ADON: Application-Driven Overlay Network-as-a-Service for Data-Intensive Science

Ronny Bazan Antequera, Prasad Calyam, Saptarshi Debroy, Longhai Cui,
Sripriya Seetharam, Matthew Dickinson, Trupti Joshi, Dong Xu, Tsegereda Beyene
University of Missouri-Columbia, USA; Cisco Systems, USA
Email: {*rcb553, calyamp, debroysa, lcyvb, ssn68, dickinsonmg, joshitr, xudong*}*@missouri.edu, tbeyene@cisco.com*

***Abstract*—Campuses are increasingly adopting hybrid cloud architectures for supporting data-intensive science applications that require "on-demand" resources, which are not always available locally on-site. Policies at the campus edge for handling multiple such applications competing for remote resources can cause bottlenecks across applications. These bottlenecks can be proactively avoided with pertinent profiling, monitoring and control of application flows using software-defined networking and pertinent selection of local or remote compute resources. In this paper, we present an "Application-driven Overlay Network-as-a-Service" (ADON) that manages the hybrid cloud requirements of multiple applications in a scalable and extensible manner by allowing users to specify requirements of the application that are translated into the underlying network and compute provisioning requirements. Our solution involves scheduling transit selection, a cost optimized selection of site(s) for computation and traffic engineering at the campus-edge based upon real-time policy control that ensures prioritized application performance delivery for multi-tenant traffic profiles. We validate our ADON approach through an emulation study and through a wide-area overlay network testbed implementation across two campuses. Our workflow orchestration results show the ADON effectiveness in handling temporal behavior of multi-tenant traffic burst arrivals using profiles from a diverse set of actual data-intensive applications.**

***Keywords*—Overlay Network-as-a-Service, Distributed Resource Orchestration, Data-intensive Applications Multi-tenancy**

I. Introduction

Data-intensive applications in science fields such as bioinformatics, climate modeling, particle physics and genomics generate vast amounts (peta-byte scale) of data that needs to be processed in some cases with real-time analysis. The general data processing facilities and specialized compute resources do not always reside at the data generation sites on campus, and data is frequently transferred in real-time to geographically distributed sites (e.g., remote instrumentation site, federated data repository, public cloud) over wide-area networks. Moreover, researchers share workflows of their data-intensive applications with remote collaborators for multi-disciplinary initiatives on multi-domain networks [1] leading to a shift in design of advanced network architectures [2].

Current campus network infrastructure policies place stringent security policies at the edge router/switch and install firewalls to defend the campus local-area network (LAN) from potential cyber attacks. Such defense mechanisms significantly impact research traffic especially in the case of data-intensive science applications whose flows traverse wide-area network (WAN) paths. This has prompted campuses to build Science DMZs (de-militarized zones) [1] with high-speed (1 - 100 Gbps) programmable networks to provide dedicated network infrastructures for research traffic flows that need to be handled in parallel to the regular enterprise traffic, due to the inflexible nature of traditional policies.

This work was supported by the National Science Foundation under awards: ACI-1246001 and ACI-1245795, and Cisco Systems. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Cisco Systems.

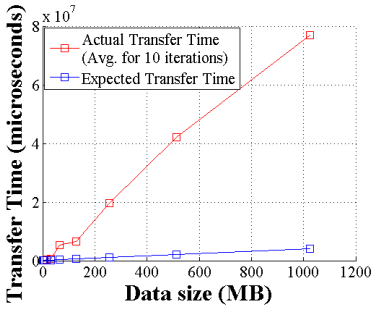


Fig. 1: Illustration to show the need for application performance visibility and control over a wide-area network path

The advanced network infrastructure components in Science DMZs that help with high-performance networking to remote sites and public clouds include: (i) software-defined networking (SDN) based on programmable OpenFlow switches [3], (ii) RDMA over Converged Ethernet (RoCE) implemented between zero-copy data transfer nodes [4] for data transport acceleration, (iii) multi-domain network performance monitoring using perfSONAR active measurement points [5], and (iv) federated identity and access management using Shibboleth-based entitlements [6].

In addition to network infrastructure that is being shared, there is an increasing push to share computational resources federated across enterprises that are connected with high-speed network connections. These hybrid cloud-like computational resources make use of virtualization and management platforms such as OpenStack [7]. The ability to schedule resources, including both network and computation in an end-to-end workflow scenario across multiple domains allows the completion of complex tasks in a shorter timeframe.

Moreover, when multiple applications that access hybrid cloud resources compete for the exclusive and limited Science DMZ resources, the policy handling of research traffic can cause a major bottleneck at the campus edge router and impact the performance across applications. Figure 1 illustrates an actual problem scenario we faced when we initiated a file transfer as part of a research application (Advanced Data Transfer Service) using RoCE protocol on the high-speed overlay between the University of Missouri (MU) and the Ohio State University (OSU) [8]. As shown, the achieved transfer time was substantially low compared to the expected theoretical transfer time. Upon investigation, we discovered

that our application's traffic was being rate limited to 2 Gbps at OSU edge router even though the link's capacity was capable of 10 Gbps speeds. This highlights two issues: i) the complexity of inter-domain network configuration is frequently not known by all parties concerned, thus leading to inefficient and often problematical network performance and ii), the RoCE protocol assumes a 10 Gbps link availability and is highly sensitive to packet loss and, as such, application performance suffers severely.

As evident from the above mentioned scenario, there is a need to provide dynamic Quality of Service (QoS) control of Science DMZ network resources versus setting a static rate limit affecting all applications. The dynamic control should have awareness of research application flows with urgent or other high-priority computing needs, while also efficiently virtualizing the infrastructure for handling multiple diverse application traffic flows. The virtualization obviously should not affect the QoS of any of the provisioned applications. Further, advanced network services should be easy-to-use for data-intensive application users, who are frequently not versed in the underlying infrastructure resource architecture and configuration.

Our work in this paper aims to solve the network path provisioning problem to meet unique needs of data-intensive applications in an federated resource environment, while making network programmability related issues a non-factor for the users. More specifically, we present a new "Application-Driven Overlay Network-as-a-Service" (ADON) architecture to intelligently provision on-demand network resources by performing a direct binding of applications to infrastructure. Additionally, the path selection for the network is based upon a cost-calculation dependent on the available network and compute resource requirements for a particular application.

The novelty and contributions of our work are as follows: we detail how "network personalization" can be performed using a concept of "custom templates" to catalog and handle unique profiles of application workflows. We also detail a multi-tenant architecture for real-time policy control of an "Overlay Network-as-a-Service" through a "Network Flowvisor - Virtual Tenant Handler" (NF-VTH) and "Compute Hypervisor - Virtual Tenant Handler" (CH-VTH). Both leverage awareness of the overall "load state" at the campus edge, and the individual application "flow state" using software-defined performance monitoring integration within the overlay network paths. Using the concepts of custom templates, NF-VTH and CH-VTH, ADON can manage the hybrid cloud requirements of multiple applications in a scalable and extensible manner. It ensures predictable application performance delivery by scheduling transit selection (choosing between regular Internet or high-performance Science DMZ paths), selection of compute processing in terms of local or remote location dependent upon the availability of computation resources, and traffic engineering (e.g., rate limit queue mapping based on application-driven requirements) at the campus-edge. We highlight the application of our ADON approach in the context of several real-time data-intensive applications, and an exemplar data-intensive knowledge based application.

The remainder paper is organized as follows: Section II presents related work. Section III describes the network personalization through custom templates for exemplar application workflows. Section IV details ADON's architecture featuring NF-VTH and CH-VTH. Section V presents ADON-orchestrated network and compute resource allocation algorithms. Section VI describes ADON effectiveness evaluation with an emulation study and a real-testbed implementation. Section VII concludes the paper.

II. Related Work

There is a need for simple and scalable end-to-end network architectures and implementations that enable applications to use wide-area networks most efficiently; and possibly control intermediate network resources to meet Quality of Service (QoS) demands [10]. A number of approaches, such as [11] and [12] have been developed that allow applications to bypass campus firewall rulesets by adopting programmable management technologies such as SDN. In [11], the authors describe a protocol implementation as part of a communication middleware offering flow control, connection management, and task synchronization, thereby maximizing the parallelism of RDMA operations. Whereas, authors in [12] propose an incrementally deployable Data-Centric Network Architecture (DCNA) between the application and transport layers to efficiently connect these layers and support mobility, multi-homing, and ease the adoption of new applications and networking technologies. Legacy data transfer methods have fundamental scalability issues [13] with regards to data transfer to the cloud for processing and data archival. Existing studies such as [14] recognizes similar application-driven network virtualization issues at network-edges, and have proposed new architectures based on SDN principles. In [14], application level requirements are programmed using an inter-domain controller implemented using OpenFlow [3], and a custom-built extensible session protocol is used to provide end-to-end virtual circuits across campus DMZs.

Although these schemes propose novel cyberinfrastructure deployment strategies for efficient resource management, data-intensive application QoS requirements-specific resource management techniques were lacking. Studies such as [15]–[18] are among a few that address the data-intensive applications' QoS requirement issues. Among these, in [15], a work closely related to ours, QoS parameters are programmed dynamically based on high-level requirements of different kinds of application traffic. The authors argue (similar to our argument in context of Figure 1) that there is a need for dynamic QoS configuration based on application needs, and current practice of manual configuration by network administrators hinders application performance. In [16], the authors propose a new controller design for QoS based routing of multimedia traffic flows. In [17], QoS requirements are discussed in an SDN environment for end-to-end scientific applications. Many QoS aware automated network convergence schemes are proposed for a purely cloud computing context [15]. Authors in [18] propose new methods for the estimation of execution time when distributed computing resources are utilized in a dynamic provisioning setup.

There are some notable works such as [19]–[24] for dynamic resource provisioning in cloud environment that take into consideration the diverse resource requirements of data-intensive applications. In particular, authors in [19] have identified that flexible resource management is a key factor in any data center for efficient provisioning of resources. Hence, orchestration of hybrid cloud infrastructure requires fine-grained control of resources, especially when resource discovery and reservation are integral in the provisioning process. In this context, cloud infrastructure scheduling plays an important role during on-demand and dynamic provisioning of computing resources [20] in single or large heterogeneous multi-cloud infrastructures [21]. Authors in [22] and [23] consider implications of high-performance computing (HPC) and cloud computing paradigms for data-intensive applications in order to provide Infrastructure-as-a-Service for data processing workflows in public clouds. In [22], the authors connect high-bandwidth radar sensor networks with computational and

storage resources in a cloud platform to orchestrate an end-to-end data-intensive application. Authors in [23] propose advanced virtualization techniques for a HPC cloud platform by enabling advanced accelerators such as GPUs and high-speed, low latency communication through InfiniBand between virtual machines. In [24], the issues surrounding the automatic deployment of HPC and database applications in a hybrid cloud environment are discussed.

In contrast to these above works, using SDN for "personalization" as discussed in our preliminary work [25] within hybrid cloud computing architectures for on-demand and concurrent application handling for accelerated performance is still under-explored. Network overlays for data-intensive applications with real-time or urgent computing require more fine-grained control over network resources. This is because the main goal of such application support involves fast and reliable movement of data-intensive transfers with guaranteed QoS based upon the specific requirements of the application prioritization algorithm driven by the priorities. Our work in this paper is focused on the requirements and challenges at the campus-edge. We uniquely handle Science DMZ resources and related high-performance network configurations (e.g., rate limit queue mappings) that need to be controlled in real-time for meeting hybrid cloud computing needs of diverse data-intensive application flows.

III. Network Personalization for Applications within ADON

As discussed previously, handling diverse data-intensive application requirements at the campus edge can be challenging due to the complexity of their cyberinfrastructure needs. In order to address such challenges, efficient resource management through "application performance visibility and control" within the provisioned resources for individual applications is fundamental. This can be achieved by simple and scalable resource provisioning based on incremental experiences stored in templates for its reuse and/or customization according to new applications' requirements. Thus, our approach involves maintaining a catalog of application profiles in terms of Resource Specifications ($RSpecs$) and Quality Specifications ($QSpecs$). In addition, policies should be determined for the extent to which programmable capabilities at the campus edge can be used to "personalize" the network overlay setup based on: (a) the individual application $RSpecs$ and $QSpecs$, and (b) the temporal behavior of multi-tenant traffic burst arrivals.

In the following subsections, we first describe the concept of custom templates that can be used within ADON to develop a catalog of application profiles. Following this, we apply the custom template concept for exemplar data-intensive application workflows with diverse QoS requirements such as Real-Time applications and Knowledge Base applications.

A. Custom Templates

Our concept of custom templates within ADON is similar to the best practices such as Amazon Web Services (AWS) Machine Image (AMI) [26] and $RSpecs$ in the NSF-supported Global Environment for Network Innovations (GENI) [27]. Studies such as [28] also suggest the value of using of templates that can allow for composing and executing workflow pipelines for data-intensive applications in a reusable manner.

Figure 2 shows how custom templates can be used as part of the sequential steps of ADON auto-orchestration during on-demand resource provisioning for a data-intensive application flow that needs an overlay network path. The details of the steps in ADON orchestration are as follows: First, a researcher

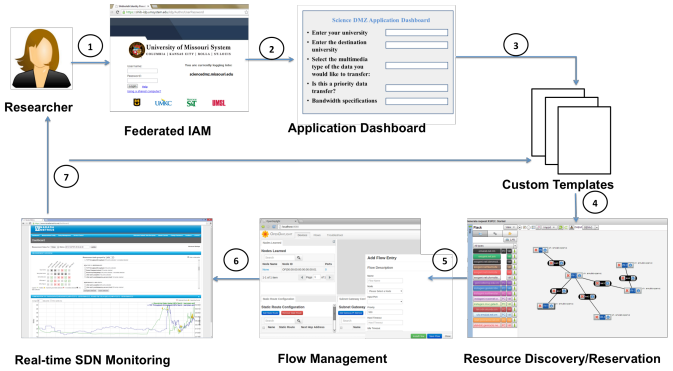
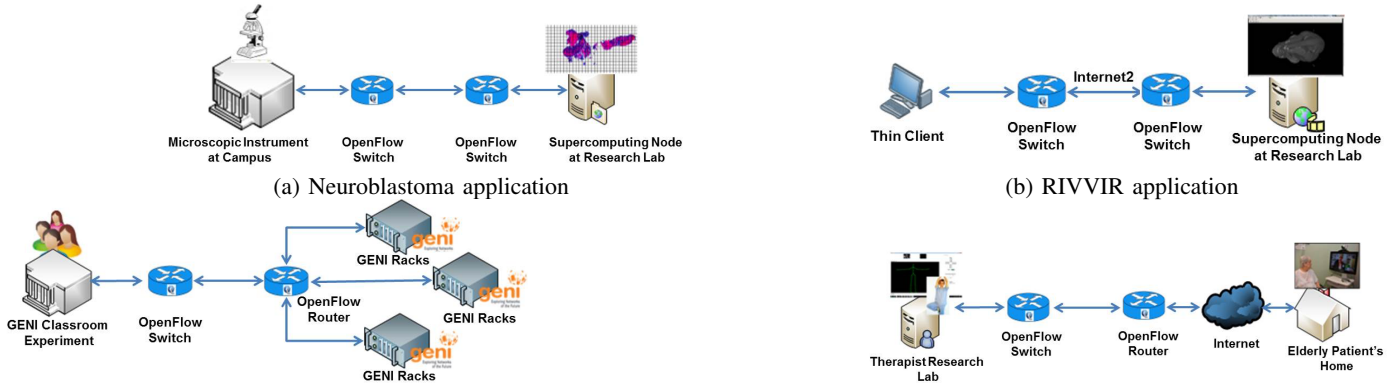


Fig. 2: Sequential workflow of ADON during on-demand resource provisioning for a data-intensive application flow

of a data-intensive application can securely request the ADON by authenticating with a Federated Identity and Access Management (Federated IAM) system that uses Shibboleth-based entitlements [6]. Such Federated IAM systems are necessary to handle multi-institutional policy specifications pertaining to cases such as: (i) How can a data-intensive application user at Campus A be authenticated and authorized to reserve HPC resources or other scientific instruments at a remote Campus B? (ii) How can a OpenFlow controller at one campus be authorized to provision flow spaces within a backbone network in an on-demand manner? (iii) Who can subscribe to the performance measurements related to a data-intensive application to monitor workflow status and track/troubleshoot any bottleneck anomaly events?

Subsequently, the researcher provides his/her data intensive application handling specifications through a simple and intuitive application dashboard mashup. The specifications can include details such as destination host (i.e., remote collaborator or remote instrument site) and application type (e.g., remote interactive volume visualization, video streaming, file transfer or compute resource reservation). Next, the application specifications are subsequently matched to a custom template that corresponds to the requested application type for resource discovery/reservation of the necessary compute and network resources.

The custom template can then be pre-configured by a "Performance Engineer" to apply specific resource descriptions and associated campus policies that can be interpreted by a network flowvisor (i.e., proxy for OpenDaylight, POX [29] or Ryu [30]) to instantiate flows on intermediate OpenFlow switches, and by a compute hypervisor to instantiate virtual machines within a data center. We refer to a Performance Engineer as a person who serves as the primary "keeper" and "helpdesk" of the Science DMZ equipment, and the success of this role is in the technician's ability to augment traditional System/Network Engineer roles on campuses and serve high-throughput computing needs of researchers. In addition to helping the Performance Engineer with the resource configuration, custom templates also help in configuring real-time network monitoring within the overlay network path to provide the performance visibility to define triggers for dynamic resource adaptation. Moreover, performance bottlenecks such as those observed in Figure 1 can be avoided through use of custom templates, and in exception cases where end-to-end QoS configuration is not possible, bottlenecks can be relatively more easily discovered and overcome. Manual



(a) Neuroblastoma application

(b) RIVVIR application

(c) GENI classroom experiments application

(d) ElderCare-as-a-Service application

Fig. 3: Data movement scenarios of various data-intensive applications that need network personalization

interventions that require the Performance Engineer attention can be minimized in cases where custom templates can be automatically re-applied through user 'self-service', if a similar specification was successfully handled previously.

B. Custom Templates for Real-Time Applications

1) Neuroblastoma Data Cutter Application: As shown in Figure 3(a), the workflow of the Neuroblastoma application [8] consists of a high-resolution microscopic instrument on a local campus site (at MU) generating data-intensive images that need to be processed in real-time to identify and diagnose Neuroblastoma (a type of cancer) infected cells. The processing software and HPC resources are available remotely at OSU, and hence images from MU need to be transferred in real-time to the remote OSU campus. To handle the large scale data transfers, the application relies on advanced file transfer protocols such as RoCE and GridFTP technologies that support parallel TCP flows between the two campuses. A corresponding Neuroblastoma application template can be given as: (i) $RSpec$ - high-resolution microscope instrument that is connected to remote GPU and storage resources, and (ii) $QSpec$ - high flow throughput with no packet loss and high flow priority to provide fast-enough application response time for a histopathological evaluator.

2) Remote Interactive Volume Visualization Application (RIVVIR): As shown in Figure 3(b), the RIVVIR application [31] at OSU deals with real-time remote volume visualization of 3D models of small animal imaging generated by MRI scanners. When such an application needs to be accessed for remote steering and visualization by thin-clients, the network path between the two sites should have as much available bandwidth as possible. A corresponding RIVVIR application template can be: (i) $RSpec$ - HPC compute with GPU and storage resources to process high resolution images at research lab, and (ii) $QSpec$ - low latency/jitter flow with high bandwidth and medium flow priority to help with interactive analysis with a thin-client.

3) GENI Classroom Lab Experiments: As shown in Figure 3(c), a class of 30 students conducting lab experiments at MU in a Cloud Computing course [29] require resources across multiple GENI racks. As part of the lab exercises, multiple virtual machines need to be reserved and instantiated on remotely located GENI racks. There can be sudden bursts of application traffic flows at campus edge router, especially the evening before the lab assignment submission deadline. A corresponding GENI Classroom application template can be: (i) $RSpec$ - virtual or physical remote machines with UNIX distribution, and (ii) $QSpec$ - low packet loss and medium flow priority to allow students to finish the lab exercises.

4) ElderCare-as-a-Service Application: As shown in Figure 3(d), the ElderCare-as-a-Service application [29] consists of an interactive video streaming session between a therapist on MU campus and a remotely residing elderly patient at a Kansas City residence for performing physiotherapy exercises as part of Telehealth interventions. During a session, the quality of application experience for both users is a critical factor (especially in skeletal images from Kinect sensors), and the application demands strict end-to-end QoS requirements to be usable. A corresponding ElderCare-as-a-Service application template can be: (i) $RSpec$ - remote HPC resources with abundant storage, and (ii) $QSpec$ - consistent high available bandwidth with very low jitter and high flow priority for elder to closely follow postures being exercised in the session.

C. Custom Templates for Knowledge Base Applications

Having a Knowledge Base (KB) of certain transactional genomics, multi-omics and molecular breeding datasets allows users and researchers to have a single point of information source. It can enable the collaboration of experts in different areas such as Grass KB (GrassKB), Rice KB (RiceKB), and Soybean KB (SoyKB). An exemplar is the SoyKB [32], which is a comprehensive all-inclusive web resource for soybean translational genomics and breeding. SoyKB handles the management and integration of soybean genomics and multi-omics data along with gene function annotations, biological pathway and trait information. The process consists of large data sets being transferred to MU for pre-processing and subsequently being transferred to either local or remote HPC sites (at: ISI (Information Sciences Institute) [33], TACC (Texas Advanced Computing Center) [34] or XSEDE [35]) for analysis as shown in Figure 4. Following computation completion, the resultant processed data is transferred to MU and also to the iPlant Collaborative [9] storage, in addition to becoming available for user browsing in SoyKB.

The above process can suffer from long run-time for the overall process due to workloads at HPC centers and slow network transmission of large data sets despite high bandwidth

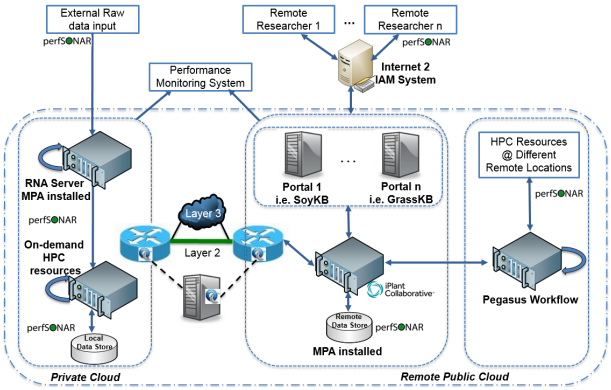


Fig. 4: SoyKB workflow

connectivity over paths involving Internet2 national high-speed network backbone. End-to-end performance monitoring is achieved through instrumentation of the infrastructure with perfSONAR measurement points based upon our Narada Metrics framework [36]. Various custom metrics at the system, network and application-levels are integrated into relevant measurement archives in order to obtain timely and accurate performance intelligence (e.g., log analysis and anomaly event notifications). A corresponding SoyKB application template can be: (i) $RSpec$ - HPC compute environment with local site and remote location speedup processing capabilities of raw genomic data, and (ii) $QSpec$ - high flow throughput with little or no packet loss to provide fast-enough data transfer to justify wide-area network path selection to move data to remote compute resources.

IV. ADON Architecture

In this section, we describe the policy implementation architecture of ADON that leverages the custom templates for fine-grained control and customization of programmable resources. Figure 5 shows the ADON architecture, which consists of the application, middleware and the infrastructure layers within which individual components interact with each other to provision resources for incoming application requests.

The high-level application requirements along with $RSpecs$, $QSpecs$ and application priority are captured in the application layer. Depending upon the resources being requested in the infrastructure layer, and the campus policy rules (maintained by the Performance Engineer), the routing and queue policy assignments are applied in the middleware layer for each application being provisioned. Real-time performance monitoring of individual flows can be used to configure adaptation triggers within already provisioned flows, or even to reject a new application flow if the required QoS levels cannot be met given the load of already provisioned flows. Such middleware layer functions can be implemented with: (i) Control Module, (ii) Network Flowvisor, and (iii) Compute Hypervisor. In this paper, we mainly focus on the Control Module's 'Custom Template Catalog', Network Flowvisor module's 'Virtual Tenant Handler' (NF-VTH) and Compute Hypervisor 'Virtual Tenant Handler' (CH-VTH) (highlighted in red color boxes in Figure 5) that are necessary to implement ADON functionalities.

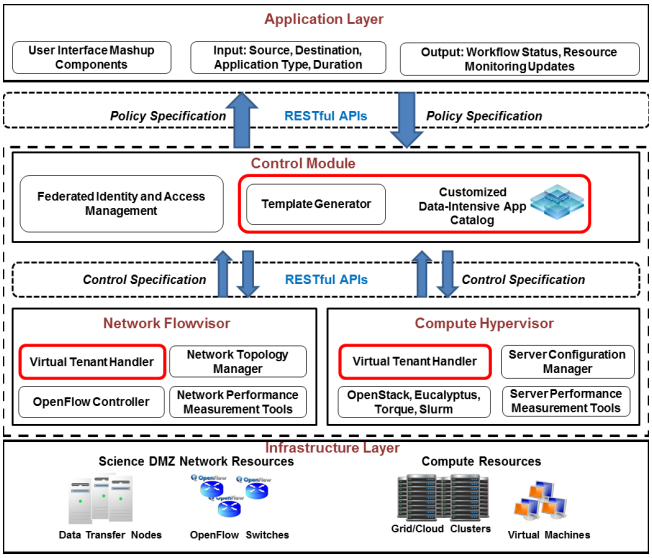


Fig. 5: ADON reference architecture

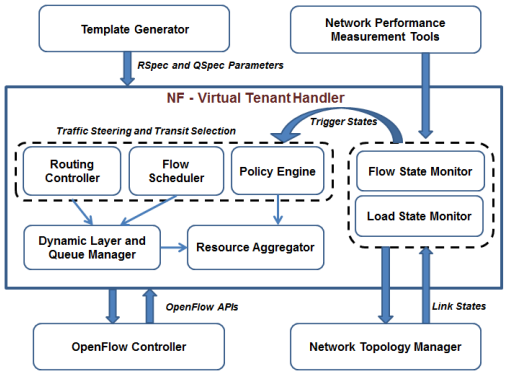


Fig. 6: Individual components of the Network Flowvisor-Virtual Tenant Handler

A. Custom Template Catalog

The Control Module consists of the Template Generator component which exposes RESTful APIs for configuring application type, application priority and routing specifications that can be programmed within the application layer. The Template Generator module also allows the Performance Engineer to save a successfully configured template in a custom template catalog database, which allows re-use for future flow provisioning instances. The QoS and application priority parameters are then fed into the Network Flowvisor module by programming the required REST APIs such as: queue policies and bandwidth requirements. The Federated IAM component within the Control Module features an 'entitlement service' module for all campuses that federate their Science DMZ infrastructures using third-party frameworks such as the Internet2 Incommon federation [37]. It also allows for centrally managing entitlements based on mutual protection of privacy policies between institutions to authorize access to different multi-domain infrastructure components.

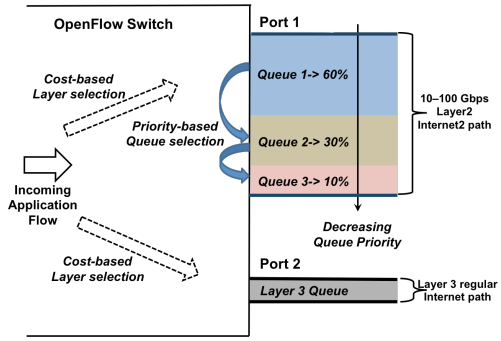


Fig. 7: Illustration of dynamic layer selection and queue management

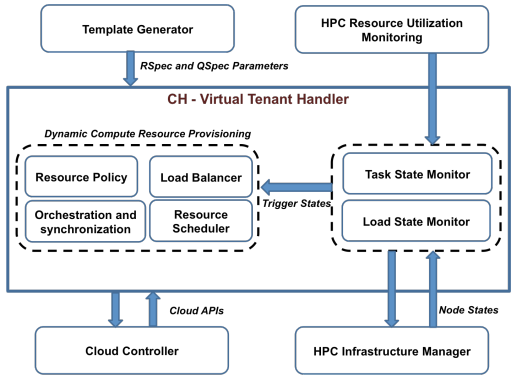


Fig. 8: Individual components of the Compute Hypervisor Virtual Tenant Handler components

B. Network Flowvisor - Virtual Tenant Handler

The NF-VTH is responsible for dynamically handling incoming application flows and providing the intelligence for adaptive network resource allocation. As shown in Figure 6, the Policy Engine interacts with the Template Generator component on the top layer for accepting the $QSpec$ parameters along with application priority within the custom templates. The Policy Engine then interacts with the Flow Scheduler to compute the relative costs of assigning the application flow either to Layer 2 (Science DMZ path over Internet2 AL2S) or Layer 3 (regular Internet) infrastructure should the application flow be provisioned any network resources. The Flow scheduler also compares the priority of the new application with the existing applications in order to decide on the appropriate queue assignment within Layer 2. The Routing Controller is responsible for assigning network configuration for the hybrid-port of the OpenFlow switch depending on the Flow Scheduler decision on the network resource allocation.

The next component is the Dynamic Queue Manager which is responsible for provisioning the right QoS policies for a given application flow. To accomplish such right provisioning, we utilize the *minimum rate* and *maximum rate* properties of queue configuration on OpenFlow switches as provided by the OpenFlow 1.3 specification [38]. The configured queues on the OpenFlow switch ports are then mapped to incoming application flows using the *set-queue* action of the OpenFlow specification. As shown in Figure 7, the queue slots are mapped based on the application priority as specified in the high-level application requirements. A higher priority application is mapped to a queue with the maximum available bandwidth.

In the case that the desired queue is not available, the application is mapped down the queue priority level to be provisioned in the next best available priority queue. The mapping is performed until a queue slot that can provide an acceptable QoS is selected. If found, the flow is provisioned on the selected queue, else it is pushed to the Flow Scheduler component. If none of the slots are available, the flow can be pushed again to the Flow Scheduler to be retrieved later once the appropriate queue is available. The Dynamic Queue Manager then interacts with the Resource Aggregator to update the available resources once a given flow is provisioned on its overlay network.

The NF-VTH also monitors the load states of each queue and flow states of each application using Flow State Monitor and Load State Monitor components. These components receive inputs from the Network Performance Measurement module powered by Narada Metrics SDN monitoring that provides network health status notifications such as flow statistics and other such QoS impacting factors. The module also provides options to reconfigure existing paths based on the custom template directed resource reservations. In cases where the load state in terms of number of applications contending for Layer 2 service creates a link saturation, NF-VTH can steer the traffic on Layer 3 if acceptable QoS parameters can allow a push of the new flows into the Flow Scheduler.

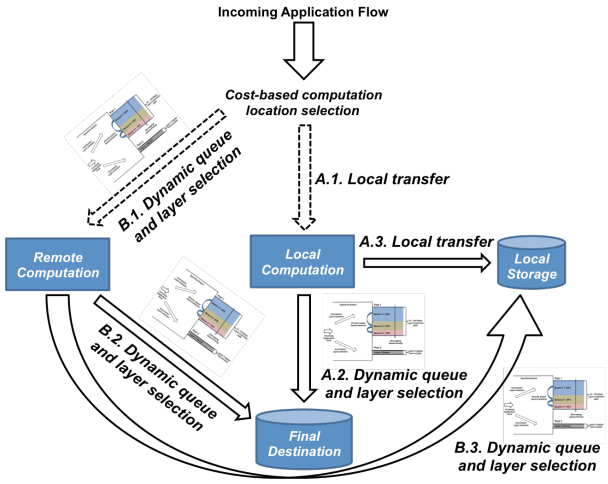


Fig. 9: Illustration of computation location selection

The NF-VTH further interacts with the underlying OpenFlow controller for installing the required flows on the OpenFlow switches. The Network Topology Manager (part of the OpenFlow controller) implements shortest available path algorithms with weighted and non-weighted deterministic routes to remote WAN-accessible sites and provides graphical abstractions to the NF-VTH module with topology link states.

C. Compute Hypervisor - Virtual Tenant Handler

The CH-VTH is responsible for local and remote HPC provisioning process using suitable HTCondor [39] configurations and OpenStack based on detailed resources utilization and performance as show in Figure 8. Template Generator module collects all users' $RSpec$ requirements that are needed for the 'Dynamic Compute Resources Provisioning', where Resource

Policy identifies and considers the cluster resource boundaries. The multi-tenant aware, Load Balancer in this module manages the elastic processes in conjunction with Orchestration and Synchronization for different nodes deploying the process. The Resource Scheduler administers the deployment of HPC jobs, and this process is controlled by the Cloud Controller module. On the other hand, HPC Resource Utilization Monitoring is a real-time monitoring system that provides information of the available Task State and Load State i.e., computing resources, run-time processes, and detailed information of executed jobs and storage usage. Finally, HPC Infrastructure Manager initializes HPC resources considering Compute and Storage Information based on State Monitors.

D. Computation Location Selection

Through the help of NF-VTH and CH-VTH, we achieve the end-to-end overall network and computational resource optimization as illustrated in Figure 9. Here we show a typical data-intensive application life-cycle where the final repository of processed data is reached through two possible paths involving either local (A.1) or remote computation (B.1). The choice of computation location as well as storage destination is mandated by a comprehensive cost optimization which takes into consideration factors such as: availability of computation resources, residual utilization left over by existing applications, and $RSpec$ of the new application. At each stage of network transfer of either raw or processed data, NF-VTH performs dynamic layer and queue selection thereby optimizing network resources. In the case that the data is processed locally, the resultant data will be locally stored (A.3) and a copy will be transferred to the Final Destination (A.2). Alternately, if the data is processed remotely, the resultant data will be stored in the Final Destination (B.2) and a copy will be transferred to Local Store (B.3). The applied algorithm details that are critical in the NF-VTH and CH-VTH are described in the following section.

V. ADON-Supported Resource Allocation Algorithms

The NF-VTH and CH-VTH receive resource allocation requirements of any application a_x from the Template Generator represented as a vector $\{QS_{a_x}, R_{a_x}, P_{a_x}\}$, where QS_{a_x} is the k-tuple vector ***QSpec***, RS_{a_x} is the 2-tuple representing ***RSpec***, and P_{a_x} is the priority of the application a_x. The Performance Engineer is responsible for translating the resource requirements into $QSpec$ and $RSpec$ and forwards the translated information to the respective NF-VTH and CH-VTH. For both, $QSpec$ of application a_x is a QoS vector of k elements (i.e., k QoS metrics) with each vector being represented with an optimal q^o_{xk} and a lower bound q^{lb}_{xk} value required by the application itself, whereas, $RSpec$ is a vector $\{RS^C_x\}$ denoting the computational resource requirement of the application.

Upon receipt of any such application resource allocation request, the NF-VTH and CH-VTH are responsible for the allocation of optimal network and computational resources from a set of available resource $\mathcal{R}$ so that the allocation not only satisfies the application QoS needs, but also minimizes the allocation cost. This allocation cost function can be designed by the NF-VTH and CH-VTH to minimize any performance or economic metric or a function of different metrics specific to the overall system requirements. In our implementation of ADON to be discussed in Section V-D, we use a specific cost metric custom designed for the applications we aim to support.

The algorithm makes use of a generic basis for overall application flow, each part of which has a specific cost that is able to be calculated by the custom cost metric function as follows: (i) data is moved from source to point of computation (using one or more network paths), (ii) computation occurs at an appropriate compute location, and (iii) data is moved to a storage location (using one or more network paths). By including a custom cost metric function for each step of the process, we allow the ability for selection of network path or compute, not only for performance reasons but also other arbitrary decisions such as $\$cost$ or availability of a resource.

A. Optimization Function

Let us assume that $R_{a_x} = \{R^C_x, R^N_x\}$ represents the optimal resource allocation for application a_x, with R^C_x being the allocated computational resource and R^N_x being the allocated network resource. The resource allocation algorithm should then guarantee R^N_x to satisfy the application $QSpec$ and R^C_x to satisfy the application $RSpec$. Thus, with $\mathcal{C}()$ being the allocation cost function and $\mathcal{Q}^k()$ being the QoS translation function for metric k, the objective function of the resource allocation can be written as -

$$\begin{aligned} &\text{minimize} && \mathcal{C}(R_{a_x}) \\ &\text{subject to} && q^{lb}_{xk} < \mathcal{Q}^k(R^N_x) \leq q^o_{xk} \quad \forall \ k \\ & && \& \quad R^C_x \geq RS^C_x \end{aligned} \tag{1}$$

B. Edge Optimization Algorithm

We solve the optimization problem in Equation (1) using a heuristic approach explained through a set of algorithms. Algorithm 1 explains the Edge Optimization which takes a new application a_{new} with the corresponding $RSpec$ and $QSpec$ requirements as inputs, compares the local, and remote processing costs (both network and compute) using *computeLocal()* and *computeRemote()* functions. The corresponding outputs are then compared to take the final decision on allocating compute resources, suitable Layer 2 or Layer 3 network resources, and assignment of the proper queue within the network tunnel. The Edge Optimization algorithm thus solves the cost computation problem to choose the optimal network and compute resources. It also solves the dynamic queue assignment problem by choosing the most appropriate queue within the network resource using the new application priority $P_{a_{new}}$. Further, it satisfies Equation 1 through a set of End-to-End Optimization algorithms which are responsible to compute the stepwise costs for each possible computation and ensuing networking scenarios.

C. End-to-End Optimization Algorithms

The end-to-end optimization ensures the local or remote processing outcome of the application. Thus, such optimization is a direct consequence of local and remote computational resource availability, and corresponding available network resources. Although apparently remote processing seems undesirable for unpredictability of ensuing remote networking issues, factors such as: local resource availability, time period of the available resources, CPU core requirements of the application, data transfer time and available network bandwidth for data transfer - influence the application computation location decision and thus affect the overall resource allocation. The end-to-end optimization is ensured through Algorithm 2 and Algorithm 3 where we compute the local and remote processing costs, respectively.

TABLE I: Notations used

$\mathbb{R}$	Entire resource pool
a_x	Any application x
$\mathcal{R} = \{\mathcal{R}_C^L, \mathcal{R}_C^R, \mathcal{R}_N^{L2}, \mathcal{R}_N^{L3}\}$	4-tuple representing available resources
$\mathcal{R}_C^L$	Available local compute resource
$\mathcal{R}_C^R$	Available remote compute resource
$\mathcal{R}_N^{L2}$	Available L2 network resource
$\mathcal{R}_N^{L3}$	Available L3 network resource
$RS_{a_x} = \{RS_x^C\}$	*RSpec* of application a_x represented by computational resource requirement RS_x^C
QS_{a_x}=$\{\{q_{x1}^o, q_{x1}^{lb}\}, \{q_{x2}^o, q_{x2}^{lb}\}, \cdots, \{q_{xk}^o, q_{xk}^{lb}\}\}$	k-tuple vector *QSpec* of application a_x
$\{q_{xk}^o, q_{xk}^{lb}\}$	QoS metric k 2-tuple with optimal and lower bound values respectively
P_{a_x}	Priority of application a_x
$A = \{A_{L2}, A_{L3}\}$	Set of all allocated applications
A_{L2}	Set of all applications allocated L2 resources
A_{L3}	Set of all applications allocated L3 resources
$R_{a_x} = \{R_x^C, R_x^N\}$	2-tuple representing resource allocated to application a_x
R_x^C	Compute resource allocated to application a_x
R_x^N	Network resource allocated to application a_x
R_T	Threshold resource needs to be available for a new application to be allocated

Algorithm 1 Edge Optimization Algorithm

Input: *RSpec* $RS_{a_{new}}$, *QSpec* $QS_{a_{new}}$, and Priority $P_{a_{new}}$ of new application a_{new}
Compute: $\mathcal{R}$=$\mathbb{R}$ - $\sum_{a_i \in A} R_{a_i}$
Compute: $P_{min} = min(P_{a_i}) \quad \forall \quad a_i \in A_{L2}$
Output: Resource allocation $R_{a_{new}}$
begin procedure
if $\mathcal{R} \geq R_T$ **then**
  /*Cost for local computation*/
  $C_L = computeLocal(QS_{a_{new}}, RS_{a_{new}}, P_{a_{new}}, P_{new})$
  /*Cost for remote computation*/
  $C_R = computeRemote(QS_{a_{new}}, RS_{a_{new}}, P_{a_{new}}, P_{new})$
  /*Compare costs and final decision*/
  if $C_L < C_R$ **then**
    $\mathcal{R}^L = \mathcal{R}^L - \{RS_{new}^C, RS_{new}^N\}$
    Include a_{new} to A
  else
    $\mathcal{R}^R = \mathcal{R}^R - \{RS_{new}^C, RS_{new}^N\}$
    Include a_{new} to A
  end if
else
  Push a_{new} to resource scheduler
end if
end procedure

Algorithm 2 Local Processing Cost Computation Algorithm

Input: *RSpec* RS_{a_i}, *QSpec* QS_{a_i}, and Priority P_{a_i} of application a_i
Output: Local computation cost C_L
begin procedure
if $\mathcal{R}_C^L \geq RS_i^C$ **then**
  /*Compute cost for L3 transfer*/
  if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L3}, RS_{a_i}) < q_{ik}^o \quad \forall$ QoS metrics k **then**
    $C_L^{L3} = computeCost(\mathcal{R}_C^L, \mathcal{R}_N^{L3}, RS_i^C, RS_i^N)$
  else
    $C_L^{L3} = \infty$
  end if
  /*Compute cost for L2 transfer*/
  if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2}) < q_{ik}^o \quad \forall$ QoS metrics k **then**
    $C_L^{L2} = computeCost(\mathcal{R}_C^L, \mathcal{R}_N^{L2}, RS_i^C, RS_i^N)$
  else if $P_{a_i} > P_{min}$ **then**
    if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2} + R_{min}^N) < q_{ik}^o \quad \forall$ QoS metrics k **then**
      $terminate(a_{min})$
      $C_L^{L2} = computeCost(\mathcal{R}_C^L, \mathcal{R}_N^{L2} + R_{min}^N, RS_i^C, RS_i^N)$
    end if
  else
    $C_L^{L2} = \infty$
  end if
  $return\ min(C_L^{L3}, C_L^{L2})$
else
  $return\ \infty$
end if
end procedure

1) Local Processing: In Algorithm 2, we compute the cost incurred for local computation of scientific data and the ensuing networking cost to transfer the processed data to a remote location. Such local computation is only possible if sufficient computational resources that are available at the local site meet the $RSpec$ requirements of the application. Another requirement is that the available bandwidth to transfer the processed data to the final destination site should meet the $QSpec$ requirements of the application, which is computed using *estimateQoS()* function. Such transference of the processed data is carried out either through the legacy best effort Layer 3 infrastructure i.e., public Internet, or through a dedicated high-speed Layer 2 infrastructure such as Internet2 AL2S. The choice of network infrastructure depends upon factors such as: the capacity of the network, ensuing network performance metrics such as loss or latency, and the current utilization as well as future availability of network resources. Thus, for both Layer 2 and Layer 3 scenarios, cost of computation is calculated using *computeCost()* function and the minimum is returned to the Edge Optimization Algorithm.

We designed the ADON-supported resource allocation in such a way that the NF-VTH first tries to accommodate an incoming application in Layer 3, if the available Layer 3 resources meet all the performance metric requirements specified in the application $QSpec$. The argument behind such a greedy provisioning is to conserve the precious/expensive Layer 2 services and dedicate them for future applications needing higher bandwidth and stricter QoS guarantees. However, if the new application $QSpec$ requirements are not met by the available Layer 3 resources, Layer 2 resources are provisioned. If enough Layer 2 resources are not available to meet the $QSpec$ requirements, the NF-VTH tries to force deallocate the Layer 2 scheduled application with lowest priority, provided that the new application is of higher priority than the application being deallocated. Although such provisioning satisfies the optimization conditions from Equation 1, and intelligently prioritizes precious network resources only for deserving applications, it fails to satisfy the optimization function of Equation 1. Thus, to satisfy the cost minimization, we compute costs for both Layer 2 and Layer 3 allocation scenarios and choose the option that incurs minimal cost.

2) Remote Processing: The remote computation analysis is very similar to that of local computation. However, when science applications are computed remotely for lack of local computation resources, the eventual processing can be either done at the final destination or at an intermediate HPC location with the need to duplicate the processed data locally. Such duplication requirements may incur more costs in terms of possible Layer 2 or Layer 3 transfer of the processed data to the local site, which eventually affects the final local versus remote computation decision. Thus in Algorithm 3, we make such provisions by performing a step-by-step cost computation taking into account: the cost incurred for transferring the unprocessed data to that remote HPC facility, cost of computation at a remote HPC facility, cost of transferring the processed data to the final destination, and the possible duplication cost, i.e., transferring the processed data back to the local site again.

Algorithm 3 Remote Processing Cost Computation Algorithm

Input: *RSpec* RS_{a_i}, *QSpec* QS_{a_i}, and Priority P_{a_i} of application a_i
Output: Remote computation cost C_R
begin procedure
if $\mathcal{R}_C^R \geq RS_i^C$ **then**
  /*Compute cost for L3 transfer*/
  if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L3}) < q_{ik}^{o}$ $\forall$ QoS metrics k **then**
    $C_R^{L3} = computeCost(\mathcal{R}_C^R, \mathcal{R}_N^{L3}, RS_i^C, RS_i^N)$
  else
    $C_R^{L3} = \infty$
  end if
  /*Compute cost for L2 transfer*/
  if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2}) < q_{ik}^{o}$ $\forall$ QoS metrics k **then**
    $C_R^{L2} = computeCost(\mathcal{R}_C^R, \mathcal{R}_N^{L2}, RS_i^C, RS_i^N)$
  else if $P_{a_i} > P_{min}$ **then**
    if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2} + R_{min}^N) < q_{ik}^{o}$ $\forall$ QoS metrics k **then**
      $terminate(a_{min})$
      $C_R^{L2} = computeCost(\mathcal{R}_C^R, \mathcal{R}_N^{L2} + R_{min}^N, RS_i^C, RS_i^N)$
    end if
  else
    $C_R^{L2} = \infty$
  end if
  /*Check for duplication cost*/
  if Duplication of process data needed **then**
    /*Compute cost for L3 transfer*/
    if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L3}) < q_{ik}^{o}$ $\forall$ QoS metrics k **then**
      $C_D^{L3} = computeCost(\mathcal{R}_C^R, \mathcal{R}_N^{L3}, RS_i^C, RS_i^N)$
    else
      $C_D^{L3} = \infty$
    end if
    /*Compute cost for L2 transfer*/
    if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2}) < q_{ik}^{o}$ $\forall$ QoS metrics k **then**
      $C_D^{L2} = computeCost(\mathcal{R}_C^R, \mathcal{R}_N^{L2}, RS_i^C, RS_i^N)$
    else
      $C_D^{L2} = \infty$
    end if
    $C_R = min(C_R^{L3}, C_R^{L2}) + min(C_D^{L3}, C_D^{L2})$
  else
    $return\ min(C_R^{L3}, C_R^{L2})$
  end if
else
  $return\ \infty$
end if
end procedure

The output of algorithms 2 and 3 are compared in the Edge Optimization Algorithm (Algorithm 1) where the final decisions on computation location and ensuing resource allocation are made. Thus, the final output from Algorithm 1 minimizes the cost of allocation and satisfies the $QSpec$ and $RSpec$ requirements from Equation 1.

D. Algorithm Implementation for SoyKB Use Case

In our ADON mathematical model, the function *computeCost()* is a generic abstraction for calculating the cost of different applications. It can be defined and represented in different forms based on the specific requirements and heuristics of each application. The purpose of calculating this cost is to decide: i) Should the computation be performed locally at MU or at remote HPC resources, such as ISI, TACC or XSEDE? and ii) Should we use AL2S or regular Internet for data transfer the data to the iPlant data store? In the SoyKB application, for example, it is essential for users to get the analyzed results in a timely manner after they run the application with a large size of input data, which will speed up their bioinformatics research process.

To meet these requirements, we can define the cost from Edge Optimization Algorithm (Algorithm 1) as the total time taken, which is mainly comprised of data computation time and network transfer time. Once the application is formalized as a workflow, the Pegasus Workflow Management Service [40] can map it onto available compute resources and execute the steps in appropriate order. However, it is possible that the service cannot find enough resources and needs to put the task in a prioritized queue. When there are many application tasks waiting in the Workflow Management Service queue, the SoyKB application has to wait for a long time to start the Pegasus workflow. The Mapper component in Pegasus is responsible for dividing the jobs and mapping these jobs to corresponding nodes to optimize performance. Since we are also able to get the data size that needs to be computed, number of the worker nodes assigned and the average throughput of each worker node, we can easily calculate the total estimated computation time by adding the waiting time and actual computation time. It is to be noted that if currently any local HPC resources with Pegasus workflow are not available for SoyKB, the local queuing time is set to infinity as the workflow will always use the remote HPC resources at ISI, TACC or XSEDE, as shown in Figure 4.

Once processed, all the processed SoyKB research data is stored in the iPlant Data Center and is available to query through a website hosted in iPlant Atmosphere cloud platform that is similar to Amazon EC2. The iPlant collaborative uses iRODS [41] to manage and transfer the data, which creates multiple TCP streams for transferring data. Since our network performance monitoring system can provide us the round-trip time (RTT) delay of the transfer, we can configure the TCP buffer size and length of the processor input queue at the end nodes accordingly by following well-known TCP tuning practices [42] to achieve throughput that is close to the theoretical maximum throughput. However, both the AL2S and regular Internet are not always able to provide the maximum bandwidth desired for SoyKB, especially when there are high levels of cross-traffic passing through the shared links, or source to destination physical distance is large enough to impact TCP behavior. Nevertheless, the computation time is significant and the network performance can often play a smaller role in the overall time to task completion.

In this scenario, the network monitoring system can again play an important role and provide the maximum bandwidth and the current throughput utilization of the network links to calculate the actual available bandwidth left for the SoyKB data transfer, or directly measure single thread TCP throughput and treat it as approximate available bandwidth. We can compare the theoretical TCP throughput and actual available bandwidth to get the estimated network transfer time, given that the size of data to be transferred is known to our ADON-related support services. We particularly remark that the analyzed data will be significantly reduced after the computation process and could be only ≈1/10th of the original raw data size. This indicates that even if the local HPC resources is not as powerful as the remote resources at TACC, it might still provide an increased overall performance by significantly reducing the

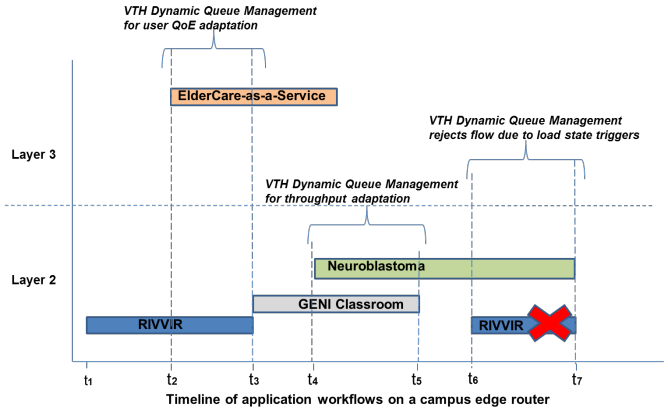
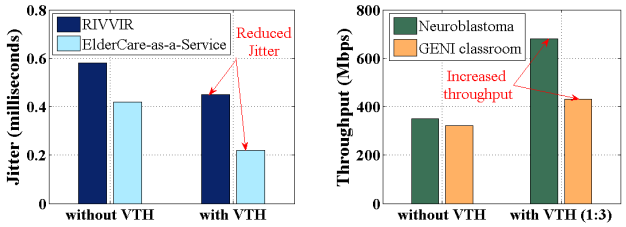


Fig. 10: Timeline of a campus edge router handling multiple data-intensive application workflows



(a) Jitter comparison (b) Throughput comparison

Fig. 11: SoyKB Layer 2 performance without contending traffic

estimated wait-time in the task queue for the computation part, and by reducing the size of the data to be transferred over the wide-area network segments. Such characteristics does impact the outcome of the Local and Remote processing algorithms (Algorithms 2 and 3, respectively) which in turn decide between local or remote processing.

VI. Experimental Evaluation

In this section, we first present a emulation study with application workflows to analyze the ADON-supported algorithm results while handling temporal behavior of multi-tenant traffic burst arrivals. Next, we describe a case study featuring validation experiments of our ADON implementation with the SoyKB data-intensive science application on a wide-area testbed across OSU and MU campuses with regular Internet (Layer 3) and Internet2 AL2S (Layer 2).

A. Emulation Study

We used Mininet network emulator for NF-VTH experiments that involved synthetic traffic flows using the "tc" and "iperf" network utility tools. Figure 10 shows the timeline from time t_1 to t_7 as seen by an edge OpenFlow router/switch handling application workflows as they enter and exit the NF-VTH module. RIVVIR application workflow (see Figure 3(b)) was initiated as a UDP flow at time t_1 with a guaranteed bandwidth of 10 Mbps (typical requirements of remote desktop access with raw encoding) and latency of 50 ms (RTT between OSU and MU). At time t_2, ElderCare-as-a-Service application workflow (see Figure 3(d)) was started as a new UDP flow with a guaranteed bandwidth of 100 Mbps (typical requirement of a Kinect video stream) and latency of 30 ms (RTT between MU and Kansas City).

At this moment, the Dynamic Queue Manager within the NF-VTH instantiated the queues on the OpenFlow switch to provide the required QoS guarantees for each of the concurrent flows. The total jitter observed when both flows coexisted on the link are captured with and without NF-VTH dynamic queue management in Figures 11(a). We can see that the jitter is significantly reduced (improved performance) for both the applications, especially for the video flow using the NF-VTH module with application-specific queue policies. This is due to the fact that the NF-VTH reduced the external fragmentation of available bandwidth caused by static policy management on the switch ports. The dynamic queue mapping of the UDP flows to the requested bandwidth provided performance isolation to each of the flows, and hence reduced their jitter values.

At time t_3, GENI Classroom workflow (see Figure 3(c)) was started as a TCP flow with burst traffic pattern (burst rate - 1 Mbps and buffer size - 10 KB similar to web traffic) parallel to the RIVVIR workflow. Both flows co-existed without affecting each other's QoS policies as both flows were assigned their individual priorities on the queues. At time t_4, Neuroblastoma workflow (see Figure 3(a)) was started as a parallel TCP flow with 5 parallel streams to simulate a GridFTP application for image file transfer. NF-VTH then triggered the dynamic queue configuration for assigning a prioritized bandwidth of 600 Mbps for Neuroblastoma and 360 Mbps bandwidth for GENI Classroom experiments (on a 1 Gbps link).

Figure 11(b) shows throughput of the two workflows achieved with and without NF-VTH. Bandwidth is equally split when there is no dynamic queue management for both flows. However with NF-VTH, internal fragmentation of bandwidth is reduced. The total bandwidth is sliced between the two flows as per their individual priorities ensuring each flow is only utilizing the requested bandwidth as provided in the application QoS templates. This slicing happens until time t_5 when the GENI Classroom flow exits and resources are released. However, when a new RIVVIR workflow starts again at time t_6 while the Neuroblastoma application is currently provisioned, the new flow is rejected and pushed to the Flow Scheduler. This is because the new flow's QoS requirements cannot be guaranteed and mapped in the Dynamic Queue Manager. This scenario occurs due to the resource unavailability of the link which is fully utilized by the prioritized data-intensive Neuroblastoma application flow.

Thus, we can conclude from the above experiments that the NF-VTH is effective in scheduling transit selection and traffic engineering at the campus-edge based on real-time policy control that ensures predictable application performance delivery, when handling temporal behavior of multi-tenant traffic burst arrivals corresponding to a diverse set of data-intensive applications.

B. Real-Network Case Study

The testbed setup for the real-network case studies is as shown in Figure 12, which consists of the OSU and MU campuses connected through an extended VLAN overlay that involves an Internet2 AL2S connection by way of local regional networks of OARnet and GPN/MOREnet in Ohio and Missouri, respectively. Each Science DMZ has a matching DTN equipped with dual Intel E5-2660, 128GB of memory,

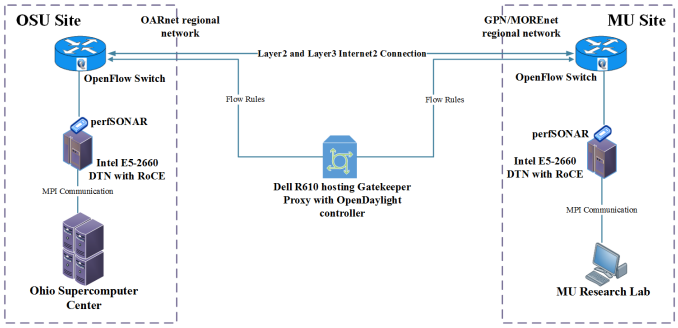


Fig. 12: Collaborative Science DMZ testbed between two campuses for handling multi-tenancy of data-intensive applications

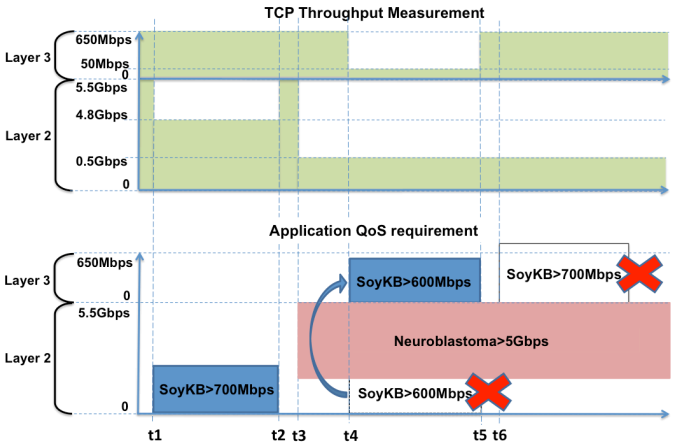
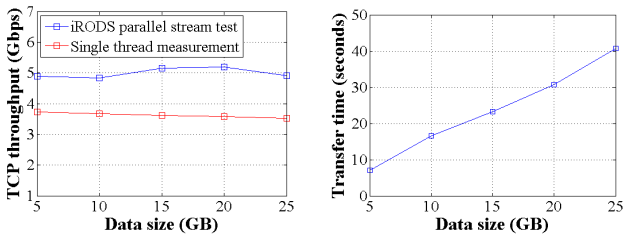


Fig. 13: Timeline of OpenFlow switch handling SoyKB application workflows based on application $QSpecs$ and periodic throughput measurement



(a) Throughput (b) Transfer time

Fig. 14: SoyKB Layer 2 performance without contending traffic

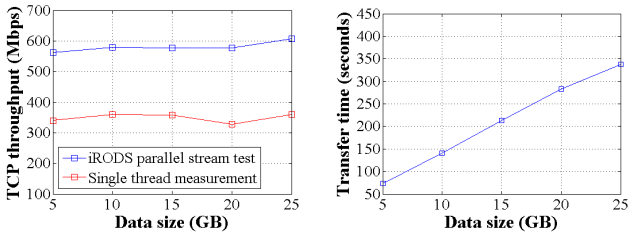
300GB PCI-Express solid state drive, and dual Mellanox 10 Gbps network cards with RoCE support. Each Science DMZ has perfSONAR measurement points powered by Narada Metrics for continuous monitoring at 1 - 10 Gbps network speeds. A common Dell R610 node in the OSU Science DMZ is used to run the OpenFlow controller based on OpenDayLight for the NF-VTH functionality. The NEC switch on the OSU end is connected to OSU's 100 Gbps Cisco Nexus router at 10 Gbps, but has the ability to scale to 40 Gbps as the Science DMZ grows to support future researchers and applications. At MU, the Science DMZ features a Brocade VDX 8770 switch to attach various nodes in the Science DMZ, and a 100 Gbps Brocade MLXE router at 10 Gbps interface speeds, with the ability to scale to 100 Gbps speeds.

We used the SoyKB application use case in experiments to comprehensively validate the effectiveness of the ADON algorithm implementation in the MU-OSU Science DMZ testbed. We mainly focused on improving the SoyKB data transfer using ADON adaptations that couple with data computation. Our goal was to improve the SoyKB-related processing tasks with and without contending real-time applications such as Neuroblastoma that have much higher bandwidth requirements. Initially, we perform SoyKB data transfer with and without the ADON service to show how our network measurement system can help to intelligently and confidently make decisions for SoyKB application workflow with and without contending traffic to get the best performance, while meeting the $QSpecs$. Following this, we present the performance improvement after TCP tuning process and with the help of our SDN monitoring system to provide a better Custom Template configuration for the SoyKB system.

Unlike the real-time applications such as Neuroblastoma or ElderCare-as-a-Service whose goals are to reduce the jitter in order to provide better user experience, the SoyKB application focus is more on the time taken for the data transfer and data analysis across the distributed computing resource sites. Regarding the testbed configuration, we set the TCP buffer size to 256 MB on both sides of MU and OSU data transfer nodes. Prior to the experiment, we made sure that the TCP buffer size and processor input queue sizes were big enough to ensure that our path selection decision is solely dependent on the available bandwidth when the application is executed by the SoyKB users.

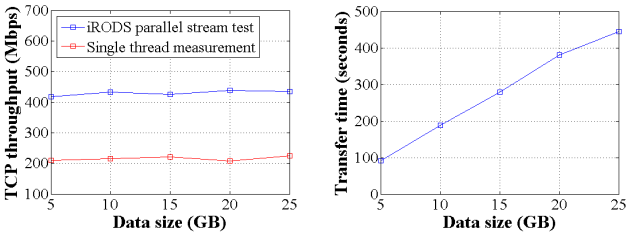
Figure 13 shows the orchestration timeline of data-intensive applications as seen by our ADON along with periodically sampled TCP throughout measurements. At time $t1$, the SoyKB application workflow is initiated, whose $QSpec$ requires to have at least 700 Mbps of single TCP throughput in the event that iRODS is not able to create more than 1 thread. Predictably, the 10 Gbps Internet2 AL2S network connection between MU and OSU has significantly better throughput (around 5.5-to-6 Gbps) than the regular Layer 3 network connection with only around 650 Mbps achievable throughput when there are no other contending applications. Since ADON can get updated information from our Narada Metrics SDN monitoring system, it makes the decision to use AL2S for SoyKB data transfer, and calls a function in the OpenFlow controller for transferring traffic to a specific VLAN created for AL2S connection with a higher priority.

Figure 14 shows the overall performance of the SoyKB transfer with different data sizes tested during the experiment run period from $t1$ to $t2$. The figure shows that due to absence of any other contending traffic, the SoyKB flow can use the entire AL2S bandwidth achieving close to the highest achievable TCP throughput of 5.5 Gbps with the total transfer time in the order of tens of seconds. After $t1$, the achievable TCP throughput measured by Narada Metrics becomes about 4.8 Gbps from the original 5.5 Gbps, since the bandwidth test applications will compete against the SoyKB applications for the network bandwidth resource. At time $t3$, Neuroblastoma workflow which requires extremely high throughput and higher flow priority starts and the single TCP stream throughput test only shows around 500 Mbps, which does not meet the $QSpec$



(a) Throughput (b) Transfer time

Fig. 15: SoyKB Layer 3 performance without contending traffic



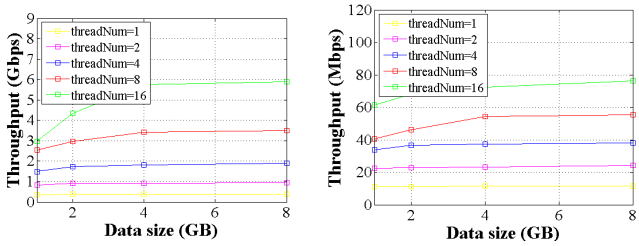
(a) Throughput (b) Transfer time

Fig. 16: SoyKB Layer 2 performance with contending traffic

of another SoyKB application workflow started at time $t4$ with at least 600 Mbps throughput requirements. That is when the ADON algorithm recalculates new estimated transfer times and achievable TCP throughput in Layer 3 (which currently had no other contending data-intensive traffic), and decides to use Layer 3 for SoyKB data transfer by making the priority of rule for the AL2S VLAN lower than the rule for Layer 3.

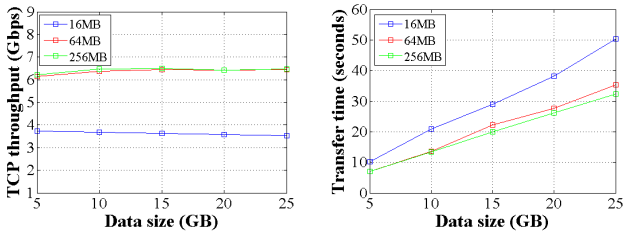
Figure 15 shows the performance of SoyKB application transfer without any other contending data-intensive flows in Layer 3. The data throughput reduced when compared to the Figure 14. However counter-intuitively, it still has considerably higher performance if we continued the SoyKB data transfer through Layer 2, as shown in Figure 13. The transfer performance in such a scenario is shown in Figure 16 where the SoyKB data is transferred through Layer 2 along with the contending Neuroblastoma traffic and does not even meet the minimum SoyKB $QSpec$. Thus through intelligent provisioning with ADON service, we can save more than 100 seconds when transferring a 25 GB size file. In bioinformatics, it is very important for the users to transfer and analyze the data as soon as possible, and the time difference becomes very significant since the actual research data size can be up to several TBs for important applications such as the SoyKB. When we initiate another SoyKB application workflow requiring at least 700 Mbps single throughput at time $t6$, the ADON algorithm will reject the transfer task and push it to the Flow Scheduler, since neither network links can meet the $QSpec$ of the new workflow. It is relevant to note that SoyKB transfer time performance (from Figure 16b) for 5 GB data with contending traffic is as good as 1 GB data transfer on the same testbed with no contention without ADON scheme as previously shown in Figure 1; thus validating the overall effectiveness of the proposed ADON scheme in terms of performance improvements.

Next, we perform additional experiments to demonstrate the performance improvements after the TCP tuning process following guidelines in [42]. While performing above experiments, we found that the network infrastructure does not provide full advertised throughput due to protocol overhead, physical distance between the source and destination data transfer nodes, and end-point hardware limitation related issues. The MU-OSU AL2S link, which advertises 10 Gbps capacity, only had a maximum TCP throughput of 6 Gbps. When using tools such as iRODS that support parallel streams, or have multiple data transfers in parallel, we can configure the system settings such as Linux auto tuning TCP buffer limit and length of the processor input queue in both the MU and OSU data transfer nodes to achieve closer to maximum TCP throughput in order to have fair sharing between flows. We



(a) Layer 2 performance (b) Layer 3 performance

Fig. 17: SoyKB transfer performance with different thread counts



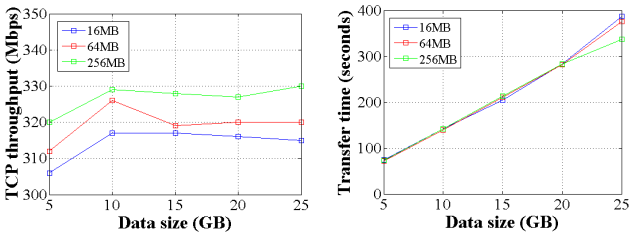
(a) Throughput (b) Transfer time

Fig. 18: SoyKB Layer 2 transfer performance with different TCP buffer sizes

found that it is important to change the configurations at both the source and destination edge nodes in order to ensure that the parallel stream setting performs at peak rates.

Through Figure 17, we observe that the overall throughput increases when we use an increased number of TCP threads to move the data. However, the current iRODS system limits the maximum number of parallel threads to 16. Therefore, to achieve higher speed in next-generation networks, such as 100 Gbps Internet2 infrastructures, the iRODS system might need to support a increased number of parallel TCP threads. Nevertheless, ADON capitalizes on the information regarding the number of threads that can be effectively used for SoyKB data throughput acceleration within the given Science DMZ environments.

Figures 18 and 19 show the performance of Layer 2 and Layer 3 network connections with different TCP buffer sizes



(a) Throughput (b) Transfer time

Fig. 19: SoyKB Layer 3 transfer performance with different TCP buffer sizes

configured in both MU and OSU edge data transfer nodes. For both the cases, we keep the thread count at 16 in order to compare the respective maxima. In Layer 3, we were not able to improve the actual data throughput since the theoretical maximum throughput after TCP tuning in the data transfer nodes exceeds the throughput that the physical regular IP network infrastructure can provide. In contrast, when we increase the TCP buffer size to 64 MB from the original 16 MB, the data throughput increases by almost 50%. The maximum throughput of Layer 2 is greater than the theoretical maximum TCP throughput with 16 MB TCP buffer size, which indicates that there is significant room for improvement by using multiple threads for TCP transfer with bigger buffer sizes (as shown with our experiments with increments of buffer size up to 256 MB). Hence, we can conclude that no further significant improvement of network performance can be obtained while transferring data through the 10 Gbps Layer 2 link beyond using the 64 MB TCP buffer size. Such a knowledge of the achievable peak performance over a certain wide-area overlay network path can be updated within the corresponding ADON custom template, and can be used with the SDN monitoring system to manage expectations of the peak performance achievable for a given application workflow.

VII. Conclusion and Future Work

In this paper, we presented a novel ADON architecture with an application-driven overlay network-as-a-service approach to support multi-tenant data-intensive application flows with hybrid cloud resource needs. With the pent-up resource requirements of data-intensive application flows, traditional network and compute infrastructures are not scalable or flexible for effectively handling such flows, especially in cases with urgent or real-time computing requirements. Using our ADON architecture, we showed that the application-specific policies can be effectively controlled at the campus edge based on individual application flow requirements, and the 'friction' imposed due to firewalls for enterprise traffic flows can be overridden for data-intensive science applications.

The novelty of our work is in our approach for "network personalization" that can be performed using a concept of "custom templates" that helps a Performance Engineer to catalog and handle unique profiles of application workflows in an automated and repeatable manner. We also presented design details and validation experiments of a multi-tenant architecture featuring "Network Flowvisor - Virtual Tenant Handler" (NF-VTH) and " Compute Hypervisor - Virtual Tenant Handler" (CH-VTH) for real-time policy control of an "Overlay Network-as-a-Service" within a campus Science DMZ environment. Our testbed for ADON validation featured high-performance networking capabilities such as OpenFlow switches and RoCE-based data transfer nodes, as well as local and remote HPC resources for real-time and knowledge base application use cases. Our experiment results have demonstrated how our ADON architecture and implementation is capable of providing predictable performance to data-intensive applications, without any changes to existing campus network infrastructure designed for regular enterprise traffic.

As part of the future work, we plan to work with additional data-intensive application use cases having diverse computing and networking needs - e.g., a compute-intensive application that has a compute complexity that is greater than the data transfer complexity over an overlay network path. We also plan to extend ADON to jointly orchestrate nodes and links mapping across multiple geographically-distributed campuses for efficient virtual network embedding to satisfy multiple QoS constraints of data-intensive applications.

References

[1] E. Dart, L. Rotman, B. Tierney, M. Hester, J. Zurawski, "The Science DMZ: A Network Design Pattern for Data-Intensive Science", *Proc. of IEEE/ACM Supercomputing*, 2013.

[2] H. Yin, Y. Jiang, C. Lin, Y. Luo, Y. Liu, "Big Data: Transforming the Design Philosophy of Future Internet", *IEEE Network Magazine*, 2014.

[3] N. McKeown, T. Anderson, H. Balakrishnan, et. al., "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, 2008.

[4] P. Lai, H. Subramoni, S. Narravula, A. Mamidala, D. K. Panda, "Designing Efficient FTP Mechanisms for High Performance Data-Transfer over InfiniBand", *Proc. of ICPP*, 2009.

[5] A. Hanemann, J. Boote, E. Boyd, et. al., "perfSONAR: A Service Oriented Architecture for Multi-Domain Network Monitoring", *Proc. of Service Oriented Computing*, 2005.

[6] R. Morgan, S. Cantor, S. Carmody, W. Hoehn, K. Klingenstein, "Federated Security: The Shibboleth Approach", *EDUCAUSE Quarterly*, 2004.

[7] OpenStack - http://www.openstack.com

[8] P. Calyam, A. Berryman, E. Saule, H. Subramoni, P. Schopis, G. Springer, U. Catalyurek, D. K. Panda, "Wide-area Overlay Networking to Manage Accelerated Science DMZ Flows", *Proc. of IEEE ICNC*, 2014.

[9] S. Goff, et al., "The iPlant Collaborative: Cyberinfrastructure for Plant Biology", *Frontiers in Plant Science*, 2011.

[10] X. Yi, F. Liu, J. Liu, H. Jin, "Building a Network Highway for Big Data: Architecture and Challenges", *IEEE Network Magazine*, 2014.

[11] Y. Ren, T. Li, D. Yu, S. Jin, T. Robertazzi, B. Tierney, E. Pouyoul, "Protocols for Wide-area Data-intensive applications: Design and Performance Issues", *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.

[12] H. Luo, H. Zhang, M. Zukerman, C. Qiao, "An incrementally deployable network architecture to support both data-centric and host-centric services", *IEEE Network Magazine*, 2014.

[13] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, F. Lau, "Moving Big Data to The Cloud: An Online Cost Minimizing Approach", *IEEE Journal on Selected Areas in Communications*, 2013.

[14] I. Monga, E. Pouyoul, C. Guok, "Software-Defined Networking for Big Data Science", *Proc. of IEEE/ACM Supercomputing*, 2012.

[15] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S. Lee, P. Yalagandula, "Automated and Scalable QoS Control for Network Convergence", *Proc. of INM/WREN*, 2010.

[16] H. Egilmez, S. Dane, K. Bagci, A. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks", *Proc. of APSIPA ASC*, 2012.

[17] M. Fatih, G. Haldeman, M. Parashar, "Flexible Scheduling and Control of Bandwidth and in-transit Services for end-to-end Application Workflows", *Proc. of IEEE Workshop on Network-Aware Data Management*, 2014.

[18] A. Chirkin, A. Belloum, S. Kovalchuk, M Makkes, "Execution Time Estimation for Workflow Scheduling", *Proc. of IEEE Workshop on Workflows in Support of Large-Scale Science*, 2014.

[19] B. Hu, H. Yu, "Research of Scheduling Strategy on OpenStack", *Proc. of International Conference on Cloud Computing and Big Data*, 2013.

[20] O. Litvinski, A. Gherbi, "Openstack Scheduler Evaluation using Design of Experiment Approach", *Proc. of IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2013.

[21] J. Castillo, K. Mallichan, Y. Al-Hazmi, "OpenStack Federation in Experimentation Multi-cloud Testbeds", *Proc. of IEEE Conference on Cloud Computing Technology and Science*, 2013.

[22] D. Irwin, P. Shenoy, E. Cecchet, M. Zink, "Resource Management in Data-Intensive Clouds: Opportunities and Challenges", *Proc. of IEEE LANMAN Workshop*, 2010.

[23] A. Younge, G. Fox, "Advanced Virtualization Techniques for High Performance Cloud Cyberinfrastructure", *Proc. of IEEE/ACM CCGrid*, 2014.

[24] C. Bunch, C. Krintz, "Enabling Automated HPC / Database Deployment via the Appscale Hybrid Cloud Platform", *Proc. of HPCDB*, 2011.

[25] S. Seetharam, P. Calyam, T. Beyene, "ADON: Application-Driven Overlay Network-as-a-Service for Data-Intensive Science", *Proc. of IEEE Cloud Networking*, 2014.

[26] Amazon Web Services - http://aws.amazon.com

[27] M. Berman, J. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, "GENI: A federated testbed for innovative network experiments", *Elsevier Computer Networks Journal*, 2014.

[28] D. Gunter, L. Ramakrishnan, S. Poon, G. Pastorello, V. Hendrix, D. Agarwal, "Designing APIs for Data-Intensive Workflows: Methodology and Experiences from Tigres", *IEEE e-Science*, 2013.

[29] P. Calyam, S. Seetharam, R. Antequera, "GENI Laboratory Exercises Development for a Cloud Computing Course", *Proc. of GENI Research and Educational Experiment Workshop*, 2014.

[30] Ryu - Component-based software-defined networking framework; http://osrg.github.io/ryu

[31] P. Calyam, A. Berryman, A Lai, M. Honigford, "VMLab: Infrastructure to Support Desktop Virtualization Experiments for Research and Education", *VMware Technical Journal*, 2012.

[32] T. Joshi, M. Fitzpatrick, S. Chen, Y. Liu, H. Zhang, R. Endacott, E. Gaudiello, G. Stacey, H. Nguyen, D.Xu, "Soybean Knowledge Base (SoyKB): A web resource for integration of soybean translational genomics and molecular breeding", *Nucl. Acids Res*, 2014.

[33] Information Science Institute - http://www.isi.edu

[34] Texas Advanced Computing Center - https://www.tacc.utexas.edu

[35] Extreme Science and Engineering Discovery Environment - https://www.xsede.org

[36] P. Calyam, S. Kulkarni, A. Berryman, K. Zhu, M. Sridharan, R. Ramnath, G. Springer, "OnTimeSecure: Secure Middleware for Federated Network Performance Monitoring", *IEEE Conf. on Network and Service Management (CNSM)*, 2013. (https://www.naradametrics.com)

[37] Internet2 InCommon - https://incommon.org

[38] OpenFlow Switch Specification - https://www.opennetworking.org/sdn-resources/onf-specifications/openflow

[39] HTCondor Resource Manager - http://research.cs.wisc.edu/htcondor

[40] E. Deelman, J. Blythe, et. al., "Pegasus: Mapping Scientific Workflows onto the Grid", *Springer LNCS - Grid Computing*, 2004.

[41] Integrated Rule-Oriented Data System (iRODS) - http://irods.org

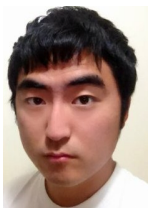
[42] ESnet Fasterdata Knowledge Base - http://fasterdata.es.net/host-tuning/linux

Ronny Bazan Antequera received his MS degree in Computer Science from University of Missouri-Columbia in 2014. He received his BS degree in Computer Science from San Andres University, Bolivia in 2005. He is currently pursuing his Ph.D. degree in Computer Science at University of Missouri-Columbia. His current research interests include hybrid cloud computing and software-defined networking.

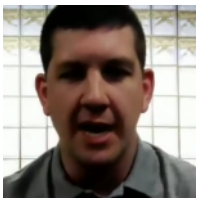
Prasad Calyam received his MS and PhD degrees from the Department of Electrical and Computer Engineering at The Ohio State University in 2002 and 2007, respectively. He is currently an Assistant Professor in the Department of Computer Science at University of Missouri-Columbia. His current research interests include distributed and cloud computing, computer networking, and cyber security.

Saptarshi Debroy received his PhD degree in Computer Engineering from University of Central Florida in 2014, MTech degree from Jadavpur University, India in 2008, and BTech degree from West Bengal University of Technology, India in 2006. He is currently a Post Doctoral Fellow at University of Missouri-Columbia. His current research interests include Cloud Computing, Big Data Networking, and Cognitive Radio Networks.

Sripriya Seetharam received the BE degree in Computer Science from Visvesvaraya Technological University in 2008. She is currently pursuing her MS degree in Computer Science at University of Missouri-Columbia. Her current research interests include distributed systems and networking.

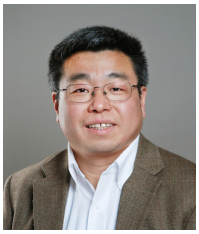


Longhai Cui received his B.S. degree in Beijing Jiaotong University, China in 2013. He is currently pursuing his M.S. degree in Computer Science from University of Missouri-Columbia. His current research interests include cloud computing and software-defined networking.

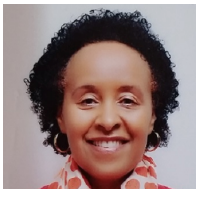


Matthew Dickinson received his M.S. degree in Computer Science from University of Missouri-Columbia in 2009. He is currently pursuing his Ph.D. degree in Computer Science at University of Missouri-Columbia. His current research interests include cloud computing and unmanned aerial systems.

Trupti Joshi received her PhD from the University of Missouri-Columbia in 2013. She received her MS from University of Tennesse-Knoxville in 2003. She is currently a Director, Translational Bioinformatics at University of Missouri-Columbia. Her current research interests include bioinformatics, computational systems biology and genomics.



Dong Xu received his PhD from University of Illinois at Urbana-Champaign. He received his MS and BS from Peking University. He is currently the James C. Dowell Professor and Chair of Computer Science department at University of Missouri-Columbia. His current research interests include bioinformatics, protein structure prediction and computational systems biology.



Tsegereda Beyene received her ME in Systems Engineering from the University of Virginia, and MS in Operations Research from The American University. She is currently a Technical Leader in Cisco Systems, USA. Her current research interests include software-defined networking and smart cities.