Toward a Dynamic Network-centric Distributed Cloud Platform for Scientific Workflows: A Case Study for Adaptive Weather Sensing

Eric Lyons[‡], George Papadimitriou[†], Cong Wang[*], Komal Thareja[*]
Paul Ruth[*], J. J. Villalobos[§], Ivan Rodero[§]
Ewa Deelman[†], Michael Zink[‡], Anirban Mandal[*]

[*]RENCI, University of North Carolina at Chapel Hill, NC, USA
[†]Information Sciences Institute, University of Southern California, CA, USA
[‡]Electrical and Computer Engineering Department, University of Massachusetts at Amherst, MA, USA
[§]Rutgers Discovery Informatics Institute, NJ, USA

***Abstract*—Computational science today depends on complex, data-intensive applications operating on datasets from a variety of scientific instruments. A major challenge is the integration of data into the scientist's workflow. Recent advances in dynamic, networked cloud resources provide the building blocks to construct reconfigurable, end-to-end infrastructure that can increase scientific productivity. However, applications have not adequately taken advantage of these advanced capabilities. In this work, we have developed a novel network-centric platform that enables high-performance, adaptive data flows and coordinated access to distributed cloud resources and data repositories for atmospheric scientists. We demonstrate the effectiveness of our approach by evaluating time-critical, adaptive weather sensing workflows, which utilize advanced networked infrastructure to ingest live weather data from radars and compute data products used for timely response to weather events. The workflows are orchestrated by the Pegasus workflow management system and were chosen because of their diverse resource requirements. We show that our approach results in timely processing of Nowcast workflows under different infrastructure configurations and network conditions. We also show how workflow task clustering choices affect throughput of an ensemble of Nowcast workflows with improved turnaround times. Additionally, we find that using our network-centric platform powered by advanced layer2 networking techniques results in faster, more reliable data throughput, makes cloud resources easier to provision, and the workflows easier to configure for operational use and automation.**

***Index Terms*—adaptive weather sensing, network-centric platform, distributed cloud infrastructure, dynamic network and resource provisioning, malleable data flows, scientific workflow automation**

I. Introduction

Computational science today depends on many complex, data-intensive applications operating on distributed datasets that originate from a variety of scientific instruments and repositories. A major challenge for these applications is the integration of data into the scientist's workflow. These workflows are composed of a set of dependent tasks, each of which has different compute, network, storage, and input requirements that determine where and how each task can run. They may require specialized access to resources for large transfers between tasks, with data residing in different domains. This necessitates an integration of two or more existing infrastructures (e.g., instruments, compute resources, and data repositories) using high-performance networks and data management software in order to increase the scientific output. Currently, such integration is either not available, or is purpose-built manually for a specific scientific application or community. However, recent advances in dynamic networked cloud infrastructure (e.g., ExoGENI [1]) provide the technical building blocks to construct and manage such integrated, reconfigurable, end-to-end infrastructure, built-to-order with performance guarantees that satisfy workflow compute and data movement requirements.

Data-driven applications and workflows have not adequately taken advantage of the rich set of capabilities offered by dynamic, networked infrastructures. They are not designed to utilize adaptive features offered by state-of-the-art, networked cloud infrastructures, especially with respect to delivering end-to-end, high-performance data flows. As a result, scientists in weather modeling, ocean sciences, seismology, etc. are struggling to analyze data from community resources. They are often downloading the data to their own environment, processing it at limited scales in modest chunks, losing crucial time to react to the observed phenomenon and/or missing longitudinal patterns.

In parallel, resource requirements for these workflows are increasing in scale. Traditional approaches of statically provisioned, dedicated, pre-configured compute and network infrastructure are expensive, hard to adapt, and hard to manage. The bursty computational and network demands for these workflows warrant flexible processing solutions on diverse infrastructures for computing, and malleable, high-performance data movements for expeditious delivery. While dynamic provisioning mechanisms exist, these are not offered to the scientists at the right level of abstraction, making them difficult to use, with no guarantee of a proper resource match. Additionally, managing the execution of workflow ensembles over the provisioned infrastructure remains a significant challenge.

In this work, we address some of the above challenges

by developing a system called DyNamo, which enables high-performance, adaptive, performance-isolated data-flows across a federation of distributed cloud resources and community data repositories. DyNamo facilitates the provisioning of appropriate compute and storage resources for observational science workflows from diverse, national-scale cyberinfrastructure (CI). Driven by specific needs of science applications, these capabilities are presented to applications and users at the right level of abstraction through an easily understandable, high-level interface. Efforts to use, configure, and reconfigure resources are significantly simplified by this approach. Through integration with the Pegasus Workflow Management System [2], the DyNamo system offers automation and orchestration of data-driven workflows on the provisioned infrastructure.

In this paper, we make the following contributions:

- We present a case study of a data-driven science application called Collaborative and Adaptive Sensing of Atmosphere (CASA), and describe its specific computing and networking challenges, which are addressed by the DyNamo system (cf. Section II).
- We present our approaches to infrastructure federation and mechanisms used to enable malleable, high-performance data flows between diverse, distributed, national-scale cloud platforms (ExoGENI and Chameleon [3]) and the CASA data repository (cf. Section III).
- We present new features in a network-centric platform called Mobius, which bridges the abstraction gap by presenting an appropriate, high-level interface to scientists for dynamic, end-to-end resource provisioning (cf. Section III).
- We show how CASA workflows can be parallelized as ensembles and automatically executed on the provisioned infrastructure by Pegasus (cf. Section IV).
- We present a performance evaluation of our system with multiple CASA workflows under different deployment scenarios and infrastructure conditions (cf. Section IV).
- We show that our system is used operationally by CASA scientists for real weather events (cf. Section IV).

II. Challenges: Collaborative and Adaptive Sensing of Atmosphere (CASA) Application

CASA has the goal to improve our ability to observe, understand, predict, and respond to hazardous weather events. To achieve this goal, CASA has created a technology that allows the sensing of the lowest part of the atmosphere through networks of small, dual polarized, X-band Doppler radars [4]. Since 2012, CASA has operated a network of seven radars in the Dallas/Fort Worth (DFW) Metroplex. Deployment and operation of these meteorological sensors is only the first step, since the project goals depend on efficient distribution and scalable processing of the volumetric radar data. Dozens of meteorological products are generated in near real time, some 24/7/365, others on demand, based on the characteristics of the ongoing weather regime. First order processes include calculating rainfall rate and accumulations, short term nowcasts (0-30min), hydrometeor classifications (rain/hail/snow), hydrological products (runoff, streamflow), and network wind products. In addition, various post-processing routines operate on the gridded product data, including raster image generation, contouring, format conversions, and end user driven, GIS based data extraction. Timely generation of these products is essential for the warning process and requires significant network and compute resources.

Setting up a CASA workflow is complex [5] due to asynchronous input data, and dynamic computational processing loads which are often a function of the nature of the ongoing weather. CASA workflows have traditionally been processed across 16 servers, one dedicated server at each radar site, and the rest at the DFW Radar Operations Center (DROC) at NOAA Southern Region Headquarters in Fort Worth, Texas. Processing tasks have been assigned to each server based on extensive performance analysis to ensure that the largest loads can be handled when severe weather is present. As new types of processing algorithms have been introduced and existing algorithms improved, the system has been very carefully managed to integrate these without risking existing workflow performance. In recent years, dynamic provisioning solutions have reduced some unnecessary processing [5], however static architectures must still be chosen in advance based on an estimated requirement, and effort is needed to distribute workflow processes evenly across an arbitrary collection of VMs. The preferred solution is to be able to quickly acquire appropriate resources in an automated fashion based on specific triggers and measured loads, with a simple interface that allows for architecture reconfiguration as resource demand fluctuates. With its ability to provide on-demand, configurable, high-bandwidth network paths to distributed, national scale resources, its support of parallel execution of algorithms, and its workflow automation and management, DyNamo has the potential to greatly improve system design and simplify operations.

We briefly introduce the weather products that are generated by the CASA workflows described in this paper:

A. Nowcast

Nowcasts are short-term advection forecasts that are computed by mosaicking asynchronous individual radar reflectivity data, accumulating the composite grids over a short duration, and projecting into the future by estimating the derivatives of motion and intensity with respect to time [6], [7]. Every minute the CASA nowcasting system generates 31 grids of predicted reflectivity, one for each minute into the future from minutes 0-30. Nowcasts are valuable for short term planning and estimating the arrival or departure of precipitation by giving the user an estimation of the timing and trajectory of moving weather features. They can assist in such tasks as route planning, deployment of spotters, and keeping emergency responders out of harm's way. Before DyNamo, CASA dedicated a server with substantial resources to generate nowcasting data and required a second server to extract contours

for user notification purposes. Upon generation, nowcast data are transmitted to the CASA data repository, which serves as gateway between the public internet and ExoGENI provisioned layer-2 VLANs.

B. Wind Speed

A Doppler radar is able to estimate the velocity of moving objects based on a phase shift that occurs if the objects are moving toward or away from the radar beam. Components of velocity perpendicular to the beam are not sensed. For a given radar this means that there will be substantial underestimations of true wind speed over portions of the sensing domain where certain directional components of the winds are not able to be sampled. However with an overlapping network of radars, areas not adequately sampled by one radar are often better sampled by other radars with different relative angles. Therefore CASA's maximum observed velocity workflow ingests the single radar base data from all of the radars in the network and creates a gridded product representing the maximum observed wind speeds. As part of this workflow, areas of severe winds are identified and checked against the location of known infrastructure, in our case hospitals, with email alerts sent out should they be impacted. This class of workflow that blends together voluminous radar based data as an initial processing step is much more network intensive than other workflows that operate on derived products. Input data rates can approach 10Mbps per radar, of which CASA currently operates 7.

C. Contouring

Similar to contours on a topographical map that enclose areas above a certain elevation, in CASA contours indicate areas where values within the grid exceed a certain threshold [8]. Contouring produces 2D closed GIS style polygon 'objects' in GeoJSON format that describe geographical areas of observed or forecast weather risk. The contouring algorithm is used to describe areas with high radar reflectivity as a proxy for storm location in the nowcasting workflow, and areas of severe winds that are likely to be causing damage in the wind workflow. While the contouring process is a highly useful mechanism for communicating risk locations to users, generating valid, well ordered polygons out of large grids of imperfect data is a CPU intensive process. The CASA contouring procedure uses the CONREC algorithm [9] to generate a series of unordered isolines, followed by a custom approach to connect them into ordered concave polygons.

Visualization & Data dissemination. For each of the 31 grids produced per minute by the nowcasting algorithm and the grid produced by the Wind Speed algorithm a PNG image are created and disseminated back to the CASA data portal, where they are transferred for user visualization on CASA's Google Maps based website. In contrast to the computation of the gridded data, the generation of visualization, contouring, and notification can be performed in a distributed fashion and the postprocessing workflows are well suited for a HTCondor [10] compute cluster, instantiated at either Chameleon or an ExoGENI rack.

III. DyNamo System Description

The challenges described in Section II are common to several data-driven scientific workflows. Hence, we have developed the DyNamo system to address some of those challenges. The DyNamo system offers the following capabilities: (a) programmatic resource provisioning on an integrated, diverse infrastructure federation, which enables on-demand access to multiple national-scale computational cloud resources for science workflows, (b) end-to-end network management for malleable data movement across infrastructures and external science data repositories, (c) a network-centric platform called Mobius that hides the complexity of resource and network provisioning from the scientists and transforms high-level application-aware, infrastructure-agnostic resource requests to low-level infrastructure provisioning actions, and (d) support for workflow automation for science applications on the provisioned infrastructure using Pegasus. These DyNamo system capabilities are described as follows.

A. Integrated, Multi-Cloud Resource Provisioning

Data-driven workflows like those from the CASA application need to quickly acquire appropriate resources in an automated fashion to satisfy their bursty computational and network demands. The nature of ongoing or expected weather, the number of available sensors, and end user defined triggers all contribute to load variability. DyNamo enables CASA scientists to transparently acquire cloud resources from multiple cloud providers based on high-level resource requirements. Through network integration and programmatic provisioning of specific cloud resources using their native APIs, we are shielding the scientists from interacting with diverse cloud providers or from using low-level provisioning commands.

We initially target two national scale, federally funded cloud resource providers and make them accessible to CASA scientists using high-level resource requests.

- **ExoGENI** [1], a networked Infrastructure-as-a-Service (IaaS) testbed supported by NSF, links cloud racks at 20 sites on campuses across the US through regional and national transit networks (Internet2, ESnet, etc.). ExoGENI allows users to dynamically provision mutually isolated "slices" of interconnected infrastructure from multiple independent providers (compute, network, and storage) by taking advantage of existing virtualization mechanisms and integrating various resources together: layer2 global dynamic-circuit networks like Internet2 and ESnet, and private clouds like OpenStack [11]. ExoGENI provides value added over dynamic-circuit providers by permitting users to instantiate virtual, distributed topologies, and by provisioning the appropriate network resources corresponding to the topologies, thereby creating end-to-end layer2 paths.
- **NSF Chameleon Cloud** [12] is a large, deeply programmable testbed designed for systems and networking experiments. Similar to ExoGENI, it leverages OpenStack to deploy isolated slices of cloud resources for user

experiments. However, where ExoGENI scales in geographic distribution, Chameleon scales by providing large amounts of compute, storage, and networking resources spread across only two sites. In total, Chameleon provides over 15K cores and 5 PB storage across the University of Chicago and the Texas Advanced Computing Center. Users can provision bare metal compute nodes with custom system configuration connected to user-controlled OpenFlow switches operating at up to 100 Gbps. In addition, Chameleon networks can be stitched to external partners including ExoGENI slices.

By utilizing the Mobius platform (cf. Sect. III-C), it is possible to provision appropriate compute resources from ExoGENI and Chameleon using a high-level, application-aware API. By sending requests to Mobius, scientists can provision resources from different clouds and connect them for use by a single workflow. The Mobius platform also provides capabilities to easily instantiate application-specific environments on top of the multi-cloud provisioned resources, e.g., a distributed HTCondor [10] cluster.

B. Malleable Data Movement Across Infrastructures and External Data Repositories

Integrating data movements with computations is essential, but often overlooked, for data-driven science workflows. While provisioning resources from multiple clouds provides scientists with significant compute flexibility, supporting high-performance data movements into, across, and out of the provisioned distributed compute infrastructure is another important problem that we address in this work. Mobius allows flexible, programmatic provisioning of high-bandwidth network paths/circuits across and within different infrastructures and science data repositories.

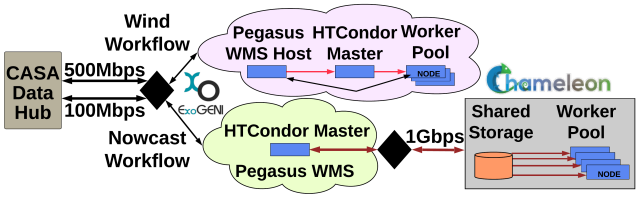


Fig. 1: Data movement and resource provisioning across CI federation.

Figure 1 shows the deployment scenario for supporting malleable data movements. We leverage the ExoGENI network overlay to connect data repositories like CASA with national scale CI resources, Chameleon and ExoGENI racks currently, with plans for connecting to XSEDE JetStream, Open Science Grid and Amazon AWS in the future, through ExoGENI-provisioned, dynamically reconfigurable network paths. The connections between ExoGENI and the external resources and data repositories use ExoGENI stitchports [13].

Stitchport: A general stitchport abstraction is a foundational building block of this deployment. What lies beyond a stitchport is assumed to be an IP-based subnet including infrastructure like data transfer nodes for large compute clusters, data repository endpoints, nodes connected to storage arrays for instrument data, or any other sources or sinks of scientific data sets. Stitchports enable high-performance connectivity to external infrastructure outside of ExoGENI.

UNT stitchport. We deployed a stitchport at the location of the CASA data repository at the University of North Texas in Denton, TX. The server behind this stitchport hosts the weather data from the 7 CASA radars. This server is plumbed to a static pool of VLANs set up and connected to the ExoGENI network fabric by transiting through the LEARN regional network. This makes it possible to programmatically connect the data repository to any of the 20 ExoGENI racks or other external infrastructure that ExoGENI can connect to, with a dynamic bandwidth specification.

Chameleon Stitching. Chameleon supports the creation of isolated OpenFlow networks controlled by users. In addition to connecting Chameleon compute resources, these networks can be stitched to external partners, including ExoGENI, using dedicated layer2 circuits [14]. Stitching uses a set of ExoGENI stitchports which can be allocated to Chameleon networks. A Chameleon user can create an isolated network that includes one of the ExoGENI stitchports. Then ExoGENI users can create slices that include the allocated stitchport. After the stitchport has been allocated on both infrastructures, layer2 traffic will pass between the networks.

Hence, our system, through stitchports, connects the CASA data repository, via dynamically provisioned layer2 overlay networks, to ExoGENI resources, which are then stitched to Chameleon. This creates layer2, end-to-end, virtualized, performance isolated data movement paths for CASA workflows.

A proof-of-concept has been developed that enables Chameleon and ExoGENI slices to connect to other partners including Amazon AWS Direct Connect using Internet2's Cloud Connect service [15].

C. DyNamo Network-centric Platform: Mobius

We have developed new capabilities in a network-centric platform called Mobius [16], [17]. These include support for integrated, multi-cloud resource provisioning and high-performance science data flows across diverse infrastructures, and enhanced mechanisms for interacting with higher level application and workflow management systems and transforming high-level resource requests to low-level provisioning actions.

The Mobius platform has been implemented as a Spring framework [18] based REST server and exposes a REST API [17] for automated provisioning of network and compute resources. It consumes high-level application-aware resource requests from scientists or workflow systems and automatically provisions resources using the native APIs of different providers. Essentially, the applications can specify to Mobius their resource requirements over time in the form of a Gantt chart. Scientists can easily set up application-specific environments by invoking the Mobius REST API. Figure 2 shows the different components of the Mobius platform.

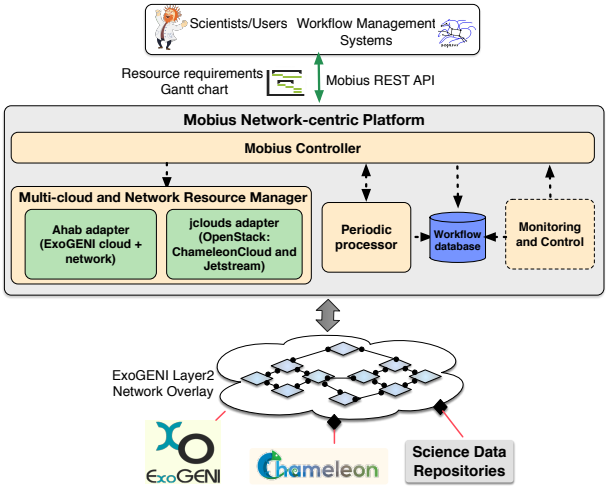


Fig. 2: Mobius network-centric platform.

Multi-cloud and Network Resource Manager. At this layer, Mobius translates application requests to native cloud specific requests. Since we are leveraging the ExoGENI network overlay to set up data flow paths, the application-level data movement requests get translated to ExoGENI network provisioning requests. The Multi-cloud and Network Resource Manager consists of two native cloud specific adapters to provision resources on our target infrastructures.

Ahab adapter: *Ahab* [19] is a collection of graph-based Java libraries designed to allow applications to control, modify and manage the state of ExoGENI slices. *Ahab* includes *libndl*, which provides a graph-based abstraction for interacting with ExoGENI *slices*. It primarily handles the conversion of an abstract topology graph consisting of *ComputeNodes*, *networks*, *stitchports*, *storage*, etc. into native ExoGENI resource requests (NDL-OWL [20]), which are then sent to ExoGENI using another library called *libtransport*. The Mobius Ahab adapter leverages the Ahab library functionalities to instantiate compute resources on ExoGENI racks and to create network paths between stitchports, ExoGENI racks and other cloud providers like Chameleon.

Jclouds adapter: Apache jclouds [21] is an open source multi-cloud toolkit that allows the creation of portable applications across different cloud providers while maintaining full control to use cloud-specific features. We have implemented a Mobius jclouds adapter for OpenStack to provision resources on Chameleon and XSEDE JetStream. We also plan to implement a Mobius jclouds adapter for Amazon EC2 to provision resources on Amazon AWS, which can then be used in conjunction with the stitchport for AWS Direct Connect to move data in and out of the EC2 provisioned resources.

Workflow Database. The information about all the resources provisioned for a workflow or an application on different clouds and the corresponding application request parameters is maintained in the Workflow Database.

Periodic Processor. High level application requests can be represented as a Gantt chart of required resources for a particular application or workflow. The periodic processor triggers the provisioning of the resources at the scheduled time. It also monitors the provisioning status of all the resources instantiated for various application workflows and triggers notifications to applications/workflow management system.

Monitoring and Control. The Monitoring and Control module is designed to transparently maintain the quality of service of the provisioned end-to-end infrastructure through continuous monitoring and control (e.g. growing and shrinking compute or storage resource pools and changing network properties of links). This module is currently under development.

Mobius Controller. The Mobius controller orchestrates all the above components and processes the incoming REST requests to trigger appropriate Mobius components.

D. Workflow Automation

The DyNamo system supports workflow automation on the provisioned infrastructure by leveraging Pegasus, which provides the necessary abstractions for scientists to create workflows, and allows for transparent execution of these workflows on a range of execution platforms. Workflows are described abstractly as directed acyclic graphs (DAGs), where nodes represent individual compute tasks and the edges represent data and control dependencies between tasks. During planning, Pegasus translates the abstract workflow into an executable workflow, determining executables, data, and computational resources required for execution. Individual workflow tasks are managed by a scheduler (HTCondor), which supervises their execution on local and remote resources.

In this work, we have developed Pegasus workflows for the CASA application, which are executed on end-to-end infrastructure provisioned using Mobius. While designing these workflows we wanted to achieve three main goals: (1) ease of deployment, (2) portability and (3) scalability. The Mobius platform simplifies resource provisioning and helps in ease of deployment. Taking advantage of Pegasus' container support provides a unified execution environment and guarantees portability across multiple clouds. The abstract workflow definitions describe the computational tasks in their simplest form (one task triggers one executable). Even though this approach results in more tasks in the generated DAG, it increases the versatility of the workflow. Taking into consideration characteristics of the tasks (e.g., runtime or resource usage) and the provisioned resources, we can instruct Pegasus to automatically cluster multiple tasks together, creating larger jobs and optimizing for data movement and resource utilization.

IV. Evaluation of DyNamo System

A. CASA Workflow Description

For the evaluation of DyNamo we have selected CASA workflows that produce nowcasts and wind speed estimates as described in Section II. Figure 3 shows the deployment scenario for the CASA workflows used for the evaluation. The workflow processes include input data collection and product generation, visualization, contouring into polygon

objects, spatial comparisons of identified weather features with infrastructure, and dissemination of notifications.

Pegasus Workflows and Testcases. The generated Pegasus workflow for nowcast [22] consists of 63 compute tasks. There is a preprocessing task that splits the input data into 31 grids, and then 62 independent tasks compute the reflectivity and the respective contour images. All tasks run within a Docker container that is managed by Pegasus and has a size of 476MB. For the evaluation of the nowcast workflow we are using 30 minutes of pre-captured data (individual file size 9.6MB, total size 287MB), which we replay using an accumulation interval of 1 minute. On the other hand, the wind workflow [23] has a variable size. The preprocessing phase is responsible to unzip any zipped input files, and the number of tasks depends on that. This phase is followed by four compute tasks that output the wind products and notify points of interest for severe weather. These four tasks are running within a Docker container, 523MB in size. For the evaluation of the wind workflow we are using 40 minutes of pre-captured data (individual file size ∼12MB, total size ∼6GB), which we replay using an accumulation interval of 5 minutes. With these in mind we classify nowcast as a *compute intensive* workflow and wind speed as a *data intensive* workflow.

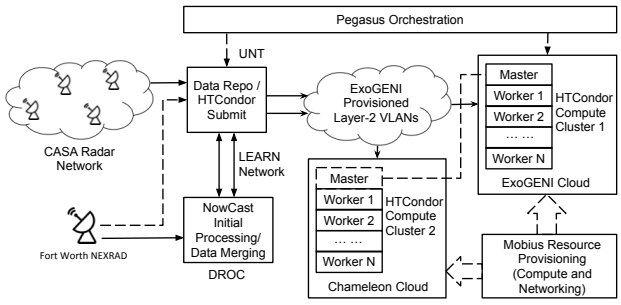


Fig. 3: CASA workflow deployment.

B. Experimental Infrastructure

We deployed a realistic scenario that is similar to the real CASA operational radar data processing setup, on ExoGENI and Chameleon testbeds using the DyNamo system. It is worth mentioning that rather than direct performance comparison between ExoGENI and Chameleon testbeds, our major goal in this paper is to demonstrate the effectiveness of DyNamo workflow execution over heterogeneous compute and networking infrastructures based on their resource availability. In our setup, the ExoGENI compute cluster is located in Jacksonville, FL, and contains 13 VM nodes, with 1 master node, 11 worker nodes, and 1 Network File System (NFS) storage node. Each node has 4 virtual cores, 12 GB RAM, and is connected to 1 Gbps network. The Chameleon compute cluster contains 5 nodes and is located in Chicago, IL. It is comprised of 4 worker nodes and 1 NFS storage node. Each node has 24 cores, 192 GB RAM, 250GB SSD and is connected to a shared 10 Gbps network. The two configurations provide 44 compute slots and 96 compute slots respectively.

Because the master node doesn't require much processing power for a small-sized cluster, we decided it is suitable to maintain it within ExoGENI and connect all the workers to the same HTCondor pool. In order to specify which workers we want to dispatch jobs to, we use HTCondor's job requirements filter and request workers on either ExoGENI or Chameleon based on their hostnames.

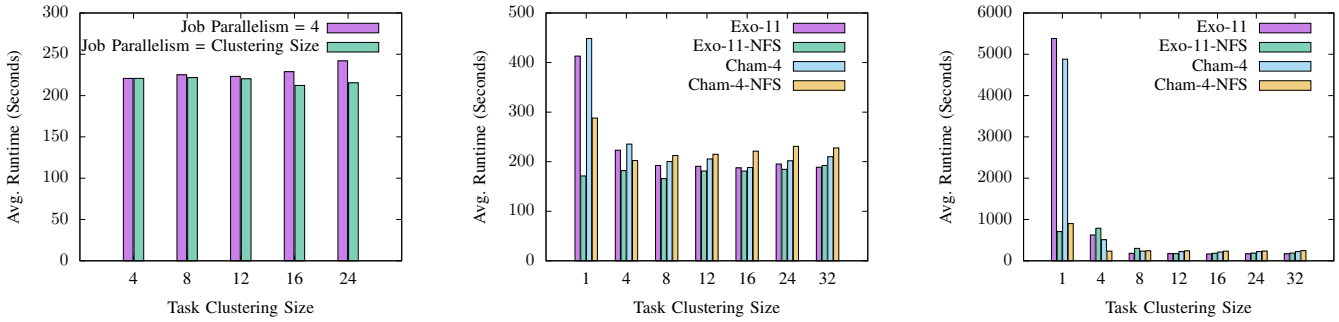
Finally, we deployed the Pegasus submit node at the University of North Texas (UNT) in Denton, TX, which receives data from CASA's data sources via the TX LEARN research network and triggers new workflows as new data arrive. Raw data from the CASA radars and the KFWS NEXRAD radar are directly transmitted to the submit node. Nowcasts are first computed on the DROC and then transmitted to the UNT submit node. The UNT submit node is connected to the ExoGENI and Chameleon processing clusters via 1 Gbps stitchable layer-2 VLANs, which are dynamically provisioned on ExoGENI by Mobius.

Software. On all of the nodes we have installed HTCondor v8.6, and we have customized its configuration to match the role of each node. In this setup, the workers are configured with partitionable slots, that allows us to request multiple cores per job request. Additionally, on the submit node we have installed Pegasus v4.9.1 and all of the workers use Docker CE v.18.09.5. Mobius was used to provision compute resources on ExoGENI and Chameleon, and the network connections between ExoGENI, Chameleon and the CASA repository.

C. Experimental Results

During our preliminary tests with the nowcast workflow, we observed that transferring and loading the Docker container for every task had a severe impact on performance. To alleviate this, we configured the nowcast workflow for an environment that does not have access to shared storage and another one that shares NFS folders across the worker nodes. In the first environment, Pegasus uses HTCondor's file transfer feature to move data in and out of the worker nodes, and back to the submit node. For the NFS environment, Pegasus uses its own data transfer recipes to stage in data to a *shared scratch* location on the NFS shared folder and orchestrates the jobs on the workers to *symlink* against them. When the final output products are generated, Pegasus pulls the data back to the submit node, via *scp*. This approach allows us to remove frequent transfers between the submit node and the workers, and enables us to place large common files (e.g., the container images) closer to the workers. For both cases, we tested varying sizes of task clusters, grouping together tasks of the same type, and evaluating the effect for each configuration.

1) Effect of cluster parallelism in nowcast: Figure 4a presents the average workflow makespan, which is defined as the average runtime of the worklow, for executions of the nowcast workflow (with 63 tasks) on Chameleon nodes, while having NFS storage enabled. Pegasus allows to select a task clustering size (X-axis), which is the number of workflow tasks grouped in a job, and the desired job parallelism, which is the number of cores allocated to the job for running the tasks in the

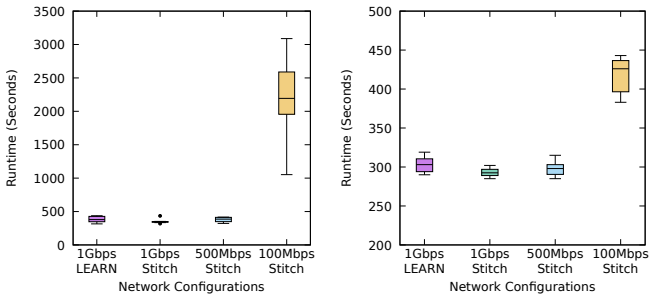


(a) Exploring Cluster Parallelism. (b) Single Workflow Runs. (c) Workflow Ensemble Runs.

Fig. 4: Nowcast: Stitchport 1Gbps.

job. For each task clustering size, we plot the runtimes for two values of job parallelism - a fixed value of 4 and for a value equal to the task clustering size. We observe that there are marginal gains in workflow runtimes, ranging between 3 and 28 seconds for all task clustering sizes, but these benefits are quickly overshadowed by the increased need for more compute slots, blocking other jobs from being executed concurrently. Thus, for the rest of our nowcast experiments, when Pegasus clusters tasks together, a fixed *parallelism of 4* is used.

2) Performance study of nowcast workflows: In Figures 4b and 4c, we present the average nowcast workflow makespan, while targeting different execution environments on ExoGENI and Chameleon, with varying task clustering sizes. Figure 4b shows results from running a single nowcast workflow without any other workflows competing for resources for different cluster sizes. Due to the size of the nowcast workflow (63 tasks) we could fit the non-clustered workflow at once in Chameleon (96 slots available), but there were insufficient resources on ExoGENI (44 slots available). By submitting the workflow without clustering (cluster size 1) there is a noticeable impact on the performance in the cases where we do not use NFS, observed on both sites. This is due to the increased network traffic created by transferring the application container. Additionally, in all cases, Chameleon executions tend to be a bit slower than the ExoGENI ones. This is an effect of loading the Docker images to the workers for every task. Having 11 nodes on ExoGENI helps to spread out the load imposed by Docker. However, as we increase the number of clustered tasks, the performance advantage is almost negligible since for cluster sizes >12 the runtimes of the configurations are 30-45 seconds apart. Figure 4c presents results from replaying the test case data, which results in executing a workflow ensemble, which is defined as a set of workflows running the same computations on different input data, possibly arriving at different times. Similar to the single workflow runs, non clustered configurations perform very poorly as they saturate the submit node's link for prolonged periods, while overlapping workflows in the ensemble competing for the same resources, makes the situation even worse. *By increasing the number of clusters to 12 or more, we improve the average runtime by more than 4500 seconds, reaching a stable point, on both ExoGENI and Chameleon.*



(a) Exogeni - HTCondor Transfers (b) Exogeni - NFS

Fig. 5: Wind: Workflow Ensemble Runs - No Clustering.

3) Better data movement performance: Figure 5 depicts the distribution of the wind workflow makespans while either using TX LEARN public layer 3 network or the stitchport with variable bandwidths. Figure 5a displays makespan statistics without using NFS, while executions in Figure 5b use NFS to optimize data placement. For both cases, we observe that the performance of the 1Gbps stitchport is more consistent than the 1Gbps LEARN, which is not performance isolated. As we decrease the stitchport bandwidth to 500Mbps and 100Mbps, the makespan increases and its distribution spreads, since the wind workflow, being data-intensive, is sensitive to network bandwidth. We also observe that the median makespans for 1Gbps LEARN and 1Gpbs or 500Mbps stitchport are similar, proving that 500Mb/s bandwidth is sufficient to allow our processing infrastructure to keep up with the incoming data. *These results show the value of dynamically provisioned, performance isolated networks for data-intensive workflows.*

4) Compute slot utilization: Another aspect with great operational interest is the amount of resources that have to be allocated for a compute intensive workflow like nowcast. Figure 6 presents the number of active compute slots while replaying the nowcast testcase, using task clustering sizes of 4, 16 and 32, on Chameleon nodes, with and without an NFS server. In both cases, *increasing task cluster size decreases the number of active compute slots, and thus decreases the number of nodes that need to be provisioned.* We observe that clustering 4 tasks creates high demand, with spikes close to 80 slots (NFS case). Increasing the task clustering to 16

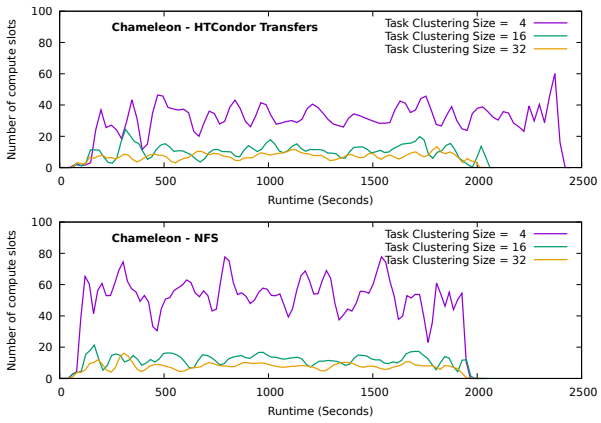


Fig. 6: Nowcast: Workflow Ensemble Runs - Chameleon Compute Slot Utilization.

or 32, decreases peak compute slot demand to just 20 slots. The runtime is most with task cluster size 4 (HTCondor case) because that case has total 16 jobs moving containers vs. 4 and 2 jobs for task clustering 16 and 32.

D. Operational Experience

The architecture and the workflows were used for CASA operations during the spring of 2019, with ongoing severe weather, and live radar and nowcast data sent to the DyNamo server and into the DyNamo framework for product generation. Cloud resources were provisioned with Mobius. Initial experiences lead us to migrate several parameters out of statically defined json configuration files, and into command line arguments for ease of scripted integration and automation, without the need for error prone string replacements in files. Among these were lease duration times, network domain addresses, number of worker nodes, and the ability to specify directories containing instructions for different workflows. What remained in json file structures were only parameters unlikely to be changed with regularity such as layer 2 bandwidth definitions, memory, and CPU types. In general, there is only a brief interval from when resources are requested to when they are online, configured, and begin processing data. This is an important consideration as in some cases where unstable atmosphere thunderstorms can develop rapidly, resulting in a small lead time from initial detection to when the workflow data are urgently needed. We used observed radar data as the cue to procure and release resources, but the preferred manner may eventually be to use short term forecast data to provide enough lead time to not miss early stage activity. Provisioning ExoGENI nodes only was found to take 2-4 minutes from request to being ready for operations. A hybrid setup using both ExoGENI and Chameleon nodes took up to 10 minutes at times, with the selected number of worker nodes contributing to the variability. Because it can be impossible or expensive to reserve external IP addresses on different clouds, it was decided for operational purposes to dedicate particular VLANs to specific workflows. This was convenient, as our secure publish/subscribe data transfer software, Unidata LDM [24], requires the explicit listing of allowed downstream IP addresses in advance for data forwarding. When Mobius provisions the stitchport it uses the VLAN tag associated with the workflow in question, and assigns an IP address in that range to the cloud instance on ExoGENI. LDM configuration scripts are generated in advance with VLAN IP addresses and the naming convention patterns defined for workflow input and output data. In addition, the VLANs are matched to the input/output data rates of individual workflows. The DyNamo server uses a single 1Gbps link upon which up to 10 VLANs may be used. As the number of workflows increase, it will be of greater importance to partition the links according to workflow I/O. For these initial experiments a high bandwidth 500Mbps link for the wind workflow that operates on voluminous radar moment data was selected, and a 100Mbps link which is more than sufficient for the nowcast workflow. Often, the algorithmic runtimes of the workflow processes are dependent on the nature of the weather regimes contained in the data. Whereas the test cases removed variability with respect to weather to isolate network, compute, and HTCondor/Pegasus parameters, the realtime operations quickly demonstrated the variability of algorithmic load over time. During periods of clear weather, there is very little processing to be done, and runtime drops sharply. During these times, workers can be released back into the cloud pool, or the workflow could be ceased altogether and the instances terminated. As weather develops and coverage and intensity increases, it will be important to monitor processing times and increase worker nodes to ensure that they stay within tolerable latencies, and conversely, that unnecessary resources are not allocated.

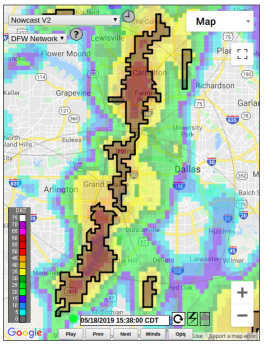


Fig. 7: Nowcast output with DyNamo from live storm event (05/18/19).

Upon the development of weather in the radar network, a single Mobius command was used to spawn our predefined architectures. First we tested with a master, a submit, and 6 worker nodes on the University of Houston ExoGENI cloud. A stitchport is created from the CASA server at UNT and the submit node at UH. Later we tested with a single node at UH ExoGENI, and four worker nodes and an NFS shared data node located at the Chameleon site at the University of Chicago (UC). The latter setup requires a second stitchport to be created by Mobius to connect UH and UC servers, along with simple routing rules defined to move data back and forth across the networks. In both cases, as soon as the architecture is spawned and post-processing scripts are executed, no further intervention is necessary and the workflow progresses as intended. After algorithm output is returned from the workers, monitoring routines ensure that data is moved out and back to UNT for transport to the CASA Google Maps based website for live display (Figure 7).

In the case of the max wind product, a single image of the

highest observed velocities from various radars in the network was generated every 90 seconds, along with GeoJSON files containing contoured windspeed levels. These were compared with hospital locations in the DFW metroplex and automated email notifications sent when winds exceeded severe levels. This workflow was selected because of its large input data bandwidth for networking purposes, and to evaluate the ease with which a complex chain of processes could be scripted for the Pegasus workflow management system.

In the case of Nowcasting, 31 image files and GeoJSON formatted polygons are calculated every minute. Before utilizing the DyNamo framework, we only had available processing power to script the workflow for 3 of the 31 grids produced every minute by the Nowcasting algorithm, without risking increasing backlog during the worst cases with widespread weather. With Dynamo and 6 ExoGENI workers, or 4 Chameleon workers, we could run the workflow on all 31 grids per minute (representing 0-30 minutes into the future) without steadily increasing latency. Were the nowcasting algorithm extended beyond 30 minutes, the workflow could remain identical, requiring only a scaling of the number of worker nodes in the Mobius request! Looking ahead to larger, national-scale networks and increasingly complex algorithms, this has substantial potential to reduce costs by matching compute with demand and simplifying integration efforts.

V. Related Work

The related work can be classified into three categories: research cloud platforms, resource provisioning for workflows, and workflow management for science applications.

Cloud platforms. A number of public cloud providers, such as Amazon EC2 [25], Microsoft Azure [26], Rackspace [27], offer IaaS abstractions and some ability to orchestrate them together with networks through mechanisms like CloudFormation [28] and Heat [29]. However, their closed nature and difficulty of moving data in and out of their infrastructure, limit their uses in science applications. This is especially true when science applications need to be deployed on a multi-cloud environment [13]. Existing cloud research testbeds like FutureGrid [30] are not programmable from a networking perspective. GlobusOnline [31] project permits users to efficiently move data from one computing resource to another, however, it does not provide unified environments for science workloads. We have focused on integration of scalable, reconfigurable distributed testbeds, including ExoGENI [1] and Chameleon [3] with emphasis on data movements.

Resource provisioning for science workflows. Recent survey papers [32]–[34] focused on the effectiveness of IaaS cloud resource provisioning for executing scientific workflows. Wang et al. [35] propose an approach to build and run scientific workflows on a federation of multiple clouds using Kepler and CometCloud. Moreover, there has been strategies for workflow systems focusing on the deployment of virtual machines in the cloud with limited support for on demand provisioning and elasticity, while none or minimal support to infrastructure optimization is enabled. In particular, data placement/movement and network configuration/provisioning decisions are crucial to achieving high performance for big data applications [36]. Ostermann et al. [37] discussed a set of VM provisioning policies to acquire and release cloud resources for overflow grid jobs from workflows, and the impact of those policies on execution time and overall cost. In a previous work, we presented dynamic provisioning techniques that spawn resources based on compute elasticity using Mobius [16]. Our current work differs from above by presenting easy-to-use, on-demand resource provisioning mechanisms for malleable data movement and compute provisioning for inter-cloud workflows.

Science workflow management systems. Several workflow management systems focus on the execution and management of science applications on cloud platforms. Malawski et al. [38] focused on cost optimization modeling for scheduling workflows on public clouds to minimize the cost of workflow execution under deadline constraints. Abrishami et al. [39] presented workflow scheduling algorithms based on partial critical paths, which also optimize for cost of workflow execution while meeting deadlines. With the rise of multi-clouds, many workflow management systems have focused on this type of platform. Matthew et al. [40] discuss workflow management on multi-cloud brokering among multi-cloud domains with heterogeneous security postures. Senturk et al. [41] deals with bioinformatics applications on multi-clouds with focus on resource provisioning. In this paper, we propose a set of new approaches to enable dynamic resource provisioning which is integrated with workflow management systems, with example deployments with science applications.

VI. Conclusions and Future Work

We presented the DyNamo system that addresses the computing and networking challenges for CASA distributed, atmospheric science workflows by enabling high-performance, adaptive data flows and coordinated access to distributed cloud resources and data repositories. We showed how the Mobius platform encapsulates the different DyNamo capabilities and makes it easier for scientists to provision end-to-end infrastructure. Through performance evaluation of our system executing CASA workflows orchestrated by Pegasus, we have shown that our approach results in timely processing of Nowcast workflows under different infrastructure configurations and network conditions. We also show the effect of workflow task clustering on throughput of an ensemble of Nowcast workflows. We found that using our network-centric platform powered by advanced layer2 networking techniques results in faster, more reliable data throughput, makes cloud resources easier to provision, and the workflows easier to configure for operational use and automation. In future, we plan to extend the DyNamo system by stitching to other resource providers, supporting streaming workflows, developing new CASA workflows, developing policies and mechanisms for monitoring and control to transparently maintain the QoS of the provisioned infrastructure, and incorporating virtual

software defined exchanges [19] as a basis for link adaptation, flow prioritization and traffic control between endpoints.

ACKNOWLEDGMENT

This work is funded by NSF award #1826997. We thank Mert Cevik (RENCI), engineers from UNT and LEARN for the UNT stitchport setup. Results in this paper were obtained using Chameleon and ExoGENI testbeds supported by NSF.

REFERENCES

[1] I. Baldin, J. Chase, Y. Xin, A. Mandal, P. Ruth, C. Castillo, V. Orlikowski, C. Heermann, and J. Mills, "Exogeni: A multi-domain infrastructure-as-a-service testbed," in *The GENI Book*, R. McGeer, M. Berman, C. Elliott, and R. Ricci, Eds. Springer International Publishing, 2016, pp. 279–315.

[2] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, no. 0, pp. 17–35, 2015.

[3] J. Mambretti, J. Chen, and F. Yeh, "Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn)," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, Oct 2015, pp. 73–79.

[4] D. McLaughlin, D. Pepyne, V. Chandrasekar, B. Philips, J. Kurose, M. Zink, K. Droegemeier, S. Cruz-Pol, F. Junyent, J. Brotzge, D. Westbrook, N. Bharadwaj, Y. Wang, E. Lyons, K. Hondl, Y. Liu, E. Knapp, M. Xue, A. Hopf, K. Kloesel, A. DeFonzo, P. Kollias, K. Brewster, R. Contreras, B. Dolan, T. Djaferis, E. Insanic, S. Frasier, and F. Carr, "Short-wavelength technology and the potential for distributed networks of small radar systems," *Bulletin of the American Meteorological Society*, vol. 90, no. 12, pp. 1797–1818, 2009. [Online]. Available: https://doi.org/10.1175/2009BAMS2507.1

[5] E. J. Lyons, M. Zink, and B. Philips, "Efficient data processing with exogeni for the casa dfw urban testbed," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2017, pp. 5977–5980.

[6] L. Li, W. Schmid, and J. Joss, "Nowcasting of motion and growth of precipitation with radar over a complex orography," *Journal of Applied Meteorology*, vol. 34, no. 6, pp. 1286–1300, 1995. [Online]. Available: https://doi.org/10.1175/1520-0450(1995)034¡1286:NOMAGO¿2.0.CO;2

[7] E. Ruzanski and V. Chandrasekar, "Weather radar data interpolation using a kernel-based lagrangian nowcasting technique," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 6, pp. 3073–3083, June 2015.

[8] P. R. Mahapatra and V. V. Makkapati, "Studies on a high-compression technique for weather radar reflectivity data," in *2005 5th International Conference on Information Communications Signal Processing*, Dec 2005, pp. 895–899.

[9] P. Bourke, "A contouring subroutine," *Byte*, vol. 12, pp. 143–150, 06 1987.

[10] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[11] "Openstack." [Online]. Available: https://www.openstack.org/

[12] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing: From Petascale toward Exascale*, 1st ed., ser. Chapman & Hall/CRC Computational Science, J. Vetter, Ed. Boca Raton, FL: CRC Press, 2018, vol. 3, ch. 5.

[13] I. Baldin, P. Ruth, C. Wang, and J. S. Chase, "The future of multi-clouds: A survey of essential architectural elements," in *2018 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC)*, Oct 2018, pp. 1–13.

[14] M. Cevik, P. Ruth, K. Keahey, and P. Riteau, "Wide-area software defined networking experiments using chameleon," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2019.

[15] I. Sara Aly, "Secure, private, dedicated connection offering available to aws direct connect, google cloud dedicated interconnect, and microsoft azure expressroute," Press Release, April 2019, https://www.internet2.edu/news/detail/17118/.

[16] A. Mandal, P. Ruth, I. Baldin, Y. Xin, C. Castillo, G. Juve, M. Rynge, E. Deelman, and J. Chase, "Adapting scientific workflows on networked clouds using proactive introspection," in *IEEE/ACM Utility and Cloud Computing (UCC)*, 2015.

[17] Mobius Github Repository, https://github.com/RENCI-NRIG/Mobius.

[18] Spring Framework, https://spring.io/.

[19] A. Mandal, P. Ruth, I. Baldin, R. F. Da Silva, and E. Deelman, "Toward prioritization of data flows for scientific workflows using virtual software defined exchanges," in *2017 IEEE 13th International Conference on e-Science (e-Science)*, Oct 2017, pp. 566–575.

[20] J. van der Ham, F. Dijkstra, P. Grosso, R. van der Pol, A. Toonk, and C. de Laat, "A distributed topology information system for optical networks based on the semantic web," *Optical Switching and Networking*, vol. 5, no. 2, pp. 85 – 93, 2008, advances in IP-Optical Networking for IP Quad-play Traffic and Services. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1573427708000064

[21] Apache jclouds, https://jclouds.apache.org/.

[22] Scitech, "CASA Nowcast Pegasus Workflow," https://github.com/pegasus-isi/casa-nowcast-workflow.

[23] ——, "CASA Wind Pegasus Workflow," https://github.com/pegasus-isi/casa-wind-workflow.

[24] Unidata LDM, https://www.unidata.ucar.edu/software/ldm/.

[25] Amazon Elastic Compute Cloud, http://www.amazon.com/ec2.

[26] Microsoft Azure Cloud, https://azure.microsoft.com/en-us/.

[27] Rackspace Cloud, https://www.rackspace.com/.

[28] AWS CloudFormation, http://aws.amazon.com/cloudformation.

[29] OpenStack Heat Project, https://wiki.openstack.org/wiki/Heat.

[30] FutureGrid, https://portal.futuregrid.org/.

[31] I. Foster, "Globus online: Accelerating and democratizing science through cloud-based services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, May 2011.

[32] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "A survey of data-intensive scientific workflow management," *Journal of Grid Computing*, vol. 13, no. 4, pp. 457–493, Dec. 2015.

[33] G. Galante, L. C. Erpen De Bona, A. R. Mury, B. Schulze, and R. Rosa Righi, "An analysis of public clouds elasticity in the execution of scientific applications: A survey," *Journal of Grid Computing*, vol. 14, no. 2, pp. 193–216, Jun. 2016.

[34] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: a survey," *annals of telecommunications - annales des télécommunications*, vol. 70, no. 7, pp. 289–309, Aug 2015.

[35] J. Wang, M. AbdelBaky, J. Diaz-Montes, S. Purawat, M. Parashar, and I. Altintas, "Kepler + cometcloud: Dynamic scientific workflow execution on federated cloud resources," *Procedia Computer Science*, vol. 80, pp. 700 – 711, 2016, international Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

[36] R. F. da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications," *Future Generation Computer Systems*, vol. 75, pp. 228 – 238, 2017.

[37] S. Ostermann, R. Prodan, and T. Fahringer, "Dynamic cloud provisioning for scientific grid workflows," in *2010 11th IEEE/ACM International Conference on Grid Computing*, Oct 2010, pp. 97–104.

[38] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski, "Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization," *Scientific Programming*, vol. 29, pp. 158–169, Jan. 2015.

[39] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158 – 169, 2013.

[40] M. Dickinson, S. Debroy, P. Calyam, S. Valluripally, Y. Zhang, R. Bazan Antequera, T. Joshi, T. White, and D. Xu, "Multi-cloud performance and security driven federated workflow management," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[41] I. F. Senturk, P. Balakrishnan, A. Abu-Doleh, K. Kaya, Q. Malluhi, and mit V. atalyrek, "A resource provisioning framework for bioinformatics applications in multi-cloud environments," *Future Generation Computer Systems*, vol. 78, pp. 379 – 391, 2018.