Cluster Computing
https://doi.org/10.1007/s10586-021-03457-3

Fair sharing of network resources among workflow ensembles

George Papadimitriou[1] · Eric Lyons[2] · Cong Wang[3] · Komal Thareja[3] · Ryan Tanaka[1] · Paul Ruth[3] · Ivan Rodero[4] · Ewa Deelman[1] · Michael Zink[2] · Anirban Mandal[3]

Received: 3 May 2021 / Revised: 12 October 2021 / Accepted: 19 October 2021

Abstract
Computational science 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 reconfiguration, end-to-end infrastructure that can increase scientific productivity, but applications are not taking advantage of them. In our previous work, we introduced DyNamo, that enabled CASA scientists to improve the efficiency of their operations and effortlessly leverage capabilities of the cloud resources available to them that previously remained underutilized. However, the provided workflow automation did not satisfy all the operational requirements of CASA. Custom scripts were still in production to manage workflow triggering, while multiple layer 2 connections would have to be allocated to maintain network QoS requirements. To address these issues, we enhance the DyNamo system with advanced network manipulation mechanisms, end-to-end infrastructure monitoring and ensemble workflow management capabilities. DyNamo's Virtual Software Defined Exchange (vSDX) capabilities have been extended, enabling link adaptation, flow prioritization and traffic control between endpoints. These new features allow us to enforce network QoS requirements for each workflow ensemble and can lead to more fair network sharing. Additionally, to accommodate CASA's operational needs we have extended the newly integrated Pegasus Ensemble Manager with event based triggering functionality, that improves managing CASA's workflow ensembles. The Pegasus Ensemble Manager, apart from managing the workflow ensembles can also create conditions for a more fair resource usage, by employing throttling techniques to reduce compute and network resource contention. We evaluate the effects of the DyNamo's vSDX policies by using two CASA workflow ensembles competing for network resources, and we show that traffic shaping of the ensembles can lead to a fairer sharing of the network links. Finally, we study how changing the Pegasus Ensemble Manager's throttling for each of the two workflow ensembles affects their performance while they compete for the same network resources, and we assess if this approach is a viable alternative compared to the vSDX policies.

Keywords Network-centric platform · Distributed cloud infrastructure · Scientific workflow automation · Dynamic network and resource provisioning · Virtual software defined exchange · Ensemble manag

✉ George Papadimitriou
georgpap@isi.edu

Ivan Rodero
ivan.rodero@utah.edu

1 Information Sciences Institute, University of Southern California, Los Angeles, CA, USA

2 Electrical and Computer Engineering Department, University of Massachusetts at Amherst, Amherst, MA, USA

3 RENCI, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA

4 SCI Institute, University of Utah, Salt Lake City, USA

1 Introduction

Computational sciences rely on complex, data-intensive applications to manage the processing of distributed datasets that are produced by a diverse set of scientific instruments and reside in geographically scattered locations. One of the biggest challenges these applications face, is the efficient data movement between the heterogeneous compute and storage resources, and the integration of the data into the scientists' workflows. These workflows might depend on specialized resources, such as hardware accelerators (e.g., GPUs and FPGAs), that are available in different compute locations, require access to input data hosted under different domains, and produce a significant amount of intermediate data that have to be transferred between tasks to facilitate their execution. To accommodate the needs of these scientific applications and increase scientific productivity, two or more infrastructure domains must be integrated and offered transparently to the scientists. Mechanisms to support this integration are currently not readily available, and scientific communities that require such integration have resorted to their own custom solutions that don't provide flexibility to generalize their approach. However, recent advances in dynamic networked cloud infrastructure, such as ExoGENI [1], provide the technical building blocks to construct and manage such integrated, reconfigurable, end-to-end infrastructure, built-to-order with isolated resources 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 a new set of dynamic, networked infrastructures. They are not designed to utilize adaptive features offered by state-of-the-art, networked cloud infrastructures, especially with respect to managing end-to-end, high-performance data flows. As a result, domain scientists in weather modeling, ocean sciences, seismology, etc., struggle to analyze data available in community resources. They often download 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.

Additionally, managing the execution of workflow ensembles over the sophisticated inter-domain infrastructures remains a significant challenge. Traditional workflow management approaches make use of statically provisioned, dedicated, pre-configured compute and network infrastructure. Such approaches are often associated with high cost, since the resources are usually provisioned such that the highest workload can be handled. This imposes extra cost when the system stays idle. Therefore, the bursty computational and network demands for science workflows warrant flexible processing solutions on diverse infrastructures for computing, and malleable, high-performance data movements for efficient data delivery.

In a previous work, we presented the DyNamo system [2] which addresses the above challenges and we focused on its networking capabilities that enable high-performance, adaptive, performance-isolated data-flows across a federation of distributed cloud resources and community data repositories. Even though this system introduced a robust way of connecting distributed data repositories to cloud compute resources with guaranteed performance, and automated the deployment of weather modeling workflows, it didn't address all the operational needs of weather modeling scientists.

In this paper, we extend the DyNamo system with more advanced capabilities in layer 2 network resource allocation. We integrate DyNamo with the virtual Software Defined Exchange (vSDX) [3] architecture, which serves as a virtual interconnect among different domain infrastructures providing flexible, high-performance data transfer over dedicated network circuits. Additionally, we enhance the workflow management capabilities of DyNamo with a workflow ensemble manager featuring automatic triggering of workflow ensembles and improved ensemble control. We also introduce a third party tool for end-to-end infrastructure monitoring and visualization.

Specifically, in this paper we make the following contributions:

- We present a data-driven science application, named Collaborative Adaptive Sensing of the Atmosphere (CASA), and describe its revised requirements and challenges that need to be addressed by the DyNamo system.
- We briefly present the architectural components of the DyNamo system, which provides federated infrastructure support to enable malleable, high-performance data flows between diverse, distributed, national-scale research cloud platforms (ExoGENI [1] and Chameleon [4]) and the CASA data repository.
- We present the architecture of the vSDX network infrastructure, which enables high-performance data transfer among heterogeneous compute and storage infrastructures, and we describe the newly introduced functionality that allows link adaptation, flow prioritization and traffic shaping.
- We present new features of the Pegasus Ensemble Manager that enable event based workflow triggering, and discuss how CASA's workflow ensembles can benefit from its functionality
- We provide an in-depth evaluation and analysis of the network performance on the inter-domain, multi-cloud infrastructures. While network resource sharing is

unavoidable, we discuss how DyNamo can create an environment that promotes a more fair utilization of the network resources either by employing features of the vSDX component or the Pegasus Ensemble Manager.

The rest of this paper is organized as follows: Sect. 2 provides an overview of the related work. Section 3 discusses background information for the DyNamo components. Section 4 introduces the components of the CASA weather forecasting application. Section 5 presents the extended components that work together to support science workflows. In Section 6, we evaluate the performance of the DyNamo ensemble manager and of the enhanced DyNamo networking features. Finally, Section 7 concludes the paper.

2 Related work

Cloud services allow users to easily spawn and dismiss resources around the globe upon their realtime needs. With its great flexibility, cloud computing has rapidly emerged as one of the most popular approaches for compute intensive and data intensive applications. There has been extensive prior work on the topics of cloud support for various types of science applications. In this section, we review the related works, which can be classified into three categories: cloud platforms, inter-domain networking and compute infrastructure provisioning for science workflows, and science workflow management systems.

2.1 Cloud platforms

A lot of work has been done on the development of research and commercial cloud infrastructures. A number of public cloud providers, such as Amazon EC2 [5] and Microsoft Azure [6], offer IaaS abstractions and some ability to orchestrate them together with networks through mechanisms like CloudFormation [7] and Heat [8]. However, data movement among different cloud providers and infrastructures is expensive and hard to implement, which significantly limits the use of commercial clouds in science applications [9]. The Globus [10] project provides users the ability to efficiently move data from one computing resource to another, however, it does not provide unified environments for science workloads. In the work presented in this paper, we focus on integration of scalable, reconfigurable distributed testbeds, including ExoGENI [1] and Chameleon [4] with emphasis on data movement and optimization of network resource sharing.

2.2 Inter-domain networking and compute infrastructure provisioning for science workflows

Resource management and provisioning for distributed applications has been the subject of many research efforts. There have been extensive survey papers [11–13] in regards to provisioning IaaS cloud resource for scientific workflows. Wang et al. [14] propose an approach to build and run scientific workflows on a federation of clouds using Kepler and CometCloud. Moreover, there have been strategies for workflow systems to deploy virtual machines in the cloud with limited support for on-demand provisioning and elasticity, while none or minimal support to infrastructure optimization is enabled. Ostermann et al. [15] discussed a set of VM provisioning policies to acquire and release cloud resources for overflow grid jobs from workflows, and characterized the impact of those policies on execution time and overall cost. In prior work [2], we presented dynamic provisioning techniques that spawn resources based on compute elasticity using Mobius [16].

On the perspective of networking between the compute, storage and instrument sites, Macker et al. [17] describe workflow paradigms to address network edge workflow scenarios. Ramakrishnan et al. [18] present experience for virtualized reservations for batch queue systems, as well as coordinated usage of TeraGrid, Amazon EC2 and Eucalyptus (cloud) resources with fault tolerance through automated task replication. Liu et al. [19] developed the Virtual Science Network Environment (VSNE) that emulates the multi-site host and network infrastructure, wherein software can be tested based on mininet with SDN capabilities.

As an important factor, many of the prior works have thrived to achieve a satisfactory Quality of Service (QoS) for the provisioned resources, as indicated by many recent survey papers [20, 21]. Varshney et al. [21] proposed QoS based workload scheduling mechanism by considering energy consumption, execution cost and execution time as QoS parameters. The Department of Energy's ESNet has proposed an On-Demand Secure Circuits and Advance Reservation System [22], which provides software system for booking time and resources on high-speed science networks used by large teams of researchers to share vast amounts of data.

Our work presented in this paper differs from the above by presenting easy-to-use, on-demand resource provisioning mechanisms for malleable data movement and compute provisioning for inter-cloud workflows. We provide dedicated network connections among multiple cloud provider sites with guaranteed performance and QoS policies enforced by a virtual software defined exchange (vSDX).

2.3 Science workflow management systems

Several workflow management systems focus on the optimization of science application management on cloud platforms. Islam et al. [23] presented a scalable workflow management system specifically for Hadoop applications. Senturk et al. [24] deal with bioinformatics applications on multi-clouds with a focus on resource provisioning. Malawski et al. [25] presented cost optimization modeling for scheduling workflows on public clouds to minimize the cost of workflow execution under deadline constraints. Abrishami et al. [26] 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. [27] discuss workflow management on multi-cloud brokering among multi-cloud domains with heterogeneous security postures. In this paper, we propose a new approach to enable dynamic resource provisioning in the clouds, which is integrated with a workflow management system coupled with advanced workflow ensemble management, and demonstrated through deployments with science applications.

3 Background

3.1 Pegasus WMS

Pegasus [28] is a popular workflow management system that enables users to design workflows at a high-level of abstraction. The Pegasus workflow descriptions are independent of the resources available to execute the workflow tasks and are also independent of the location of data and executables. Pegasus transforms these abstract workflows into executable workflows that can be deployed onto distributed and high-performance computing resources such as Leadership Computing Facilities (e.g., NERSC [29] and OLCF [30]), shared computing resources (e.g., XSEDE [31], OSG [32]), local clusters, and commercial (e.g., Amazon AWS [33]) and academic clouds (e.g., ExoGENI [1], Chameleon [34]). During the compilation process, Pegasus performs data discovery, locating input data files and executables. Data transfer tasks are automatically added to the executable workflow and perform two key functions: (1) move input files to staging areas associated with the target computing resources, and (2) transfer the generated outputs back to a user-specified location. Additionally, data cleanup (removal of data that is no longer required by the workflow at the execution site) and data registration tasks (that catalog the output files) are also added to the workflow. To manage user data, Pegasus interfaces with a wide variety of backend storage systems that use different data access and transfer protocols.

Pegasus relies on HTCondor [35] DAGMan as its workflow execution engine to run and manage the generated executable workflows. DAGMan in turn, submits the workflow jobs, as they become ready to run (when all parent jobs have completed successfully) to the internal job queue managed by HTCondor. During workflow execution, provenance information from workflow and job logs is automatically parsed and stored in a relational datastore by a monitoring daemon [36].

3.2 HTCondor

HTCondor [35] is a comprehensive job management system. In contrast to other batch systems such as PBS [37] and SLURM [38], it is particularly suited for distributed high throughput computing (HTC) environments, where one can setup a compute pool of nodes connected over a local area network or a wide area network. HTCondor provides users with a local job queue managed by a daemon *HTCondor Schedd* to which users submit jobs. Furthermore, HTCondor supports matchmaking [39] that allows users to match their jobs with compute nodes that support specific resources. The matchmaking takes place during the negotiation of the resources and is based on HTCondor ClassAds advertised by the compute nodes. Finally, in addition to submitting jobs to HTCondor managed compute resources, HTCondor also provides a component, called HTCondor-G [40], that allows users to submit jobs to other types of schedulers.

3.3 Mobius

A network-centric platform called Mobius [41] depicted in (Fig. 1) includes (a) support for integrated, multi-cloud resource provisioning and for high-performance science data flows across diverse infrastructures, and (b) enhanced mechanisms for interacting with higher level application and workflow management systems and transforming high-level resource requests to low-level provisioning actions, thereby bridging the abstraction gap between data-driven science applications and resource provisioning systems, and (c) transparently maintain the quality of service of the provisioned end-to-end infrastructure through continuous monitoring and control. Mobius was enhanced in our previous work [2] to support the provisioning of network connections between compute resources across sites/clouds and modulating the bandwidth on these network connections.

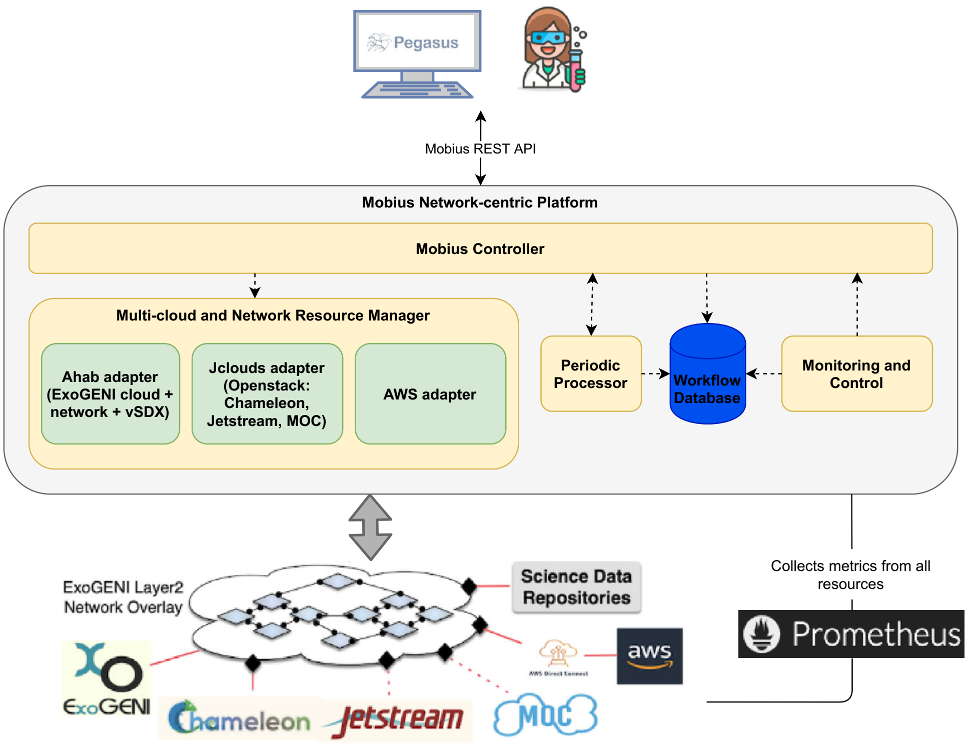

Pegasus
Mobius REST API
Mobius Network-centric Platform
Mobius Controller
Multi-cloud and Network Resource Manager
Ahab adapter (ExoGENI cloud + network + vSDX)
Jclouds adapter (Openstack: Chameleon, Jetstream, MOC)
AWS adapter
Periodic Processor
Workflow Database
Monitoring and Control
ExoGENI Layer2 Network Overlay
Science Data Repositories
aws
Collects metrics from all resources
Prometheus
ExoGENI
Chameleon
Jetstream
MOC

Fig. 1 Mobius—network centric platform overview

3.4 DyNamo

Data-driven workflows need to automatically and flexibly provision resources to satisfy scientists' bursty computational and network demands. In the case of CASA workflows (Sect. 4), the nature of ever-changing weather events, the number of available sensors, and end user-defined triggers all contribute to load variability.

As presented in previous work [2], DyNamo enables CASA scientists to transparently acquire cloud resources from multiple cloud providers based on high-level resource requirements. As depicted in Fig. 2, DyNamo provides network integration and programmatic provisioning of specific cloud resources using their native APIs. With this approach, domain scientists no longer need to directly interact with diverse cloud providers. To achieve this goal, DyNamo brings together the 3 major components defined earlier in this section: *Pegasus WMS* is used to provide workflow automation to the applications. *HTCondor* is used to manage the computational resources and distribute the computations. *Mobius* is used to allocate compute and network resources and create the interconnect between data sources and execution sites. Later in Sect. 5, we will present additional components for DyNamo, making it an integrated, network-aware instrument and monitoring tool for data-driven science applications in multi-cloud environments.

3.5 Target cyberinfrastructure

In this paper, we make use of two national scale research cloud providers: ExoGENI and the Chameleon cloud.

- *ExoGENI* [1] is a networked Infrastructure-as-a-Service (IaaS) testbed that links 20 cloud sites on campuses across the US through regional and national transit networks, such as Internet2 [42] and ESnet [43]. ExoGENI allows users to dynamically provision isolated "slices" of compute and networking resources from multiple sites and to integrate various resources using layer 2 global dynamic-circuit networks like Internet2 and ESnet, and private clouds like OpenStack [44]. ExoGENI allows users to instantiate customized, distributed topologies, and by provisioning the appropriate network resources corresponding to the topologies, thereby creating end-to-end layer-2 paths.
- *NSF Chameleon Cloud* [34] is a large-scale, 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, ExoGENI scales in geographic distribution, while Chameleon scales by providing large amounts of compute, storage, and networking resources spread across two sites: University of Chicago (UC) and the Texas Advanced Computing Center (TACC). Chameleon provides over 15K cores and 5 PB storage across the two sites. Users can

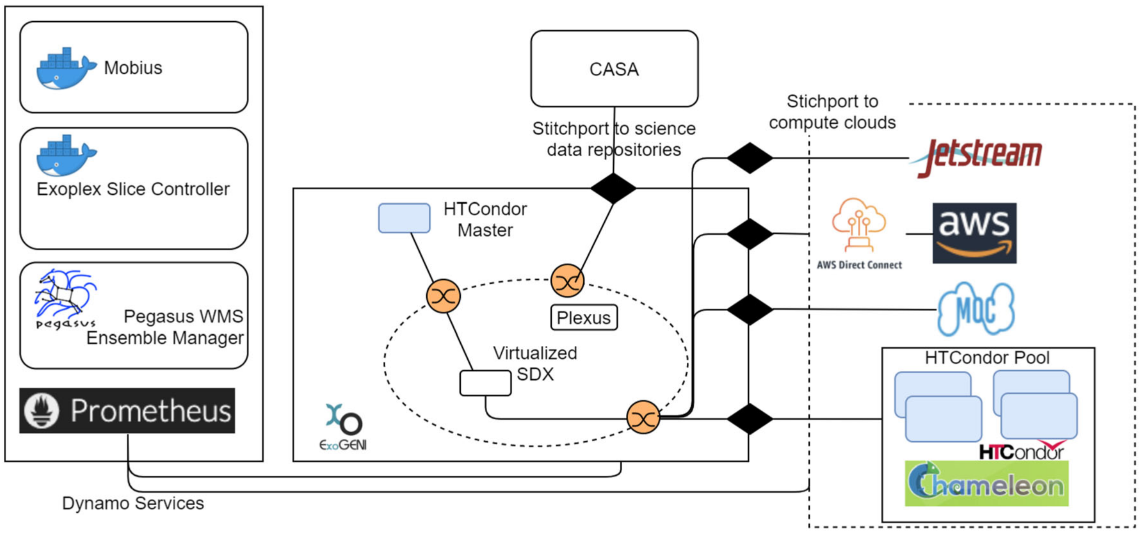


Fig. 2 Dynamo framework

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.

4 CASA—motivation

The NSF Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere (CASA) was formed to study the lower atmosphere with networks of high resolution Doppler weather radars with the goal to improve severe weather awareness [45]. The volumetric data produced by these continuously operating remote sensors must be distributed to processing servers quickly and efficiently such that analysis can occur in near real time for the sake of warning the public to fast developing threats such as tornadoes and high winds. The networked radar concept requires that asynchronous raw data from multiple sources are blended together to create value-added meteorological products. At any given time the characteristics of the ongoing weather regime determine the necessity and priority of certain products. For example, a hail detection algorithm takes on high importance only when strong thunderstorms are ongoing, whereas forecasting algorithms may be of more importance well in advance of such severe weather events and perhaps somewhat less so once the event has started.

For years, CASA's scientific workflows associated with product creation have been executed on dedicated servers existing at individual radar sites and at compute centers at NOAA Southern Region Headquarters and at the University of Massachusetts Amherst. Servers have been assigned dedicated processing tasks carefully tailored to their hardware and networking resources through trial and error with estimates made regarding the largest likely compute loads associated with each task. The careful management required implies that reconfiguration is highly complex and not feasible by an operator on short notice during an event. To help mitigate this limitation, and to create a more scalable system, in recent years CASA has developed several containerized scientific workflows for calculating these weather products that can be deployed and prioritized as needed [2]. CASA workflows are generally multi-step processes that can include a collection of necessary radar and non-radar sensor data access, grid transformations, format conversions, derived product creation, raster image generation, contouring, GIS based data extraction, and customized notification and alerting [46]. These require complex scheduling and in some cases significant resource consumption, especially during widespread impactful weather when they take on their greatest utility to the end users. For these workflows, CASA now relies on Mobius to provision and modulate compute and networking resources on demand, and uses the Pegasus Workflow Management System to manage the execution of the workflow steps [2].

In the following subsections, we briefly introduce the weather products that are generated by the CASA workflows studied in this paper.

4.1 Nowcast

Nowcasts are short-term advection forecasts that use observed reflectivity data from multiple radars, composite them for a certain number of minutes, and project into the future by estimating the derivatives of motion and intensity with respect to time [47, 48]. Every minute the CASA nowcasting system generates 31 grids of predicted reflectivity, one for each minute into the future from minutes 0 to 30. The workflow associated with Nowcasting creates raster images for all 31 grids every minute, and also contours for multiple reflectivity levels on each of these grids. The contours are sent to a database where they are used for notification purposes as simplified boundaries containing forecast reflectivity levels of importance for particular applications such as route planning, deployment of spotters, and keeping emergency responders out of harm's way. Nowcast rasters and contours are sent to CASA's data repository over layer 2 stitchports [1] where they are used in web and mobile applications.

4.2 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 (as in CASA's case), 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, contoured, and checked against the location of known infrastructure, with email alerts sent out to locations likely to be affected. Workflows that use the large single radar raw data as input have a substantially higher network bandwidth requirement than those operating on derived data. Input rates of over 100Mbps are common, and given that high winds, which are associated with tornadoes and downbursts are often short lived, one must minimize transmission delays as much as possible to adequately provide warnings for users downstream of the observations.

5 Approach—DyNamo extensions

In order to accommodate different application QoS policies and make a more efficient and fair use of the infrastructure among the workflow ensembles, we are extending the DyNamo system (Fig. 2) with a more sophisticated network configuration component, end-to-end infrastructure monitoring and advanced workflow management techniques.

5.1 vSDX module

A Virtual Software Defined Exchange (vSDX) is defined as a virtual interconnect point between multiple adjacent domains, e.g, instruments, compute resources, or data/storage systems. Like a static SDX, a vSDX uses Software Define Networking (SDN) within the exchange to enforce different network policies.

In our case, the vSDX support is provided by the Exo-Plex [49] network architecture depicted in (Fig. 3). Exo-Plex uses an elastic slice controller to coordinate dynamic circuits and the Zeek (formerly Bro) [50] security monitors via Ahab [51]. The controller runs outside of the vSDX slice and exposes a REST API for clients to request network stitching and connectivity and to express QoS parameters. Clients (i.e. Mobius) invoke this API to bind named subnets under its control to the vSDX via L2 stitching and request bandwidth provisioned connectivity with other subnets. The vSDX slice is comprised by virtual compute nodes running OpenVSwitch [52], OpenFlow controllers [53], and Zeek traffic monitors. Traffic flow and routing within the vSDX slice are governed by a variant of the Ryu [54] rest router [55] SDN controller. The vSDX slice controller computes routes internally for traffic transiting through the vSDX network, and invokes the SDN controller API to install them. The SDN controller runs another Ryu module (rest ofctl) to block traffic from offending senders. If a Zeek node detects that traffic violates a Zeek policy, it blocks the sender's traffic by invoking a rest ofctl API call via the Zeek NetControl plugin.

As client requests for bandwidth provisioned connectivity arrive at the vSDX, the slice controller instantiates slice resources as needed to carry the expected traffic. These resources include peering stitchport interfaces at each point of presence (PoP), the OVS nodes that host these vSDX edge interfaces, Zeek (Bro) nodes to monitor the traffic, and backplane links to carry the traffic among the PoPs. The controller reuses existing resources in the slice if they have sufficient idle capacity to carry the newly provisioned traffic, and instantiates new resources as needed. In particular, it adapts the vSDX backplane topology by allocating and releasing dynamic network circuits as needed to meet its bandwidth assurances to its customers. The flows are inspected by out of band Zeek network security monitor appliances to detect intrusion. As a simple form of intrusion prevention, it uses Zeek's NetControl framework to interrupt all traffic from the source of a suspect flow. The vSDX controller deploys Zeek instances elastically to scale capacity.

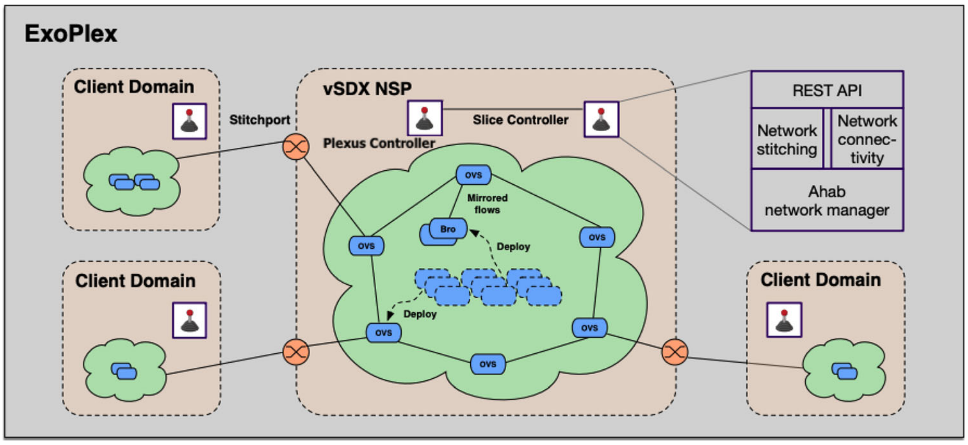


Fig. 3 Virtual software defined exchange (SDX) network architecture

In our scenario, the Exoplex Slice controller [56] runs as a docker container. Mobius has been enhanced to communicate with the ExoPlex Slice controller via its REST API to establish network connectivity between ExoGENI and Chameleon via layer 2 networks and to allocate bandwidth to individual workflows. Once connectivity is established, Mobius triggers REST API calls to publish network prefixes, sets up routes between network prefixes and dynamically applies different bandwidths as needed. Additionally, we have implemented a Python based interface that can be used to provision the required resources. This interface enables programmatic resource provisioning and is capable of spinning up resources, establishing connectivity and implementing network QoS policies on a per workflow ensemble level.

5.2 Pegasus ensemble manager

The Pegasus WMS can manage collections of related workflows, commonly referred to as ensembles, through a service called the Pegasus Ensemble Manager (Pegasus-EM) [57]. Pegasus-EM supports ensemble creation, workflow prioritization, workflow submission, throttling of concurrent executions, and ensemble level monitoring capabilities.

To support dynamic execution of workflow ensembles based on the continuous flow of data obtained from various sources, we have have extended Pegasus-EM with workflow triggering capability that supports three triggering modes (a) cron, (b) monitoring for local files, and (c) monitoring for web files.

– *Cron* This mode is similar to a cron job. On a predefined interval specified during the trigger's creation, Pegasus-EM executes a user-defined script that generates a new Pegasus workflow, which is in turn added to the targeted ensemble.
– *Monitoring local files* In this mode Pegasus-EM monitors a local directory for new files. Based on an interval specified during its creation, it checks for new files that match a file pattern and passes them to a user-defined workflow generation script that dynamically creates and plans a Pegasus workflow based on the incoming data. Pegasus-EM executes the workflow generation script and queues up the generated workflow for execution.
– *Monitoring web files* This triggering mode is similar to the local file mode. In this case, however, Pegasus-EM will monitor a remote web location (HTTP) for new files that match the provided file patterns.

An example of a Pegasus-EM trigger monitoring for web files is presented on Fig. 4. In the definition of the trigger the following parameters need to be specified.

– Ensemble: The targeted ensembe to which Pegasus-EM will queue up the new workflow
– Trigger: A unique name for the trigger
– Interval: The polling period that Pegasus-EM will check for changes
– Script: User-defined script that handles workflow generation
– Web_location: Web url of the remote repository

```
#!/bin/bash

pegasus-em create wind-ensemble

pegasus-em web-file-pattern-trigger
    --ensemble wind-ensemble
    --trigger wind_1min
    --interval 60s
    --script run_script.sh
    --web_location https://data.casa.umass.edu
    --file_patterns .*netcdf.tar.gz
    --timeout 60m
    --args -n 10
```

Fig. 4 Pegasus-EM web file trigger example

– File_patterns: A list of regex patterns that will be checked against the file names
– Timeout: After an optional timeout time has elapsed and no new files have appeared, the trigger will be deleted
– Args: An optional parameter for any extra arguments that need to be passed to the user-defined script

5.3 Prometheus monitoring

The Prometheus monitoring system [58] has been added to the DyNamo ecosystem. Mobius automatically configures the Prometheus node exporter [59] on each compute node to push system metrics to a Prometheus server hosted at RENCI. The metrics collected by

Prometheus give us the opportunity to dynamically take actions to ensure the infrastructure QoS. The actions include enabling compute, storage and network elasticity, i.e., growing and shrinking compute or storage resource pools and increasing or decreasing network properties of links. To visualize the collected data in a comprehensive and easy to understand way, an instance of Grafana [60] has been configured to pull the metric data from Prometheus and plot various graphs on a dashboard depicted in Fig. 5. To persist the data for long periods of time, we store the Prometheus collected metrics into an Elasticsearch [61] instance.

5.4 Operational effect on CASA's workflows

CASA workflows, due to their nature, can benefit from all of these enhancements to the DyNamo framework. As described in Sect. 4, CASA workflows need to process and respond to a continuous flow of weather radar data arriving at different rates. With the additions to the Pegasus-EM, CASA workflows can be started automatically as new files arrive at CASA's remote data repositories, with direct support by the DyNamo framework. In the past, this functionality was implemented using perl scripts that were invoked manually at the processing initiation stage. On top of this Pegasus-EM can alleviate pressure from the compute and network resources via its throttling mechanisms, by limiting ensembles that can flood the resources and allowing other ensembles to compete for their fair share. Moreover, with the introduction of the vSDX capabilities CASA workflow ensembles can now share the same layer 2 link in an isolated fashion. I.e, traffic from one workflow can only consume the maximum assigned bandwidth without impacting the network resources assigned to other workflows. CASA's workflows have different requirements that not only depend on the data being processed and the pipeline, but also the workflow configuration. With the vSDX, CASA can reserve a single layer 2 circuit to its data repository while distributing the network bandwidth based on the network subnet each worker node resides in. Each worker is assigned a specific CASA workflow ensemble by

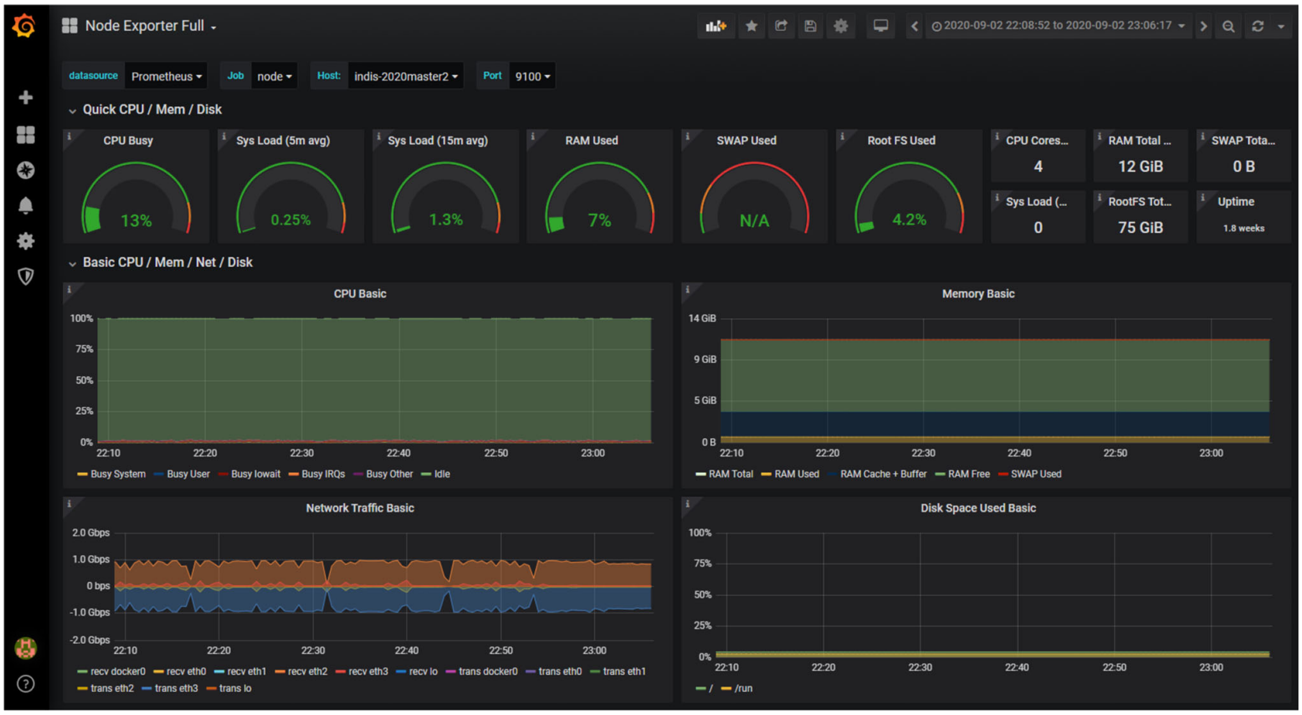


Fig. 5 Grafana dashboard depicting prometheus metrics

advertising a target workflow tag in its HTCondor advertisements. Previously this functionality was supported by reserving multiple layer 2 circuits on CASA's data repository, but due to the limited number of the available links this couldn't be achieved consistently.

6 Evaluation

6.1 CASA Pegasus workflows description

For the evaluation of the QoS impact we have selected two CASA workflows that produce nowcasts and wind speed estimates as described in Sect. 4. The workflow tasks include input data collection and product generation, visualization, contouring into polygon objects, spatial comparisons of identified weather features with infrastructure, and dissemination of notifications.

6.1.1 Nowcast

The Pegasus Nowcast workflow [62] computes short-term advection forecasts, as described in Sect. 4.1, by splitting grided reflectivity data into 31 grids and computing reflectivity predictions over the next 30 min. An abstract version of the workflow's DAG is presented in Fig. 6, which reveals that the size of the workflow doesn't depend on the input, and the number of *compute* tasks is fixed. The nowcast workflow contains 63 compute tasks in total, 1 task for splitting the input data into 31 individual grids, and then 62 independent tasks that compute the reflectivity and the respective contour images. All tasks run within a Singularity container that is managed by Pegasus and has a size of 153MB.

6.1.2 Wind

The Pegasus Wind Speed workflow [63] computes the maximum wind velocity, by combining multiple single radar output to account for single radar measurement inaccuracies (Sect. 4.2). An abstract version of this workflow's DAG is depicted in Fig. 7. To construct the input for the wind speed pipeline (preprocessing phase), single radar data files are accumulated over a variable time window (minimum 1 min), which regulates how often CASA produces maximum wind velocity contours, but also affects the size of the input of a single workflow run. As a result the first level of tasks (unzipping any zipped files) in the wind speed workflow (Fig. 7) depends on the number of input files, and thus the workflow has a variable number of tasks. The unzipping 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 Singularity container, 163MB in size.

6.1.3 Workflow testcases

To conduct our evaluation, both workflows are processing 30 min of pre-captured real weather data, which we replay as if they were arriving in real-time to simulate a production scenario from CASA's operations. The individual files consumed by the nowcast workflow are 9.6MB in size and the total size is 287MB. On the other hand the dataset for the wind workflows is comprised by files with individual size of $\sim$12MB, and the total dataset size is $\sim$6GB. For the

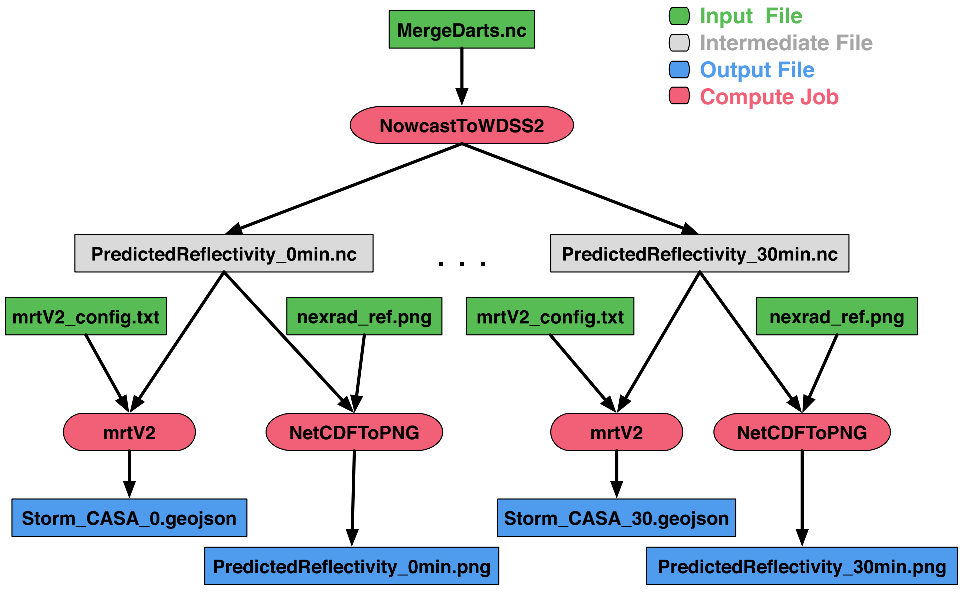


Fig. 6 CASA nowcast Pegasus workflow

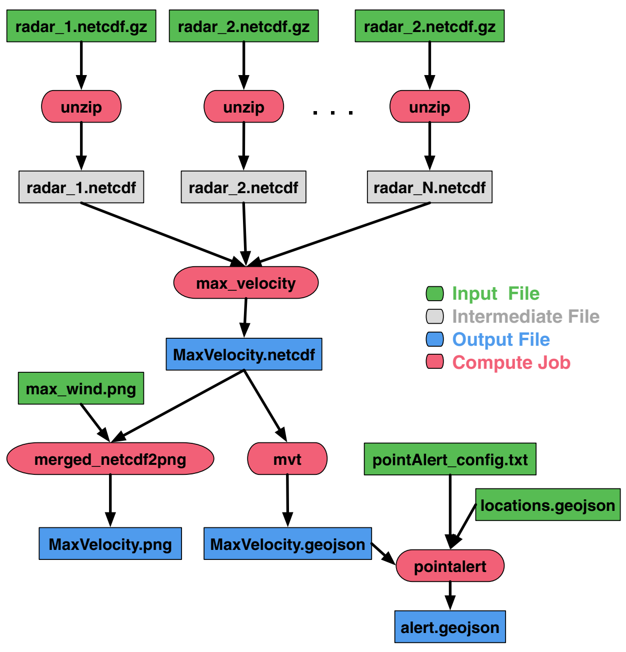


Fig. 7 CASA wind Pegasus workflow

two workflows we replay the data using an accumulation interval of 1 min and we are using Pegasus-EM to identify the newly added files and queue nowcast or wind workflow to their respective ensembles.

6.2 Experimental infrastructure

For evaluation, we used the DyNamo system to deploy a production scenario that is similar to CASA’s day to day operational radar data processing setup, and spreads across both ExoGENI and Chameleon testbeds (Fig. 8). In our setup Mobius and the vSDX controller are running within Docker containers at our USC Information Sciences Institute (ISI) Docker cluster.

Additionally, we are using one of CASA’s operational nodes at the University of North Texas (UNT) in Denton, TX, to host the data and submit the Pegasus workflows. The vSDX nodes and the workflow master node are located on ExoGENI at the University of Massachusetts Amherst (UMass) rack, on separate slices, while the compute nodes are located on Chameleon at TACC. To establish the layer 2 connectivity between the sites, Mobius “stitched” the UNT server to the workflow master node and instructed the vSDX controller to stitch the same node to the Chameleon nodes via the vSDX slice. The Chameleon compute cluster contains 5 nodes, 4 compute nodes and 1 storage node. 3 of the compute nodes reside in the 192.168.40.0/24 subnet while the other compute node and the storage node reside in subnet 192.168.30.0/24. Each node has 24 physical cores with hyperthreading (48 threads), 192GB RAM, 250GB SSD and is connected to a shared 10Gbps network. During the experiments we did not use the storage node to optimize for network traffic, but it was used as a next hop to route traffic from the subnet (192.168.40.0/24) that did not match the Chameleon stitchport’s subnet.

As we have shown in our initial evaluation of the DyNamo system [2] 144 and 48 HTCondor compute slots are enough to execute the nowcast and the wind speed workflow ensembles, respectively, without any compute imposed delays. Using HTCondor tags, the 3 compute nodes residing on the subnet 192.168.40.0/24 have been assigned to nowcast workflow tasks, while the node on the subnet 192.168.30.0/24 has been assigned to the wind speed workflow. Finally, all the stitchable networks were created with a network bandwidth of 1Gbps.

6.2.1 Software

On the submit node (where parts of the Dynamo system reside), the master node and the worker nodes we have installed HTCondor v8.8.9, and we customized its configuration to match the role of each node. In this setup, the workers are configured with partitionable slots and they advertise a workflow tag so they can be matched to the correct workflow. Additionally, on the submit node we have installed the nightly build of Pegasus v5.0.0 and the Apache HTTP server, to allow the workers to retrieve input files, configuration files and the application containers over HTTP. All of the workers use Singularity v3.6.1, and Mobius was used to provision compute resources on ExoGENI and Chameleon, and establish the network connections between ExoGENI, Chameleon and the CASA repository.

6.3 Workflow ensembles—network requirements

The two workflow ensembles present different network requirements due to the amount of tasks and the container transfers they instantiate. We profile the network utilization on CASA’s data repository at UNT, during the execution of the two workflow ensembles, using a dedicated 1Gbps layer 2 connection and the testcase datasets described in Sect. 6.1.

Figure 9 shows that the wind workflow ensemble is executed for $\sim$2100 s, has an average bandwidth usage of $\sim$200Mbps with a peak close to 240Mbps, while the total amount of data transferred is $\sim$44GBs.

Figure 10 depicts the network utilization imposed by the nowcast workflow ensemble. The nowcast calculations are occupying resources for $\sim$3200 s and they lead the network to congestion for prolonged periods of time. The average network utilization is close to 900Mbps with spikes reaching 960Mbps, and the total amount of data transferred is $\sim$280GBs.

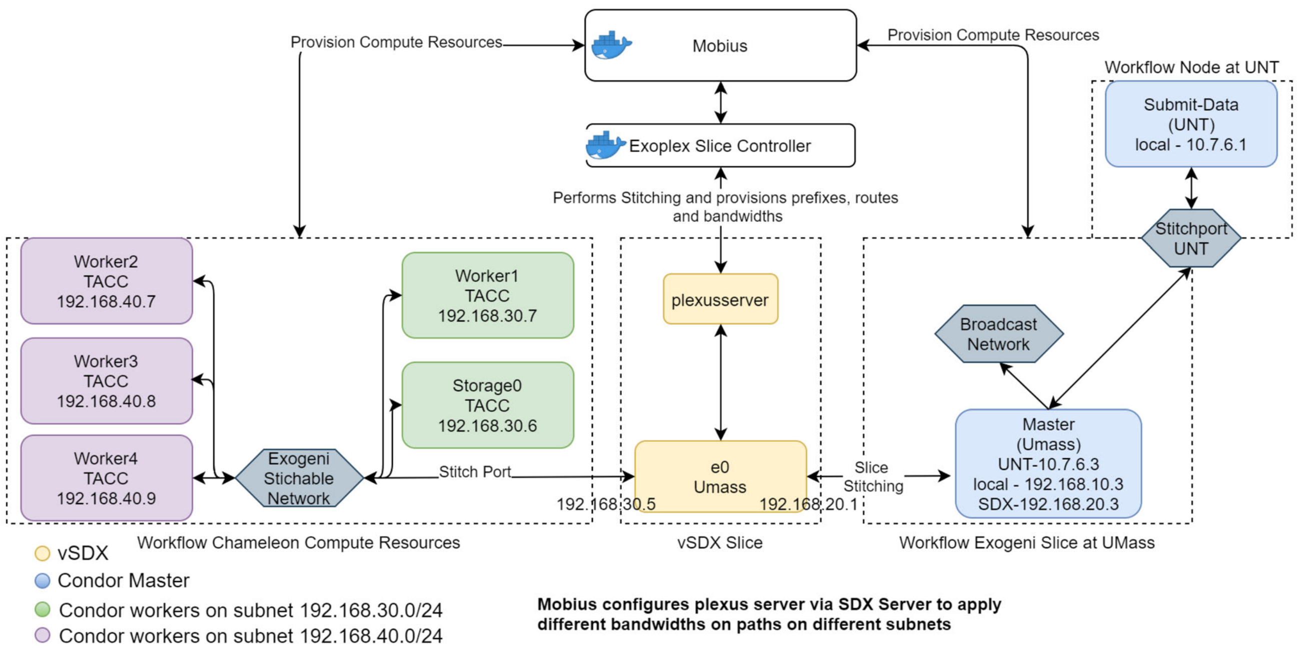

Mobius
Provision Compute Resources
Provision Compute Resources
Exoplex Slice Controller
Performs Stitching and provisions prefixes, routes and bandwidths
Workflow Node at UNT
Submit-Data (UNT) local - 10.7.6.1
Stitchport UNT
Worker2 TACC 192.168.40.7
Worker3 TACC 192.168.40.8
Worker4 TACC 192.168.40.9
Worker1 TACC 192.168.30.7
Storage0 TACC 192.168.30.6
Exogeni Stichable Network
Stitch Port
plexusserver
e0 Umass
192.168.30.5
192.168.20.1
Slice Stitching
Broadcast Network
Master (Umass) UNT-10.7.6.3 local - 192.168.10.3 SDX-192.168.20.3
Workflow Chameleon Compute Resources
vSDX Slice
Workflow Exogeni Slice at UMass
vSDX
Condor Master
Condor workers on subnet 192.168.30.0/24
Condor workers on subnet 192.168.40.0/24
Mobius configures plexus server via SDX Server to apply different bandwidths on paths on different subnets

Fig. 8 CASA vSDX workflow deployment

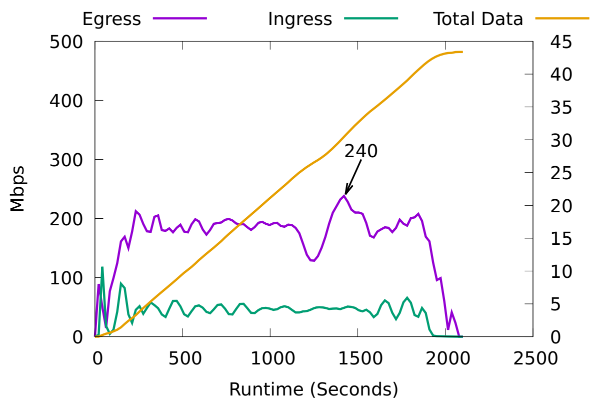


Fig. 9 Wind ensemble—network utilization

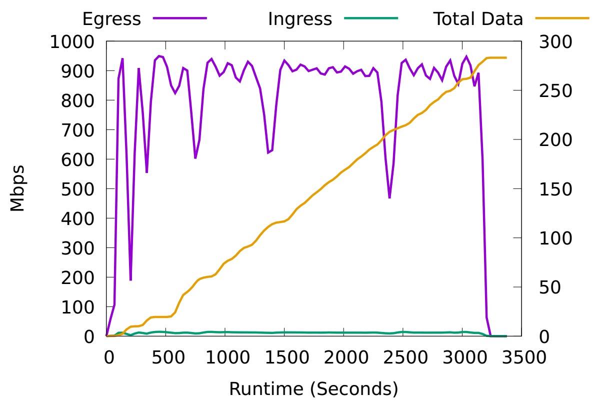


Fig. 10 Nowcast ensemble—network utilization

From Figs. 9 and 10 it is clear that the two workflow ensembles cannot fairly share the shame network resources without one of them impacting the other's QoS constraints, since the nowcast workflow ensemble will lead to prolonged network congestion. In our previous work [2] we used workflow runtime optimizations provided by Pegasus (e.g., task clustering) in order to lower nowcast's network requirements, but we did not apply them to this study since it is our goal to evaluate the effectiveness of DyNamo's new network QoS capabilities and ensemble management throttling techniques.

6.4 Experimental results

To conduct our vSDX study, we used three scenarios without throttling the ensembles via Pegasus-EM.

- 1 Gbps Dedicated: Each workflow ensemble had a dedicated 1 Gbps layer 2 link provisioned with Mobius, connecting the data repository to the compute resources.
- 1 Gbps vSDX-Shared: Both workflow ensembles shared the same 1 Gbps layer 2 link provisioned with Mobius, connecting the data and compute. No QoS policies where applied.
- 1 Gbps vSDX-Shared-QoS: Both workflow ensembles shared the same 1 Gbps layer 2 link provisioned with Mobius, connecting the data and compute. Individual QoS policies were applied to each workflow ensemble. 300Mbps for Wind and 700Mbps for Nowcast.

For each of the 3 scenarios, we repeated the workflow ensemble executions 5 times, leading to 900 workflow submissions and over 240,000 file transfers generating over 4TBs of network traffic. Figs. 11 and 12 present makespan statistics of the individual workflows of the ensembles, while Figs. 13 and 14 present statistics of the individual data transfers of the workflow ensembles.

To explore the space for different ensemble throttling configurations, we executed the two workflow ensembles under 42 unique configurations, varying the number of maximun concurrent wind and nowcast workflows. In every configuration we make sure that wind ensemble concurrency is greater or equal of the nowcast ensemble concurrency. We justify this decision to prune some of the possible configurations, on the fact that nowcast is congesting the network while wind doesn't, and by exploring that space we won't get useful insight.

- Wind ensemble: Concurrency was altered from 1 to 16 with a step of 2.
- Nowcast ensemble: Concurrency was altered from 1 to 12 with a step of 2.

This resulted in over 2000 additional workflow executions and over 13TBs of data transfers. Figs. 15 and 16 present heatmaps of the average individual workflow makespans of the two ensembles. Figs. 17 and 18 present heatmaps of the makespans of the two workflow ensembles. These results are discussion in Sect. 6.4.4

6.4.1 Dedicated link performance

To conduct our vSDX analysis, we first executed the nowcast and wind workflow ensembles under the best conditions possible, using 1 Gbps dedicated layer 2 connection and no Pegasus-EM throttling. For the wind ensemble Figs. 11 and 13 show a very consistent workflow duration ($\sim$300 s) and file transfer duration with very little deviation. On the other hand, since the nowcast workflow was creating network congestion we observe a noticeable deviation in both the workflow and file transfer durations

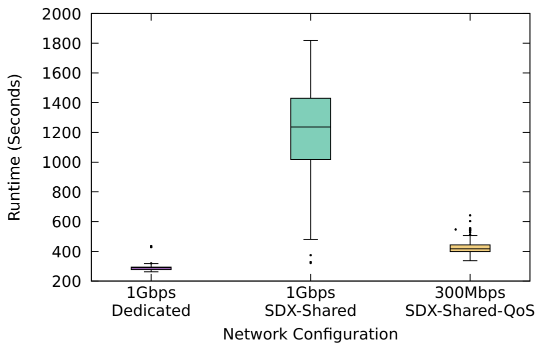


Fig. 11 Wind ensemble workflow makespans

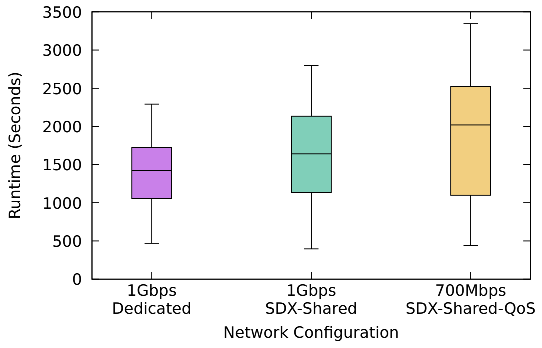


Fig. 12 Nowcast ensemble—workflow makespans

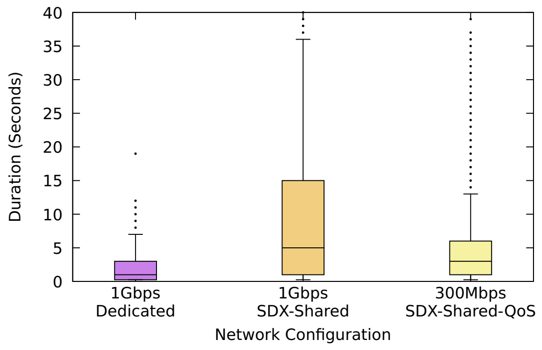


Fig. 13 Wind ensemble—data transfer durations

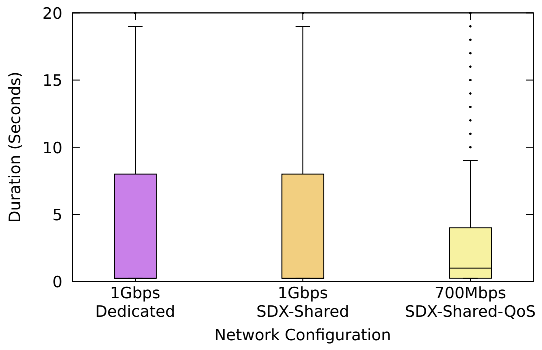


Fig. 14 Nowcast ensemble—data transfer durations

(Figs. 12 and 14). More than half of the workflows in the nowcast ensemble are completed within less than 1500 s. However, there are workflow executions that take from 500 s all the way to $\sim$2,400 s.

6.4.2 Uncontrolled network sharing

When we allow the two workflow ensembles to share the same network resources without any QoS policy, then we observe a very noticeable increase to the workflow makespans (Figs. 11, 12 middle). The most impacted are the workflows of the wind ensemble, where the average workflow duration increases from 300 s to over 1000 s,

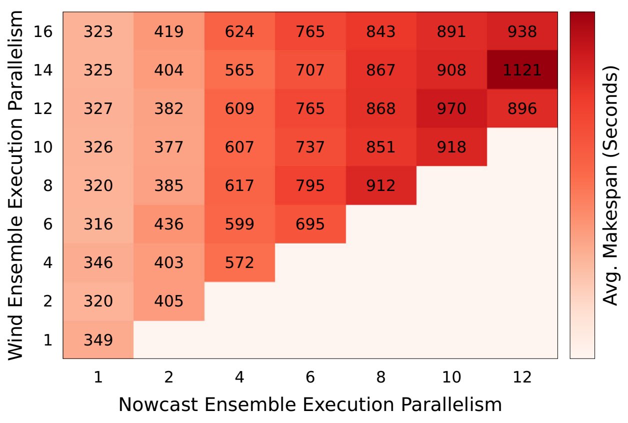


Fig. 15 Wind average workflow makespans

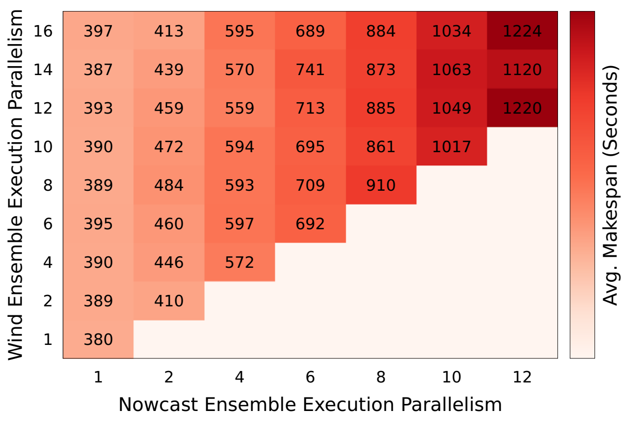


Fig. 16 Nowcast average workflow makespans

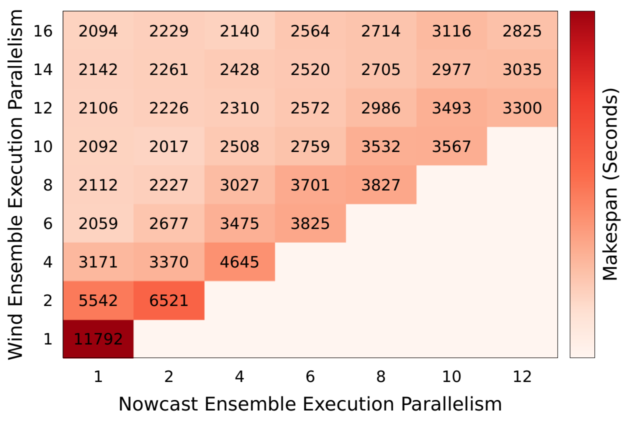


Fig. 17 Wind ensemble makespans

with some workflows completing execution close to 1800 s. This is an increase of over 500%. The impact of the additional network overhead is also visible in the nowcast workflows, although more subtle. The median nowcast workflow duration increased by about 200 s, while there were more workflows to the far ends of the spectrum.

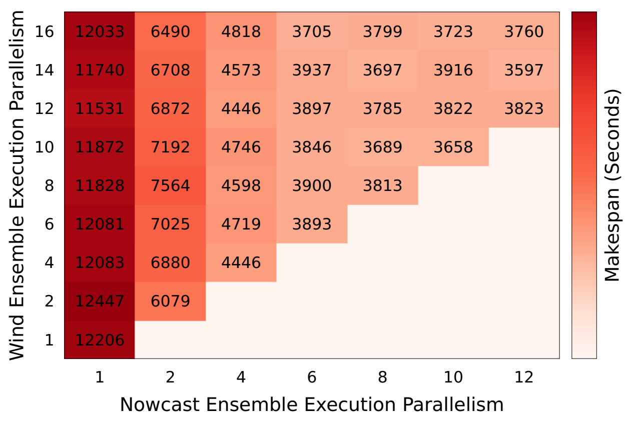


Fig. 18 Nowcast ensemble makespans

6.4.3 Applying SDX QoS policies

Finally, based on the network profiles presented in Figs. 9 and 10 we allocated 300Mbps of the available network bandwidth to the wind workflow ensemble and 700Mbps to the nowcast workflow ensemble, in an attempt to accommodate any network spikes of the wind ensemble. Both Figs. 11 and 13 (right) show an improvement of the wind workflow median makespan and data transfer durations. The wind ensemble's statistics have returned to a more consistent and predictable state with small deviation, similar to the execution conditions when a dedicated network link was used. Meanwhile, as it was expected, the median runtime of the nowcast workflows has increased since there is less available bandwidth (700Mbps) than what the workflow would optimally require ($\sim$900Mbps). However, the relative increase in comparison to the dedicated link runtimes is less than 60%. Something we did not expect to see was that even though the median duration of the file transfers in the nowcast ensemble increased by a few seconds, the transfers became more consistent, reducing the duration of the slowest transfers.

6.4.4 Applying QoS policies using Pegasus-EM

DyNamo, through the Pegasus Ensemble Manager throttling capabilities, it offers another opportunity to apply QoS policies on workflow ensembles that share network resources. We executed the wind workflow ensemble with workflow execution parallelism ranging from 1 to 16, and the nowcast ensemble with workflow execution parallelism ranging from 1 to 12. Figures 15 and 16 present the average workflow makespans of the two ensembles and we can distinguish a pattern for both cases. As we increase the concurrency of the nowcast ensemble (moving to the right) the average workflow execution time increases in both cases, and affects the turnaround times. For the wind workflows there is a 350% worst case increase, and for the nowcast workflows there is a 320% worst case increase. On the other hand increasing the concurrency of the wind ensemble (moving to the top) doesn't affect the execution times, which was expected.

Figures 17 and 18 show the makespans of the whole ensembles. In Fig. 17 as we increase the wind ensemble parallelism (moving to the top) and maintaining the nowcast ensemble parallelism equal to 1, the makespan of the wind ensemble decreases, but only until max concurrency equals 6. After that there is no improvement. However as we increase the nowcast ensemble parallelism (moving to the right) we need to increase wind ensemble parallelism again to improve the makespans. In Fig. 18 the makespan of the nowcast ensembles is governed only by the execution parallelism set for them. As we increase the nowcast ensemble execution parallelism (moving to the right) the makespan of the ensembles is reduced, until max concurrency of 8 is reached. After this point we don't observe any significant improvements, the network gets significantly saturated and the workflow turnaround time is tripled (Figs. 15, 16) compared to the non-congested state.

One thing that is notable with the Pegasus-EM throttling, is that we were able to get better workflow

turnaround times for the nowcast ensemble under a shared resource scenario, than in the 1 Gbps dedicated link scenario (Fig. 12 left boxplot). By setting the nowcast ensemble's workflow execution parallelism to 8 the average workflow turnaround time is under 1000 s (Fig. 16). This can be explained due to the fact that the nowcast ensemble overly saturates the network (Fig. 10) and by pacing the rate of the dispatched nowcast workflows we can achieve better average workflow execution times.

6.4.5 Discussion

Based on our experimentation DyNamo can aid to maintain the QoS of workflow ensembles when they are facing unfair network contention. Even though, TCP congestion algorithms attempt to provide a fair share of the network to all of the flows occupying it [64], they cannot provide it at the level of workflow ensembles. Workflow ensembles that flood the network with transfers are claiming a bigger chunk of the available bandwidth, impacting other ensembles with fewer transfers, which struggle to gain their network share. When vSDX policies can be enabled, DyNamo, through well-thought infrastructure deployment-design, can identify the individual flows that belong to specific workflow ensembles and effectively allocate bandwidth to meet their QoS expectations. In the case that vSDX policies cannot be applied, DyNamo, through the Pegasus-EM, can throttle workflow ensembles that congest the network and allow a more fair network allocation to the rest of the ensembles competing for their share.

In our current implementation all the aforementioned policies and techniques are considered not adaptive. Even though they can be changed and take effect on the fly during the execution of the workflow ensembles, there's no automated mechanism adapting the policies to maximize the utilization of the resources. A simple example is having one workflow ensemble actively using the resources. In this case it is not an optimal strategy to throttle the workflow ensemble's network bandwidth in the name of a future ensemble that needs higher network priority. The incorporation of Prometheus monitoring in the DyNamo framework opens up the possibility of applying reactive QoS policies by monitoring the state of the infrastructure in combination with workflow level feedback.

7 Conclusion

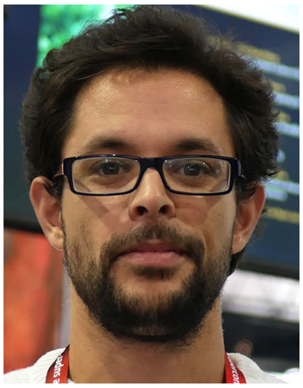
In this paper we introduced three new additions to the DyNamo system. These extensions address monitoring, infrastructure and operational challenges of CASA's distributed, atmospheric science workflows. The newly added Virtual Software Defined Exchange (vSDX) capabilities provide fine-grained control over the dynamically established networks, via link adaptation, flow prioritization and traffic control between endpoints. These policies can be an effective way to avoid unfair use of network resources. Even if single workflow ensembles are capable of flooding and congesting the network, other ensembles can maintain their own QoS requirements. To evaluate the QoS polices we deployed two of CASA's workflow ensembles (wind speed and nowcast) and we showed that even though the nowcast ensemble is capable of interfering with the wind ensemble, by applying the QoS policies the interference is removed and the wind ensemble's performance returns to levels close to the ones observed using a dedicated network link. Another contribution was the Pegasus Ensemble Manager (Pegasus-EM) extension. Pegasus-EM now supports file and time-based workflow triggering logic that allows CASA to automatically execute its workflows as new data arrive while managing the number of the concurrent workflows being executed. In our evaluation we showed that Pegasus-EM provides an alternative way of applying QoS policies using DyNamo and can promote a fairer sharing of both network and compute resources. This is essential for the infrastructures that don't offer Software Defined Network (SDN) support and QoS policies need to be applied. Finally we incorporated the Prometheus monitoring system into the DyNamo framework, providing comprehensive information about the status of the network and the compute resources, allowing CASA scientists to better understand the performance of their provisioned resources across the clouds. In the future, we plan to extend the DyNamo system's capabilities by stitching to more resource providers, supporting streaming workflows, developing new CASA workflows and provide mechanisms that will allow the applications to automatically evaluate the current pressure applied on the provisioned resources and make adjustments to the infrastructure without user intervention (e.g., change the QoS policies of workflow ensembles).

Acknowledgements 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. Baldin, I., Chase, J., Xin, Y., Mandal, A., Ruth, P., Castillo, C., Orlikowski, V., Heermann, C., Mills, J.: ExoGENI: a multi-domain infrastructure-as-a-service testbed, pp. 279–315. Springer, Cham (2016)
2. Lyons, E., Papadimitriou, G., Wang, C., Thareja, K., Ruth, P., Villalobos, J., Rodero, I., Deelman, E.,Zink, M., Mandal, A.: Toward a dynamic network-centric distributed cloud platform for scientific workflows: A case study for adaptive weather sensing. In: 2019 15th International Conference on eScience (eScience), pp. 67–76. (2019)
3. Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S.P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., Katz-Bassett, E.: Sdx: a software defined internet exchange. SIGCOMM **44**, 551–562 (2014)
4. Mambretti, J., Chen, J., Yeh, F.: Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn), In: 2015 international conference on cloud computing research and innovation (ICCCRI), pp. 73–79. (2015)
5. Amazon Elastic Compute Cloud. http://www.amazon.com/ec2
6. Microsoft Azure Cloud. https://azure.microsoft.com/en-us/
7. AWS CloudFormation. http://aws.amazon.com/cloudformation
8. OpenStack Heat Project. https://wiki.openstack.org/wiki/Heat
9. Baldin, I., Ruth, P., Wang, C., Chase, J. S.: The future of multiclouds: a survey of essential architectural elements, In: 2018 international scientific and technical conference modern computer network technologies (MoNeTeC), pp. 1–13. (2018)
10. Foster, I.: Globus online: accelerating and democratizing science through cloud-based services. IEEE Internet Comput. **15**(3), 70–73 (2011)
11. Liu, J., Pacitti, E., Valduriez, P., Mattoso, M.: A survey of data-intensive scientific workflow management. J. Grid Comput. **13**(4), 457–493 (2015)
12. Galante, G., Erpen De Bona, L.C., Mury, A.R., Schulze, B., Rosa Righi, R.: An analysis of public clouds elasticity in the execution of scientific applications: a survey. J. Grid. Comput. **14**(2), 193–216 (2016)
13. Coutinho, E.. F.., de Carvalho Sousa, F.. R.., Rego, P.. A.. L.., Gomes, D.. G.., de Souza, J.. N..: Elasticity in cloud computing: a survey. Ann.Telecommun. - annales des telecommunications **70**(7), 289–309 (2015)
14. Wang, J., AbdelBaky, M., Diaz-Montes, J., Purawat, S., Parashar, M., Altintas, I.: "Kepler + cometcloud: Dynamic scientific workflow execution on federated cloud resources (international Conference on Computational Science 2016, ICCS 2016, 6–8 June 2016. San Diego, California, USA), Proced. Comput. Sci. **80**, 700–711 (2016)
15. Ostermann, S., Prodan, R., Fahringer, T.: Dynamic cloud provisioning for scientific grid workflows. In: 2010 11th IEEE/ACM international conference on grid computing, pp. 97–104. (2010)
16. Mandal, A., Ruth, P., Baldin, I., Xin, Y., Castillo, C., Juve, G., Rynge, M., Deelman, E., Chase, J.: Adapting scientific workflows on networked clouds using proactive introspection, In: IEEE/ACM Utility and Cloud Computing (UCC). (2015)
17. Macker, J.P., Taylor, I.: Orchestration and analysis of decentralized workflows within heterogeneous networking infrastructures. Future Gener. Comput. Syst. **75**, 388–401 (2017)
18. Ramakrishnan, L., Koelbel, C., Kee, Y., Wolski, Y., Nurmi, Y., Gannon, D., Obertelli, G., YarKhan, A., Mandal, A., Huang, T. M., Thyagaraja, T. M., Zagorodnov, D.: Vgrads: enabling e-science workflows on grids and clouds with fault tolerance. In: Proceedings of the conference on high performance computing networking, storage and analysis, pp. 1–12. (2009)
19. Liu, Q., Rao, N. S. V., Sen, S., Settlemyer, B. W., Chen, H.-B., Boley, J. M., Kettimuthu, R., Katramatos, D.: Virtual environment for testing software-defined networking solutions for scientific workflows. In: Proceedings of the 1st international workshop on autonomous infrastructure for Science, ser. AI-Science'18. New York, NY, USA: Association for Computing Machinery. (2018). https://doi.org/10.1145/3217197.3217202
20. Ghahramani, M.H., Zhou, M., Hon, C.T.: Toward cloud computing qos architecture: analysis of cloud systems and cloud services. IEEE/CAA J. At. Sin. **4**(1), 6–18 (2017)
21. Varshney, S., Sandhu, R., Gupta, P.K.: Qos based resource provisioning in cloud computing environment: a technical survey. In: Singh, M., Gupta, P., Tyagi, V., Flusser, J., Ören, T., Kashyap, R. (eds.) Advances in computing and data sciences, pp. 711–723. Springer, Singapore (2019)
22. On-demand secure circuits and advance reservation system. https://doi.org/10.1145/2443416.2443420
23. Islam, M.,Huang, A. K., Battisha, M., Chiang, M., Srinivasan, S., Peters, C., Neumann, A.,Abdelnur,a.: Oozie: Towards a scalable workflow management system for hadoop. In: Proceedings of the 1st ACM SIGMOD workshop on scalable workflow execution engines and technologies, ser. SWEET '12. Association for Computing Machinery, New York, (2012). https://doi.org/10.1145/2443416.2443420
24. Senturk, I. F., Balakrishnan, P., Abu-Doleh, A., Kaya, K., Malluhi, Q., ., Çatalyürek, Ümit. V.: A resource provisioning framework for bioinformatics applications in multi-cloud environments. Future Gener. Comput. Syst. **78**, 379–391 (2018)
25. Malawski, M., Figiela, K., Bubak, M., Deelman, E., Nabrzyski, J.: Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. Sci. Pogram. **29**, 158–169 (2015)
26. Abrishami, S., Naghibzadeh, M., Epema, D.H.: Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Gener. Comput. Syst. **29**(1), 158–169 (2013)
27. Dickinson, M., Debroy, S., Calyam, P., Valluripally, S., Zhang, Y., Bazan Antequera, R., Joshi, T., White, T., Xu, D.: Multi-cloud performance and security driven federated workflow management. IEEE Trans.Cloud Comput. **9**, 240–257 (2018)
28. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., Wenger, K.: Pegasus: a workflow management system for science automation (funding Acknowledgements: NSF ACI SDCI 0722019, NSF ACI SI2-SSI 1148515 and NSF OCI-1053575). Future Gener. Comput. Syst. **46**, 17–35 (2015)

29. National Energy Research Scientific Computing Center (NERSC). https://www.nersc.gov
30. Oak Ridge Leadership Computing Facility. https://www.olcf.ornl.gov
31. Extreme science and engineering discovery environment (xsede). http://www.xsede.org
32. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., Foster, I., Gardner, R., Wilde, M., Blatecky, A., McGee, J., Quick, R.: The open science grid. J.Phys. Conf.Ser. **78**, 012057 (2007)
33. Amazon.com, Inc.: Amazon Web Services (AWS). http://aws.amazon.com
34. Keahey, K., Riteau, K., Stanzione, D., Cockerill, K., Mambretti, J., Rad, P., Ruth, P.: "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
35. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurr. Comput. **17**(2–4), 323–356 (2005)
36. Gunter, D., Deelman, E., Samak, T., Brooks, C., Goode, M., Juve, G., Mehta, G., Moraes, P., Silva, F., Swany, M., Vahi, K.: "Online workflow management and performance analysis with stampede," in *7th International Conference on Network and Service Management (CNSM-2011)*, (2011)
37. Bayucan, A., Henderson, R. L., Lesiak, C., Mann, B., Proett, T., Tweten, D.: Portable batch system: external reference specification. In: Technical report, MRJ technology solutions, vol. 5, (1999)
38. Simple Linux Utility for Resource Management. http://slurm.schedmd.com/
39. Raman, R., Livny, M., Solomon, M.: "Matchmaking: distributed resource management for high throughput computing," in *Proceedings. The Seventh International Symposium on High Performance Distributed Computing (Cat. No.98TB100244)*, pp. 140–146(1998)
40. Frey, J., Tannenbaum, T., Foster, I., Livny, M., Tuecke, S.: "Condor-G: A computation management agent for multi-institutional grids," in Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC), pp. 7–9. California, August, San Francisco (2001)
41. Mobius Github Repository. https://github.com/RENCI-NRIG/Mobius
42. Internet 2. https://www.internet2.edu/
43. The energy science network. https://www.es.net/
44. OpenStack Cloud Software. http://openstack.org
45. McLaughlin, D., Pepyne, D., Chandrasekar, V., Philips, B., Kurose, J., Zink, M., Droegemeier, K., Cruz-Pol, S., Junyent, F., Brotzge, J., Westbrook, D., Bharadwaj, N., Wang, Y., Lyons, E., Hondl, K., Liu, Y., Knapp, E., Xue, M., Hopf, A., Kloesel, K., DeFonzo, A., Kollias, P., Brewster, K., Contreras, R., Dolan, B., Djaferis, T., Insanic, E., Frasier, S., Carr, F.: Short-wavelength technology and the potential for distributed networks of small radar systems. Bull. Am. Meteorol. Soc. **90**(12), 1797–1818 (2009). https://doi.org/10.1175/2009BAMS2507.1.
46. Lyons, E. J., Zink, M.,Philips, B.: Efficient data processing with exogeni for the casa dfw urban testbed. In: 2017 IEEE international geoscience and remote sensing symposium (IGARSS), pp. 5977–5980, (2017)
47. Li, L., Schmid, W., Joss, J.: Nowcasting of motion and growth of precipitation with radar over a complex orography. J. Appl. Meteorol. **34**(6), 1286–1300 (1995)
48. Ruzanski, E., Chandrasekar, V.: Weather radar data interpolation using a kernel-based lagrangian nowcasting technique. IEEE Trans. Geosci. Remote Sens. **53**(6), 3073–3083 (2015)
49. Yao, Y., Cao, Q., Farias, R., Chase, J., Orlikowski, V., Ruth, P., Cevik, M., Wang, C., Buraglio, N.: Toward live inter-domain network services on the exogeni testbed. In: IEEE INFOCOM 2018—IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 772–777, (2018)
50. Zeek Github Repository. https://github.com/zeek/zeek
51. Ahab Github Repository. https://github.com/RENCI-NRIG/ahab
52. Linux Foundation Collaborative Projects. https://www.openvswitch.org/
53. Open flow SDN Controllers. https://en.wikipedia.org/wiki/List_of_SDN_controller_software/
54. Ryu SDN Controller. https://ryu-sdn.org/
55. Ryu Rest Router. https://github.com/faucetsdn/ryu/blob/master/ryu/app/rest_router.py
56. Exoplex Github Repository. https://github.com/RENCI-NRIG/CICI-SAFE
57. Pandey, S., Vahi, K., Ferreira da Silva, R., Deelman, E., Jian, M., Harrison, C., Chu, A., Casanova, A.: Event-based triggering and management of scientific workflow ensembles, In: 2018, poster presented at the HPC Asia 2018: Tokyo, Japan. http://sighpc.ipsj.or.jp/HPCAsia2018/poster/post102s2-file1.pdf
58. Prometheus. https://prometheus.io/
59. Node Exporter. https://prometheus.io/docs/guides/node-exporter/
60. Grafana. https://grafana.com/
61. ELK stack. (2018). https://www.elastic.co/elk-stack
62. Scitech, CASA Nowcast Pegasus Workflow. https://github.com/pegasus-isi/casa-nowcast-workflow
63. Scitech: CASA Wind Pegasus Workflow. https://github.com/pegasus-isi/casa-wind-workflow
64. Hasegawa, G., Murata, M., Miyahara, H.: Fairness and stability of congestion control mechanisms of tcp. In: IEEE INFOCOM '99. Conference on computer communications. Proceedings. Eighteenth annual joint conference of the IEEE computer and communications societies. The future is now (Cat. No.99CH36320), vol. 3, pp. 1329–1336, (1999)



George Papadimitriou is a Computer Science PhD student at the University of Southern California, and a Graduate Research Assistant in the Science Automation Technologies group at the USC Information Sciences Institute.His research interests lie within the intersection of Data Intensive Applications and Distributed Computing. He received his BS in Electrical and Computer Engineering from the National Technical University of Athens.

Eric Lyons is a Research Fellow at the University of Massachusetts Amherst in the Department of Electrical and Computer Engineering (ECE). Eric has been a member of the Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere (CASA) since 2004. He has served as the radar operations lead and since 2011 also the chief Systems and Software engineer for the CASA DFW living lab in north Texas and is additionally responsible for IT, security, and data management. Eric's research also extends into cloud computing and networking, with a focus on scalable workflow management and the development of toolsets to assist data scientists. In the last few years, Eric has lead development of tailored weather extractions for GIS and aviation and created a flight path routing suite to dynamically steer unmanned aircraft around impactful meteorological and non-meteorological objects.

Cong Wang is a senior network and systems researcher at RENCI, University of North Carolina at Chapel Hill. His research focuses on cloud computing, networking, and distributed systems. He obtained his PhD in department of Electrical and Computer Engineering at University of Massachusetts Amherst.

Komal Thareja is a Distributed Systems Software Engineer at RENCI, University of North Chapel Hill.

Ryan Tanaka is a research programmer in the Science Automation Technologies group at ISI. He received his Master's degree in Computer Science from the University of Hawaii at Manoa in 2019. His interests include distributed systems and data intensive applications.

Paul Ruth is an Assistant Director in Network Research and Infrastructure at RENCI, UNC-Chapel Hill. His research interests include building and using dynamic cloud computing and network testbeds for software defined exchanges targeted at data driven scientific workflows. He earned his Ph.D. degree in Computer Science from Purdue University in 2007.

Ivan Rodero is a Research Computer Scientist at the Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research focuses on data-driven science and engineering, high performance parallel and distributed computing and advanced cyberinfrastructure. He has received various awards for his research and publications, including the IEEE TCSC Young Achievers in Scalable Computing Award. He is senior member of IEEE and ACM.

Ewa Deelman is a Research Professor at the USC Computer Science Department and a Research Director of the Science Automation Technologies group at the USC Information Sciences Institute (ISI). Her group has lead the design and development of the Pegasus Workflow Management software and conducts research in job scheduling and resource provisioning in distributed systems, workflow performance modeling, provenance capture, reproducibility, and the use of cloud platforms for science. Her group has also experience in deploying and leveraging other advanced cyberinfrastructure for science. Dr. Deelman received her PhD in Computer Science from the Rensselaer Polytechnic Institute in 1998 and before joining ISI in 2000 she held a postdoc at the UCLA Computer Science Department. She is an AAAS and IEEE Fellow.

Michael Zink is an Associate Professor in the Electrical and Computer Engineering Department at the University of Massachusetts in Amherst. He received his PhD in 2003 from the Multimedia Communications Laboratory at Darmstadt University of Technology. He works in the areas of future multimedia systems, Internet architectures, and sensor networks.

Anirban Mandal serves as the Assistant Director for network research and infrastructure at Renaissance Computing Institute (RENCI) at University of North Carolina, Chapel Hill. He leads several efforts in cyberinfrastructure research in support of science. His research interests lie in the areas of distributed systems, cloud computing, networking, and data-driven scientific workflows. His research deals with resource provisioning, scheduling, performance analysis, machine learning, and anomaly detection for large scale scientific cyberinfrastructures, next generation networks and experimental testbeds. Prior to joining RENCI, he earned his PhD degree in Computer Science from Rice University in 2006 and a Bachelor's degree in Computer Science & Engineering from IIT Mumbai, India in 2000.