ADON: Application-Driven Overlay Network-as-a-Service for Big Data Networking

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

***Abstract*—Campuses are increasingly adopting hybrid cloud architectures for supporting science Big Data 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 principles. 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 using features such as: programmable "custom templates" and "virtual tenant handler". 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 predictable application performance delivery for multi-tenant traffic profiles. We validate our ADON approach by implementing it on a wide-area overlay network testbed across two campuses, and present a workflow that eases the orchestration of network programmability for campus network providers and data-intensive application users. We present an emulation study of the ADON effectiveness in handling temporal behavior of multi-tenant traffic burst arrivals using profiles from a diverse set of actual data-intensive applications. Lastly, we present a real-life comprehensive science Big Data experimental use case which validates the optimization of end-to-end network and computational resources.**

I. Introduction

Data-intensive and Big Data applications in research fields such as bioinformatics, climate modeling, particle physics and genomics generate vast amounts (peta-byte scale) of data that need to be processed 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 physical networks [1].

Current campus network infrastructures 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.

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 [2], (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 being shared, there is an increased push to share computational resources on a campus-level. These cloud-like computational resources make use of virtualization and management platforms such as OpenStack [20]. 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. Furthermore, the complexity in the understanding and configuration of these inter-domain systems causes the system as a whole to be unnecessarily-complex for researchers to effectively use the resources available.

However, 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) [7]. 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. Since RoCE protocol assumed a 10 Gbps link availability and is highly sensitive to packet loss, our application performance suffered severely.

Frequently, the Science DMZ has to compete with traditional general purpose "enterprise" network (i.e., Layer 3/IP network) resource usage. With multiple applications accessing hybrid cloud resources and coupled with traditional campus traffic network, the result can be a significant amount of "friction" for science Big Data movement and can easily lead to performance bottlenecks. Figure 2 illustrates the periodic nature of the enterprise traffic with total bandwidth utilization and the session count of wireless access points at MU through-

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.

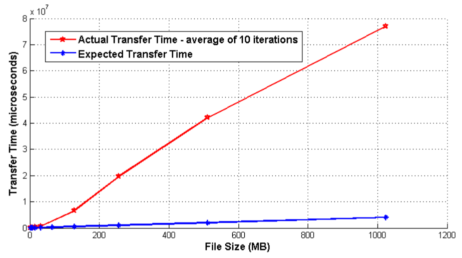
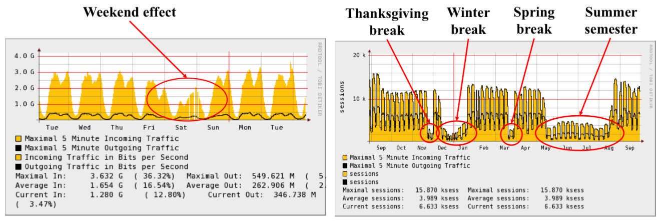


Fig. 1: Illustration to show the need for application performance visibility and control over a wide-area network path



(a) Weekly campus bandwidth utilization (b) Yearly wireless access point session counts

Fig. 2: Illustration of typical network utilization at MU

out the year. Figure 2a shows the overall bandwidth usage is dependent on the time of the day with a significant dip during the latter hours of night and during the weekends. For wireless access points session counts shown in Figure 2b, the frequent student movements around the campus leads to a large number of association and authentication processes to wireless access points, with peaks observed during Fall and Spring semesters. It is obvious that sharing such traditional campus networks with daily and seasonally fluctuating cross-traffic trends causes significant amount of 'friction' for science Big Data movement and can easily lead to performance bottlenecks.

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, and also advanced services should be easy-to-use for data-intensive application users, who should not be worrying about configuring underlying infrastructure resources.

Our work in this paper aims to solve the path, method and network selection problem for data intensive applications in an inter-domain environment, while making network programmability related issues a non-factor for data-intensive application 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 requirements of a particular application in terms of available network and compute resources.

The choice of where (and when) to initiate computation is coupled with the choice of how to move the data, as such the computation of the overall cost, both network and computation must be calculated.

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 "Virtual Tenant Handler" (VTH). The VTH leverages 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 custom templates and VTH concepts, 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), location 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 using a real-world Big Data Science application with which the cost of where to perform computation is also important for the overall end-to-end computation time to completion.

To model the ADON architecture, we devise a generic algorithm to handle a typical workflow consisting of data transfer followed by computation and finally data transfer. This algorithm then allows finer-grain control for individual applications by allowing cost-computation within the context of each application. We validate our ADON implementation by means of optimizing the time-to-completion of a complex generic description of SoyKB [34], utilizing local compute resources at MU, remote compute resources available in the public cloud and finally transfer of processed data for public dissemination in the iPlant [39] environment. In order to achieve this optimization, there are a number of things to consider, including the cost of the network transfer and the availability of compute resources.

The remainder paper organization is as follows: Section II presents related work. Section IV details custom templates for exemplar application workflows. Section III describes ADON's modular architecture with Virtual Tenant Handler. Section V describes the algorithms. Section VI describes ADON implementation on an actual testbed, and in an emulation study. Section VII concludes the paper.

II. Related Work

Existing works such as [8], [9] and [10] recognize similar application-driven network virtualization issues at network-edges, and have proposed new architectures based on SDN principles. In [8], application level requirements are programmed using an inter-domain controller implemented using OpenFlow, and a custom-built extensible session protocol is used to provide end-to-end virtual circuits across campus DMZs. In [9] that is closely related to our work, 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 [10], the authors propose a new controller design for QoS based routing of multimedia traffic flows.

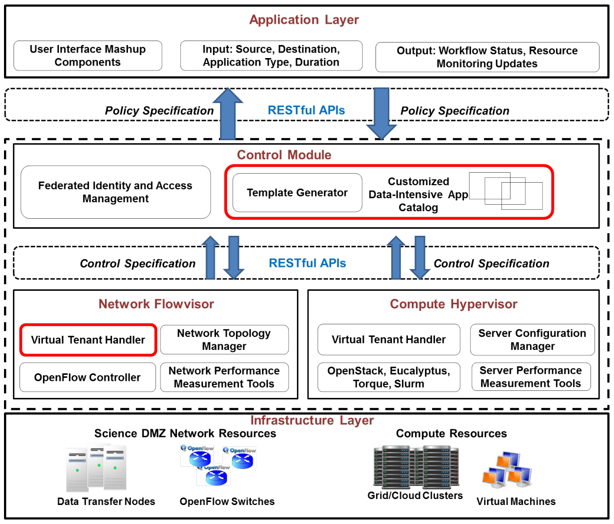


Fig. 3: ADON reference architecture

NaaS models have been studied in existing works such as [8] - [3] that provide various approaches to network virtualization especially for data intensive application. In [8], application level requirements are programmed using an inter-domain controller implemented using OpenFlow, and a custom-built extensible session protocol is used to provide end-to-end virtual circuits in campus DMZs. Works such as [3] have focused on building a new proprietary OpenFlow controller that builds on the flowvisor idea of campus network slicing.

Network overlays for data-intensive applications with real-time or urgent computing require more fine-grained control over network resources because the main goal of such application support involves fast and reliable movement of data-intensive transfers with guaranteed Quality of Service(QoS). In [8] application level requirements are programmed using an inter-domain controller implemented using OpenFlow, and a custom-built extensible session protocol is used to provide end-to-end virtual circuits in campus DMZs. Works such as [9] and [10] underline the need for dynamically programming QoS especially for multimedia applications. In [13], QoS requirements are discussed in a SDN environment for end-to-end scientific applications. [15] proposes methods for the estimation of execution time.

Furthermore, an efficient and flexible resource management is a key factor in any datacenter due to the provisioning of resources [11], hence hybrid cloud infrastructure requires to have detailed control of the resources especially when reservation concept is including as part of the whole process. In that sense, cloud infrastructure scheduling plays an important role during on-demand and dynamic provisioning and reservation of computing resources [12] in single or large heterogeneous multi-cloud infrastructure [16]. Authors in [17] and [18] consider implications of HPC and cloud computing paradigms for data intensive applications in order to Infrastructure-as-a-Service cloud model workflows for data processing on the public cloud. In [14], the issues surrounding the automatic deployment of HPC and database applications in a hybrid cloud environment is discussed.

III. ADON Architecture

In this section, we describe the policy implementation architecture of ADON that leverages the custom templates

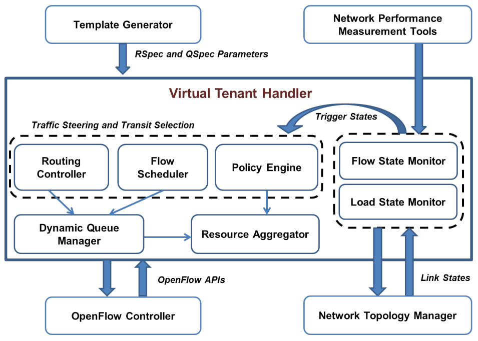


Fig. 4: Individual components of the Virtual Tenant Handler

for fine grain control and customization of programmable resources. Figure 3 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' and Network Flowvisor module's 'Virtual Tenant Handler' (highlighted in red in Figure 6) that are necessary to implement ADON, and the Compute Hypervisor issues are beyond this paper scope.

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 [25]. 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.

B. *Virtual Tenant Handler*

The Virtual Tenant Handler (VTH) is responsible for dynamically handling incoming application flows and providing the intelligence for adaptive network resource allocation. As

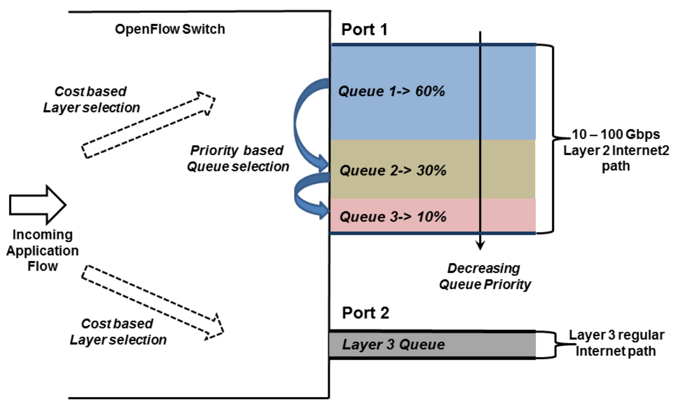

OpenFlow Switch
Cost based Layer selection
Priority based Queue selection
Incoming Application Flow
Cost based Layer selection
Port 1
Queue 1-> 60%
Queue 2-> 30%
Queue 3-> 10%
10 – 100 Gbps Layer 2 Internet2 path
Decreasing Queue Priority
Port 2
Layer 3 Queue
Layer 3 regular Internet path

Fig. 5: Illustration of dynamic layer selection and queue management

shown in Figure 4, the Policy Engine interacts with the Template Generator component on the top layer for accepting the $RSpec$ and $QSpec$ 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 the port number 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 [26]. The configured queues on the OpenFlow switch ports are then mapped to incoming application flows using the *set-queue* action set of the OpenFlow specification. As shown in Figure 5, 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 availability 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, or else the flow 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 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

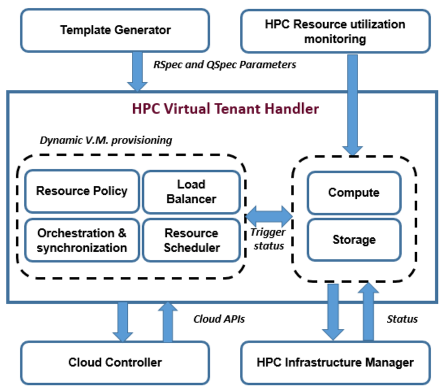

Template Generator
HPC Resource utilization monitoring
RSpec and QSpec Parameters
HPC Virtual Tenant Handler
Dynamic V.M. provisioning
Resource Policy
Load Balancer
Orchestration & synchronization
Resource Scheduler
Trigger status
Compute
Storage
Cloud APIs
Status
Cloud Controller
HPC Infrastructure Manager

Fig. 6: HPC Virtual Tenant Handler components

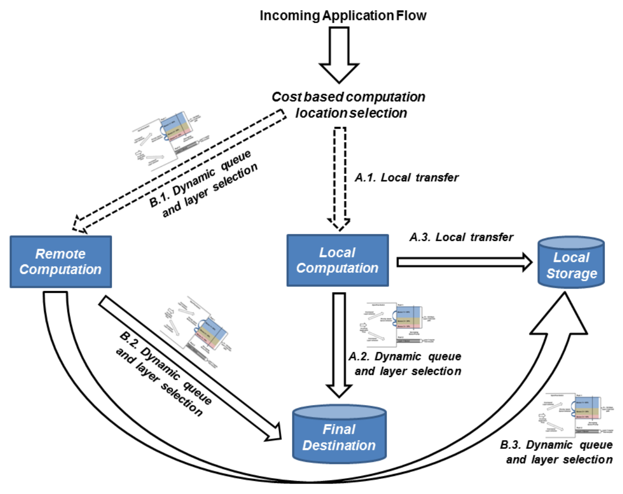

Incoming Application Flow
Cost based computation location selection
B.1. Dynamic queue and layer selection
A.1. Local transfer
Remote Computation
Local Computation
A.3. Local transfer
Local Storage
B.2. Dynamic queue and layer selection
A.2. Dynamic queue and layer selection
Final Destination
B.3. Dynamic queue and layer selection

Fig. 7: Illustration of computation location selection

inputs from the “Network Performance Measurement” module which is a real-time SDN monitoring system that provides network health status notifications such as flow statistics and other such QoS impacting factors. The module provides options to reconfigure existing paths based on the custom template directed resource reservations. In case the load state in terms of number of applications contending for Layer 2 service creates a link saturation, VTH can steer the traffic on Layer 3 if acceptable QoS parameters can allow a push of the new flows into the Flow Scheduler.

The 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 VTH module with topology link states. The algorithm used in the VTH is formally discussed Section V.

~~*C. HPC Virtual Tenant Handler*~~

The HPC Virtual Tenant Handler is responsible for local and remote HPC provisioning process using suitable HTCondor [43] configurations and OpenStack based on detailed resources utilization and performance as show in Figure 6. Template Generator module collect all users requirements (RSpec and QSpec parameters) that are used by Resource Policy for the best asset selection of the cluster, Load Balancer manages scalability process in conjunction with Orchestration & Synchronization for different nodes deployed where multi-tenant aware is considered as well as maximum HPC capabilities and Resource Scheduler administers deployment HPC jobs. All this process controller by Cloud Controller module. On the other hand, HPC Resource Utilization Monitoring is a real time HPC monitoring system that provides information of the available computing resources, execution and remaining time, and detailed information of executed jobs and Storage. HPC Infrastructure Manager initialize HPC resources considering Compute and Storage Information.

Through the help of VTH and HPC-VTH, we achieve the end-to-end overall network and computational resource optimization as illustrated in Figure 7. Here we show a typical science Big Data application life-cycle where the final repository of processed data is reached through two possible paths involving either local or remote computation. The choice of computation location is mandated by a comprehensive cost optimization which takes into consideration factors like availability of computation resources, rescue utilization by existing applications, and $RSpec$ of the new application. At each stage of network transfer of either raw or processed data, VTH performs dynamic layer and queue selection thereby optimizing network resources.

IV. Network Personalization for Applications within ADON

Figure 8 shows how data-intensive applications can co-exist on top of a shared wide-area physical infrastructure topology, with each application demanding local/remote network or compute resources with unique end-to-end requirements. In order to effectively handle the diverse requirements at the campus edge router, several challenges need to be addressed. One of the important challenges is the need to have "application performance visibility and control" within the provisioned resources for individual applications. This requires 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.

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) [19] and $RSpecs$ in the NSF-supported Global Environment for Network Innovations (GENI) [21]. Works such as [22] 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 9 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

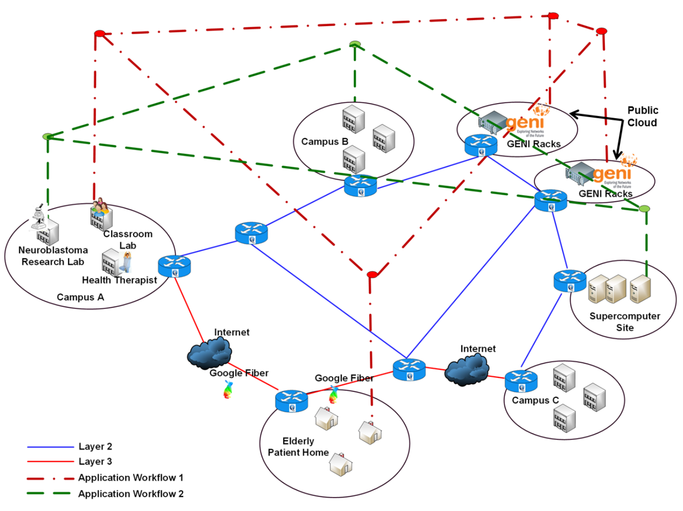


Fig. 8: Multi-tenant application workflows on a shared wide-area physical infrastructure

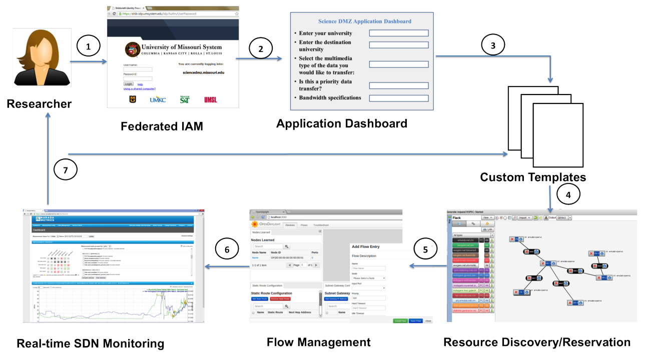

1
2
3
4
5
6
7
Researcher
Federated IAM
Application Dashboard
Science DMZ Application Dashboard
Enter your university
Enter the destination university
Select the multimedia type of the data you would like to transfer:
Is this a priority data transfer?
Bandwidth specifications
Custom Templates
Resource Discovery/Reservation
Flow Management
Real-time SDN Monitoring

Fig. 9: Sequential workflow of ADON during on-demand resource provisioning for a data-intensive application flow

steps in ADON orchestration are as follows: First, a researcher 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? or (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 be pre-configured by a Performance Engineer to apply specific resource descriptions and associated campus policies that can be interpreted by for e.g., a network flowvisor (i.e., proxy for OpenDaylight or POX [24]) to instantiate flows on intermediate OpenFlow switches, and by a compute hypervisor to instantiate virtual machines within a data center. We refer to Performance Engineer as the one 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. The Science DMZ is capable of functioning without a Performance Engineer, however that individual is able to communicate with researchers ahead of time and pre-plan templates that could otherwise be complex to alleviate the burden on researchers.

In addition to helping the Performance Engineer with the resource configuration, custom templates also help in configuring real-time network performance monitoring within the overlay network path to provide the application performance visibility that can be used 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.

Real-time network performance monitoring on the campus edge router will provide application visibility and control to the Performance Engineer for dynamic resource adaptation. The manual intervention performed by the Performance Engineer can be omitted and the custom template can be automatically re-applied if a similar specification was successfully handled. The Performance Engineer is also able to adapt templates in advance of events such as scheduled maintenance or upgrades, that the ADON system would not necessarily have intrinsic knowledge of.

1) Neuroblastoma Data Cutter Application: As shown in Figure 10(a), the workflow of the Neuroblastoma application [7] 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 required for processing these images 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 highly 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) $RSpec$ - parallel TCP flows with high bandwidth bursts, 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 10(b), the RIVVIR application [23] 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) $RSpec$ - Layer 2 network steering (over Internet2) of application traffic, 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 10(c), a class of 30 students conducting lab experiments at MU in a Cloud Computing course [24] require resources across multiple GENI racks. As part of the lab exercises, multiple VMs need to be reserved and instantiated on remotely located GENI racks. There can be a 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) $RSpec$ - Multiple layer 2 flows to remote sites with low bandwidth requirements, and (ii) $QSpec$ - Low packet loss and medium flow priority to allow students to finish the lab exercises.

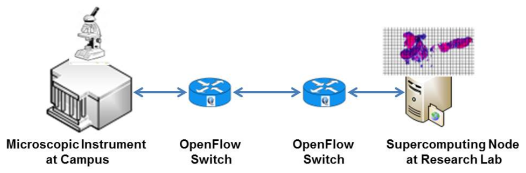
4) ElderCare-as-a-Service Application: As shown in Figure 10(d), the ElderCare-as-a-Service application [24] consists of an interactive video streaming session between a therapist on MU campus and a remotely residing elderly patient at a Kansas City residence for performing physiotherapy exercises as 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) $RSpec$ - Deterministic flow path with high-resilience for network link failures on Layer 3 (regular Internet with Google Fiber last-mile), and (ii) $QSpec$ - Consistent high available bandwidth with very low or no jitter and high flow priority to allow elder to closely follow the postures being exercised in the session.

5) SoyKB: Soybean Knowledge Base (SoyKB), 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 data being transferred to MU for pre-processing and subsequently being transferred to remote HPC sites for analysis as shown in Figure 11. The process can suffer from long run-time for the overall process due to workloads at HPC centers and slow network transmission of large datasets despite high bandwidth connectivity, such as Internet2. A corresponding application template can be (i) $RSpec$ - Layer2 or Layer3 connection to remote compute/data storage site dependent on path selection and (ii) $QSpec$ - high flow throughput with little to no packet loss to provide fast-enough transfer to validate the path selection.

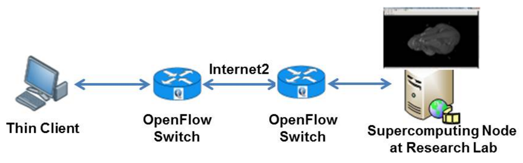
V. ADON Resource Allocation

The VTH receives 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 forward the translated information to the VTH. For the VTH, $QSpec$ 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.

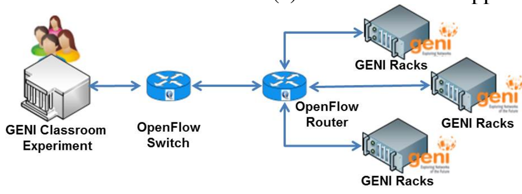
Upon receipt of any such application resource allocation request, the VTH is responsible for the allocation of op-


Microscopic Instrument at Campus
OpenFlow Switch
OpenFlow Switch
Supercomputing Node at Research Lab

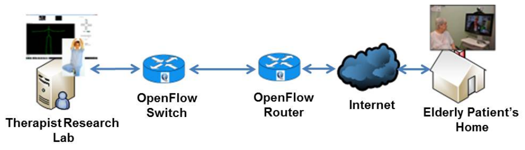
(a) Neuroblastoma application



(b) RIVVIR application



(c) GENI classroom experiments application



(d) ElderCare-as-a-Service application

Fig. 10: Topologies of various data-intensive applications that need network personalization

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

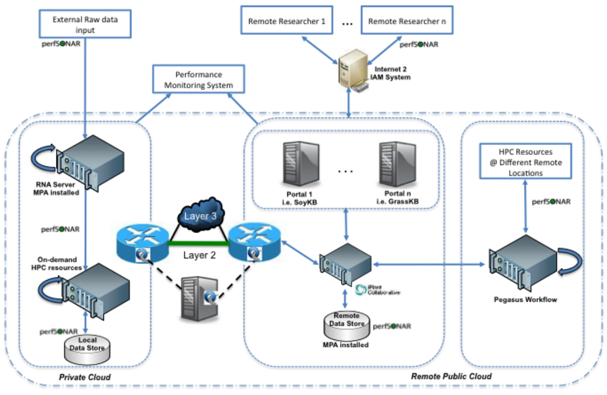


Fig. 11: SoyKB wokflow

timal network and computational resource from the 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 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 VI-C we use a specific cost metric custom designed for the applications we cater.

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 : i) data is moved from source to point of computation (using one or more network paths). ii) computation occurs at an appropriate compute location. 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 it allows 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 $R_{a_x} = \{R_x^C, R_x^N\}$ represents the optimal resource allocation for allocation a_x with R_x^C being the allocated computational resource and R_x^N being the allocated network resource. The resource allocation algorithm should then guarantee R_x^N to satisfy the application $QSpec$ and R_x^C 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,

$$\begin{aligned} &\text{minimize} \quad \mathcal{C}(R_{a_x}) \\ &\text{subject to} \quad q_{xk}^{lb} < \mathcal{Q}^k(R_x^N) \le q_{xk}^{o} \quad \forall \;\; k \\ &\qquad\qquad\qquad \& \quad R_x^C \ge RS_x^C \end{aligned} \tag{1}$$

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) and takes the final decision on allocating compute resources, suitable Layer 2 or Layer 3 network resources, and assigning the proper queue within the network tunnel.

The Edge Optimization algorithm thus solves the cost computation problem to chose the optimal network, and optimal compute resources and also solves the dynamic queue assignment problem by choosing the most appropriate queue with the network resource using the new application priority $P_{a_{new}}$.

Algorithm 1 ADON 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 $\mathcal{R} \ge R_T$ then
  /*Check for local computation*/
  $C_L = computeLocal(QS_{a_{new}}, RS_{a_{new}}, P_{a_{new}}, P_{new})$
  /*Check for remote computation*/
  $C_R = computeRemote(QS_{a_{new}}, RS_{a_{new}}, P_{a_{new}}, P_{new})$
  /*Compare computation costs*/
  if $C_L < C_R$ then
    return $\{R_{new}^C = RS_{new}^C, R_{new}^N = RS_{new}^N\}$
    $\mathcal{R}^L = \mathcal{R}^L - \{R_{new}^C, R_{new}^N\}$
    Include $a_{new}$ to $A$
  else
    return $\{R_{new}^C = RS_{new}^C, R_{new}^N = RS_{new}^N\}$
    $\mathcal{R}^R = \mathcal{R}^R - \{R_{new}^C, R_{new}^N\}$
    Include $a_{new}$ to $A$
  end if
else
  Push $a_{new}$ to resource scheduler
end if
end procedure
```

The Edge Optimization Algorithm 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, and transfer time and available network bandwidth for transfer influenSoyKBce the computation location decision and thus affects 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.

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 available at the local site meets the $RSpec$ requirements of the application, and also if the available bandwidth to transfer the processed data to the final destination site meets the $QSpec$ requirements of the application. Such transference of the processed data is carried out either through the legacy best effort Layer 3 infrastructure, i.e., Internet, or through dedicated high speed Layer 2 infrastructure such as Internet2. The choice of network infrastructure depends upon factors such as the capacity of the network, ensuing network performance metrics like loss or latency, and current utilization and availability of future network resources etc. We designed the ADON resource allocation in such a way that the 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 greedy provisioning is saving the precious Layer 2 services dedicated for future application with higher bandwidth requirements 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 VTH tries to force deallocate the Layer 2 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 with fewer incurred 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 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. It goes without saying that all the transference costs involve both Layer 2 and Layer 3 transfer scenarios.

The output of algorithms 2 and 3 are compared in the

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
    /*Check for L3 transfer*/
    if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L3}, RS_{a_i}) < q_{ik}^o \ \forall$ 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
    /*Check for L2 transfer*/
    if $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2}) < q_{ik}^o \ \forall$ 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 $q_{ik}^{lb} < estimateQoS(\mathcal{R}_N^{L2} + R_{min}^N) < q_{ik}^o \ \forall$ 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
```

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.

VI. Experimental Evaluation

In this section, we first describe a case study featuring validation experiments of our ADON implementation on a wide-area overlay network testbed across OSU and MU campuses connected with Internet2 AL2S. Next, we present a detailed emulation study we conducted with application workflows to analyze the VTH algorithm results while handling temporal behavior of multi-tenant traffic burst arrivals.

A. Testbed Setup

The testbed setup as shown in Figure 12 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 in Ohio, and GPN/MoreNet in Missouri, respectively. Each Science DMZ has a matching DTN equipped with dual Intel E5-2660, 128GB of memory, 300GB PCI-Express solid state drive, and dual Mellanox 10 Gbps network cards with RoCE support. Each Science DMZ has perfSONAR measurement points for continuous monitoring at 1 - 10 Gbps network speeds. A common Dell R610 node in the OSU Science DMZ is used to run the VTH module along with OpenDaylight OpenFlow controller that controls both the OSU and MU Science DMZ OpenFlow switches.

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
    /*Check 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
    /*Check 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
        /*Check 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
        /*Check 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 Science DMZ at OSU contains a heterogeneous mix of OpenFlow switches from NEC and HP. The NEC switch on the OSU end is connected to OSU's100 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 performed an experiment by providing the application requirements such as the source site, and destination site - along with application type, which is a real-time image trans-

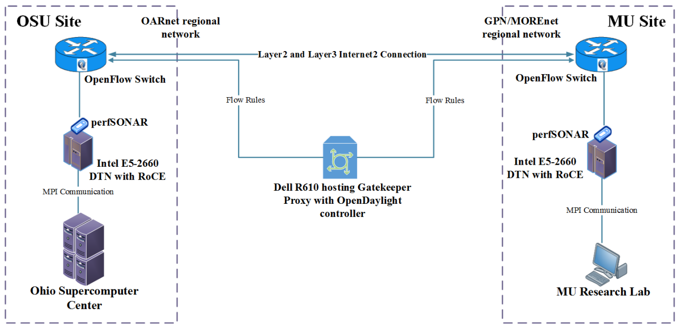

OSU Site
OARnet regional network
GPN/MOREnet regional network
MU Site
Layer2 and Layer3 Internet2 Connection
OpenFlow Switch
Flow Rules
Flow Rules
OpenFlow Switch
perfSONAR
perfSONAR
Intel E5-2660 DTN with RoCE
Intel E5-2660 DTN with RoCE
MPI Communication
MPI Communication
Dell R610 hosting Gatekeeper Proxy with OpenDaylight controller
Ohio Supercomputer Center
MU Research Lab

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

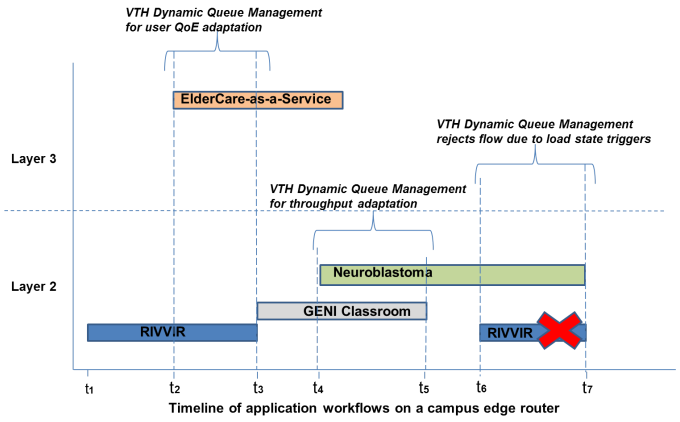

VTH Dynamic Queue Management for user QoE adaptation
ElderCare-as-a-Service
VTH Dynamic Queue Management rejects flow due to load state triggers
Layer 3
VTH Dynamic Queue Management for throughput adaptation
Layer 2
Neuroblastoma
GENI Classroom
RIVVIR
RIVVIR
t1
t2
t3
t4
t5
t6
t7
Timeline of application workflows on a campus edge router

Fig. 13: Timeline of a campus edge router handling multiple data-intensive application workflows

fer for processing of medical images within Neuroblastoma application - by following the steps illustrated previously in Figure 9. The application requirements and policy specifications were configured as high bandwidth, Layer 2 routing with priority queue assignment in the corresponding custom template. The compute instances on Owens cluster at OSU were reserved, including 6 running GPU nodes and memory space of 3000 MB at both ends. We also introduced tcp cross traffic using Iperf tool to simulate a parallel flow along with research application flow.

These specifications were then mapped by the VTH module to lower-layer control policies in order to instantiate the flow rules on the OpenFlow switches at both campus edges. Once the flow rules were in place with our ADON implementation, the data flow was automatically instantiated on the infrastructure layer with desired QoS and no manual intervention. The research application flow was instantiated on Layer 2 (Internet2 domain) while the cross traffic was steered on Layer 3 (regular Internet) by the OpenFlow switch from MU end. The RoCE traffic was detected based on the ether type 0x8915 and Iperf as 0x800. A 500 MB image file was transferred from MU to OSU Owens cluster in 5.13 seconds. The time taken to process the image was about 4 seconds (i.e., compute time) on the cluster and the transfer of processed image back to MU took around 5.2 seconds (i.e., communication time) aggregating to a total of 14.4 seconds of "compute plus communication" time within the ADON provisioned path.

B. *Emulation Study*

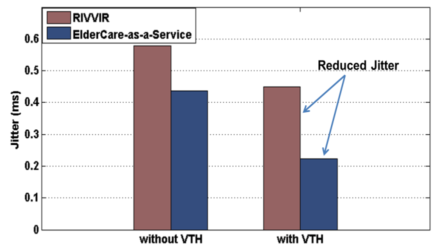
We used Mininet network emulator for VTH experiments that involved synthetic traffic flows using the "tc" and "iperf" network utility tools. Figure 13 shows the timeline from time t_1 to t_7 as seen by an edge OpenFlow router/switch handling application workflows as they enter and exit the VTH module. RIVVIR application workflow (see Figure 10(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 10(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 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 VTH dynamic queue management in Figures 14(a). We can see that the jitter is significantly reduced (improved performance) for both the applications, especially for the video flow using the VTH module with application-specific queue policies. This is due to the fact that the 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 QoS metric values.

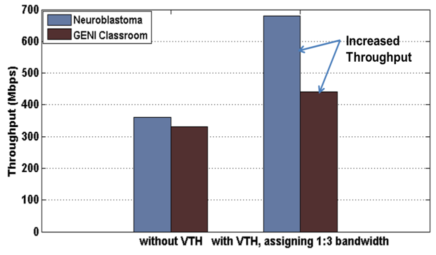
At time t_3, GENI Classroom workflow (see Figure 10(c)) was started as a TCP flow with burst traffic pattern (burst rate - 1 Mbps and buffer size- 10 KB similar to web traffic) parallel to the RIVVIR workflow. Both flows co-existed without affecting each other's QoS policies as both flows were assigned their individual priorities on the queues. At time t_4, Neuroblastoma workflow (see Figure 10(a)) was started as a parallel TCP flow with 5 parallel streams to simulate a GridFTP application for image file transfer. 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 14(b) shows throughput of the two workflows achieved with and without VTH. Bandwidth is equally split when there is no dynamic queue management for both flows. However with 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 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



(a) Jitter comparison



(b) Throughput comparison

Fig. 14: Results of dynamic queue assignment by VTH

burst arrivals corresponding to a diverse set of data-intensive applications.

C. SoyKB Implementation Case Study

In the first section, we detail the original SoyKB implementation and the resulting lack of performance due to architectural constraints. In the second section, we detail the modified architecture available by making use of the ADON model system.

1) SoyKB implementation: The raw Next Generation Sequencing (NGS) data is mailed from partner institutions (e.g., China), pre-processed by the team at MU, and later transferred to remote HPC sites for analysis. For the resequencing analysis, the team uses three available iPlant connected HPC infrastructures at: ISI (Information Sciences Institute) [36] TACC (Texas Advanced Computing Center) [37] and XSEDE [38]. The iPlant resources at the University of Arizona [32] serve as SoyKB's Data Store, and can be accessed using Integrated Rule-Oriented Data System (iRODS) [35] Data can be replicated from the University of Arizona to the servers at other HPC sites over Internet2, which allows low latency data access when running the workflows on HPC resources. The Pegasus workflow [33] system is used to control data movements, computations and execution in HPC environments. This includes user defined tasks, such as BWA [40], Picard [42], and GATK, as well as Pegasus added tasks such as data staging. The execution environments are based on HTCondor [43], a specialized workload management system (i.e., job scheduler) for compute-intensive jobs in distributed environments, allowing users to place their jobs into a queue for batch computation. Finally, the analyzed data is sent to an iPlant virtual machine ("Earnshaw Server"), which hosts the SoyKB website for researchers to view/download the data. In order to scale the SoyKB infrastructure and services to meet the KBCommons need to provide a comprehensive web resource at the front-end and backend powered by adequate resources to handle multi-omics data integration for a given species, there are certain limitations to be overcome in expansion activities.

Limitations in legacy SoyKB infrastructure that motivated the hybrid cloud requirements and proposed activities around a hybrid cloud architecture design and related services implementation include the following:

i) The first, important step involves data transfer between MU and other remote compute sites such as iPlant, which can take several days using traditional file transfer tools even over Internet2 paths for large data. Faster data transfer capability over the network through Science DMZ environments with advanced data transfer protocols and data transfer nodes that use services such as Globus Online [41] to interface with the HTCondor execution environment of a particular KB for a given species, can significantly improve efficiency and multiple KB workflows setup.

(ii) Secondly, there was some manual coordination to reserve resources, and configure the Pegasus workflow. Increasing the level of automation is necessary to accomplish resource discovery and KB environment setup by using cloud operating system stacks such as OpenStack.

(iii) The setup did not provide easy access to resource availability and performance information (such as availability, status, wait times, network health etc.), nor flexible control of resources to address special needs or jobs prioritization, causing significant delays in workflow completion due to other jobs. To overcome these performance visibility limitations, there needs to be a dedicated effort to build a "measurement plane" across the KBCommons infrastructure to instrument resources with active (e.g., perfSONAR) and passive (e.g., sFlow) tools, and having services to effectively collect and analyze various workflow related custom metrics. For flexible control of compute resources within KBCommons, a "control plane" needs to be developed that allows local MU HPC resources to be added on-demand to the Pegasus workflow related resource pools, whose job execution policy can be managed with priorities based on the individual KB needs. In cases where flexible network control is desirable for important paths that handle large data flows between the private cloud and public cloud, we can implement software-defined networking as part of the control plane functions. The implementation will involve development of suitable OpenFlow controllers that can be integrated with federated "flowspaces" of MU, Internet2 and iPlant.

(iv) Lastly, the legacy setup did not facilitate creation of "templates" that can provide information about previous successfully executed workflow configurations and access control rules for a particular KB use case. In order to truly support "self-service" capabilities within KBCommons, we will need to create services that help with re-use of workflows for repeatable execution for different resource configurations (e.g., pipeline with a different data store, or compute at a different HPC site). In addition, user authentication and authorization through Federated IAM using Shibboleth-based entitlements will need to be implemented as part of a "security plane" to manage multi-tenancy access within KBCommons.

2) SoyKB visibility and performance monitoring: As discussed, there is a need for end-to-end performance monitoring. To achieve this, we instrument the infrastructure with perfSONAR measurement points based upon our OnTimeSecure framework [44]. By using a system such as this, it will enable performing various custom metrics integration at system, network and application-levels into relevant measurement archives, and obtain timely and accurate performance intelligence (e.g., log analysis and anomaly event notifications). In fact, we have developed a production level software-defined measurement (SDM) and performance monitoring system - NaradaMetrics [45] based on OnTimeSecure framework from which we are collecting our measurement data for the following experiments.

3) Algorithm implementation design 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 actual specific requirements and heuristics of each application. The purpose of calculating this cost is to decide: is computation performed locally (at MU) or does it use remote HPC resources (at ISI, TACC or XSEDE); and, do we use AL2S or regular IP network connection to transfer the data to iPlant data store. In the SoyKB application, for example, it is vital for users to get the analyzed results as soon as possible after they run the application with a large size of input data, which will speed up their bio-informatics research process. To meet this requirement, we can define the cost 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 can map it onto available compute resources and execute the steps in appropriate order. However, it is possible that the service cannot find enough resource and puts the task in a prioritized queue. When there are many application task waiting in the Workflow Management Service queue, the SoyKB application has to wait for a long time to start Pegasus workflow. As such, we have to take this time into consideration and we denote it as T_q. 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 S_c, number of the worker nodes assigned N and the average throughput of each worker nodes Γ_c, we can easily calculate the total estimated computation time by adding the waiting time and actual computation time with the equation below. One thing to note is that, currently the local HPC resources with Pegasus workflow are not available for SoyKB, which means the value of T_q is infinite and we are always using remote HPC resources at ISI, TACC or XSEDE.

All of the SoyKB research data is stored in the iPlant Data Center and is available to query through website hosted in iPlant Atmosphere cloud platform that is similar to Amazon EC2. The iPlant collaborative is using iRODS 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 delay time RTT of the transfer, we can configure the TCP buffer size and length of the processor input queue at the end nodes accordingly by following TCP tuning process to achieve theoretical maximum throughput Γ_t. However, the AL2S and regular Layer 3 network connection is not always able to provide the maximum bandwidth for SoyKB especially when there is a lot of other traffic going through the links.

Again, the network monitoring system can 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 Γ_a or directly measure single thread TCP throughput and treat it as approximate available throughput left. 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 S_t is known to our middleware service. Another important thing to note is that, the analyzed data will be significantly reduced after the computation process and will be only approximately 1/10th of the original raw data size. This indicates that even if the local HPC cloud resource is not as powerful as remote cloud as TACC, it might still provides an increased overall performance by significantly reducing the estimated wait-time in the task queue for the computation part and reducing the size of the data to be transferred for the network transfer part.

TABLE II: Notations used for the SoyKB Algorithm

T_t	Estimated total time for getting results
T_c	Estimated data computation time
T_n	Estimated network transfer time
T_q	Estimated waiting time in the task submission queue
S_c	Size of the data to be computed
S_t	Size of the data to be transferred
N	Number of the worker nodes in Pegasus workflow used for computation
Γ_c	Average compute throughput of each worker node in Pegasus workflow
Γ_t	Theoretical maximum throughput after TCP tuning in DTNs
Γ_a	Available TCP throughput measured by monitoring system

Algorithm 4 SoyKB application local or remote cost computation

```
begin procedure
/*computeCost()*/
$T_t = T_c + T_n$
$T_c = T_q + S_c / (N * \Gamma_c)$
$T_n = S_t / \min(\Gamma_t, \Gamma_a)$
end procedure
```

4) SoyKB experimental results: In this section, we take the SoyKB application as a use-case example and evaluate the performance of the ADON algorithms and implementation for this specific use-case. Since the Science DMZ infrastructure such as Openflow switches at the iPlant data center (where the actual big bioinformatics data is transferred) and high bandwidth AL2S network connection are not operational yet, we are conducting these experiments in the MU-OSU science DMZ testbed (see VI-A for the details). Also due to the reason that the HPC computation resources, such as TACC, are not yet ready to run the Pegasus workflow for SoyKB bioinformatics data, we focus more on improving the data transfer rather than data computation.

First we perform SoyKB data transfer with and without the ADON middleware service to show how our network measurement system can help to intelligently and confidently make decisions for SoyKB application workflow to get the best performance while meeting the QSpecs with the help of our SDM system. Then, we present the performance improvement after TCP tuning process to provide a better Custom Template configuration for SoyKB system. Then, we present the performance improvement after TCP tuning process to provide a better Custom Template configuration for SoyKB system. For both experiments we are monitoring both Layer 2 and Layer 3 connection between MU and OSU and mainly getting measurement data - one thread TCP throughput since in SoyKB use-case the most important thing that the researchers care about is the how fast they can get the data transferred to remote side for analysis.

Experiment to demonstrate ADON effectiveness: Unlike the real time Neuroblastoma or ElderCare-as-a-Service application whose goals are to reduce the jitter in order to provide better QoE for the users, SoyKB application focus more on the time

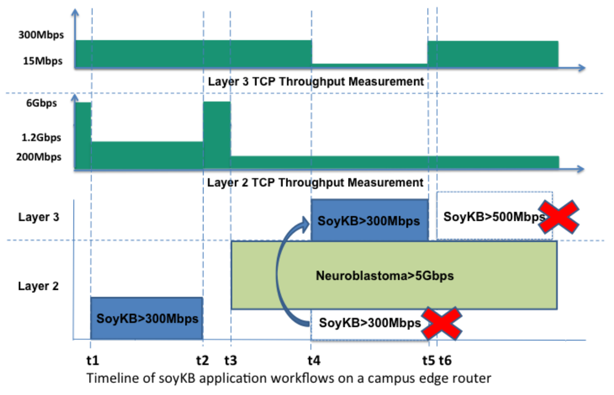
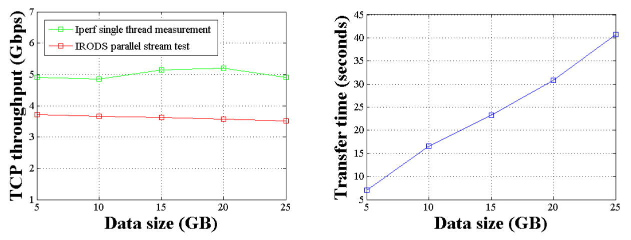

300Mbps
15Mbps
Layer 3 TCP Throughput Measurement
6Gbps
1.2Gbps
200Mbps
Layer 2 TCP Throughput Measurement
Layer 3
Layer 2
SoyKB>300Mbps
SoyKB>500Mbps
Neuroblastoma>5Gbps
SoyKB>300Mbps
SoyKB>300Mbps
t1
t2
t3
t4
t5 t6
Timeline of soyKB application workflows on a campus edge router

Fig. 15: Timeline of Openflow switch handling SoyKB application workflows based on quality specification of application and periodic throughput measurement



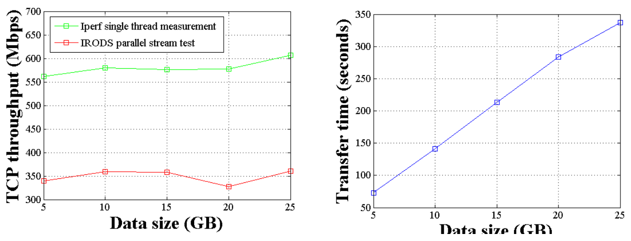
(a) Transfer speed, throughput (b) Transfer time

Fig. 16: Layer 2 performance without contending traffic

taken for the data transfer and data analysis. Regarding the testbed configuration, we set the TCP buffer size 256MB in both sides of MU and OSU DTN. For the experiment, we make sure that the TCP buffer size is always bigger than Γ_a in Algorithm 4, which means our path selection decision is solely dependent on the available bandwidth or an approximated value when the application is run by the SoyKB researchers.

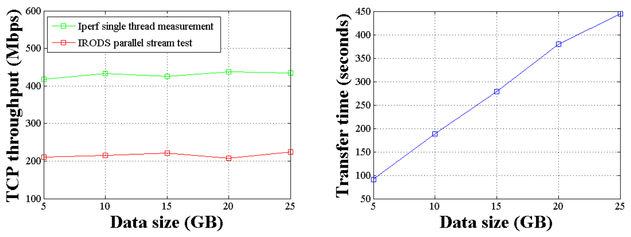
Figure 15 shows the timeline from time $t1$ to $t5$ as seen by our ADON middleware service along with periodic single stream TCP throughout measurement. At time $t1$ SoyKB application workflow is initiated, whose QSpec requires to have at least 300Mbps of single TCP throughput in case iRODS is not able to create more than 1 thread in some worst situations. Predictably, the 10Gbit AL2S network connection between MU an OSU has significantly better throughput than the regular Layer 3 network connection when there are no other application workflows. Since the ADON middleware service can get updated information from the measurement 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 16 shows the overall performance of SoyKB transfer with different data size tested during the above mentioned period. After $t3$, the TCP throughput measured by NaradaMetrics becomes only about 1.5 Gbps from original 6 Gbps since the bandwidth test applications will compete against the research applications for the bandwidth. 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 200Mbps which does not meet another SoyKB workflow started at time $t4$. That is when the ADON middleware service recalculates new estimated transfer time 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.



(a) Transfer speed, throughput (b) Transfer time

Fig. 17: Layer 3 performance without contending traffic

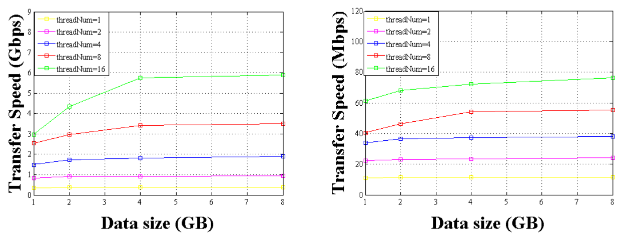


(a) Transfer speed, throughput (b) Transfer time

Fig. 18: Layer 2 performance with contending traffic

Figure 17 shows the performance of SoyKB transfer without any other competing flows in Layer 3. The data transfer speed became slower when compared to the Figure 16, however, it still has way better performance compared to that when continue to transfer SoyKB data in Layer 3 without our resource provisioning algorithm as shown in Figure 18 where it does not even meet the minimum QSpec. Since there is no extra traffic in the Layer 3 connection from MU to OSU DTN, the performance remains the same and we can save more than 100 seconds when transferring a 25GB size file. In bioinformatics, it is very important for the researchers to get transfer and analyze the data as soon as possible and the time difference becomes more significant since the actual research data can be several TB for Big Data application like SoyKB system. When we initiate another SoyKB application workflow requiring at least 500Mbps single throughput at time $t6$, the ADON middleware service will reject the task and put the task in the Flow Scheduler since neither network links can meet the QSpec of the new workflow.

Performance benefits with ADON: Next we perform experiments to demonstrate the performance improvements after the TCP tuning process. While performing these experiments, it was found that the network infrastructure does not provide full advertised throughput. The MU-OSU AL2S link, which advertises a bandwidth up to 10 Gbps, only had a bandwidth of 5 Gbps. When using tools like iRODS that support parallel streams, or have multiple data transfers in parallel, we can



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

Fig. 19: SoyKB transfer performance with different thread counts

configure the system settings such as Linux auto tuning TCP buffer limit and length of the processor input queue in both DTNs to achieve closer to maximum TCP throughput in order to have fair sharing between flows. It is important to change the configurations at both the two edge nodes to ensure that the parallel stream performs well.

Through Figure 19, we observe that the overall transfer speed 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 thread to 16. Therefore, to achieve higher speed in next generation networks, such as 100G Internet2 infrastructures, the iRODS system might need to support a increased number of parallel TCP threads. Thus, the number of threads used for data transfer plays an important role in data transfer speed.

Figures 20 and 21 show the performance of Layer 2 and Layer 3 network connections with different values for the TCP buffer sizes configured in both edge DTNs. For both the cases, we keep the thread count at 16 in order to compare respective maxima. In Layer 3, we are not able to improve the actual data transfer speed since the theoretical maximum throughput after TCP tuning in DTNs exceeds the throughput that the physical regular IP network infrastructure can provide. In contrast, when we increase the TCP buffer size to 64MB from the original 16MB, the data transfer speed increase by almost 50%.

The maximum throughput of Layer 2 is greater than the theoretical maximum TCP throughput with 16MB TCP buffer size, which means there is plenty of room for improvement by using multiple threads for TCP transfer with bigger buffer sizes. As a result, we can observe significant improvement of network performance while transferring data through the 10G Layer 2 link with 64 MB TCP buffer size. However, as it is already very close to the limit of Layer 2 physical infrastructure, further increase of TCP buffer size will not further improve its performance.

VII. Conclusion

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 infrastructures are not scalable or flexible for effectively handling such flows, especially in cases that 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 a "Virtual Tenant Handler" (VTH) for real-time policy control of an "Overlay Network-as-a-Service" within a campus Science DMZ environment with high-performance networking capabilities such as OpenFlow switches and RoCE-based data transfer nodes. Further, we demonstrated how our ADON architecture and implementation were capable of providing predictable performance to data-intensive applications, without any changes to existing campus network infrastructure designed for regular enterprise traffic.

Our future work includes integrating multiple geographically-distributed campuses to the ADON architecture approach as a community model, and conducting additional wide-area overlay network and emulation experiments.

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

[3] Cisco Systems - http://www.cisco.com

[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] 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.

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

[9] 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.

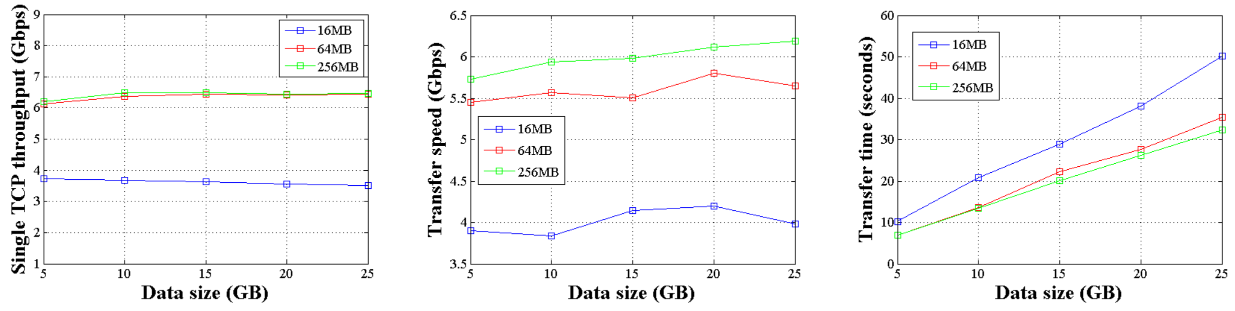
[10] H. Egilmez, S. Dane, K. Bagci, A. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks", *Proc. of APSIPA ASC*, 2012.

[11] Bin Hu; Hong Yu, "Research of Scheduling Strategy on OpenStack," Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on , vol., no., pp.191,196, 16-19 Dec. 2013

[12] Litvinski, O.; Gherbi, A., "Openstack scheduler evaluation using design of experiment approach," Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on , vol., no., pp.1,7, 19-21 June 2013

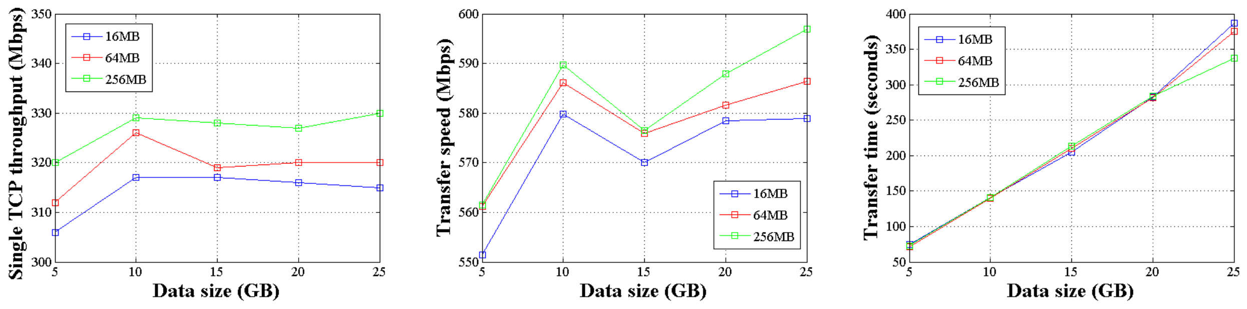
[13] Mehmet Fatih Aktas, Georgiana Haldeman, and Manish Parashar. 2014. Flexible scheduling and control of bandwidth and in-transit services for end-to-end application workflows. In Proceedings of the Fourth International Workshop on Network-Aware Data Management (NDM '14). IEEE Press, Piscataway, NJ, USA, 28-31. DOI=10.1109/NDM.2014.9 http://dx.doi.org/10.1109/NDM.2014.9

[14] Chris Bunch and Chandra Krintz. 2011. Enabling automated HPC / database deployment via the appscale hybrid cloud platform. In Proceedings of the first annual workshop on High performance computing meets databases (HPCDB '11). ACM, New York, NY, USA, 13-16. DOI=10.1145/2125636.2125642 http://doi.acm.org/10.1145/2125636.2125642



(a) Throughput (b) Transfer speed (c) Transfer time

Fig. 20: SoyKB Layer 2 transfer performance with different TCP buffer sizes



(a) Throughput (b) Transfer speed (c) Transfer time

Fig. 21: SoyKB Layer 3 transfer performance with different TCP buffer sizes

[15] Artem M. Chirkin, A. S. Z. Belloum, Sergey V. Kovalchuk, and Marc X. Makkes. 2014. Execution time estimation for workflow scheduling. In Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science (WORKS '14). IEEE Press, Piscataway, NJ, USA, 1-10. DOI=10.1109/WORKS.2014.11 http://dx.doi.org/10.1109/WORKS.2014.11

[16] del Castillo, J.A.L.; Mallichan, K.; Al-Hazmi, Y., "OpenStack Federation in Experimentation Multi-cloud Testbeds," Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on , vol.2, no., pp.51,56, 2-5 Dec. 2013

[17] Resource Management in Data-Intensive Clouds: Opportunities and Challenges, Irwin, D. ; Comput. Sci. Dept., Univ. of Massachusetts, Amherst, Amherst, MA, USA ; Shenoy, P. ; Cecchet, E. ; Zink, M., Local and Metropolitan Area Networks (LANMAN), 2010 17th IEEE Workshop on.

[18] Younge, A.J.; Fox, G.C., "Advanced Virtualization Techniques for High Performance Cloud Cyberinfrastructure," Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on , vol., no., pp.583,586, 26-29 May 2014 doi: 10.1109/CCGrid.2014.93

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

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

[21] 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.

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

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

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

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

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

[27] Yufei Ren, Tan Li, Dantong Yu, Shudong Jin, Thomas Robertazzi, Brian L. Tierney, and Eric Pouyoul, "Protocols for wide-area data-intensive applications: design and performance issues", In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12).

[28] Hongbin Luo, Hongke Zhang, M. Zukerman, Chunming Qiao, "An incrementally deployable network architecture to support both data-centric and host-centric services", IEEE Network Megazine, vol.28, no.4, pp.58,65, July-August 2014.

[29] Hao Yin, Yong Jiang, Chuang Lin, Yan Luo, Yunjie Liu, "Big data: transforming the design philosophy of future internet", IEEE Network Megazine, vol.28, no.4, pp.14,19, July-August 2014.

[30] Xiaomeng Yi, Fangming Liu, Jiangchuan Liu, Hai Jin, "Building a network highway for big data: architecture and challenges", IEEE Network Megazine, vol.28, no.4, pp.5,13, July-August 2014.

[31] A. Rajendran, P. Mhashilkar, Hyunwoo Kim; D. Dykstra, G. Garzoglio, I. Raicu, "Optimizing Large Data Transfers over 100Gbps Wide Area Networks", IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 2013.

[32] http://uanews.org/story/ua-led-research-collaborative-awarded-50m-to-advance-cyberinfrastructure-for-the-life-sciences

[33] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, Daniel S. Katz. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems - Scientific Programming Journal, Vol 13(3), 2005, Pages 219-23

[34] Joshi T, Fitzpatrick MR, Chen S, Liu Y, Zhang H, Endacott RZ, Gaudiello EC, Stacey G, Nguyen HT, Xu D. Soybean knowledge base (SoyKB): a web resource for integration of soybean translational genomics and molecular breeding. Nucl. Acids Res. (1 January 2014)42 (D1): D1245-D1252. doi: 10.1093/nar/gkt905

[35] Integrated Rule-Oriented Data System (iRODS) - The iRODS Consor-

tium, which at the time of writing consists of RENCI (Renaissance Computing Institute) and DICE (Data Intensive Cyber Environments Center) at the University of North Carolina at Chapel Hill, DDN (DataDirect Networks), the Wellcome Trust Sanger Institute, EMC Corporation, and Seagate.

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

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

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

[39] Goff, Stephen A. et al., ”The iPlant Collaborative: Cyberinfrastructure for Plant Biology,” Frontiers in Plant Science 2 (2011), doi: 10.3389/fpls.2011.00034.

[40] Li, H. and R. Durbin, Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 2009. 25(14): p. 1754-60.

[41] I. Foster, ”Globus Online: Accelerating and Democratizing Science through Cloud-Based Services”, IEEE Internet Computing, Vol. 15, No. 3, pp. 70-73, 2011.

[42] http://picard.sourceforge.net/, P.

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

[44] 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) (Short Paper), 2013.

[45] NaradaMetrics Software Defined Measurement and Monitoring: https://www.naradametrics.net/.