming and runtime support for cross-layered power and resilience management are needed so that application programmers can choose the minimum level of resilience required in each code segment, as well as to balance trade-offs and meet power budgets. The community needs to develop a generic method that provides efficient error detection capabilities for both systematic and nonsystematic errors. Research in new adaptive and resilient WMSs can allow mitigating error propagation and failure recovery across software layers in a way that enables users to understand application and system behavior and make online recovery possible.

3.5.3. Accuracy and scientific reproducibility. Reproducibility refers to "closeness of agreement among repeated simulation results under the same initial conditions over time," and accuracy refers to "conformity of a resulted value to an accepted standard (or scientific laws)" (Considine and Considine, 1999). The definitions of conformity and agreement often depend on the community or even the individual scientist's point of view. They may range from stringent bitwise reproducibility to overall agreement of scientific conclusions.

The inability to exactly reproduce data at the application level can be the result of arithmetic and algorithm factors (Balaji and Kimpe, 2013; Chiang et al., 2013). In response to these challenges, mathematical techniques can be applied to mitigate the degree to which aggregated results exhibit sensitivity to operation order. Such techniques can range from simple fixed-reduction orders (Yelick, 2012) to interval arithmetic (Revol and Theveny, 2014) and to extended precisions (Bailey, 2005). Compensated summation algorithms (Kahan, 1965), composite precision (Chapp et al., 2015; Taufer et al., 2010; Thall, 2006), and prerounded algorithms (Arteaga et al., 2014; Demmel and Nguyen, 2015) show promising results but require some level of code modification and performance tuning (Gustafson, 2015).

Reproducibility, enabled by provenance, is a fundamental requirement for the validation of complex workflows. Automatic annotations embedded in multilayer and modular workflows are desired, where workflows provide documentation of paths taken and resources used, with the aim of making workflows reproducible. While applications must continue to be responsible for providing results of acceptable accuracy; in many instances, the workflow is the right place to validate and propagate performance and accuracy expectations and achievements.

By establishing the required accuracy of components a priori, workflow systems can monitor specific variables to see whether the workflow is progressing as intended. The use of data mining, machine learning, and statistical methods can identify deviations and possibly correct them. Open questions include to what degree the scientist can be added in the loop and what role can she play in determining accuracy (i.e. what deviation is acceptable) and in deciding on suitable actions. The more that workflows can document the paths taken and check that the overall simulation is behaving correctly, the better they will be able support the scientist in validating and reproducing science data.

4. Conclusions

In the context of scientific computing, a workflow is the orchestration of multiple computing tasks over the course of a science campaign. Examples are computational simulations, experimental observations, and data analysis and visualization software working in concert to test a hypothesis and arrive at a conclusion. A large-scale science campaign consists of numerous such codes working together.

Workflows and workflow systems are common in distributed cloud and grid computing but relatively obscure in HPC that features batch jobs consisting of single codes. One of the objectives of the Workshop on the Future of Scientific Workflows (2015) was to bring these two communities together to share their expertise. The mission of the workshop was to develop requirements for workflow methods and tools in a combined HPC and distributed work environment to enable science applications to better manage their end-to-end data flow.

This article described the findings of the scientific workflow community as represented in that meeting (Ewa Deelman et al., 2016). The highest priority findings in the areas of application requirements, hardware systems, software systems, WMS design and execution, programming and usability, provenance capture, and validation are listed below.

4.1. Application requirements

Workflows for both computational and experimental sciences need investigation, as do workflows to support scientific collaboration. Sharing workflows, migrating workflows between different computing environments, accommodating different user roles, and combining different languages and software tools used by various users all require further research.

4.2. Hardware systems

Extreme-scale hardware challenges for workflows—power, performance, resilience, and productivity—

require significant planning and investment in system software, programming environments, applications, and WMSs. Heterogeneous nodes, new memory systems including NVM, and little growth in bandwidth to external storage systems or networks dictate new use patterns for WMS that leverage in situ analytics and heterogeneous programming models for individual workflow tasks.

4.3. System software

Supercomputers today are intended to run single-program batch jobs. Workflows, in contrast, are collections of multiple programs whose execution must be coordinated. Workflows require resource allocators to treat storage, I/O, and network capacity as first-class resources to be allocated, managed, and measured to the same degree as computing capacity is today. Scheduling HPC and distributed resources over several systems will require cooperative schedulers that can coordinate with the WMS.

4.4. Programming and usability

Major challenges include lack of standardization between numerous programming models for tasks and workflows, the interconnection between the programming of individual tasks and entire workflows, and the portability of both code and data across different locations in a potentially heterogeneous workflow that spans both HPC and distributed resources. Human interaction with the workflow (e.g. to steer a computation in one program based on a result in another) is another challenge.

4.5. WMS design and execution

WMSs need to expand to deal with data management challenges in the transport, layout, attributes, and provenance of data. To provide efficient execution, one needs to examine the interplay of resource provisioning and workflow management. Then, based on the available resources and the needs of the workflow tasks, one needs to develop mapping and scheduling strategies that can handle distributed HPC resources in a consistent and integrated manner and can optimize execution across these resources. Research is required into developing automatic, online, and possibly local recovery mechanisms that can handle the scales, complexities, and failures rates of emerging systems. Exploring application-aware mechanisms will be critical. WMSs also need to be resilient to failures and provide mechanisms for error detection and failure information propagation.

4.6. Task coupling

The efficient management of distributed and HPC workflows present challenges at multiple levels of software. At the base layer, coupling control and data flow between heterogeneous components requires expanding workflow links into data flows that can buffer, prefetch, aggregate, and distribute data. At a higher level of abstraction, managing the workflow must address concurrency, locality, and system topology if data movement is to be optimized at extreme scale.

4.7. Monitoring

WMSs offer a unique opportunity for provenance capture, because they encapsulate the process of solving a computational problem. The increasing scale and complexity of hardware and software systems, however, coupled with the composition of multiple tasks by workflows, complicates provenance capture. The velocity of provenance data generated at extreme scale requires new methods to compress, mine, analyze, store, and share it.

4.8. Validation and reproducibility

The increasing complexity of workflows and their computational environments makes it critical to provide the approaches, methods, and tools to ensure that workflows are executed with sufficient reproducibility in terms of performance and accuracy. Validation of expected performance is a complex high-dimensional space of metrics over a heterogeneous computing architecture. Needed research directions include extending single-application performance validation tools to workflows of applications and developing methods to learn what levels of differences in science results are statistically significant.

Acknowledgments

The authors would like to thank all workshop participants for their contributions: Greg Abram, Gagan Agrawal, James Ahrens, Pavan Balaji, Ilya Baldin, James Belak, Amber Boehnlein, Shreyas Cholia, Alok Choudhary, Constantinos Evangelinos, Ian Foster, Geoffrey Fox, Foss Friedman-Hill, Jonathan Gallmeier, Al Geist, Berk Geveci, Gary Grider, Mary Hall, Ming Jiang, Elizabeth Jurrus, Gideon Juve, Larry Kaplan, Dimitri Katramatos, Dan Katz, Darren Kerbyson, Scott Klasky, Jim Kowalkowski, Dan Laney, Miron Livny, Allen Malony, Anirban Mandal, Folker Meyer, David Montoya, Peter Nugent, Valerio Pascucci, Wilfred Pinfold, Thomas Proffen, Rob Ross, Erich Strohmaier, Christine Sweeney, Douglas Thain, Craig Tull, Tom Uram, Dean Williams, Matt Wolf, John Wright, John Wu, Frank Wuerthwein, and Dantong Yu. The authors also thank Lucy Nowell and Richard Carlson for sponsoring the workshop on scientific workflows.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: Argonne National Laboratory is managed by UChicago Argonne, LLC, for the U.S. Department of Energy Office of Science under contract DE-AC02-06CH11357. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. This work was funded in part by the US Department of Energy under Contract #DESC0012636, "Panorama -Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows".

Note

1. http://tpeterka.github.io/maui-project/

References

Abbasi H, Wolf M, Eisenhauer G, et al. (2009) DataStager: scalable data staging services for petascale applications. In: *18th ACM international symposium on high performance distributed computing*, 2009, pp. 39–48. DOI:10.1007/s10586-010-0135-6.

Altintas I, Berkley C, Jaeger E, et al. (2004) Kepler: an extensible system for design and execution of scientific workflows. In: *16th international conference on scientific and statistical database management*, Santorini Island, Greece, 21–23 June 2004.

Aktas MF, Haldeman G and Parashar M (2014) Flexible scheduling and control of bandwidth and in-transit services for end-to-end application workflows. In: *Fourth international workshop on network-aware data management*, New Orleans, Louisiana, USA, 16 November 2014, pp. 28–31.

Apache Drill – Schema-free SQL for Hadoop, NoSQL and Cloud Storage [Online]. Available at: https://drill.apache.org/ (accessed November 2016).

Arteaga A, Fuhrer O and Hoefler T (2014) Designing bit-reproducible portable high-performance applications. In: *IEEE 28th international parallel and distributed processing symposium*, Phoenix, Arizona, USA, 19–23 May 2014, pp. 1235–1244.

Bavoil L, Callahan SP, Crossno PJ, et al. (2005) VisTrails: enabling interactive multiple-view visualizations. In: *In IEEE visualization 2005*, pp. 135–142. DOI:10.1109/VISUAL.2005.1532788.

Bergmann R and Gil Y (2014) Similarity assessment and efficient retrieval of semantic workflows. *Information Systems* 40: 115–127.

Balderrama JR, Simonin M, Ramakrishnan L, et al. (2014) Combining workflow templates with a shared space-based execution model. In: Johan M and Ian JT (eds.) *Proceedings of the 9th workshop on workflows in support of large-scale science*, New Orleans, LA, USA, 16–21 November 2014, pp. 50–58.

Biddiscombe J, Geveci B, Martin K, et al. (2007) Time dependent processing in a parallel pipeline architecture. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1376–1383.

Benedict MN, Mundy MB, Henry CS, et al., (2014) Likelihood-based gene annotations for gap filling and quality assessment in genome-scale metabolic models. *PLoS Computational Biology* 10(10): e1003882.

BDEC Committee (2016) The BDEC 'Pathways to Convergence' Report. Available at: http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/bdec2017pathways_v2.pdf.

Brin S and Page L (1998) The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30(1–7): 107–117.

Balaji P and Kimpe D (2013) On the reproducibility of MPI reduction operations. In: Erol G, Guojun W and Qingping Z (eds.) *10th IEEE international conference on high performance computing and communications & 2013 IEEE international conference on embedded and ubiquitous computing*, Zhangjiajie, China, 13–15 November 2013, pp. 407–414.

Barga R, Jackson J, Araujo N, et al. (2008) The trident scientific workflow workbench. In: Geoffrey F (ed.) *Proceedings of the 2008 fourth ieee international conference on eScience*, Indianapolis, IN, USA, 10–12 December 2008, pp. 317–318.

Bui P, Yu L and Thain D (2010) Weaver: integrating distributed computing abstractions into scientific workflows using python. In: Salim H and Kate K (eds.) *Proceedings of the 19th ACM international symposium on high performance distributed computing*, Chicago, IL, USA, 21–25 June 2010, pp. 636–643.

Bhat VN (2008) *Autonomic Management of Data Streaming and In-Transit Processing for Data Intensive Scientific Workflows*. PhD thesis, The State University of New Jersey, Rutgers.

Bennett JC, Abbasi H, Bremer PT, et al. (2012) Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In: Jeffrey KH (ed.) *Proceedings of the international conference on high performance computing, networking, storage and analysis*, Los Alamitos, CA, USA, 10–16 November 2012, pp. 491–499. IEEE Computer Society Press.

Bailey DH (2005) High-precision floating-point arithmetic in scientific computation. *Computer Science Engineering* 7(3): 54–61.

Burtscher M, Kim BD, Diamond J, et al. (2010) PerfExpert: an easy-to-use performance diagnosis tool for HPC applications. In: Barry VH (ed.) *Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis*, New Orleans, LA, USA, 13–19 November 2010, pp. 1–11. IEEE Computer Society.

Cappello F (2009) Fault tolerance in petascale/exascale systems: current knowledge, challenges and research opportunities. *International Journal of High Performance Computing Applications* 23(3): 212–226.

Chapp D, Johnston T and Taufer M (2015) On the need for reproducible numerical accuracy through intelligent runtime selection of reduction algorithms at the extreme scale. In: Pavan B and Michela T (eds.) *Presented at the IEEE cluster conference*, Chicago, IL, USA, 8–11 September.

Chiang WF, Gopalakrishnan G, Rakamaric Z, et al. (2013) Determinism and reproducibility in large-scale HPC systems. In: *Workshop determinism correctness parallel programming (WoDet)*, Houston, TX, USA, 16–20 March 2013.

Churches D, Gombas G, Harrison A, et al. (2006) Programming scientific and distributed workflow with Triana services: research articles. *Concurrency and Computation: Practice and Experience* 18(10): 1021–1037.

Cattell R (2011) Scalable SQL and NoSQL data stores. *ACM Sigmod Record* 39(4): 12–27.

Cuevas-Vicenttín V, Dey S, Wang ML, et al. (2012) Modeling and querying scientific workflow provenance in the D-OPM. In: Jeffrey KH (ed.) *High performance computing, networking storage and analysis*, Los Alamitos, CA, USA, 10–16 November 2012, pp. 119–128. IEEE Computer Society Press.

Clarke JA and Mark ER (2007) Enhancements to the eXtensible data model and format (XDMF). In: *DoD high performance computing modernization program users group conference*, 18–21 June 2007, pp. 322–327. Washington, DC: IEEE Computer Society.

Considine DM and Considine GD (1999) *Van Nostrand's Scientific Encyclopedia*, 5th ed. New York: Wiley Producer.

Deelman E, Vahi K, Juve G, et al. (2015) Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* 46: 17–35.

Demmel J and Nguyen HD (2015) Parallel reproducible summation. *IEEE Transactions on Computers* 64(7): 2060–2070.

Dean J and Ghemawat S (2004) MapReduce: simplified data processing on large clusters. In: *Proceedings of the 6th conference on symposium on operating systems design & implementation – Volume 6*, San Francisco, CA, 6–8 December 2004, pp. 10–10. USENIX Association Berkeley.

De Oliveira D, Ocaña KACS, Baião F, et al. (2012) A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *Journal of Grid Computing* 10(3): 521–552.

Docan C, Parashar M, Cummings J, et al. (2011) Moving the code to the data – dynamic code deployment using active spaces. In: Sussman A (ed) *IEEE international parallel & distributed processing symposium*, Anchorage, Alaska, USA, 16–20 May 2011, pp. 758–769.

Dorier M, Dreher M, Peterka T, et al. (2015) Lessons learned from building in situ coupling frameworks. In: Bethel EW, Vishwanath V, Weber GH and Wolf M (eds) *First workshop on in situ infrastructures for enabling extreme-scale analysis and visualization*, Austin, TX, 16 November 2015, pp. 19–24.

Docan C, Parashar M and Klasky S (2010) DataSpaces: an interaction and coordination framework for coupled simulation workflows. In: Hariri S and Keahey K (eds) *Proceedings of the 19th ACM international symposium on high performance distributed computing*, New York, NY, USA, 2010, pp. 25–36. New York: ACM.

Dayal J, Bratcher D, Eisenhauer G, et al. (2014) Flexpath: type-based publish/subscribe system for large-scale science analytics. In: Raicu I (ed) *2014 14th IEEE/ACM international symposium on cluster, cloud and grid computing*, Chicago, IL, USA, 26–29 May 2014, pp. 246–255.

Deelman E, Blythe J, Gil Y, et al. (2004) Pegasus: mapping scientific workflows onto the grid. In: de Compostela S (ed) *Across grid conference*. Spain, 13–14 February, Berlin, New York: Springer.

Deelman E, Singh G, Su MH, et al. (2005) Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*. 13(3): 219–237.

Dreher M and Peterka T (2016) Bredala: semantic data redistribution for in situ applications. In: Dongarra J, Matsuoka S and King C-T (eds) *2016 IEEE international conference on cluster computing (CLUSTER)* Taipei, Taiwan, 13–15 September 2015, pp. 279–288.

Dayal J, Cao J, Eisenhauer G, et al. (2013) I/O Containers: managing the data analytics and visualization pipelines of high end codes. In: Bougé L and Hong B (eds) *IEEE 27th international symposium on parallel and distributed processing workshops and PhD forum*, Boston, Massachusetts, USA, 20–24 May 2013, pp. 2015–2024.

Eaton B, Gregory J, Drach B, et al. (2011) NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.6, 5 December. Available at: http://cfconventions.org

Ewa Deelman, et al. (2016) The Future of Scientific Workflows Report of the DOE NGNS/CS Scientific Workflows Workshop. U.S. Department of Energy, Office of Science, 20–21 April 2015.

Ekanayake J, Pallickara S and Fox G (2008) MapReduce for data intensive scientific analyses. In: Fox G (ed) *Proceedings of the 2008 fourth IEEE international conference on eScience*, Indianapolis, IN, USA, 10–12 December 2008, pp. 277–284.

Fox G, Qiu J and Jha S (2014) *High Performance High Functionality Big Data Software Stack*. Available at: http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf

Gamell M, Katz DS, Kolla H, et al (2014) Exploring automatic, online failure recovery for scientific applications at extreme scales. In: New Orleans, LA, USA, 16–21 November 2014, *International conference for high performance computing, networking, storage and analysis*, pp. 895–906. Piscataway, NJ: IEEE Press.

Garijo D and Gil Y (2012) *Towards Open Publication of Reusable Scientific Workflows: Abstractions, Standards, and Linked Data*. Pennsylvania: Citeseer.

Giardine B, Riemer C, Hardison RC, et al. (2005) Galaxy: a platform for interactive large-scale genome analysis. *Genome Research* 15(10): 1451–1455.

Goecks J, Nekrutenko A, Taylor J, et al. (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology* 11(8): R86.

Graves R, Jordan TH, Callaghan S, et al. (2011) CyberShake: a physics-based seismic hazard model for southern california. *Pure Applied Geophysics* 168(3): 367–381.

Gustafson JL (2015) *The End of Error: Unum Computing*. Boca Raton: CRC Press.

Habib S, Pope A, Finkel H, et al. (2016, January)HACC: Simulating sky surveys on state-of-the-art supercomputing architectures, *New Astronomy* 42: 49–65. Available at: http://dx.doi.org/10.1016/j.newast.2015.06.003

Jacob JC, Katz DS, Berriman GB, et al. (2010) Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. *Computing Research Repository*

Juve G, Chervenak A, Deelman E, et al. (2013) Characterizing and profiling scientific workflows. *Future Generation Computer Systems* 29(3): 682–692.

Jha S, Qiu J, Luckow A, et al. (2014) A tale of two data-intensive paradigms: applications, abstractions, and architectures. In: Hofmann P, Ming-Chien Shan M-C and Chou W (eds) *2014 IEEE international congress on big data*, Anchorage, AK, USA, 27 June–2 July 2014, pp. 645–652. IEEE Computer Society Press.

Khan F, Hammonds J, Narayanan S, et al. (2013) Effective end-to-end management of data acquisition and analysis for X-ray photon correlation spectroscopy. In: Marshall C, Fisher J and Schaa VRW (eds) *Presented at the ICALEPCS*. Available at: http://accelconf.web.cern. ch/AccelConf/ICALEPCS2013/

Kleese van Dam K, Stephan EG, Raju B, et al. (2015) *Enabling Structured Exploration of Workflow Performance Variability in Extreme-Scale Environments*. In: No. PNNL-SA-120941. Pacific Northwest National Laboratory (PNNL), Richland, WA, US.

Kahan W (1965) Pracniques: further remarks on reducing truncation errors. *Communications of the ACM* 8(1): 40.

LANL, NERSC and SNL (2016, July) APEX workflows. Available at: http://www.nersc.gov/assets/apex-workflows-v2.pdf, SAND2016-2371 O, LA-UR-15-29113

Lofstead JF, Klasky S, Schwan K, et al. (2008) Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In: Kim Y and Li XA (eds) *Proceedings of the 6th international workshop on challenges of large applications in distributed environments*, Boston, MA, USA, 23–27 June 2008, pp. 15–24. New York, NY, USA: ACM.

Logan J, Klasky S, Abbasi H, et al. (2012) Understanding I/O performance using I/O skeletal applications. In: Kaklamanis C, Papatheodorou T and Spirakis PG (eds) *Euro-Par 2012 parallel processing* 7484, Berlin, Heidelberg, 2012, pp. 77–88. Berlin: Springer

Lim C, Lu S, Chebotko A, et al. (2011) Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases. *Future Generation Computer Systems* 27(6): 781–789.

Lofstead JF, Klasky S, Schwan K, et al. (2008) Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In: Kim Y and Li XA (eds) *6th international workshop on challenges of large applications in distributed environments*, Boston, MA, USA, 23–27 June 2008, pp. 15–24. New York, NY, USA: ACM.

Moustakas D, Lang PT, Pegg S, et al. (2006) Development and validation of a modular, extensible docking program: DOCK 5. *Journal of Computer-Aided Molecular Design* 20(10–11): 601–619.

Moreau L, Clifford B, Freire J, et al. (2011) The open provenance model core specification (v1.1). *Future Generation Computer Systems* 27(6): 743–756.

Melnik S, Gubarev A, Long JJ, et al. (2010) Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment* 3(1–2): 330–339.

Muniswamy-Reddy KK, Braun U, Holland DA, et al. (2009) Layering in provenance systems. In: Voelker GM and Wolman A (eds) *Proceedings of the 2009 conference on USENIX annual technical conference*, San Diego, CA, 14–19 June 2009, pp. 10–10.

Malewicz G, Austern MH, Bik AJ, et al (2010) Pregel: a system for large-scale graph processing. In: Elmagarmid A and Agrawal D (eds) *Proceedings of the 2010 ACM SIGMOD international conference on management of data*, Indianapolis, IN, USA, 6–11 June 2010, pp. 135–146. New York, NY, USA: ACM.

Mathog DR (2003) Parallel BLAST on split databases. *Bioinformatics* 19(14): 1865–1866.

Maechling P, Deelman E, Zhao L, et al. (2006) SCEC CyberShake workflows—automating probabilistic seismic hazard analysis calculations. In: *Workflows for e-Science* (eds I Taylor,
E Deelman, D Gannon and M Shields) Berlin: Springer, pp. 143–163.

Meyer L, Mattoso M, Wilde M, et al. (2013) *WGL: A Workflow Generator Language and Utility*, Technical Report, University of Chicago.

Maheshwari K, Kelly D, Krieder SJ, et al. (2013) Reusability in science: from initial user engagement to dissemination of results. In: *3rd workshop on sustainable software for science: Practice and experiences (WSSSPE3)*, 28–29 September 2015, Boulder, CO.

Oinn T, Greenwood M, Addis M, et al. (2006) Taverna: lessons in creating a workflow environment for the life sciences: research articles. *Concurrency and Computation: Practice and Experience* 18(10): 1067–1100.

Parker SG and Johnson CR (1995) SCIRun: a scientific programming environment for computational steering. In: Karin S (ed) *Supercomputing, 1995. Proceedings of the IEEE/ACM SC95 conference*, San Diego, CA, USA, 4–8 December 1995, pp. 52–52. New York, NY: ACM.

Prabhat and Koziol Q. (2014) *High Performance Parallel I/O*. Chapman & Hall/CRC Computational Science. CRC Press. Available at: http://dl.acm.org/citation.cfm?id = 27 00549

Qui J, Jha S, Luckow A, et al. (2014) Towards HPC-ABDS: an initial high-performance big data stack. Presented at *the Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data*, SDSC.

Ramakrishnan L, Poon S, Hendrix V, et al. (2014) Experiences with user-centered design for the Tigres workflow API. In: *10th IEEE international conference on e-Science*, Guarujá, SP, Brazil, 20–24 October 2014, pp. 290–297. IEEE Computer Society Press.

Revol N and Theveny P (2014) Numerical reproducibility of high-performance computations using floating-point or interval arithmetic. Presented at *the Challenges in 21st Century Experimental Mathematical Computation*, Brown University, 21 July 2014.

Silva T, Freire J and Callahan SP (2007) Provenance for visualizations: reproducibility and beyond. *Computing in Science & Engineering* 9(5): 82–89.

Samak T, Gunter D, Goode M, et al. (2011) *Failure prediction and localization in large scientific workflows*. In: Taylor IJ and Montagnat J (eds) *Proceedings of the 6th Workshop on workflows in support of large-scale science*, Seattle, WA, USA, 12–18 November 2011, pp. 107–116. New York, NY, USA: ACM.

Scheidegger CE, Vo HT, Koop D, et al. (2008) Querying and re-using workflows with VsTrails. In: Lakshmanan LVS, Ng RT and Shasha D (eds) *ACM SIGMOD international conference on management of data*, Vancouver, Canada, 9–12 June 2008, pp. 1251–1254. New York, NY, USA: ACM.

Shasharina S, Cary J, Durant M, et al. (2010) VizSchema – a unified visualization of computational accelerator physics data. In: Noda A, Petit-Jean-Genaz C, Schaa VRW, et al. (eds) *1st international particle accelerator conference (IPAC 2010)*, Kyoto, Japan, 23–28 May 2010, p. TUPEC069. Geneva, Switzland: inPIRE.

Stodden V, McNutt M, Bailey DH, et al. (2016) Enhancing reproducibility for computational methods. *Science* 354(6317): 1240–1241.

Sorenson H (1985) *Kalman Filter: Theory and Applications*. Vol. 38. New York: IEEE Press.

Stonebraker M, Brown P, Poliakov A, et al. (2011) The architecture of SciDB. In: Cushing JB, French J and Shawn B (eds) *Proceedings of the 23rd international conference on scientific and statistical database management*, Portland, OR, USA, 20–22 July 2011, pp. 1–16. Berlin: Springer.

Tchoua R, Choi J, Klasky S, et al. (2010) ADIOS visualization schema: a first step towards improving interdisciplinary collaboration in high performance computing. In: *IEEE 9th international conference on e-Science*, Beijing, China, 22–25 October 2013, pp. 27–34. IEEE Press. DOI: 0.1109/eScience.2013.24.

Thakur R. Parallel I/O Benchmarks, Applications, Traces. May 2015. [Online]. Available at: http://www.mcs.anl.gov/~thakur/pio-benchmarks.html (accessed July 2016)

Talukder KA, Kirley M and Buyya R (2009) Multiobjective differential evolution for scheduling workflow applications on global Grids. *Concurrency and Computation: Practice and Experience* 21(13): 1742–1756.

Truong HL, Dustdar S and Fahringer T (2007) Performance metrics and ontologies for grid workflows. *Future Generation Computer Systems* 23(6): 760–772.

Teranishi K and Heroux MA (2014) Toward local failure local recovery resilience model using MPI-ULFM. In: Dongarra J, Ishikawa Y and Hori A (eds) *Proceedings of the 21st European MPI users' group meeting*, Kyoto, Japan, 9–12 September 2014, pp. 5151–5156. New York, NY, USA: ACM.

Thall A (2006) Extended-precision floating-point numbers for GPU computation. In: Finnegan J and McGrath M (eds) *ACM SIGGRAPH 2006 research posters*, Boston, MA, USA, 30 July–3 August 2006, New York, NY, USA: ACM.

Taufer M, Padron O, Saponaro P, et al. (2010) Improving numerical reproducibility and stability in large-scale numerical simulations on GPUs. In: *IEEE International symposium on parallel distributed processing (IPDPS)*, Atlanta, GA, USA, 19–23 April 2010, pp. 1–9. IEEE Computer Society Press.

Top Ten Exascale Research Challenges (2014) DOE ASCAC Subcommittee Report, Feb. Available at: https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf

Upson C, Faulhaber TA, Kamins D, et al. (1989) The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications* 9(4): 30–42.

Wilde M, Hategan M, Wozniak JM, et al. (2011) Swift: a language for distributed parallel scripting. *Parallel Computing* 37(9): 633–652.

Weerawarana S, Curbera F, Leymann F, et al. (2005) *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Upper Saddle River: Prentice Hall PTR.

Wilke J, Bennett J, Kolla H, et al. (2014) Extreme-scale viability of collective communication for resilient task scheduling and work stealing. In: Blough D (ed) *44th annual IEEE/IFIP international conference on dependable systems and networks*, 23–26 June 2014, pp. 756–761. IEEE Computer Society

Wolstencroft K, Haines R, Fellows D, et al. (2013) The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research* 41: W557–W561.

Wilde M, Foster I, Iskra K, et al. (2009) Parallel scripting for applications at the petascale and beyond. *Computer* 42(11): 50–60.

Workshop on the Future of Scientific Workflows (2015). [Online]. Available at: http://extremescaleresearch.labworks.org/events/workshop-future-scientific-workflows. (accessed September 2016).

Yelick K (2012) Algorithmic challenges of exascale computing. Presented at *the Synchronization-reducing and Communication-reducing Algorithms and Programming Models for Large-scale Simulations Workshop*, Brown University, January 2012.

Zheng F, Abbasi H, Cao J, et al. (2011) In-situ I/O processing: a case for location flexibility. In: Maltzahn C and Bent J (eds) *Proceedings of the sixth workshop on parallel data storage*, Seattle, WA, USA, 12–18 November 2011, pp. 37–42. ACM: New York, NY, USA

Zhao Y, Hategan M, Clifford B, et al. (2007) Swift: fast, reliable, loosely coupled parallel computation. In: Zhang LJ, Yang J and Hung PCK (eds) *IEEE congress on services*, Salt Lake City, Utah, USA, 9–13 July 2007, IEEE Computer Society.

Zheng F, Abbasi H, Docan C, et al. (2010) PreDatA – preparatory data analytics on peta-scale machines. In: Phillips C (ed) *IPDPS*'10, 19–23 April 2010, pp. 1–12. IEEE Computer Society.

Zhang Z and Katz DS (2014) Using application skeletons to improve eScience infrastructure. In: Cesar RM Jr. and Oliveira MCF (eds) *10th IEEE international conference on e-Science, eScience 2014*, Sao Paulo, Brazil, 20–24 October 2014, pp. 111–118. IEEE Computer Society.

Author biographies

Ewa Deelman is a research professor at the USC computer science department and the director of science automation technologies at the USC Information Sciences Institute. Dr. Deelman's research interests include the design and exploration of collaborative, distributed scientific environments, with particular emphasis on workflow management as well as the management of large amounts of data and metadata. In 2007, Dr. Deelman edited a book, *Workflows in e-Science: Scientific Workflows for Grids*, published by Springer. She is also the founder of the annual Workshop on Workflows in Support of Large-Scale Science, which is held in conjunction with the Super Computing conference. In 1997, Dr. Deelman received her PhD in computer science from the Rensselaer Polytechnic Institute.

Tom Peterka is a computer scientist at Argonne National Laboratory, fellow at the Computation Institute of the University of Chicago, adjunct assistant professor at the University of Illinois at Chicago, and fellow at the Northwestern Argonne Institute for Science and Engineering. His research interests are in large-scale parallelism for in situ analysis of scientific data. His work has led to three best paper awards and publications in ACM SIGGRAPH, IEEE VR, IEEE TVCG, and ACM/IEEE SC, among others. Peterka received his PhD in computer science from the University of Illinois at Chicago, and he currently works actively in several DOE- and NSF-funded projects.

Ilkay Altintas is the chief data science officer at SDSC, UC San Diego, where she is also the founder and director for the Workflows for Data Science Center of Excellence. Since joining SDSC in 2001, she has worked on different aspects of scientific workflows as a principal investigator across a wide range of cross-disciplinary NSF, DOE, NIH, and Moore Foundation projects. She is a co-initiator of the popular open-source Kepler Scientific Workflow System and the coauthor of publications related to computational data science and e-Sciences at the intersection of scientific workflows, provenance, distributed computing, bioinformatics, observatory systems, conceptual data querying, and software modeling. Ilkay is the recipient of the first SDSC Pi Person of the Year in 2014, and the IEEE TCSC Award for Excellence in Scalable Computing for Early Career Researchers in 2015. Ilkay Altintas received her PhD degree from the University of Amsterdam in the Netherlands with an emphasis on provenance of workflow-driven collaborative science and she is currently an assistant research scientist at UC San Diego.

Christopher D Carothers is a faculty member in the computer science department at Rensselaer Polytechnic Institute. He received the PhD from Georgia Institute of Technology in 1997. Prior to joining RPI in 1998, he was a research scientist at the Georgia Institute of Technology. His research interests are focused on massively parallel computing which involves the creation of high-fidelity models of extreme-scale networks and computer systems. These models have executed using nearly 2,000,000 processing cores on the largest leadership-class supercomputers in the world. Professor Carothers serves as the director for the Rensselaer Center for Computational Innovations (CCI). The center provides computation and storage resources to diverse network of researchers, faculty, and students from Rensselaer, government laboratories, and companies across a number of science and engineering disciplines. The flagship supercomputer is a 1 petaFLOP IBM Blue Gene/Q system with 80 terabytes of memory, 81,920 processing cores and over 2 petabytes of disk storage.

Kerstin Kleese van Dam is the director of the Brookhaven National Laboratory (BNL) Computational Science Initiative (CSI) that focuses in its core on data science-specific research at the extreme scale. Her research interests include large-scale data management, streaming data analysis, provenance, and reproducibility. She is currently involved in research projects in workflow performance analysis (ASCR IPPD) and exascale in situ data reduction and analysis workflows (ASCR ECP CODAR).

Kenneth Moreland is a principal member of the technical staff at Sandia National Laboratories. His research interests include in situ visualization and large-scale, finely threaded scientific visualization algorithms. Dr. Moreland has a PhD in computer science from the University of New Mexico.

Manish Parashar is distinguished professor of computer science at Rutgers University, and founding director of the Rutgers Discovery Informatics Institute (RDI2). He is also founding chair of the IEEE Technical Consortium on High Performance Computing (TCHPC). His research interests are in the broad areas of Parallel and Distributed Computing and Computational and Data-Enabled Science and Engineering. Manish is fellow of AAAS, fellow of IEEE/IEEE Computer Society, and ACM distinguished scientist. For more information, please visit http://parashar.rutgers.edu/.

Lavanya Ramakrishnan is a staff scientist at Lawrence Berkeley National Lab. Her research interests are in software tools for computational and data-intensive science. Ramakrishnan has previously worked as a research staff member at Renaissance Computing Institute and MCNC in North Carolina. She has masters and doctoral degrees in computer science from Indiana University and a bachelor degree in computer engineering from VJTI, University of Mumbai. She joined LBL as an Alvarez postdoctoral fellow in 2009.

Michela Taufer is a JPMorgan Chase scholar in the department of computer and information sciences and an associate professor in the same department at the University of Delaware. She earned her doctoral degree in computer science from the Swiss Federal Institute of Technology (Switzerland). In 2003–2004, she was a La Jolla Interfaces in Science Training Program Postdoctoral Fellow at the University of California San Diego and The Scripps Research Institute, where she worked on interdisciplinary projects in computer systems and computational chemistry. Taufer's research focuses on high-performance computing (HPC) including scientific applications; performance analysis, modeling, and optimization of multi-scale applications on heterogeneous computing, cloud computing, and volunteer computing; numerical reproducibility and stability of large-scale simulations on multi-core platforms; data analytics and MapReduce.

Jeffrey Vetter, is a distinguished R&D staff member at Oak Ridge National Laboratory (ORNL). At ORNL, Vetter is the founding group leader of the Future Technologies Group in the Computer Science and Mathematics Division. Vetter also holds joint appointments at the Georgia Institute of Technology and the University of Tennessee-Knoxville. Vetter earned his PhD in computer science from the Georgia Institute of Technology. Vetter is a fellow of the IEEE and a distinguished scientist member of the ACM. In 2010, Vetter, as part of an interdisciplinary team, was awarded the ACM Gordon Bell Prize. Also, his work has won awards at major conferences including Best Paper Awards at IPDPS and EuroPar, and Best Presentation at EASC 2015. Vetter served as the SC15 Technical Program Chair. His recent books, entitled *Contemporary High Performance Computing: From Petascale toward Exascale* (Vols. 1 and 2), survey the international landscape of HPC. For more information, see his website http://ft.ornl.gov/~vetter/.