Enabling Structured Exploration of Workflow Performance Variability in Extreme-scale Environments

Kerstin Kleese van Dam*
Pacific Northwest National Laboratory
902 Batelle Boulevard
Richland, WA, 99352, USA
+15093717797
Kerstin.Kleesevandam@pnnl.gov

Eric Stephan
Pacific Northwest National Laboratory
902 Batelle Boulevard
Richland, WA, 99352, USA
+15093756977
Eric.Stephan@pnnl.gov

Bibi Raju
Pacific Northwest National Laboratory
902 Batelle Boulevard
Richland, WA, 99352, USA
+15093753623
Bibi.Raju@pnnl.gov

Ilkay Altintas
San Diego Supercomputer Center
9500Gilman Drive
La Jolla, CA, 92093-0505, USA
+18588225453
Altintas@sdsc.edu

Todd Elsethagen
Pacific Northwest National Laboratory
902 Batelle Boulevard
Richland, WA, 99352, USA
+15093754431
Todd.Elsethagen@pnnl.gov

Sriram Krishnamoorthy
Pacific Northwest National Laboratory
902 Batelle Boulevard
Richland, WA, 99352, USA
+15093726963
Sriram@pnnl.gov

ABSTRACT

Workflows are taking an Workflows are taking an increasingly important role in orchestrating complex scientific processes in extreme scale and highly heterogeneous environments. However, to date we cannot reliably predict, understand, and optimize workflow performance. Sources of performance variability and in particular the interdependencies of workflow design, execution environment and system architecture are not well understood. While there is a rich portfolio of tools for performance analysis, modeling and prediction for single applications in homogenous computing environments, these are not applicable to workflows, due to the number and heterogeneity of the involved workflow and system components and their strong interdependencies. In this paper, we investigate workflow performance goals and identify factors that could have a relevant impact. Based on our analysis, we propose a new workflow performance provenance ontology, the Open Provenance Model-based WorkFlow Performance Provenance, or OPM-WFPP, that will enable the empirical study of workflow performance characteristics and variability including complex source attribution.

General Terms

Algorithms, Measurement, Documentation, Performance, Standardization, Theory.

Keywords

Workflow, Performance, Provenance, Extreme-Scale

INTRODUCTION

High-performance computing (HPC), Grid and Cloud users are looking increasingly toward workflow solutions to orchestrate their complex application coupling, pre- and post-processing needs. Their growing popularity is particularly visible at workflow sharing sites such as MyExperiment [1] or Galaxy [2-4]. Workflows can be implemented through simple scripts, work schedulers [27] or sophisticated workflow management systems such as Kepler[22], Pegasus [40], Swift[23], Galaxy[19-21], VisTrails[24], Taverna[25] or Medici[26]. Many workflow systems have the ability to coordinate tasks spread across many different systems, indeed some offer the ability to adapt and optimize task placement at runtime [22, 40]. Higher level workflow systems hereby separate the composition of the interrelated tasks into scientific workflows, from the actual execution of specific workflow instances in a particular execution environment, and as such can help to lower the barriers to using complex infrastructures. For exascale systems with their complex memory, I/O and processor architectures, the ability to easily coordinate simulations, in-situ analysis, external processing and data access tasks [5, 6] through the usage of workflows, could play an important role in enabling more scientists to master their complexity and harness their capabilities. The reliability of the performance delivered by the workflow systems will hereby play a key role in their adoption.

Unfortunately, while workflows are in regular use today, the community lacks the ability to study, predict and optimize the performance of complex workflows, in particular in extreme-scale environments. There are no performance analyses or modeling tools available, and existing performance studies are usually centered around one off studies of specific use cases in a specific execution environment, rather than structured, analytical analysis of whole classes of workflows.

For single computational applications key performance limiting and influencing factors have been widely studied [28,29,30,31,32], and performance analysis [36,37] and modeling [30,33] tools and techniques can provide key insights into potential bottlenecks or sources of performance variability (i.e. strong reliance on I/O). These insights have been used to devise algorithms [38,39] and design templates [34,35] that enable scientists to create codes with reliable performance characteristics. OS, Runtime and Resource Management systems for extreme scale systems including exascale are optimized to support single application performance, based on the fundamental understanding of their performance characteristics. Unfortunately these capabilities have not been brought to bear on complex workflows, because these differ from single applications executed in one homogenous environment in a number of significant ways:

- Workflows combine several applications with potentially quite different programming models, data models, execution and performance characteristics that can show strong interdependencies in their execution behavior.
- Are often executed on more than one system, each with quite different architectures, execution environments, policies, performance and load characteristics, and are not under the control of a single institution.
- They regularly utilize shared resources such as file systems, archives or wide area networks, as such their performance can be strongly influenced by the availability of resources and the behavior of other users.
- Utilize a workflow management system that can adjust execution behavior at runtime based on rules or user input.

Ultimately it is the number and heterogeneity of the involved workflow and system components and their strong interdependencies that provide key challenges in understanding workflow performance variability. Furthermore their ability to react to hardware, systems, software and user created events at runtime, adds another level of complexity. Consider the following simple exemplary use case and its potential performance challenges:

We have three computational models, utilizing (MPI and PGAS) as their programming models, the codes are coupled through a workflow. Their input data is streamed from an external data archive server, as it cannot be held on the system in its entirety. As the models produce significant output an automated analysis step, utilizing a map reduce programming paradigm, has also been added. This analysis model requires access to externally held observational data for its validation component and selects and accesses this data based on the results received from the modeling programs. As the time steps of the three models are in sync, the underperformance of one of them for any reason or time interval will impact the overall workflow, bad load balancing, resilience measures, slow data access to the input data due to low rate storage or network connections would have the same effect. As all models are run at the same time utilizing the same storage and memory they could create their own contention challenges. A workflow system might need to halt the majority of the applications while it restarts on application affected by a hard, soft or silent error. The underlying runtime system might decide to reduce the processor speed to save energy, just as the workflow prepares to send more work to it to balance the overall workflow performance better, leading to a slowdown of all components and underutilization of their resources.

As can be seen, even simple workflows, such as the one described above, include many potentially performance influencing factors that would need to be analyzed and modeled. We suggest that extended provenance capture systems could provide a novel approach to providing performance monitoring capabilities that correlate information across system levels and systems itself. Initially this would be envisaged as a post hoc analysis, but the approach could be extended to provide runtime information. Furthermore the task of modeling and simulating all possible workflow and systems components that could influence workflow performance in one integrated system at any level of detail would be a formidable computational task in its own right. We propose that empirical studies based on the collected workflow performance provenance information could help to provide critical guidance on the importance and impact of specific factors for selected classes of workflows. Thus providing guidance and focus to the modeling and simulation activities. In addition these studies can illuminate key interdependencies between workflow design, execution environment, executables' computation models and system architecture.

In this paper, we investigate workflow performance goals and identify factors that could have a relevant impact. Based on our analysis, we propose a new workflow performance provenance ontology, Open Provenance Model-based WorkFlow Performance Provenance, or OPM-WFPP, that will enable the structured capture and subsequent evaluation of workflow performance characteristics and variability including complex source attribution. In addition we will introduce the basic architecture of the ProvEn provenance capture, storage and analysis infrastructure.

1. WORKFLOW PERFORMANCE GOALS

Workflow performance is commonly defined as "execution time" for a particular workflow instance—that is, the wall clock time it takes from the start of the workflow to its successful conclusion. Other variations on this theme are "always-on" workflows, which act upon data as it arrives. In which case, execution time is the time it takes for a specific data element (record, file, or data set) to pass through the workflow pipeline. In addition, response time may be considered as a performance goal: the time it takes from the request for a workflow execution to its completion. This can be a particularly important factor in time-critical decision making processes. In extreme-scale computing environments, energy use is a growing concern; therefore, low energy consumption can be another workflow performance goal.

In commercial computing systems, users need to pay for service use, making lowering monetary costs another workflow performance goal. There are concerns that also might be considered under this goal, such as optimized use of resources, which is important when available compute time, data and

network resources are limited and completion of a predefined volume of work depends on effective use of the available resources. Highly distributed workflows, workflows in cloud environments, or workflows for extreme-scale computing resources (e.g., U.S. Department of Energy [DOE]-owned Leadership Class Computing Facilities (LCFs)) are expected to encounter increasing amount of failure events during execution. Thus, reliability and resilience also may be considered additional facets of workflow performance.

Next to workflow performance dimensions related to its execution properties, scientific users also are concerned with more quality-specific performance measures, such as accuracy, reproducibility, and adaptability. It is important to involve these types of criteria in an overall performance investigation and optimization effort as they can lead to important trade-off decisions, for example: *am I prepared to give up some of my accuracy or adaptability for a faster result* or *what are the costs of user interactions in terms of speed and the quality of the scientific outcome*?

Thus, workflow performance might be characterized as subset or combination of the following features:

- Execution Time
- Total Response Time
- Resource Use
- Energy Use
- Reliability
- Resilience
- Reproducibility
- Accuracy

It is clear that many users will pursue more than one performance optimization goal, and they need to study the trade-offs in a multidimensional design space. For example, de Olievera et al. [7] define performance in a cloud environment in a three-objective cost model, considering execution time, financial cost, and reliability. Khaled Ahsan Talukder et al. [8] use quality of service parameters to define performance, such as execution time, cost, and total data transmission time. In the following we investigate which observable factors are relevant in the context of different performance goals.

2. FACTORS RELEVANT TO WORKFLOW PERFORMANCE

To study workflow performance, we first and foremost need to understand the different sources of workflow performance variability and under performance as they pertain to specific performance goals, such as speed, energy efficiency or resource usage. Our key interest is to identify patterns across different workflow classes that cause performance degradation, particularly in extreme-scale environments. As workflows have the ability to adapt at runtime, it is similarly important to investigate suitable strategies in response to performance impacting events. The ability to identify, describe and prioritize all components that could have an impact on workflow performance is fundamental to this endeavor. In the following subsections, we review existing workflow performance description mechanisms used in studies of workflow performance and suggest a number of new extensions that will be required to capture a comprehensive picture of performance-relevant components and their interdependencies.

2.1 Basic Workflow Descriptions

Workflows are commonly described as a type of graph, e.g., as directed acyclic graphs [6], incorporating the applications or tasks present in the workflow and their connection with each other, such as "A creates input for B." In addition, input and output (I/O) data sets are represented as components of the workflow. Few of the available workflow performance languages capture the workflow management system that was used to orchestrate the tasks [7], [9], [10]. This is of particular import as the workflow management system itself can introduce an additional layer of tasks and may influence overall performance by: a) introducing a certain level of overhead and b) impacting performance based on its scheduling and adaptation policies. Thus, we suggest that the workflow management system needs to be part of the overall workflow description when performance is being investigated.

2.2 Execution Environment and Systems Architectures

However to date, none of the workflow description languages utilized for performance investigations provides any mechanisms to describe either the systems software or hardware components employed in the workflow execution. With this we neglect to account for performance variability due to different hardware solutions (CPUs, storage, or networking), the impact of protocols and runtime systems on performance and resource contention or the influence system load fluctuations can have. Furthermore it is important to correlate the system descriptions with the workflow itself, so that subsequent performance studies can identify the root causes of performance degrading events and assess their impact and knock-on effects on the overall workflow performance. In consequence we propose to make the description of the execution environment and system architecture part of the core workflow description (Figure 1).

Key performance influencing factors are the execution environments and system architectures on which the workflow will be executed.

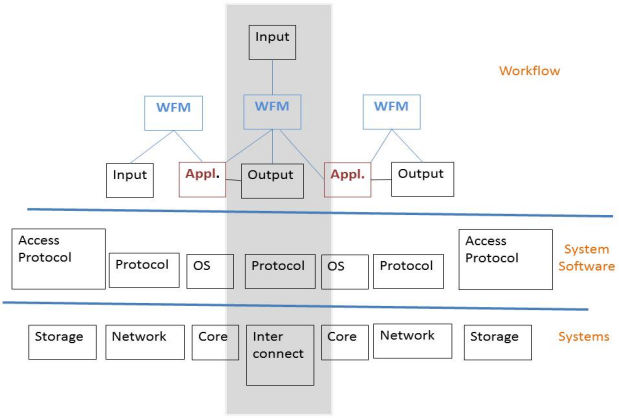

Figure 1: Comprehensive Workflow Ecosystem Representation for Performance Studies

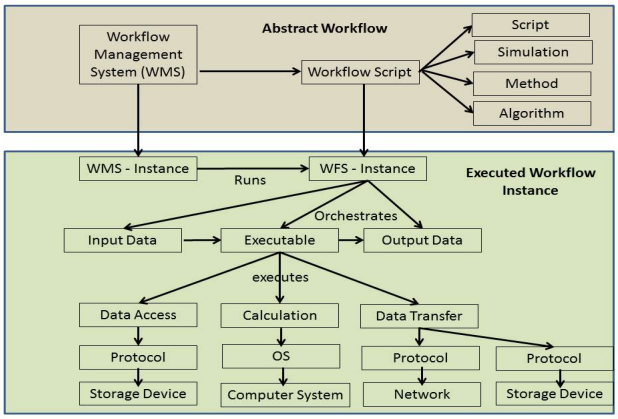

Figure 2: Principle Workflow Representation

2.3 Performance Relevant Workflow Description Details

Components involved in a workflow must be represented in sufficient detail to capture their key characteristics and potential sources of performance variability.

2.3.1 Workflows

As workflows are performed more than once and likely in changing execution environment and on different hardware platforms, we propose to introduce a basic abstraction layer in the workflow description. An abstract description of the overall workflow provides hereby an overview of all possible execution pathways; this is linked to one or more descriptions of executed workflow instances. These workflow instances might vary in their description, depending on the rule based decisions taken during its execution, however they will all be a subset of the overall abstract workflow. This workflow instance description is then linked to the actual execution environment and system architectures used during the workflow execution (Figure 2).

The abstract description will characterize the workflow management system (name, type, version) and abstract workflow scripts that can be executed by the workflow management system, e.g., input data *x* is used by simulation model Community Land Model, which produces output data of type *b*.

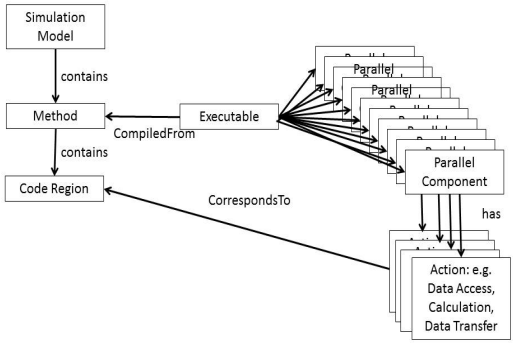

Figure 4: Representations of Parallel Programs

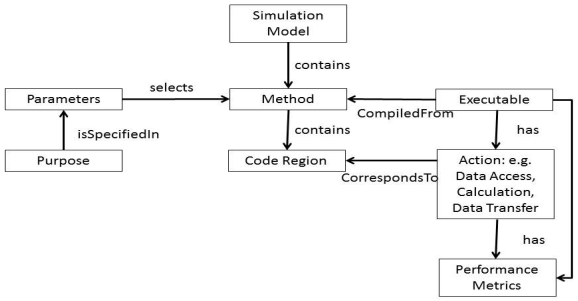

Figure 3: Simulation Model Representation

The executed workflow instance is much more detailed in its description, noting specific versions of applications used, how they were created (parameters, compiler, etc.), their programming model, the scientific methods encapsulated in them, and where they were executed. The executed workflow instance is the key construct that connects workflows, applications, system-level software, and systems.

For the performance evaluation, it is helpful to treat executables not as a single entity but to divide them into code regions with specific predominant actions as they are observed by the system level (i.e., data access, calculation, or data transfer). Similarly the utilized systems need to be described in terms of their architecture and key performance characteristics, operating and runtime system, compilers and libraries. Another aspect currently not depicted in Figure 2 are disruptive events that change the execution of the workflow during runtime. These could be system-level events (e.g., disk failure, core failure, network outage) or user-created events (e.g., kill workflow as simulation is not converging or choose a different analysis path). These types of events would be captured at both the workflow script instance system software or hardware level.

2.3.2 Scientific Simulation Models

Scientific workflows often center around simulation models. These are complex computational models that represent physical, biological, and chemical processes. Many of these codes have been developed over a long time and consist of a variety of theoretical methods. These methods can be combined in different ways to serve the investigation of a specific scientific purpose. Not all methods available are necessarily involved in each model run. Each method usually is developed on its own timescale and will have a separate version tree from other methods. A simulation model release version will contain a number of methods, each with its own version. Method version, method combination, and scientific purpose will have a critical impact on the overall model performance characteristics and need to be captured. In turn, methods have code regions that correspond to specific actions in a subsequent executable compiled from these methods. These code regions may contain actions, such as access to disk storage, numerical calculations, data transfer, or communication. It is important to identify these actions and corresponding code regions to track their specific performance, as well as the effects of changes to them (Figure 3).

2.3.3 Workflow Description for Extreme -scale Applications and Environments

Most workflow descriptions treat each application integrated in a workflow as one unit, which is associated with a set of properties, such as average memory usage. However, in highly parallel codes, each application component, executed on a different core or thread could display different performance behavior (e.g. due to uneven work distribution) or experience different events (e.g., memory corruption). Due to interdependencies between different workflow components (across applications) it is imperative to consider each parallel component as an individual contributor to the overall workflow performance and describe them separately (see Figure 4).

In addition we need to provide the ability to capture direct dependencies between specific applications (Figure 5), where performance degradation in one parallel component or in the interaction could have widespread implications. Hereby differences in programming models should also be noted as these could have a significant impact in the communication and data exchange overhead.

2.4 *Workflow as Time Series*

The combination of applications, resource requirements, and performance behavior can change significantly during the execution of a workflow, as can its execution environment and the load characteristics of the involved systems. When studying workflow performance variability, it is not enough to look at statistics for the overall workflow run. We need to capture the changing characteristics through time, meaning a time series of snapshots of correlated workflow and system topology and behavior are required (Figure 6).

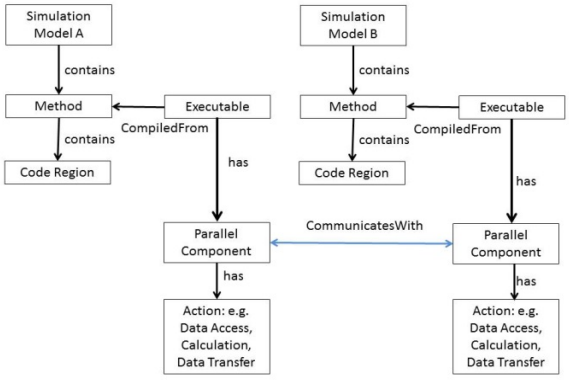

Figure 5: Capturing Interdependencies between Parallel Components at Runtime

2.5 Events Related to Workflow Performance

These can originate from the systems or execution environment the workflow is operating on, the user, the applications, or the workflow system itself. In each case, we need to capture these events, attribute them to their source, and describe their impact. For example, contention on required resources will lead to a slowdown of the workflow, so the workflow system might react by scheduling time on additional resources to overcome this bottleneck. Similarly, the workflow might experience hardware

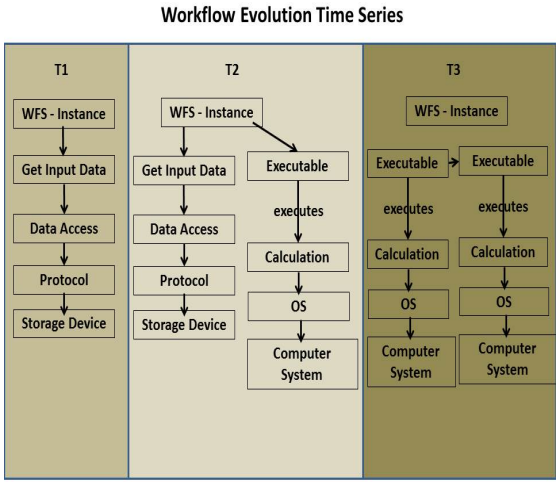

Figure 6: Workflow Evolution Time Series

power throttling events, temperature events, and fault events, leading to an unexpected slowdown in parts of a workflow. Other events at the software level might include dynamic load balancing, idle times, waiting for communication or synchronization. External events (non program/runtime controlled) could be power capping, work migration in response to fault prediction, etc. Workflow management systems are expected to increase their capabilities to adapt and optimize at runtime, therefore any action taken by the workflow system will need to be recorded and the impact of the actions studied. Hereby it will be of particular interest to observe correlations between system actions (power throttling) and workflow management system actions (resource rescheduling). For the future it is hoped that a direct interaction between the execution environment and workflow management system will be possible, enabling systems and workflows to exchange their goals and synchronize their behavior. These types of negotiations would also need to be captured and evaluated in their impact on overall workflow performance. It is further expected that users will need more direct control over their workflows to adapt them at runtime to unexpected scientific results that need specialized treatment. In those cases the user could make changes to the workflow execution pathway and thus due introduce performance influencing actions.

As such, a system describing workflow performance must capture events and event types, as well as link their source and potential actions taken. A key challenge to this involves the potential delays between the original event, its impact being noticed, and an action taken. For instance, a silent error that causes a problem in one of the workflow components that is not immediately visible but grows over time. Once it reaches a certain threshold, the affected algorithm takes mitigating actions in form of a checkpoint restart.

3. WORKFLOW PERFORMANCE METRICS

Once a sufficient structure is in place to describe workflows and their environment, provisions have to be made to link these descriptions with specific observed performance metrics. We can distinguish these metrics into two categories: 1) metrics directly associated with the workflow (i.e., execution time and energy use

of a specific application) and 2) environmental metrics (i.e.overall system load on Lustre file system) [17]. Most currently available studies only capture performance information directly associated with the workflow. Furthermore, we often find that existing workflow performance investigations (e.g., [9], [10]) tend to capture summary statistics such as overall I/O reads, peak memory usage, etc., but they do not—as we intend —capture performance metrics as time series information [18] linked to specific workflow phases, components, and events. Independently a number of initial provenance systems have been developed to capture specific system level performance characteristics, such as: provenance aware storage [43,44], distributed storage [45], file system [50], distributed file systems [49], kernel [46] and networks [47,48]. Few solutions do however cover multiple aspects and levels of the operating system. Hi-Fi [46] is one of the most comprehensive conceptual approaches, and offers a kernel level approach to trace the data flow through systems, processes and threads, across files and file systems, memory mappings, pipes, message queues and sockets. It does however not collect any application specific computational performance metrics or information about applications and workflows themselves. Other approaches [41-43] also aim to facilitate provenance capture across different system layers, however none goes as deeply into the operating and storage systems layers as Hi-Fi. To date any these systems have only been tested in small scale, single system environments.

Based on our previously defined performance goals (see Section 1) and prior research by others, we will define metrics for the relevant runtime components of the workflow and its execution environment in terms of execution time, resource use, energy use, reliability, and accuracy. Metric categories will include:

- Workflow Script Performance Determining Characteristics (incl. e.g. number of tasks)
- Workflow Script Instance Performance Metrics (incl. some outlined in [9],[10])
- Code Region Performance Metrics to be collected for each call to the code region, for each core, (selected list of metrics informed by [9], [10])
- Computer System Performance Characteristics (incl. [17])
- Computer System Performance Metrics (as collectable by e.g. SYSSTAT [18])
- Wide Area Network Performance Characteristics
- Wide Area Network Performance Metrics
- Interconnect Performance Characteristics
- Interconnect Performance Metrics
- Storage System Performance Characteristics
- Storage System Performance Metrics:

Further metrics might be introduced to capture system utilization and costs.

4. OPM-BASED PROVENANCE MODEL TO COMPREHENSIVELY CAPTURE WORKFLOW PERFORMANCE

To establish a better understanding of workflow performance, we need to routinely capture empirical information from instrumented workflow runs in a standardized form, providing detailed performance sensitivity studies or observing workflow performance during development cycles or operational use. The analysis of the empirical information captured not only across variations of the same workflow but across classes of workflows with differing characteristics and performance goals can help optimize the design of reliably performing workflows. Furthermore, we can use these data to identify early indicators for performance degradation at runtime, as well as successful methodologies to mitigate the impact of such occurrences.

In the past, much performance information was gathered in a more ad hoc fashion, especially because many studies were small and could be analyzed manually. More extensive studies relied on data structure designs that were purely focused on the direct measurements of performance data [7], [8], [10], [11]. Truong et al. [9] developed an ontology to describe and capture workflow performance metrics on five different levels: workflow, workflow region, activity, invoked application, and code region. However, their work did not include performance information for any of the utilized system components or influencing factors.

We suggest that for a more in-depth study of workflow performance, particularly as part of a larger empirical study, we need a more extensive data model. There are three different types of information to capture: 1) static descriptions of the components involved in the workflow execution, 2) slow changing description of component characteristics specific to a particular workflow instance, and 3) fast evolving metrics captured during a specific workflow execution. The information required for the former two types of information typically is captured as provenance information, whereas metrics can be described, for example, in a performance metrics ontology. In our novel approach, we propose linking the two to create a comprehensive workflow performance data model.

5.1 Workflow Performance Provenance

In 2008, the International Provenance and Annotation Working Group (IPAW) defined the core specification for an Open Provenance Model [12] to provide an extensible provenance model that can be used to exchange and integrate provenance captured in different provenance models implementations. We built our solution on this standard. There are a number of existing workflow provenance data model implementations based on the OPM, such as OPMW [13], D-OPM [16], or the work of Lim et al. [15]. All of these models focus exclusively on the workflow graph and do not describe the workflow components or their performance-determining properties in sufficient detail for a performance evaluation beyond time spent in a particular application. Moreover, they do not include system or system software level information. We have reviewed the existing models and deemed them not suitable in their layout to be extended to our much more detailed new data model, OPM-WFPP.

For this model, we will create new subclasses for existing OPM classes:

- "Agent: Agent is a contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, or affecting its execution," to describe the basic characteristics of the underpinning systems, system software, compilers, algorithms, tools and scripts.
  - SubClass: Workflow – subclasses: Workflow Management System, Workflow Script
  - SubClass: Application – subclasses: Algorithm, Script, Simulation Model, Code Region, Method, Parallel Component

SubClass: System – subclasses: Computer Systems, Network, Storage Device
SubClass: System Software – subclasses: Compiler, Operating System, Protocol, Scheduler

- "Artifact: Artifact is a general concept that represents immutable piece of state, which may have a physical embodiment in an object or a digital representation in a computer system," to describe a specific version of the underpinning hardware and software stack.
  SubClass: Configuration
  SubClass: Data – subclasses: Input Data, Output Data, Parameters, Performance Metrics
  SubClass: Descriptors – subclasses: Location, Programming Language, Programming Model, Scientific Domain, Version
  SubClass: Purpose
- "Process: Process refers to an action or series of actions performed on or caused by artifacts and resulting in new artifacts." To describe a specific version of an algorithm, tool, or script, we will create a subclass, "application instance," that will describe properties of the instance, such as version number, compiler flags, parameter, etc., and link directly to system software component artifacts, such as compiler or runtime system.
  SubClass: Action – subclasses: Calculation, Communication, Data Transfer, Data Access
  SubClass: Decision Points – subclasses: Data Driven Change, User Driven Change
  SubClass: Events – subclasses: Hard Error, Soft Error, Silent Error, Runtime System Optimization, Workflow Optimization at Runtime
  SubClass: Executable
  SubClass: Outcome – subclasses: Failure, Success with further subclasses of: Accuracy, Error Range, Uncertainty, Validity
  Subclass: State
  SubClass: Workflow Script Instance
  SubClass: Workflow Management System Instance

In addition, we will add a number of additional properties to those originally available in OPM (including examples of their use):

Workflow Definitions

- wasDefinedIn => Workflow Script *wasDefinedIn* Workflow Management System
- Orchestrates => Workflow Script *Orchestrates* Algorithm
- facilitatesUseOf => Workflow Script *facilitatesUseOf* Computer System

Semantic Description for Specific Domain Instances

- isDependentOn => CESM Executable *isDependentOn* Data Assimilation Algorithm
- communicatesWith => Shyre Analysis Executable *communicatesWith* OPA Executable
- Follows => Diagnostics package A Executable *follows* CESM Executable

During Execution

- hasInstance => Workflow Script *hasInstance* Workflow Script Instance
- hasVersion => Workflow Script *hasVersion* Version
- interactsWith => Workflow Script Instance *interactsWith* Scheduler
- wasPerformedBy => Workflow Script Instance *wasPerformedBy* Workflow Management System Instance
- wasControlledBy => Executable *wasControlledBy* Workflow Script Instance
- hasEvents => Workflow Script Instance *hasEvents* Decision Points
- hasDecissionType => Decision Point *hasDecissionType* Data Driven Change
- hasEventType => Events *hasEventType* Silent Error

Executable

- hasDomain => Simulation Model *hasDomain* Scientific Domain
- hasProgrammingLanguage => Algorithm *hasProgrammingLanguage* Programming Language
- hasProgrammingModel => Simulation Model *hasProgrammingModel* Programming Model
- contains => Simulation Model *contains* Methods
- isSpecifiedIn => Purpose *isSpecifiedIn* Parameters
- selects => Parameters *selects* Methods
- isMadeUpOf => Methods *isMadeUpOf* Code Region
- correspondsTo => Action *correspondsTo* Code Region
- wasCompiledWith => Executable *wasCompiledWith* Compiler
- wasCreatedFrom => Executable *wasCreatedFrom* Simulation Model

At Runtime

- drives => Parameters *drives* Executable
- wasExecutedOn => Executable *wasExecutedOn* Computer System
- used => Executable *used* Input Data
- created => Executable *created* Output
- hasOutcome => Executable *hasOutcome* Validity
- hasPerformanceMetrics => executable *hasPerformanceMetrics* Performance Metrics
- includesAction => Executable *includesAction* Data Transfer

Link to System

- isExecutedIn => Data Transfer *isExecutedIn* Protocol
- wasUsedOn => Protocol *wasUsedOn* Network
- hasErrors => Storage Device *hasErrors* Hard Errors
- canCause => Storage Device *canCause* Silent Error
- linksTo => Network *linksTo* Computer System

An initial ontology has been created for OPM-WFPP and is available on request.

5. PROVEN

The ProvEn capture, storage, and analysis nfrastructure consists of two principle components: the OPM- WFPP library and the ProvEn hybrid provenance servers. The two components communicate via standard messaging services as they are available in the execution environments incl. web services calls, ZeroMQ and Apache Kafka. The system is records provenance events at any system or application level and correlates these based on time and context. This approach enables us not only to study all events relevant to a particular workflow, but we can also

investigate the behavior e.g. of specific system components when utilized by a variety of workflows at the same time or follow the performance behavior of specific workflow tasks across systems and workflow classes (e.g. data transfer between workflow tasks).

The OPM-WFPP libraries are customizable libraries that can be integrated into workflows, workflow components, and system level applications and middleware. They are bundled with the ProvEn OPM-WFPP client API and a pre-registered list of structured messages, known as provenance messages used to record performance metrics. Where it is not possible to integrate the library into a key performance relevant component e.g. a network service, ProvEn will utilize its existing provenance harvesting routines to collect provenance from system level log files and applications.

The ProvEn Hybrid Server component provides a central location for the storage and collection of provenance messages that have been created by ProvEn client libraries. It is a Java EE application managing an embedded Sesame RDF repository, where all collected provenance data are stored. Web services are made available by the ProvEn Server allowing clients to register themselves with the server for exchange of their provenance messages. The data interchange format used is JSON for Linking Data (JSON-LD), it provides a simple Java API binding used by the server to perform serialization of provenance messages from JSON-LD directly into RDF for storage into ProvEn's RDF repository. All collected provenance is compartmentalized into RDF sub-graphs based on the client producing the provenance. The time series based performance metrics will be captured via the ProvEn InfluxDB services. Its integration with the ProvEn RDF repository enables the correlated capture of time series and core provenance information (see Figure 7).

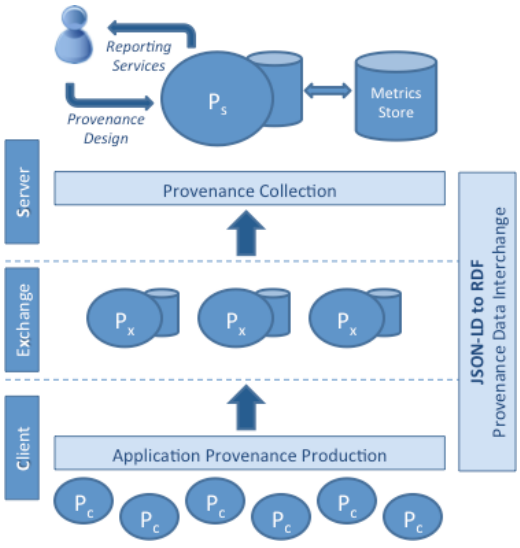

Figure 7: Conceptual depiction of ProvEn architecture.

To evaluate the initial ProvEn implementation, we are replicating a set of experiments on a highly instrumented cluster [42]. SeaPearl is a cluster with 52 nodes and instrumented with sensors that include temperature and power usage (based on the paper I provided earlier in the week). Our initial tests are repeating benchmark tests using Firestarter, a stress test tool that ensures that a constant high workload generates the maximum amount of heat that can be generated by a CPU or thermal design power (TDP) limit. For our tests Firestarter [41] is run on two nodes. Because SeaPearl is instrumented, temperature and power sensors for those two nodes are monitored at 10KHz using a Lua script running on each node that pipes streaming measurements in parallel into the InfluxDB database. Each recorded time series measurement record consists of the cluster name, node name (e.g. numbers "1"-"52"), the instrument name tag, the timestamp, and the measured floating point value. To correlate performance measures in the time series database to the provenance store a timestamp key is needed. To keep the timestamp consistent across measurements being collected across the cluster the Network Time Protocol (NTP) is relied upon as the time source. The cluster name (which for our purposes is assumed), cluster node name, and instrument name form a semantically meaningful tag name that relate to a type of performance measurements. At 10KHz each hour 36 million records are collected, for each cluster node sensor.

ProvEn's semantic store is used for collecting referential information providing details regarding:

- Benchmark tests being performed,
- Conditions of the system (cold start, successive tests etc)
- Engineering units used,
- Sensors
- Collection rate
- Codes used in the test
- Versions of the codes being used.

During the experiment ProvEn's semantic store is used for collecting:

- Start/end time of test on each node.
- When lua scripts begin/end collection on each node.
- Stages of the experiment as the stress tests occur.

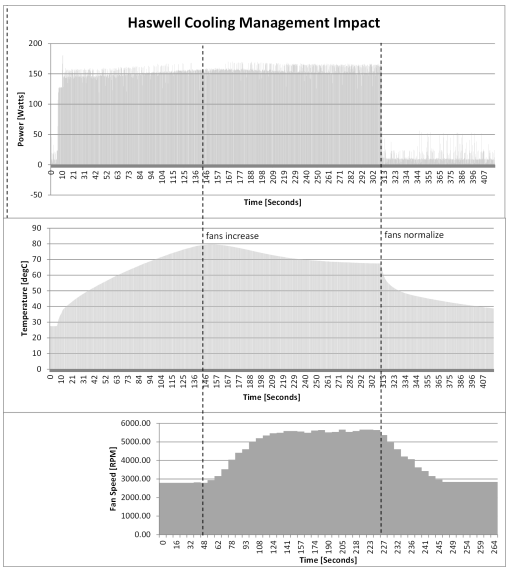

Figure 8: Depiction of affects of Firestarter stress test in initial experiments [42].

From this composite information ProvEn reporting services can monitor collection streams at collection time and provide reports either while the benchmark tests are running or post mortem. Figure 8 depicts reporting information that will be available through ProvEn reporting services.

6. FUTURE WORK

The OPM-WFPP components and ProvEn infrastructure have been implemented and undergone initial tests. We expect initial empirical study results from the SeaPearl evaluation benchmark tests in late autumn of 2015. In parallel the team is working on integrating the performance assessment process into four distinct use cases: Streaming analysis of experimental data, a complex climate modeling workflow based on Pegasus for the DOE BER ACME project (in collaboration with USC), a biological workflow based on Kepler and a high energy physics high throughput workflow for the Belle II experiment.

7. ACKNOWLEDGMENTS

The research described in this paper was funded by the DOE's Office of Advanced Scientific Computing Research Integrated End-to-End Performance Prediction and Diagnosis for Extreme Scientific Workflows (IPPD) project and, in part, by the Analysis In Motion Initiative at Pacific Northwest National Laboratory (PNNL), which is conducted under PNNL's Laboratory Directed Research and Development Program. PNNL is operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

8. REFERENCES

[1] Heinis, T.; Alonso, G. 2008. Efficient lineage tracking for scientific workflows. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1007-1018. DOI: 10.1145/1376616.1376716.

[2] Anand, M. K.; Bowers, S.; McPhillips, T.; Ludäscher, B. 2009. Efficient provenance storage over nested data collections. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* , pp. 958-969. DOI: 10.1145/1516360.1516470.

[3] Anand, M. K.; Bowers, S.; Ludäscher, B. 2010. Techniques for efficiently querying scientific workflow provenance graphs. In *Proceedings of the 13th International Conference on Extending Database Technology*, Vol. 10, pp. 287-298. DOI: 10.1145/1739041.1739078.

[4] Ikeda, R.; Park, H.; Widom, J. 2011. Provenance for generalized map and reduce workflows. In *Online Proceedings, Fifth Biennial Conference on Innovative Data Systems Research (CIDR)*, pp. 273-283.

[5] Park, H.; Ikeda, R.; Widom, J. 2011. RAMP: A System for Capturing and Tracing Provenance in MapReduce Workflows. In *Proceedings of the VLDB Endowment*, 4(12), 1351-1354.

[6] Ogasawara, E.; Dias, J.; Oliveira, D.; Porto, F.; Valduriez, P.; Mattoso, M. 2011. An algebraic approach for data-centric scientific workflows. In *Proceedings of the VLDB Endowment*, 4(12), 1328-1339.

[7] de Oliveira, D.; Ocana, K. A. C. S.; Baiao, F.; Mattoso, M. 2012. A Provenance-based Adaptive Scheduling Heuristics for Parallel Scientific Workflows in Clouds. *J. Grid Comput*. 10(3):521-552. DOI: 10.1007/s10723-012-9227-2.

[8] Khaled Ahsan Talukder, A. K. M.; Kirley, M.; Buyya, R. 2009. Multiobjective differential evolution for scheduling workflow applications on global Grids. *Concurrency Computat.: Pract. Exper.* 21(13):1742-1756. DOI: 10.1002/cpe.1417.

[9] Truong, H. L.; Dustdar, S.; Fahringer, T. 2007. Performance metrics and ontologies for Grid workflows. *Fut. Gener. Comput. Syst*. 23(6):760-772. DOI: 10.1016/j.future.2007.01.003.

[10] Juve, G.; Chervenack, A.; Deelman, E.; Bharathi, S.; Mehta, G.; Vahi, K. 2013. Characterizing and profiling scientific workflows. *Fut. Gener. Comput. Syst*. 29(3):682-692. DOI: 10.1016/j.future.2012.08.015.

[11] Samak, T.; Gunter, D.; Goode, M.; Deelman, E.; Mehta, G.; Silva, F.; Vahi, K. 2011. Failure Prediction and Localization in Large Scientific Workflows. In *Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science*, pp. 107-116. DOI: 10.1145/2110497.2110510.

[12] Moreau, L.; Clifford, B.; Freire, J.; Futrelle, J.; Gil, Y.; Groth, P.; Kwasnikowska, N.; Miles, S.; Missier, P.; Myers, J.; Plale, B.; Simmhan, Y.; Stephan, E.; Van den Bussche, J. 2011. The Open Provenance Model core specification (v1.1). *Fut. Gener. Comput. Syst*. 27(6):743-756. DOI: 10.1016/j.future.2010.07.005.

[13] Garijo, D.; Gil, Y. 2012. Towards Open Publication of Reusable Scientific Workflows: Abstractions, Standards and Linked Data. *Internal Project Report*. Accessed on March 23, 2015 at: http://www.isi.edu/~gil/papers/garijo-gil-opmw12.pdf.

[14] Missier, P.; Dey, S.; Belhajjame, K.; Cuevas-Vicenttín, V.; Ludäscher, B. 2013. D-PROV: Extending the PROV provenance model with workflow structure. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance (TaPP '13)*, Article 9, 1-7.

[15] Lim, C.; Lu, S.; Chebotko, A.; Fotouhi, F. 2010. Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases. *Fut. Gener. Comput. Syst*. 27(6):781-789. DOI: 10.1016/j.future.2010.10.013.

[16] Cuevas-Vicenttín, V.; Dey, S.; Wang, M. Y.; Song, T.; Ludäscher, B. 2012. Modeling and querying scientific workflow provenance in the D-OPM. In *High Performance Computing, Networking, Storage and Analysis (SCC)*, 2012 SC Companion Technology, pp. 119-128. DOI: 10.1109/SC.Companion.2012.27.

[17] Burtscher, M.; Kim, B.; Diamond, J.; McCalpin, J.; Koesterke, L.; Browne, J. 2010. PerfExpert: An Easy-to-Use Performance Diagnosis Tool for HPC Applications. In *Proceedings of the 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, pp. 1-11. DOI: 10.1109/SC.2010.41.

[18] SYSSTAT. Accessed on March 23, 2015 at: http://sebastien.godard.pagesperso-orange.fr/features.html.

[19] Blankenberg, D., et al.2001. Galaxy: A Web-Based Genome Analysis Tool for Experimentalists, in Current Protocols in Molecular Biology. *John Wiley & Sons, Inc*.

[20] Giardine, B., et al. 2005. Galaxy: A platform for interactive large-scale genome analysis. In *Genome Research*, 2005. 15(10): p. 1451-1455.

[21] Goecks, J., et al. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. In *Genome Biology*, 2010. 11(8): p. R86.

[22] Lud, B., et al. 2006. Scientific workflow management and the Kepler system: Research Articles. In *Concurr. Comput. : Pract. Exper.*, 2006. 18(10): p. 1039-1065.

[23] Wilde, M., et al. 2011. Swift: A language for distributed parallel scripting. In *Parallel Comput*., 2011. 37(9): p. 633-652.

[24] Scheidegger, C.E., et al. 2008. Querying and re-using workflows with VsTrails. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, ACM: Vancouver, Canada. p. 1251-1254.

[25] Wolstencroft, K., et al. 2013. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. In *Nucleic Acids Research*, 2013.

[26] Chase, J., et al. 2009. Kepler + MeDICi Service-Oriented Scientific Workflow Applications. In *Proceedings of the 2009 Congress on Services - I. 2009*, IEEE Computer Society. p. 275-282.

[27] Tsaregorodtsev, A. 2014. DIRAC Distributed Computing Services. In *Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013). Journal of Physics: Conference Series* 513 (2014) 032096. DOI: 10.1088/1742-6596/513/3/032096.

[28] Kerbyson, D. J., Alme, H. J., Hoisie, A., Petrini, F., Wasserman, H. J., & Gittings, M. 2001, November. Predictive performance and scalability modeling of a large-scale application. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing* (pp. 37-37). ACM.

[29] Williams, S., Waterman, A., & Patterson, D. 2009. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4), 65-76.

[30] Nudd, G. R., Kerbyson, D. J., Papaefstathiou, E., Perry, S. C., Harper, J. S., & Wilcox, D. V. 2000. PACE—A toolset for the performance prediction of parallel and distributed systems. *International Journal of High Performance Computing Applications*, 14(3), 228-251.

[31] Ipek, E., De Supinski, B. R., Schulz, M., & McKee, S. A. 2005. An approach to performance prediction for parallel applications. In *Euro-Par 2005 Parallel Processing* (pp. 196-205). Springer Berlin Heidelberg.

[32] Vuduc, R., Demmel, J. W., & Bilmes, J. 2001. Statistical models for automatic performance tuning. In *Computational Science—ICCS 2001* (pp. 117-126). Springer Berlin Heidelberg.

[33] Tallent, N. R., & Hoisie, A. 2014, June. Palm: easing the burden of analytical performance modeling. In *Proceedings of the 28th ACM international conference on Supercomputing* (pp. 221-230). ACM.

[34] Nishtala, R., Vuduc, R. W., Demmel, J. W., & Yelick, K. A. 2007. When cache blocking of sparse matrix vector multiply works and why. *Applicable Algebra in Engineering, Communication and Computing*, 18(3), 297-311.

[35] Yotov, K., Roeder, T., Pingali, K., Gunnels, J., & Gustavson, F. 2007, June. An experimental comparison of cache-oblivious and cache-conscious programs. In *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures* (pp. 93-104). ACM.

[36] Tiwari, A., Gamst, A., Laurenzano, M. A., Schulz, M., & Carrington, L. 2014. Modeling the Impact of Reduced Memory Bandwidth on HPC Applications. In *Euro-Par 2014 Parallel Processing* (pp. 63-74). Springer International Publishing.

[37] Solomonik, E., Carson, E., Knight, N., & Demmel, J. 2014, June. Tradeoffs between synchronization, communication, and computation in parallel linear algebra computations. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures* (pp. 307-318). ACM.

[38] Ballard, G., Buluc, A., Demmel, J., Grigori, L., Lipshitz, B., Schwartz, O., & Toledo, S. 2013, July. Communication optimal parallel multiplication of sparse random matrices. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures* (pp. 222-231). ACM.

[39] Carson, E., Knight, N., & Demmel, J. 2013. Avoiding communication in nonsymmetric Lanczos-based Krylov subspace methods. *SIAM Journal on Scientific Computing*, 35(5), S42-S61.

[40] Deelman, E., Singh, G., Su, M., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G., Good, J., Laity, A., Jacob, J., Katz, D.2005. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems, *Scientific Programming Journal*, 13:3, pp. 219-237, 2005.

[41] Hackenberg, Daniel, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. "An Energy Efficiency Feature Survey of the Intel Haswell Processor." (2015).

[42] Getov VS, DJ Kerbyson, MC Macduff, and A Hoisie. 2015. "Towards an Application-Specific Thermal Energy Model of Current Processors." In 3rd International Workshop on Energy Efficient Supercomputing Held in conjunction with SC15: The International Conference for High Performance Computing, Networking, Storage and Analysis *.(in press)*