PANORAMA: An Approach to Performance Modeling and Diagnosis of Extreme Scale Workflows

Ewa Deelman[1], Christopher Carothers[4], Anirban Mandal[3], Brian Tierney[2], Jeffrey Vetter[5], Ilia Baldin[3], Claris Castillo[3], Gideon Juve[1], Dariusz Król[1,6], Vickie Lynch[5], Ben Mayer[5], Jeremy Meredith[5], Thomas Proffen[5], Paul Ruth[3], Rafael Ferreira da Silva[1]

[1]Information Sciences Institute, University of Southern California
[2]Lawrence Berkeley National Laboratory
[3]RENCI/UNC Chapel Hill
[4]Rensselaer Polytechnic Institute
[5]Oak Ridge National Laboratory
[6] AGH University of Science and Technology, Department of Computer Science, Krakow, Poland

Abstract

Computational science is well established as the third pillar of scientific discovery and is on par with experimentation and theory. However, as we move closer toward the ability to execute exascale calculations and process the ensuing extreme-scale amounts of data produced by both experiments and computations alike, the complexity of managing the compute and data analysis tasks has grown beyond the capabilities of domain scientists. Thus, workflow management systems are absolutely necessary to ensure current and future scientific discoveries. A key research question for these workflow management systems concerns the performance optimization of complex calculation and data analysis tasks. The central contribution of this article is the PANORAMA approach for modeling and diagnosing the run-time performance of complex scientific workflows. This approach integrates extreme-scale systems testbed experimentation, structured analytical modeling and parallel systems simulation into a comprehensive workflow framework called Pegasus for understanding and improving the overall performance of complex scientific workflows.

Keywords. Performance Modeling, Extreme Scale, Scientific Workflow.

1 Introduction

Modern science often requires the processing and analysis of vast amounts of data in search of postulated phenomena and the validation of core principles through the simulation of complex system behaviors and interactions. Not surprisingly, this trait is common in fields as diverse as astronomy, bioinformatics, physics, and ocean and ice sheet modeling.

In order to support the computational and data needs of today's science, new knowledge must be gained on how to deliver the growing, distributed and high-performance computing capabilities to the scientist's desktop in an accessible, reliable and scalable way. As applications and infrastructure are growing in scale and complexity, our understanding of application behavior is not keeping up with technological and algorithmic advances. As a result, it is difficult for scientists and infrastructure providers to determine the expected and realistic behavior of a complex scientific workflow. Furthermore, when failures or performance anomalies are encountered, it is extremely difficult to pinpoint the root cause of the problem, and to correct or mitigate the issue.

Many science applications today are composed of custom scripts that tie several community or custom codes together. These scripts are frequently edited by hand to suit each new computational platform or to expand the parameter and data sets they operate on. The computing infrastructure that these applications run on is very complex and diverse. It includes NSF-funded systems, such as Open Science Grid [1], XSEDE [2], and cloud-based infrastructures [3, 4], DOE computing facilities at Oak Ridge National Laboratory, Lawrence Berkeley National Laboratory, and others. Input, output, and intermediate data are transferred over high-speed national and regional transit networks like Internet2 and ESnet.

Given the complexity of applications and infrastructures, modeling their behavior is very tedious and sometimes infeasible. Part of the solution for understanding the behavior of the applications is to formalize their structure in the form of a workflow. Workflows are able to declaratively express complex applications that are composed of several individual codes with data and control dependencies. A workflow management system can take this application description and manage and optimize its execution on a variety of computational platforms.

Tools can be created that analyze the workflow and develop models of expected behavior given a particular computing environment, such as an HPC system, clusters distributed over wide area networks, or clouds. Having a coupled model of the application and execution environment, decisions can be made about resource provisioning, application task scheduling, data management within the application, etc. As the application is executing, correlated real-time monitoring of the application and infrastructure can be used to verify the application behavior, to detect and diagnose faults, and to support adaptivity.

This paper describes the PANORAMA [5] project that aims to further our understanding of the behavior of scientific workflows as they are executing in heterogeneous environments. The central contribution of this article is the PANORAMA approach for modeling and diagnosing the run-time performance of complex scientific workflows. This approach integrates extreme-scale systems testbed experimentation, structured analytical modeling and parallel systems simulation into a comprehensive workflow framework called Pegasus for understanding and improving the overall performance of complex scientific workflows.

The paper is organized as follows: Section 2 describes applications that are motivating our work. Section 3 describes the Pegasus workflow management system that manages the application execution. Section 4 describes the tools and techniques we use to capture and profile the observed behavior of workflows. Section 5 shows our approach to analytical workflow modeling. Section 6 describes our approach to correlating the real time application and infrastructure monitoring data, while Section 7 explores anomaly detection based on that data. Section 5.3 describes how analytical models can be augmented with detailed simulations. Section 9 describes related work. Section 10 presents the work ahead and concludes the paper.

2 Motivating Examples

We have identified two important application use-cases involving advanced workflows that are the initial focus of our modeling efforts: parameter refinement workflows for the Spallation Neutron Source (SNS) and climate simulation automation for the Accelerated Climate Modeling for Energy (ACME) project.

2.1 Spallation Neutron Source (SNS)

The Spallation Neutron Source (SNS) [6] is a DOE research facility at Oak Ridge National Laboratory that provides pulsed neutron beams for scientific and industrial research. SNS uses a particle accelerator to impact a mercury-filled target with short proton pulses to produce neutrons by the process of spallation. A wide variety of experiment instruments provide different capabilities for researchers across a broad range of disciplines, including: physics, chemistry, materials science, and biology.

SNS hosts hundreds of researchers every year who conduct experiments within short reservations of a few days to a few weeks. Providing these researchers with efficient, user-friendly and highly configurable workflows that reduce the turnaround time from data collection to analysis and back is essential to the success of SNS. Figure 1 shows the data flow for a typical SNS instrument, in this case NOMAD. Neutron events scattered from the scientific sample under investigation are collected by an array of detectors. These raw events are processed into a representation familiar to the domain scientist depending on the type of experiment. For NOMAD, the reduced form is a powder diffraction pattern. This reduced data is then analyzed and compared to materials simulations to extract scientific information.

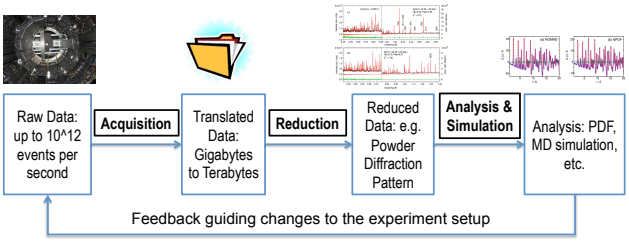

Figure 1: SNS example workflow.

In addition to workflows for processing data from SNS experiments, there are also workflows for data analysis and simulation to support and guide SNS experiments, and to validate computer models against

experimental data. These workflows automate tedious manual processes to reduce time to solution and improve researcher productivity. In collaboration with the Center for Accelerating Materials Modeling (CAMM) of SNS data, we are adapting a workflow that executes simulations to support experimental design and the validation of molecular models as a use case for the PANORAMA project. The workflow executes an ensemble of molecular dynamics and neutron scattering simulations to optimize a model parameter value. This workflow is being used to investigate temperature and hydrogen charge parameters for models of water molecules. The results are compared to data from QENS experiments using the BASIS instrument at SNS.

An illustration of this parameter refinement workflow is shown in Figure 2. For each set of parameters the workflow executes 5 batch jobs. First, each set of parameters is fed into a series of parallel molecular dynamics simulations using NAMD [7]. The first simulation calculates an equilibrium and the second inputs that equilibrium and calculates the production dynamics. Each NAMD simulation runs on 288 cores for 1 to 6 hours. The output from the MD simulations has the global translation and rotation removed using AMBER's [8] ptraj utility [9] and is passed into Sassena [10] for the calculation of coherent and incoherent neutron scattering intensities from the trajectories. Each Sassena simulation runs on 144 cores for up to 6 hours. The final outputs of the workflow are transferred to the user's desktop and loaded into Mantid [11] for analysis and visualization.

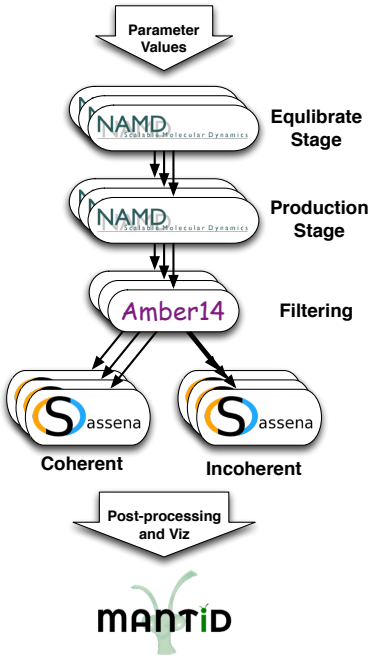

Figure 2: The SNS refinement workflow executes a parameter sweep of molecular dynamics and neutron scattering simulations to optimize the value for a target parameter to fit experimental data.

2.2 Accelerated Climate Modeling for Energy (ACME)

The Accelerated Climate Modeling for Energy (ACME) project is using coupled models of ocean, land, atmosphere and ice to study the complex interaction between climate change and societal energy requirements. One of the flagship workflows of this effort is the fully-coupled climate model running at high resolution.

The complete workflow for ACME is illustrated in Figure 3. The ACME climate modeling effort involves the interaction of many different software and hardware components distributed across computing resources at several DOE laboratories. As part of the ACME project, many of the workflow activities that were previously done manually are being automated. The goal is to have an automated, end-to-end workflow that provides full data provenance.

One important step towards that goal is to automate the small portion of the full workflow that involves running the ACME climate model. The PANORAMA project is developing a workflow that automates the manual effort involved in monitoring and resubmitting the model code in case of failures, and provides periodic reporting for validation of science outputs. The workflow, illustrated in Figure 4, divides a large climate simulation into several stages. Each stage completes a portion of the total target simulation time. For example, a 40-year simulation may be divided into 8, 5-year stages. This enables each stage of the workflow to be completed within the maximum walltime permitted for batch jobs on the target DOE leadership class computing systems. Restart files generated at the end of each stage are used as input to the next stage in order to continue the simulation. Each stage also produces history files, which are used by the workflow to automatically compute summary data called *climatologies*. This climatology data can be reviewed periodically by project scientists to ensure that the simulation is progressing as expected, so that problems can be identified, and corrections made, before computing resources are wasted. Both the history files and the climatologies are transferred to HPSS and CADES (open infrastructure) for long-term storage and future analysis.

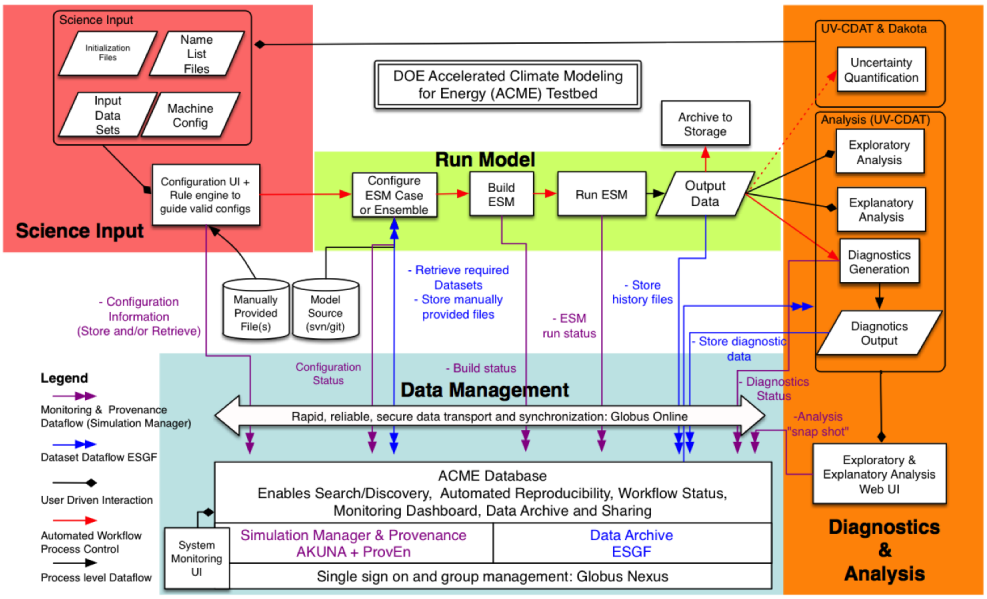

Figure 3: The complete Accelerated Climate Modeling for Energy (ACME) includes many interacting components distributed across DOE labs.

3 Workflow Execution

In order to facilitate workflow creation, scientists need to be able to formulate the workflows in a way that is meaningful to them, in a resource-independent way, using high-level abstractions to specify the structure of the analysis and the data to be operated on (via a visual or textual representation). This abstract workflow (or workflow instance) is important because it uniquely identifies the analysis to be conducted at the application level, without including operational details of the execution environment. The workflow instance can be published along with the results of a computation to describe how a particular data product was obtained. This approach supports reproducibility, a cornerstone of the scientific method. In order to support the use of abstract workflow specifications, planning technologies are needed to automatically interpret and map user-defined, abstract workflows onto the available resources.

Workflow technologies have been demonstrated to be very effective in exploiting coarse grain parallelism in applications running on distributed infrastructures such as grids and clouds [12]. Our system, Pegasus [13], focuses on scalable, reliable and efficient workflow execution on a wide range of systems, from user's desktops to leadership class machines [14, 15]. The cornerstone of our approach is the separation of the workflow description from the description of the execution environment, which results in: 1) workflows that are portable across execution environments, and 2) the ability for the workflow management system to make performance- and reliability-focused decisions at "compile time" and/or at "runtime". Pegasus pioneered the use of planning in scientific workflow systems [16]. It takes a resource-independent, or abstract, workflow description, automatically locates the input data and computational resources necessary for workflow execution, maps/plans this description onto the available execution resources, then reliably executes the plan. When errors occur, Pegasus tries to recover when possible by retrying tasks, by retrying the entire workflow, by providing workflow and task-level checkpointing, by re-planning portions of the workflow, by trying alternative data sources for staging data, and, when all else fails, by providing a rescue workflow containing a description of only the work that remains to be done. A Pegasus workflow can process millions of tasks and terabytes of data, so storage management is important. Pegasus has a sophisticated model for reasoning about and optimizing data transfers and it cleans up storage as the workflow is executed so that data-intensive workflows have enough space to execute on storage-constrained resources [17].

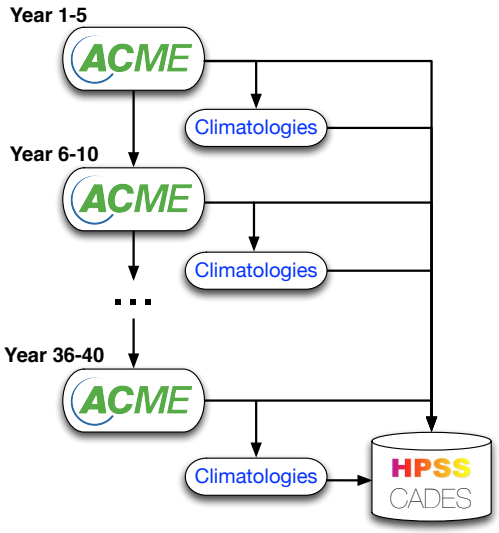

Figure 4: The ACME workflow runs one climate simulation in several stages. The output of each stage is used to compute climatologies for validation. All outputs are stored in HPSS and CADES for archiving and further analysis.

4 Characterizing Workflows

Workflow behavior modeling and analysis methods need accurate execution event traces and monitoring information to build models of system behavior and conduct anomaly detection and diagnosis. Here, we describe some of the tools and techniques we are using in PANORAMA.

Workflow Monitoring. Workflow-level monitoring collects information about events in the execution of a workflow that are critical to understanding its behavior. The STAMPEDE [18] framework collects monitoring data by parsing and correlating log files that are generated by the workflow management system. These logs contain information about the submit time of the jobs, the time when the jobs are sent to the remote system for execution, and the time when they are reported finished or failed. The logs also have information about the progress of the workflow through time, for example, when the data dependencies have been satisfied. This data can be used to compute runtimes, queue delays, and other important performance metrics.

Task Monitoring and Profiling. Task-level monitoring can be used to measure and predict the resource requirements of workflows. In addition to workflow-level data, the STAMPEDE framework also collects information about the execution of workflow tasks on remote resources using Kickstart [19], a monitoring tool launched along with the computation and data management tasks to collect information about the behavior of the tasks and their execution environment. As part of the DOE dV/dt project [20] we added functionality to Kickstart to automatically capture resource usage metrics of workflow tasks [21]. This functionality uses operating system monitoring facilities as well as system call and library call interposition to collect fine-grained profile data that includes process I/O, file accesses, runtime, memory usage, and CPU utilization. Using this information, we developed techniques based on machine learning algorithms for automatically modeling and predicting the resource usage of workflow tasks [22], such as runtime, disk usage and memory consumption. This approach combined with a Monitor Analyze Plan Execute (MAPE-K) autonomic computing loop [23], is used for online estimation of task needs as the workflow executes. Experimental results show that this online process produces more accurate estimates than a comparable offline method. Table 1 shows average estimation errors using this approach for a bioinformatics application (Epigenomics).

Application Monitoring. As part of the PANORAMA project we will develop the capability to automatically capture monitoring data generated by workflow tasks, such as progress information, performance metrics, and diagnostic messages.

Infrastructure Monitoring. The previous monitoring techniques are not sufficient for modeling the behavior of scientific workflows. Additional information about the infrastructure is also required to understand the observed performance of workflows. Network performance monitoring using tools such as perfSONAR [24], for example, is crucial to discover "soft failures" in the network, where the network seems to be up, but is performing at just a fraction of it's peak efficiency, which translates to poor performance for data-intensive workflows. perfSONAR instances can run continuous checks for latency changes and packet loss, and run periodic throughput tests using a suite of already available tools like BWCTL, OWAMP, and NDT. There are currently around 1,200 perfSONAR instances worldwide, providing us with critical network performance information.

Task	Estimation	Runtime Avg. Error (%)	I/O Write Avg.Error (%)	Memory Avg.Error (%)
fastqSplit	Offline	8.36	3.28	9.14
	Online	8.36	3.28	9.14
filterContams	Offline	59.31	109.81	102.83
	Online	29.13	5.35	8.15
sol2sanger	Offline	54.93	98.20	96.68
	Online	34.74	1.23	1.96
fast2bfq	Offline	27.13	128.18	99.98
	Online	17.09	15.11	10.65
map	Offline	23.62	0.00	21.07
	Online	1.39	0.00	3.33
mapMerge	Offline	53.74	93.34	1.01
	Online	10.22	9.39	1.00
pileup	Offline	6.00	4.17	49.42
	Online	5.11	3.87	19.31

Table 1: Average estimation errors of task runtime, I/O write, and peak memory usage for the Epigenomics workflow.

5 Analytical Models

5.1 Performance modeling with Aspen

Our **Aspen** system [25] was designed to bridge the gap between *structured analytical models* and *functional simulation*. In this work, we are extending the Aspen modeling framework to include workflow scenarios and resources to be able to develop structure analytical models for scientific workflows. We are extending Aspen's domain specific language (DSL), which was initially designed to create analytical models of traditional scientific applications and HPC architectures, to scientific workflows.

Aspen's DSL approach to analytical performance modeling provides several advantages over traditional approaches. For instance, Aspen's `kernel` construct helps to fully capture control flow, and preserves more algorithmic information than traditional frameworks [25]. Aspen's model also captures the important concepts of data capacities and data movement in very explicit terms. Similarly, the abstract machine model is more expressive than frameworks that reduce machine specifications to a small set of parameters. The formal language specification forces scientists to construct models that can be syntactically checked and consumed by analysis tools. This formal specification also facilitates collaboration between domain experts and computer scientists, and enables scientists to include application specific parameters in their model definitions, which would otherwise be difficult to infer. Finally, Aspen is modular, and therefore it is easy to compose, reuse, and extend performance models. With these features, Aspen can help answer important application-specific questions.

Aspen is complementary to other performance prediction techniques including simulation [26, 27], emulation, or measurement on early hardware prototypes. Compared to these techniques, Aspen's analytical model is machine-independent, has fewer prerequisites (e.g., architectural descriptions, application source code), and can be computed very efficiently.

Performance Modeling of Workflows

Many workflow systems have been integrated with performance monitoring and analysis tools [28–30]. These systems typically collect only coarse-grained information, such as task runtime and data size. Using Kickstart enables us to collect fine-grained profiles including I/O, memory, CPU usage, and runtime data for use in performance modeling.

Most workflow models focus on performance prediction of individual computational tasks. Data transfer and management tasks that are present in workflows are either not modeled, or are modeled very simplistically. Thus, existing performance predictions are not realistic and not detailed enough to help scientists and infrastructure providers estimate application performance, and pinpoint and diagnose issues. Profiling data, which is automatically collected during workflow execution and includes detailed runtime and resource usage statistics, can be used to create a performance model of the workflow, which is able to generate these estimates. In addition, profiling data can be used to guide resource provisioning algorithms that require estimates of resource usage [31–33].

We have been extending Aspen to include application features and resources necessary for workflows.

Compute resources. Aspen was originally designed with the capability to model large scale scientific applications running on traditional HPC systems (e.g., Titan, with hybrid CPU/GPU architecture, or Mira with the Blue Gene/Q architecture). Current Aspen constructs model computation, memory access, communication, and data size. Workflows represent a much broader set of resources, which include large storage systems of varying capability and capacity, high speed networks, and a variety of instruments and distributed computing systems and services.

Networks. One area of focus for our modeling will be networks, particularly *wide area networks*, which are often used during workflow execution. Wide area communication models must deal with network protocol issues. For example, TCP throughput, and thus data transfer throughput, is directly impacted by packet loss. Newer protocols and long haul fiber

networks will need to be represented in Aspen.

Storage systems. For data-intensive applications, storage system (e.g., GPFS, HPSS, Lustre) performance has a significant impact on overall workflow performance. The increased data movement and storage performance requirements created by large workflows can heavily impact overall workflow performance and, thus, will need to be modeled by Aspen.

5.2 Automatic Generation of Aspen Performance Models from Pegasus Descriptions

One major benefit to our approach is symmetry in representations of Pegasus workflows and Aspen performance models. In fact, we can use the structured Pegasus framework to generate Aspen performance models automatically. Because Aspen is a structured language that mirrors concepts (e.g., subroutines, loops) in traditional languages like C and Java, it is straightforward to generate performance models from other structured representations. We have started the implementation of a post-processing phase during the Pegasus compilation that will use the Pegasus intermediate abstract workflow representation to construct an Aspen model. (The infrastructure model is different from the workflow model, and is developed in a later time.)

In our initial work, we have developed a DAX (Pegasus abstract workflow format) to Aspen generator, which maps the dependencies in the workflow to an Aspen control flow sequence. In Aspen, statements within a `par` clause execute independently, i.e., with task parallelism, and statements within a `seq` clause must be executed in order, i.e., they have sequential dependencies. Our initial scheduler performed a wavefront-like algorithm, grouping each set of tasks with completed dependencies into a `par` clause, repeating within one master sequential region until all tasks completed. Because Pegasus operates at the level of individual tasks, however, this schedule was only accurate if all tasks within a group took the same amount of time. As such, we implemented an advanced scheduler, which is a more direct translation of the dependencies and more accurately matches Pegasus scheduling behavior.

Figure 2 shows an example DAX developed for a SNS workflow. It contains two sequences with different initial conditions within a single workflow, with the joint requirement that a database file must first be unpacked for Sassena to operate. Listing 1 shows the results as an Aspen kernel, with two major steps. The first step starts unpacking the Sassena database while initiating each pair of sequential equilibrium and production dynamics NAMD calculations. The second major step performs the Sassena runs in parallel with each other after ptraj run completes.

```
kernel main
{
   par {
      call unpack_database()
      seq {
         call namd_eq_200()
         call namd_prod_200()
      }
      seq {
         call namd_eq_290()
         call namd_prod_290()
      }
   }
   par {
      call ptraj_200()
      call ptraj_290()
   }
   par {
      call sassena_incoh_200()
      call sassena_coh_200()
      call sassena_incoh_290()
      call sassena_coh_290()
   }
}
```

Listing 1: Automatically-generated Aspen model for example SNS workflow

5.3 Evaluating the Use of Analytical Models

Validation of a performance model is a difficult task. The most obvious validation step would be to use the model to generate a performance prediction, and compare the performance against a measured value. This strategy has weaknesses, generally stemming from one important fact: performance is machine-dependent, and so cannot effectively disambiguate the correctness of our generated application/workflow models from that of the machine model. This means that you must generate multiple predictions and measurements in an attempt to disambiguate not only across machines, but across scales, and across scaling rates.

In the case of Aspen, two facets work in our favor. First, Aspen models are not based on runtime, but on resource usage. This means we can query the Aspen models to derive, for example, counts of floating point operations or messages. While these values might vary based on compiler or library, they are far more machine-independent and can be measured with tools like hardware counters and MPI library

interposition to give a ground truth against which we can judge our application models. Second, Aspen is an analytical tool, and can output not simply values for runtime and resource usage, but also symbolic equations. In some cases, this makes validation possible against algorithmic expectations; for example, we can use Aspen to validate that the number of floating point operations in a matrix multiplication is equal to $2n^3$.

As Section 5.1 points out, the context of workflows adds complexity. However, extending the MPI measurement to socket transmission and adding I/O capture can give us additional metrics for validating our workflow models. Sections 6 and 7 also describe additional monitoring which can aid this process, and we note that anomaly detection is not only most useful with a performance model in hand, but can assist in validating that same performance model.

6 Correlating Monitoring of Workflows and Infrastructure

An accurate attribution of the anomalies to the observed workflow performance is critical for fixing the problem, or adapting the system. In some cases the observed anomaly may have complex causes. For example, poor task performance can be caused by a slow CPU, by poor cache performance, by a slow disk, etc. Low task throughput within a workflow execution can be attributed to bottlenecks in the workflow engine, a lack of computational resources, or application errors. The end-to-end performance of data-intensive workflows is even more complex as it often depends on efficiently moving large data sets. This can be accomplished using bandwidth provisioned high-speed networks, however, performance in such networks is affected by a large number of factors, including: congestion, packet loss, end-host network and I/O performance, and end-host network tuning, among others. Thus, to diagnose these complex problems, we need to correlate observed performance with monitoring information coming from a number of systems (CPU, network, I/O, storage, etc.), and determine which of these systems is the most likely cause of the problem.

Using a combination of system monitoring utilities like the *sysstat* suite, *nethogs*, *iotop*, and the multi-domain networking monitoring tool, *perfSONAR*, we are conducting a parametric study of data transfer performance for workflows in a controlled and isolated environment using a testbed called ExoGENI [34]. We are exploring different workflow configurations based on throttling of data volumes, different data transfer modes (sharedfs, nonsharedfs/pegasus-lite, nonsharedfs-condorio), and multiple distributed domains. This will enable us to define baseline expected data-transfer performance and the monitoring manifestations of different performance profiles. We plan to combine this with other monitoring and profiling tools described in Section 4. We will use this monitoring information and the output of analytical performance models from ASPEN to conduct a thorough evaluation of these different kinds of correlations, which will help in detecting anomalies in an online fashion.

7 Anomaly Detection and Diagnosis

Anomalies should be detected when there is a deviation of observed workflow performance from the end-to-end performance model. At the same time, for accurate diagnosis, the observed performance deviations need to be traced to anomalies at the application, workflow, and infrastructure levels. A combination of offline and online strategies needs to be developed for detecting performance anomalies during workflow execution, and for diagnosing the root causes of the observed problems.

7.1 Offline Anomaly Classification

Our initial approach is to classify the different types of anomalies that can cause problems with end-to-end workflow performance, and use offline methods to determine the severity of the anomalies. *Infrastructure anomalies* contribute to a number of issues with application performance. Among them are: resource unavailability [35], poor I/O performance [36], disk failure, file system corruption, violations of disk quotas [37], poor network performance due to congestion and packet loss [38], firewalls (which often drop packets in flows over 1 Gbps), limitations in LAN switches (which may not have enough buffering to handle multiple simultaneous flows), bad optical fiber, and others. Typical *application anomalies* seen during workflow execution include: input data unavailability, application execution errors [39], and poor performance for tasks with very short execution times [40, 41]. *Workflow-level anomalies* are deviations in workflow-level performance metrics, such as: low task throughput, which may indicate problems

with resource availability; high task error rates, or significant changes in task error rates; slow end-to-end data transfers; data sets that are larger or smaller than expected; long queue wait times, which may indicate low resource availability or fairness and priority issues; and low resource utilization, which may be caused by resource failures, over-provisioning, or misconfiguration.

We need to quantify anomaly severity in discrete anomaly levels so that different sets of actions can be triggered to address different levels of the diagnosed anomaly. For example, some anomalies may be critical to workflow performance, such as input data unavailability, while others may only reduce application performance, such as long queue wait times. Anomaly levels can be determined based on thresholds that are derived by clustering performance metrics into groups. The threshold value of an anomaly level will be determined from execution traces collected and published in the workflow archive developed as part of the DOE dV/dt project, for which different thresholding approaches can be used. Anomaly levels and thresholds will, initially, be determined offline; thus they will not create any overhead on the workflow execution.

7.2 Real-time Anomaly Detection

We developed a persistent query agent (PQA) [42] that enables persistent queries on data federated from different sources, including multi-domain infrastructure monitoring data, and workflow and application performance data. Using this agent, it is possible to run continuous queries on disparate data sources and get asynchronous, real-time notifications when relevant performance events occur. We will leverage the PQA framework to combine offline anomaly analysis, Aspen performance model evaluations, and multi-domain monitoring data for online, real-time detection and scalable distribution of the triggers to the appropriate stakeholders.

It is important to define, ahead of time, which metrics are important for particular application use cases by benchmarking executions of individual applications. It is also imperative to register the application-level metrics, workflow-level metrics, and infrastructure metrics with the system so that persistent queries can refer to these metrics while constructing event conditions that might result in anomaly triggers. This will make it possible to use a unified mechanism to understand anomalies in multi-domain infrastructure, anomalies in the application, and anomalies in the application's view of the infrastructure. The Aspen analytical performance models will also help in identifying critical metrics, and hence in the design of the persistent queries. Predictions from the analytical models will also be pushed as events into the PQA making it possible to do "continuous diffs" of observed metric values with values predicted from the end-to-end models.

Another advantage of the persistent query mechanism is that the anomaly triggers immediately identify the corresponding persistent query, and hence the precise event condition that led to the trigger. The triggers are also distributed in a scalable fashion using a publish-subscribe framework, which makes it possible to inform only the interested stakeholders. For instance, different persistent queries will be relevant for different stakeholders. Since all these queries can coexist in the system, and triggers are associated with queries, only the stakeholders who are interested in the anomaly event condition will be notified. Hence, the design of persistent queries will take into account the different possible use cases—scientist use cases, workflow management system use cases, infrastructure adaptation use cases, resource provisioning use cases, and infrastructure operator use cases.

8 Hybrid Modeling: Interleaving Simulation with Analytical Modeling

Workflow models will undoubtedly have many components and interactions among those components. We will use simulation to help understand workflow components for which analytical models are not sufficiently accurate. The two simulation components are described below.

Rensselaer's Optimistic Simulation System (ROSS) (network simulation) is a framework for developing parallel discrete event simulations. ROSS has demonstrated highly scalable, massively parallel event processing capability for both conservative and optimistic synchronization approaches [43–47]. ROSS mitigates Time Warp state-saving overheads via *reverse computation* [48]. In this approach, rollback is realized by performing the inverse of the individual operations that were executed in the event computation. This eliminates the need to explicitly store prior logical process (LP), which represents a distinct component of the model state, leading to much more efficient memory utilization. We have leveraged reverse computation to demonstrate massively parallel performance of the ROSS simulation engine [44] in several models. Most recently,

ROSS optimistic event processing has demonstrated super-linear performance for the PHOLD benchmark using nearly 2 million Blue Gene/Q cores on the 120 rack Sequoia supercomputer system located at LLNL [43].

CODES storage simulation: We also developed the CODES simulation framework (based on ROSS), which combines models of storage devices, high performance networks, I/O forwarding infrastructure, and storage software into a unified parallel discrete event simulation model. CODES can simulate complete, large-scale storage systems all the way from individual application processes to individual storage device accesses.

Traditionally, users are required to create a simulation structure for the component of interest, and the interactions with those components. Since we already have the high-level structured analytical model of the target workflow in Aspen, we intend to use this Aspen model to drive the ROSS/CODES simulation of the component under consideration, and the external interactions that it expects (analytically). For example, in this combined framework, we will be able to use an Aspen model for the entire workflow (e.g., Climate model simulation), and then use the CODES simulator to simulate a component of the workflow (e.g., HPSS storage system) in great detail. The Aspen model will generate the I/O requests and transfers to the components (e.g., HPSS) in the simulator, and use analytical models for the remainder.

9 Related Work

There have been several previous efforts to characterize the workloads of distributed systems [49–52]. For instance, [49] and [50] presented analyses of Grid and HPC workload characteristics including system usage, user population, application characteristics, and characteristics of grid-specific application types. Analysis of MapReduce job characteristics such as CPU utilization, memory usage, slots allocation, I/O operations, and network transfers was presented in [51]. Cloud workload patterns (periodicity, threshold, relationship, variability, and image similarity) were identified in [52]. These studies typically use data provided by Grid and HPC workload archives [53–56]. These workloads mainly capture information about task executions, but lack critical information about scientific workflow executions such as task dependencies, task clustering, etc.

In the area of characterization and profiling, some efforts have been made to collect and publish traces and performance statistics for real scientific workflows, such as workflow-based workload traces from the Austrian grid [57, 58], provenance-based workloads from a workflow management system [59], a survey of workflow characteristics from several research domains including bioinformatics, medical informatics, weather and ocean modeling, and astronomy [60], and workflow-based workloads from a science-gateway [61]. We have characterized and profiled scientific workflows to understand and predict their resource requirements [22, 62, 63], and recently, we published traces for a few workflows executed using Pegasus [64], and synthetic workflows based on statistics from real applications for use in simulations [64, 65].

Performance modeling spans a large range of techniques, from analytical modeling to simulation. In analytical modeling, predictions can be obtained without directly using the application source code. These modeling methods range from ad-hoc, handwritten models to more structured methods like BSP [66] and LogP [67, 68]. The strength of analytical approaches when evaluating far-future architectures is shown in the HPC space in two examples considering FFT for exascale systems in [69] and [70]. Aspen extends these analytical modeling concepts to a formal DSL and adds extensibility and flexibility to analytical techniques; see [25] for a treatment of FFT on HPC architectures. Aspen is under active development and investigation for a variety of performance modeling contexts; see [71] for a more recent example of the use of Aspen in the HPC space.

10 Future Work

In addition to real workflows we will also develop synthetic workflows that emulate the behavior of the real applications, but allow us to easily vary parameters such as the input data size and the task runtimes. This will be supported by profiling the applications using our workflow profiling tools, which will be extended to support parallel applications and to collect additional metrics that characterize application behavior.

We will also identify and develop mechanisms for collecting and correlating infrastructure, workflow, and application monitoring data. This will involve developing new tools and enhancing existing tools to extract monitoring data and transport it to a central location for correlation and analysis. We plan to use the empirical monitoring data to validate the Aspen performance models for the different workflow use-cases. We will also validate the models using the ExoGENI and ESnet testbeds.

We plan to develop offline techniques for clustering and classifying anomalies based on thresholds and time-series data. We will extend our persistent query tool to use both Aspen performance models and monitoring data, and design new persistent queries for real-time detection and diagnosis of anomalies. To evaluate the accuracy of our anomaly detection approach we plan on using a combination of fault and load injection.

We will extend the Aspen analytical models and integrate them further with ROSS/CODES to facilitate detailed simulations, where analytical models lack the necessary insights.

We will also explore and implement adaptation techniques, which will be automatically triggered when anomalies and failures are detected.

Acknowledgments

This work was funded by DOE under contract #DE-SC0012636, "Panorama - Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows". The development of the neutron scattering simulation workflow was supported by the U.S. Department of Energy (DOE), Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division. The use of Oak Ridge National Laboratory's Spallation Neutron was sponsored by the Scientific User Facilities Division, Office of Basic Energy Sciences.

References

[1] Open Science Grid, http://www.opensciencegrid.org.

[2] Extreme Science and Engineering Discovery Environment, http://www.xsede.org.

[3] Chameleon cloud, http://www.chameleoncloud.org.

[4] CloudLab, http://cloudlab.us.

[5] PANORAMA: predictive modeling and diagnostic monitoring of extreme science workflows, http://sites.google.com/site/panoramaofworkflows.

[6] T. Mason, D. Abernathy, et al., The spallation neutron source in oak ridge: A powerful tool for materials research, Physica B: Condensed Matter 385 (2006) 955–960.

[7] J. C. Phillips, R. Braun, et al., Scalable molecular dynamics with NAMD, Journal of Computational Chemistry 26 (16) (2005) 1781–1802.

[8] D. Case, V. Babin, et al., AMBER 14, university of california, san francisco (2014).

[9] ptraj, http://ambermd.org/doc6/html/AMBER-sh-11.html.

[10] B. Lindner, J. Smith, Sassena: X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications 183 (7) (2012) 1491–1501.

[11] O. Arnold, J. C. Bilheux, et al., Mantid: Data analysis and visualization package for neutron scattering and sr experiments, Nuclear Instruments and Methods in Physics Research Section A 764 (2014) 156–166.

[12] E. Deelman, D. Gannon, et al., Workflows and e-science: An overview of workflow system features and capabilities, FGCS 25 (5) (2009) 528–540.

[13] E. Deelman, K. Vahi, et al., Pegasus, a workflow management system for science automation, Future Generation Computer Systemsdoi:10.1016/j.future.2014.10.008.

[14] K. Vahi, M. Rynge, et al., Rethinking data management for big data scientific workflows, in: Workshop on Big-Data and Science: Infrastruc. and Services, 2013, pp. 1–9.

[15] E. Deelman, S. Callaghan, et al., Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example, in: 2nd IEEE international Conference on E-Science and Grid Computing, 2006, pp. 4–6.

[16] E. Deelman, J. Blythe, et al., Pegasus: Planning for execution in grids, GriPhyN technical report 20 (2002) 1–6.

[17] S. Srinivasan, G. Juve, et al., A cleanup algorithm for implementing storage constraints in scientific workflow executions, in: 9th Workshop on Workflows in Support of Large-Scale Science, 2014, pp. 41–49. doi:10.1109/WORKS.2014.8.

[18] D. Gunter, E. Deelman, et al., Online workflow management and performance analysis with stampede, in: 7th International Conference on Network and Service Management, 2011, pp. 1–10.

[19] J.-S. Vöeckler, G. Mehta, et al., Kickstarting remote applications, in: 2nd International Workshop on Grid Computing Environments, 2006, pp. 1–8.

[20] dV/dT: Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science, http://sites.google.com/site/acceleratingexascale.

[21] G. Juve, B. Tovar, et al., Practical resource monitoring for robust high throughput computing, Tech. rep., University of Southern California (2014).

[22] R. Ferreira da Silva, G. Juve, et al., Toward fine-grained online task characteristics estimation in scientific workflows, in: 8th Workshop on Workflows in Support of Large-Scale Science, 2013, pp. 58–67. doi:10.1145/2534248.2534254.

[23] J. Kephart, D. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50. doi:10.1109/MC.2003.1160055.

[24] B. Tierney, J. Boote, et al., Instantiating a Global Network Measurement Framework, Tech. Rep. LBNL-1452E, Lawrence Berkeley National Lab (January 2009).

[25] K. Spafford, J. S. Vetter, Aspen: A domain specific language for performance modeling, in: ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis, 2012, pp. 84:1–84:11.

[26] C. L. Janssen, H. Adalsteinsson, et al., Using simulation to design extremescale applications and architectures: Programming model exploration, SIGMETRICS Performance Evaluation Review 38 (4) (2011) 4–8.

[27] A. F. Rodrigues, K. S. Hemmert, et al., The structural simulation toolkit, SIGMETRICS Performance Evaluation Review 38 (4) (2011) 37–42.

[28] H.-L. Truong, T. Fahringer, Scalea-g: A unified monitoring and performance analysis system for the grid, in: M. Dikaiakos (Ed.), Grid Computing, Vol. 3165 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 202–211. doi:10.1007/978-3-540-28642-4_24.

[29] S. Ostermann, K. Plankensteiner, et al., Workflow monitoring and analysis tool for askalon, in: Grid and Services Evolution, Springer US, 2009, pp. 1–14. doi:10.1007/978-0-387-85966-8_6.

[30] S. da Cruz, F. da Silva, et al., A lightweight middleware monitor for distributed scientific workflows, in: 8th IEEE International Symposium on Cluster Computing and the Grid, 2008, pp. 693–698. doi:10.1109/CCGRID.2008.89.

[31] M. Mao, M. Humphrey, Auto-scaling to minimize cost and meet application deadlines in cloud workflows, in: International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 49:1–49:12. doi:10.1145/2063384.2063449.

[32] S. Ostermann, R. Prodan, et al., Dynamic cloud provisioning for scientific grid workflows, in: 11th IEEE/ACM International Conference on Grid Computing, 2010, pp. 97–104. doi:10.1109/GRID.2010.5697953.

[33] G. Singh, C. Kesselman, et al., Application-level resource provisioning on the grid, in: 2nd IEEE International Conference on e-Science and Grid Computing, 2006, pp. 83–83. doi:10.1109/E-SCIENCE.2006.261167.

[34] I. Baldine, Y. Xin, et al., Exogeni: A multi-domain infrastructure-as-a-service testbed, in: 8th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2012, pp. 97–113.

[35] N. Russell, W. Aalst, et al., Workflow exception patterns, in: E. Dubois, K. Pohl (Eds.), Advanced Information Systems Engineering, Vol. 4001 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 288–302. doi:10.1007/11767138_20.

[36] A. Mandal, P. Ruth, et al., Evaluating i/o aware network management for scientific workflows on networked clouds, in: 3rd International Workshop on Network-Aware Data Management, 2013, pp. 2:1–2:10. doi:10.1145/2534695.2534698.

[37] E. Deelman, A. Chervenak, Data management challenges of data-intensive scientific workflows, IEEE International Symposium on Cluster Computing and the Grid (2008) 687–692doi:http://doi.ieeecomputersociety.org/10.1109/CCGRID.2008.24.

[38] M. Mathis, J. Semke, et al., The macroscopic behavior of the tcp congestion avoidance algorithm, SIGCOMM Comput. Commun. Rev. 27 (3) (1997) 67–82. doi:10.1145/263932.264023.

[39] R. Ferreira da Silva, T. Glatard, et al., Self-healing of workflow activity incidents on distributed computing infrastructures, Future Generation Computer Systems 29 (8) (2013) 2284–2294. doi:10.1016/j.future.2013.06.012.

[40] R. Ferreira da Silva, T. Glatard, et al., Controlling fairness and task granularity in distributed, online, non-clairvoyant workflow executions, Concurrency and Computation: Practice and Experience 26 (14) (2014) 2347–2366. doi:10.1002/cpe.3303.

[41] W. Chen, R. Ferreira da Silva, et al., Using imbalance metrics to optimize task clustering in scientific workflow executions, Future Generation Computer Systemsdoi:10.1016/j.future.2014.09.014.

[42] A. Mandal, I. Baldine, et al., Enabling Persistent Queries for Cross-aggregate Performance Monitoring, IEEE Communications Magazine 52(5).

[43] P. D. Barnes, Jr., C. D. Carothers, et al., Warp speed: executing time warp on 1,966,080 cores, in: ACM SIGSIM conference on Principles of advanced discrete simulation, 2013, pp. 327–336. doi:10.1145/2486092.2486134.

[44] D. W. Bauer Jr., C. D. Carothers, et al., Scalable time warp on blue gene supercomputers, in: ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation, 2009, pp. 35–44.

[45] C. D. Carothers, K. S. Perumalla, On deciding between conservative and optimistic approaches on massively parallel platforms, in: Winter Simulation Conference'10, 2010, pp. 678–687.

[46] N. Liu, J. Cope, P. Carns, et al., On the role of burst buffers in leadership-class storage systems, in: IEEE 28th Symposium on Mass Storage Systems and Technologies, 2012, pp. 1–11. doi:10.1109/MSST.2012.6232369.

[47] M. Mubarak, C. D. Carothers, et al., Modeling a million-node dragonfly network using massively parallel discrete event simulation, in: 3rd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, 2012, pp. 366–376. doi:10.1109/SC.Companion.2012.56.

[48] C. D. Carothers, K. S. Perumalla, et al., Efficient optimistic parallel simulations using reverse computation, ACM Transactions on Modeling and Computer Simulation 9 (3) (1999) 224–253. doi:http://doi.acm.org/10.1145/347823.347828.

[49] A. Iosup, D. Epema, Grid computing workloads, Internet Computing, IEEE 15 (2) (2011) 19–26. doi:10.1109/MIC.2010.130.

[50] D. L. Hart, Measuring teragrid: workload characterization for a high-performance computing federation, International Journal of High Performance Computing Applications 25 (4) (2011) 451–465.

[51] Z. Ren, X. Xu, et al., Workload characterization on a production hadoop cluster: A case study on taobao, in: IEEE International Symposium on Workload Characterization, 2012, pp. 3–13. doi:10.1109/IISWC.2012.6402895.

[52] S. Mahambre, P. Kulkarni, et al., Workload characterization for capacity planning and performance management in iaas cloud, in: IEEE International Conf. on Cloud Computing in Emerging Markets, 2012, pp. 1–7. doi:10.1109/CCEM.2012.6354624.

[53] Parallel workloads archive, http://www.cs.huji.ac.il/labs/parallel/workload.

[54] A. Iosup, H. Li, et al., The grid workloads archive, Future Gener. Comput. Syst. 24 (7) (2008) 672–686. doi:10.1016/j.future.2008.02.003.

[55] D. Kondo, B. Javadi, et al., The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems, in: 10th IEEE/ACM Inter. Conf. on Cluster, Cloud and Grid Computing, 2010, pp. 398–407.

[56] C. Germain-Renaud, A. Cady, et al., The grid observatory, IEEE International Symposium on Cluster Computing and the Grid (2011) 114–123.

[57] S. Ostermann, R. Prodan, et al., On the characteristics of grid workflows, in: CoreGRID Symposium - Euro-Par 2008, 2008, pp. 1–12.

[58] S. Ostermann, R. Prodan, et al., A trace-based investigation of the characteristics of grid workflows, in: T. Priol, M. Vanneschi (Eds.), From Grids to Service and Pervasive Computing, Springer US, 2008, pp. 191–203. doi:10.1007/978-0-387-09455-7_14.

[59] S. Madougou, S. Shahand, et al., Characterizing workflow-based activity on a production e-infrastructure using provenance data, Future Generation Computer Systems 29 (8) (2013) 1931 – 1942. doi:http://dx.doi.org/10.1016/j.future.2013.04.019.

[60] L. Ramakrishnan, D. Gannon, A survey of distributed workflow characteristics and resource requirements, Indiana University (2008) 1–23.

[61] R. Ferreira da Silva, T. Glatard, A science-gateway workload archive to study pilot jobs, user activity, bag of tasks, task sub-steps, and workflow executions, in: Euro-Par 2012: Parallel Processing Workshops, Vol. 7640, 2013, pp. 79–88. doi:10.1007/978-3-642-36949-0_10.

[62] S. Bharathi, A. Chervenak, et al., Characterization of scientific workflows, in: 3rd Workshop on Workflows in Support of Large-Scale Science, 2008, pp. 1–10. doi:10.1109/WORKS.2008.4723958.

[63] G. Juve, A. Chervenak, et al., Characterizing and profiling scientific workflows, Future Generation Computer Systems 29 (3) (2013) 682–692. doi:10.1016/j.future.2012.08.015.

[64] R. Ferreira da Silva, W. Chen, et al., Community resources for enabling and evaluating research on scientific workflows, in: 10th IEEE Inter. Conference on e-Science, 2014, pp. 177–184. doi:10.1109/eScience.2014.44.

[65] Workflow archive, http://workflowarchive.org.

[66] L. G. Valiant, A bridging model for parallel computation, Communications of the ACM 33 (8) (1990) 103–111.

[67] A. Alexandrov, M. F. Ionescu, et al., LogGP: Incorporating long messages into the LogP model, in: 7th Annual ACM Symposium on Parallel Algorithms and Architectures, 1995, pp. 95–105.

[68] D. Culler, R. Karp, et al., LogP: Towards a realistic model of parallel computation, 4th SIGPLAN Symp. Princ. and Prac. of Parall. Programming 28 (7) (1993) 1–12.

[69] H. Gahvari, W. Gropp, An introductory exascale feasibility study for ffts and multigrid, in: IEEE International Symposium on Parallel Distributed Processing, 2010, pp. 1–9. doi:10.1109/IPDPS.2010.5470417.

[70] K. Czechowski, C. Battaglino, et al., On the communication complexity of 3d ffts and its implications for exascale, in: 26th ACM International Conference on Supercomputing, 2012, pp. 205–214. doi:10.1145/2304576.2304604.

[71] K. Spafford, J. S. Vetter, et al., Modeling synthetic aperture radar computation with aspen, International Journal of High Performance Computing Applications 27 (3) (2013) 255–262.