ALTO WG
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

Q. Xiang
Tongji/Yale University
H. Newman
California Institute of Technology
G. Bernstein
Grotto Networking
H. Du
Tongji/Yale University
K. Gao
Tsinghua University
A. Mughal
J. Balcas
California Institute of Technology
J. Zhang
Tongji University
Y. Yang
Tongji/Yale University

July 3, 2017

Resource Orchestration for Multi-Domain Data Analytics
draft-xiang-alto-exascale-network-optimization-03.txt

Abstract

Data-intensive analytics is entering the era of multi-domain, geographically-distributed, collaborative computing, where different organizations contribute various resources to collaboratively collect, share and analyze extremely large amounts of data. Examples of this paradigm include the Compact Muon Solenoid (CMS) and A Toroidal LHC ApparatuS (ATLAS) experiments of the Large Hadron Collider (LHC) program. Massive datasets continue to be acquired, simulated, processed and analyzed by globally distributed science networks in these collaborations. Applications that manage and analyze such massive data volumes can benefit substantially from the information about networking, computing and storage resources from each member's site, and more directly from network-resident services that optimize and load balance resource usage among multiple data transfers and analytics requests, and achieve a better utilization of multiple resources in clusters.

The Application-Layer Traffic Optimization (ALTO) protocol can provide via extensions the network information about different clusters/sites, to both users and proactive network management services where applicable, with the goal of improving both application performance and network resource utilization. In this document, we propose that it is feasible to use existing ALTO services to provides not only network information, but also

information about computation and storage resources in data analytics networks. We introduce a uniform resource orchestration framework (Unicorn), which achieves an efficient multi-resource allocation to support low-latency dataset transfer and data intensive analytics for collaborative computing. It collects cluster information from multiple ALTO services utilizing topology extensions and leverages emerging SDN control capabilities to orchestrate the resource allocation for dataset transfers and analytics tasks, leading to improved transfer and analytics latency as well as more efficient utilization of multi-resources in sites.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Table of Contents

1. Introduction

As the data volume increases exponentially over time, data intensive analytics is transiting from single-domain computing to multi-organizational, geographically-distributed, collaborative computing,

where different organizations contribute various resources, e.g., computation, storage and networking resources, to collaboratively collect, share and analyze extremely large amounts of data. One leading example is the Large Hadron Collider (LHC) high energy physics (HEP) program, which aims to find new particles and interactions in a previously inaccessible range of energies. The scientific collaborations that have built and operate large HEP experimental facilities at the LHC, such as the Compact Muon Solenoid (CMS) and A Toroidal LHC ApparatuS (ATLAS), currently have more than 300 petabytes of data under management at hundreds of sites around the world, and this volume is expected to grow to one exabyte by approximately 2018.

With such an increasing data volume, how to manage the storage and analytics of these data in a globally distributed infrastructure has become an increasingly challenging issue. Applications such as the Production ANd Distributed Analysis system (PanDA) in ATLAS and the Physics Experiment Data Export system (PhEDEX) in CMS have been developed to manage the data transfers among different cluster sites. Given a data transfer request, these applications make data transfer decisions based on the availability of dataset replicas at different sites and initiate retransmission from a different replica if the original transmission fails or is excessively delayed. And HTCondor [HTCondor] is deployed to achieve coarse-grained data analytics parallelization across these sites. When a data analytics task is submitted, HTCondor adopts a match-making process to assign the task to a certain set of servers in one site, based on the coarse-grained description of resource availability, such as the number of cores, the size of memory, the size of hard disk, etc. However, neither dataset transfers nor data analytics task parallelization takes fine-grained information of cluster resources, such as data locality, memory speed, network delay, network bandwidth, etc., into account, leading to high data transfer and analytics latency and underutilization of cluster resources.

The Application-Layer Traffic Optimization (ALTO) services defined in [RFC7285] provide network information with the goal of improving the network resource utilization while maintaining or improving application performance. Though ALTO is not designed to provide information about other resources, such as computing and storage resources in cluster networks, in this document we propose that exascale science networks can leverage existing ALTO services defined in [RFC7285] and ALTO topology extension services defined in network graph [DRAFT-NETGRAPH], path vector [DRAFT-PV], routing state abstraction[DRAFT-RSA], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] and etc. to encode information about multiple types of resources in science networks, such as memory I/O speed, CPU utilization, network bandwidth, and provide such information to

orchestration applications to improve the performance of dataset transfer and data analytics tasks, including throughput, latency, etc.

This document introduces a unified resource orchestration framework (Unicorn), which provides efficient multi-resource allocation to support low-latency, multi-domain, geo-distributed data analytics. Unicorn provides a set of simple APIs for authorized users to submit, update and delete dataset transfer requests and data intensive analytics requests. One important proposal we make in this document is that it is feasible to use ALTO services to provide not only network information, but also information on other resources in multi-domain, geo-distributed analytics networks including computing and storage.

A prototype of Unicorn with the dataset transfer scheduling component has been implemented on a single-domain Caltech SDN development testbed, where the ALTO OpenDaylight controller is used to collect topology information. We are currently designing the resource orchestration components to achieve low-latency data-intensive analytics.

This document is organized as follows: Section 3 summarizes the change of this document since version -01. Section 4 elaborates on the motivation and challenges for coordinating storage, computing and network resources in a globally distributed science network infrastructure. Section 5 discusses the basic idea of encoding multi-resource information into ALTO path vector and abstraction services and gives an example. Section 6 lists several key issues to address in order to realize the proposal of providing multi-resource information by ALTO topology services. Section 7 gives the details of Unicorn architecture for multi-domain, geo-distributed data analytics. Section 8 discusses current development progress of Unicorn and next steps.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Changes Since Version -02

- Add an example in Section 7 to show the importance of network information in resource allocation for data analytics.

- Update the architecture of Unicorn in Section 7, i.e., add the entity locator and rename ANE aggregator to resource view extractor.

- Add detailed description of how the entity locator and the resource view extractor work.

- Minor changes in abstract and discussion sections.

4. Problem Settings

4.1. Motivation

Multi-domain, geo-distributed data analytics usually involves the participation of countries and sites all over the world. Science programs such as the CMS experiment and the ATLAS experiment at CERN are typical examples. The site located at the LHC laboratory is called a Tier-0 site, which processes the data selected and stored locally by the online systems that select and record the data in real-time as it comes off the particle detector, archives it and transfers it to over 10 Tier-1 sites around the globe. Raw datasets and processed datasets from Tier-1 sites are then transferred to over 160 Tier-2 sites around the world based on users' requests. Different sites have different resources and belong to different administration domains. With the exponentially increasing data volume in the CMS experiment, the management of large data transfers and data intensive analytics in such a global multi-domain science network has become an increasingly challenging issue. Allocating resources in different clusters to fulfill different users' dataset transfer requests and data analytics requests require careful orchestrating as different requests are competing for limited storage, computation and network resources.

4.2. Challenges

Orchestrating exascale dataset transfers and analytics in a globally distributed science network is non-trivial as it needs to cope with two challenges.

- Different sites in this network belong to different administration domains. Sharing raw site/cluster information would violate sites' privacy constraints. Orchestrating data transfers and analytics requests based on highly abstracted, non-real-time network information may lead to suboptimal scheduling decisions. Hence the orchestrating framework must be able to collect sufficient resource information about different clusters/sites in real-time as well as over the longer term, to allow reasonably

optimized network resource utilization without violating sites' privacy requirements.

- Different science programs tend to adopt different software infrastructures for managing dataset transfers and analytics, and may place different requirements. Hence the orchestrating framework must be modular so that it can support different dataset management systems and different orchestrating algorithms.

The orchestrating framework must support the interaction between the multi-resource orchestration module, the dataset transfer module, and the data analytics execution module. The key information to be exchanged between modules includes dataset information, the resource state of different clusters and sites, the transfer and analytic requests in progress, as well as trends and network-segment and site performance from the network point of view. Such interaction ensures that (1) the various programs can adopt their own data transfer and analytics systems to be multi-resource-aware, and more efficient in achieving their goals; and (2) the various orchestrating algorithms can achieve a reasonably optimized utilization on not only the network resource but also the computing and storage resources.

5. Basic Idea

5.1. Use ALTO services to provide multi-resource information

The ALTO protocol is designed to provide network information to applications so that applications can achieve better performance and the network more efficient use of resources. Different ALTO topology services including path vector, routing state abstraction, multi-cost, cost calendar, etc., have been proposed to provide fine-grained network information to applications. In this document, we propose that not only can ALTO provide network information of different cluster sites, it can also provide information of multiple resources, including computing and storage resources. To this end, the basic "one-big-switch" abstraction provided by the base ALTO protocol is not sufficient. Several examples have already been given in [DRAFT-PV] and [DRAFT-RSA] to demonstrate that. There has been a similar proposal before about using ALTO to provide resource information for data centers [DRAFT-DC]. However, that proposal requires a new information model for clusters or data centers, which may affect the compatibility of ALTO. The solution of this proposal is simpler. Its basic idea is that each computer node and storage node can be seen as an "abstract network element" defined in ALTO-path-vector [DRAFT-PV]. In this way, Unicorn can fully reuse all existing ALTO services by introducing only one cost-mode (pv) and two cost-metrics (ne and ane), instead of introducing a new information model.

5.2. Example 1: network information impacts data analytics scheduling

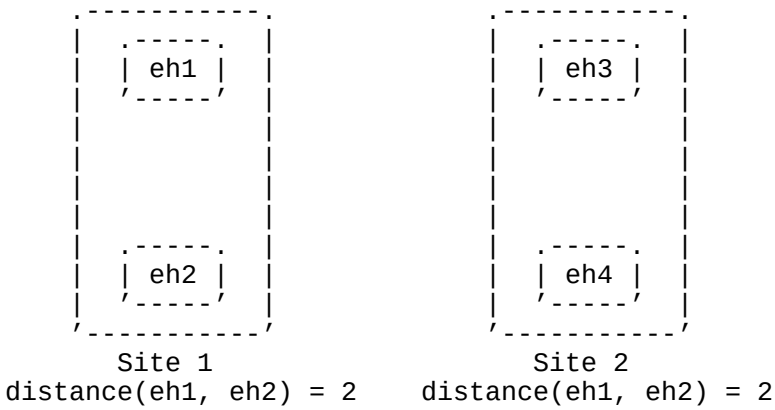

Figure 1: An Example for Data Locality Information.

We first use the example in Figure 1 to show that only network information itself has a huge impact on resource allocation for data analytics. In this scenario, a MapReduce task needs to be executed. The input data has two copies at end hosts eh1 and eh3, respectively. And the end hosts eh2 and eh4 will be the computation nodes with the same computation power, correspondingly. Using the common data analytics resource management framework such as Hadoop it can be revealed that both allocation options, i.e., eh1->eh2 and eh3->eh4, have a storage-computation distance of 2, i.e., they have the same data locality from Hadoop's view. As a result, it appears that both options would provide the same performance for this task.

However, assume that the bandwidth between eh1 and eh2 is 100Mb/s while that between eh3 and eh4 is 1Gb/s. These significant different data accessing speeds decide that these options have very different performances for the same task and the only optimal allocation option is to allocate this task to eh3->eh4. This example demonstrates the imperativeness of network information in making efficient resource allocation decisions. Such information is not provided in Hadoop or other similar or related projects such as Mesos. On the contrary, if ALTO servers are deployed at these sites, applications can retrieve such information via ALTO queries.

5.3. Example 2: encode multi-resources in abstract network elements

We use the same dumbbell topology in [DRAFT-RSA] as an example to show the feasibility of using ALTO topology service to provide multi-resource information. In this topology, we assume the bandwidth of each network cable is 1Gbps, including the cables connecting end hosts to switches. Consider a dataset transfer request which needs to schedule the traffic among a set of end host source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. Assume that the transfer application receives information from the ALTO Cost Map service that both eh1 -> eh2 and eh3 -> eh4 have bandwidth 1Gbps. In [DRAFT-RSA], it is shown that whether each of the two traffic flows can receive 1Gbps bandwidth depends on whether the routes of two flows share a bottleneck link. Path vector and routing state abstraction services provide additional information about network state encoded in abstract network elements. If the returned state is ane1 + ane2 <= 1Gbps, it means two flows cannot each get 1Gbps bandwidth at the same time. If the returned state is ane1 <= 1Gbps and ane2 <= 1Gbps, it means two flows each can get 1Gbps bandwidth.

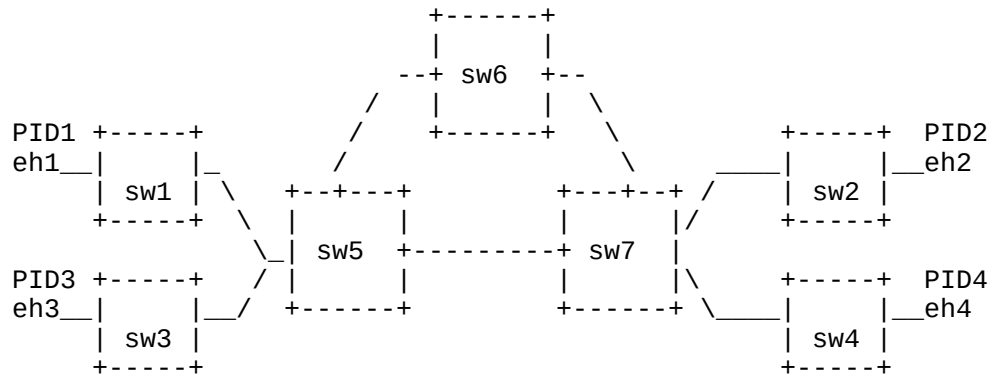

Figure 2: A Dumbbell Network Topology

Other than network resource, assume in this topology eh1 and eh3 are equipped with commodity hard disk drives (HDD) while eh2 and eh4 are equipped with SSDs. Because the bandwidth of an HDD is typically 0.8Gbps and that of SSD is typically 3Gbps. Even if the returned routing state is ane1 <= 1Gbps and ane2 <=1Gbps, the actual bottleneck of each traffic flow is the storage I/O bandwidth at source host. As a result, the total bandwidth of both traffic flows can only reach 1.6Gbps.

It has been verified in the CMS experiment, and also several studies on commercial data centers that network resource are not always the bottleneck of large dataset transfers and data analytics. Many have reported that storage resources and computing resources become the bottleneck in a fairly large percentage of dataset transfers and data analytics tasks in science networks and commercial data centers.

In this example, if we look at the end hosts as abstract network elements, the storage I/O bandwidth of each host can also be encoded as an abstract element into the path-vector. And under the storage and route settings above, the returned cluster state would be ane1 <=0.8Gbps and ane2 <=0.8Gbps, which provides a more accurate capacity region for the requested traffic flows.

6. Key Issues

The last section described the basic idea of using ALTO topology services to provide multi-resource information and gives an example to demonstrate its feasibility. Next we list and discuss several key issues to address in this proposal.

- Can ALTO topology services provide data locality information? Existing ALTO topology services do not provide such information. Many studies have pointed out that such information plays a vital role in reducing the latency of data-intensive analytics. If ALTO topology services can encode such information together with information of other resources together, data-intensive applications can benefit a great deal in terms of information aggregation and communication overhead.

- How to quickly map applications' resource allocation decision on abstract multi-resource view back to the physical multi-resource view of clusters/sites? Fine-grained resource information can be encoded into abstract network elements to reduce overhead and provide certain privacy protection of clusters. Such information can be highly compressed (see the dumbbell example used in this document as well as in [DRAFT-PV] and [DRAFT-RSA]). In preliminary evaluations on RSA, the network element compression ratio can be as high as 80 percent. This ratio is expected to be even higher in large-scale data center or cluster setting, e.g. a fat-tree topology with k=48. Therefore a fast mapping from the resource orchestration decisions based on the abstract view back to the physical view is needed to satisfy the stringent latency requirement of large dataset transfers and data-intensive analytics.

- How much privacy, including key resource configurations, raw topology, intra-cluster scheduling policy, etc., will be exposed?

Compared with the "one-big-switch" abstraction, other ALTO topology services such as path vector [DRAFT-PV] and routing state abstraction [DRAFT-RSA] provide fine-grained resource information to applications. Even if such information can be encoded into abstract network elements, it still risks exposing private information of different clusters/sites. Current internet drafts of these services did not provide any formal privacy analysis or performance measurement. This would be one of the key issues this document plans to investigate in the future.

- How does current ALTO services such as path-vector and RSA scale when they are used to provide abstract information concerning multiple resources in clusters? Another issue along this line is how to balance the liveness of fine-grained resource information and the corresponding information delivery overhead? Although encoding information of network elements into abstract network elements can achieve a very competitive information compression ratio, large dataset transfers and analytics applications always involve many network elements in multiple clusters/sites and the absolute number of involved network elements keep increasing as the scale of clusters increase. In addition, when resource information in a cluster changes, the ALTO services need to inform all related applications. In either cases, delivering fine-grained resource information would cause high communication overhead. There still lacks of an analytics or experimental understanding on the scalability of path-vector and RSA services.

7. Unified Resource Orchestration Framework

7.1. Architecture

This section describes the design details of key components of the Unicorn framework: the workflow converter, the resource demand estimator, the ALTO clients, the ALTO servers, the resource view extractor, the multi-resource orchestrator and the execution agents. Figure 3 shows the architecture of Unicorn. The overall process is as follows.

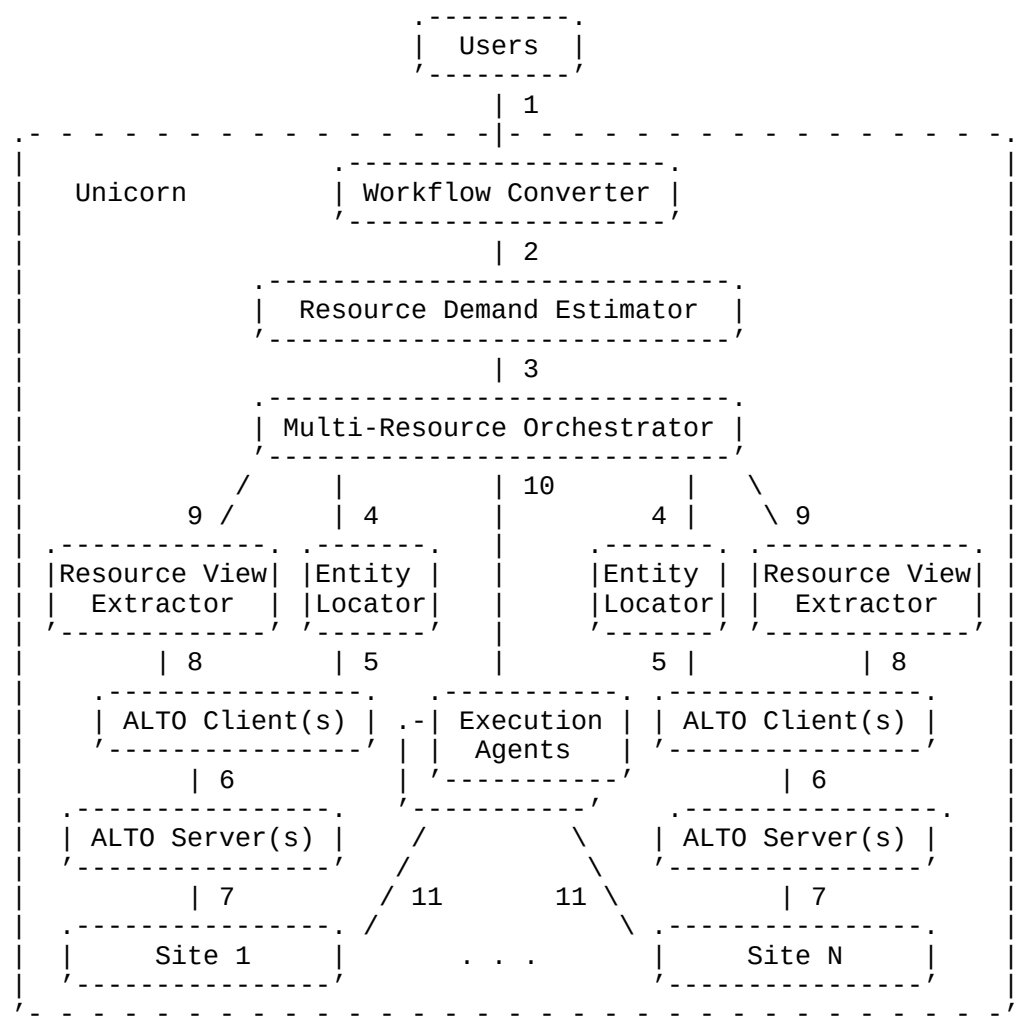

Figure 3: Architecture of Unicorn.

- STEP 1 Authorized users submit high-level data analytics workflows to Unicorn through a set of simple APIs.
- STEP 2 The workflow converter transforms the high-level data analytics workflows into low-level task workflows, i.e., a set of analytics tasks with precedence encoded in a directed acyclic graph (DAG).

- STEP 3 The resource demand estimator automatically finds the optimal configuration (resource demand) of each task, i.e., the number of CPUs, the size of memory and disk, I/O bandwidth, etc.

- STEP 4 The multi-resource orchestrator receives the resource demand of a set of tasks and sends them to the entity locator at each site.

- STEP 5 The entity locator at each site retrieves the entity addresses of the end hosts that would be allocated for the tasks to be scheduled, and passes these addresses to the ALTO clients, and asks the ALTO clients to collect information about these end hosts, i.e., properties of corresponding computing, storage and networking resources.

- STEP 6 The ALTO clients issue ALTO queries defined in the base ALTO protocol [RFC7285], e.g., EPS, ECS, Network Map, etc. and ALTO extension services, e.g., routing state abstraction (RSA) [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC], cost-calendar [DRAFT-CC] and property map [DRAFT-PM], to collect resource information.

- STEP 7 The ALTO servers at each site accept the queries from the ALTO client, collect resource information from the residing site and send back to the ALTO clients.

- STEP 8 The ALTO clients send the response from ALTO servers to the resource view extractor.

- STEP 9 The extractor uses a lightweight, optimal algorithm to compress the raw resource information provided by ALTO servers into a minimal, equivalent view of resources and sends back to the multi-resource orchestrator.

- STEP 10 The orchestrator makes resource allocation decisions, e.g., dataset transfer scheduling and analytics task placement, based on the resource demand of analytics tasks and the resource supply sent back from the resource view extractor. Decisions are then sent to the execution agents deployed in corresponding sites.

- STEP 11 The execution agents receive and execute instructions from the multi-resource orchestrator. They also monitor the status of different tasks and send the updated status to the multi-resource orchestrator.

Unicorn provides a unified, automatic solution for multi-domain, geo-distributed data analytics. In particular, its benefits include:

- On the resource demand side, it provides a set of simple APIs for authorized users to submit and manage data analytics requests and enables real-time requests' status monitoring. And it automatically converts high-level analytics workflow into low-level task workflows and finds the optimal configuration for each task.

- On the resource supply side, it collects the resource information of different sites through a common, REST based interface specified by the ALTO protocol, encodes such information into different entity abstractions and computes a minimal, yet accurate view on resource supply dynamic.

- It provides a scalable multi-resource orchestrator that makes efficient resource allocation decisions to achieve high resource utilization and low-latency data analytics.

- The architecture of Unicorn is modular to support different resource orchestration algorithms and the deployment of different ALTO servers.

7.2. Workflow converter

The converter is the front end of Unicorn. It is responsible for collecting high-level data analytics workflows from users and transforming them into low-level task workflows, e.g., HTCondor ClassAds. It provides a set of simple APIs for users to submit and manage requests, and to track the status of requests in real-time.

7.2.1. User API

- submitReq(request, [options])

  This API allows users to submit a request and specify corresponding options. The request can be a data transfer request or a data analytics request. Request options include priority, delay, etc. It returns a request identifier reqID that allows users to update, delete this request or track its status. The additional options may or may not be approved, and the relative priorities may be modified by the resource orchestrator depending on the role of users (regular users or administrators at different levels), the resource availability and the status of other ongoing requests.

- updateReq(requestID, [options])

  This API allows users to update the options of requests. It will return a SUCCESS if the new options are received by the request

parser. But these new options may or may not be approved, and may be modified by the resource orchestrator depending on the role of users (regular users or administrators), the resource availability and the status of other ongoing requests.

- o deleteReq(requestID)

  This API allows users to delete a request by passing the corresponding requestID. A completed request cannot be deleted. An ongoing request will be stopped and the output data will be deleted.

- o getReqStatus(requestID)

  This API allows users to query the status of a request by specifying the corresponding requestID. The returned status information includes whether the request has started, the assigned priority, the percentage of finished sub-requests, transmission statistics, the expected remaining time to finish, etc.

7.3. Resource Demand Estimator

The estimator leverages the fact that low-level tasks are typically repetitive or have very high similarities. It uses reinforcement learning to predict the optimal configuration for each task and passes the resource demand to the multi-resource orchestrator for further processing.

7.4. Entity Locator

The task configurations computed by the demand estimator has no knowledge on the underlying structure of topology of each site, i.e., the addresses of end hosts and network devices. Hence they cannot be directly used by the ALTO clients for querying resource information. The entity locator retrieves the entity addresses of the end hosts that would be allocated for the tasks to be scheduled, and passes these addresses to the ALTO clients, and asks the ALTO clients to collect information about these end hosts, i.e., properties of corresponding computing, storage and networking resources.

7.5. ALTO Client

The ALTO client is in the back end of Unicorn and is responsible for retrieving resource information through querying ALTO servers deployed at different sites. The resource information needed in Unicorn includes the topology, link bandwidth, computing node memory I/O speed, computing node CPU utilization, etc. The base ALTO protocol [RFC7285] provides an extreme single-node abstraction for

this information, which only allows the multi-resource orchestrator to make coarse-grained resource allocation decisions. To enable fine-grained multi-resource orchestration for dataset transfer and data analytics in cluster networks, ALTO topology extension services such as routing state abstraction (RSA) [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] are needed to provide fine-grained information about different types of resources in clusters.

7.5.1. Query Mode

The ALTO client should operate in different query modes depending on the implementation of ALTO servers. If an ALTO server does not support incremental updates using server-sent events (SSE) [DRAFT-SSE], the ALTO client sends queries to this server periodically to get the latest resource information. If the cluster state changes after one query, the ALTO client will not be aware of the change until next query. If an ALTO server supports SSE, the ALTO client only sends one query to the ALTO server to get the initial cluster information. When the resource state changes, the ALTO client will be notified by the ALTO server through SSE.

7.6. ALTO Server

ALTO servers are deployed at different sites around the world, and at strategic locations in the network itself, to provide information about different types of resources in the cluster networks in response to queries from the ALTO client. Such information include topology, link bandwidth, memory I/O speed and CPU utilization at computing nodes, storage constraints in storage nodes and etc. Each ALTO server must provide basic information services as specified in [RFC7285] such as network map, cost map, endpoint cost service (ECS), etc. To support the fine-grained multi-resource allocation in Unicorn, each ALTO server should also provide more fine-grained information about different resources in clusters through ALTO extension services such as the routing state abstraction [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] services.

7.7. Resource View Extractor

In each site, the resource information collected by the ALTO clients is not directly sent back to the orchestrator. Instead, we design a resource view extractor to compress the resource information provided by the ALTO servers into a minimal, equivalent view of all the resources, i.e., computing, storage and networking resources, that would be allocated to a set of tasks. The extractor works in the following steps.

- Resource-Task Matrix

  Depending on specific services provided by the ALTO servers, the responses collected by ALTO clients may include information of different entities, e.g., endpoints, PIDs, ane, etc. Each entity possesses a set of resources available for tasks to be scheduled. For each entity property p in the responses, such as bandwidth, delay, etc., the extractor composes an entity-task matrix M(p). Each row of this M(p) represents an entity in the responses provides information about property p and each column represents a task to be scheduled. The element m(i, j) of M(p) is a variable representing the amount of property p of entity i that task j can get.

- Resource-Task Constraints

  For each property p, the extractor composes a series of resource-task constraints. The first set of constraints is sum m(i, j) = r(p, j) for each task j. These constraints calculate r(p, j), the total amount of property p provided to j by all the entities. The exact rule of summation depends on the property p. For instance, if p is latency, the summation is through common addition operations, but if p is bandwidth, the summation is a minimum function.

  The second set of constraints is sum m(i, j) <= r(p, i) for each entity i. These constraints represent the usage of resource i on property p for all the tasks cannot exceed the capacity of resource i on this property. The exact rule of summation also depends on the property p. For instance, if p is bandwidth, the summation is through common addition operations, but if p is latency, the constraints become m(i,j) = r(p, i) for each each entity i and each task j. This means that entity i provides the same delay property for each task.

- Resource View Compression

  The whole set of resource-task constraints are linear, and express the view of resources that are available for the tasks to be scheduled. The extractor uses a lightweight, optimal algorithm to compress them into a minimal, equivalent view of resources, i.e., a minimal set of linear constraints that represent the same feasible region as the original constraints. The basic idea of this algorithm is described in [DRAFT-RSA].

7.8. Execution Agents

Execution agents are deployed at each site and are responsible for the following functions:

- Receive and process instructions from the multi-resource orchestrator, e.g. dataset transfer scheduling, data analytics task placement and execution, task update and abortion, etc.

- Monitor the status of data analytics tasks and send the updated status to the multi-resource orchestrator.

Depending on the supporting data analytics frameworks, different request execution agents may be deployed in each site. For instance, in the CMS experiment at CERN, both MPI and Hadoop execution agents are deployed.

7.9. Multi-Resource Orchestrator

The multi-resource orchestrator receives the resource demand information, i.e., a set of low-level task workflows and their configurations, from the resource demand estimator. It then asks the entity locator in each site to get the addresses of end hosts that would be allocated for the tasks to be scheduled and ALTO clients to query properties related to these end hosts. When the resource view extractor sends the response back, the orchestrator makes resource allocation decisions, e.g., dataset transfer scheduling and analytics task execution, based on both resource demand dynamic and resource supply dynamic. The dataset transfer scheduling decisions include dataset replica selection, path selection, and bandwidth allocation, etc. The analytics task execution decisions include which cluster should allocate how much resources to execute which tasks. These decisions are sent to the execution agents at different sites for execution.

7.9.1. Orchestration Algorithms

The modular design of Unicorn allows the adoption of different orchestration algorithms and methodologies, depending on the specific performance requirements. In Section 7.8.3, a max-min fairness resource allocation algorithm for dataset transfer is described as an example.

7.9.2. Online, Dynamic Orchestration

The multi-resource orchestrator should adjust the resource allocation decisions based on the progress of ongoing requests, the utilization and dynamics of cluster resources. In normal cases, the multi-

resource orchestrator periodically collects such information and executes the orchestration algorithm. When it is notified of events such as request status update, cluster state update and etc., the orchestrator will also execute the orchestration algorithm to adjust resource allocations.

7.9.3. Example: A Max-Min Fairness Resource Allocation Algorithm

In this section, we describe a max-min fair resource allocation (MFRA) scheduling algorithm which aims to minimize the maximal time to complete a dataset transfer subject to a set of constraints. To make resource allocation decisions, MFRA requires sufficient network information including topology, link bandwidth and recent historical information in some cases. In a small-scale single-domain network, an SDN controller can provide the raw complete topology information for the MFRA algorithm. However, in a large-scale multi-domain science network such as CMS, providing the raw network topology is infeasible because (1) it would incur significant communication overhead; and (2) it would violate the privacy constraints of some sites. Several ALTO extension topology services including Abstract Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA] can provide the fine-grained yet aggregated/abstract topology information for MFRA to efficiently utilize bandwidth resources in the network.

Ongoing pre-production deployment efforts of Unicorn in the CMS network involve the implementation of the RSA service. Other than topology information, the additional input of the MFRA algorithm is the priority of each class of flows, expressed in terms of upper and lower limits on the allocated bandwidth between the source and the destination for each data transfer requests.

The basic idea of the MFRA algorithm is to iteratively maximize the volume of data that can be transferred subject to the constraints. It works in quantized time intervals such that it schedules network paths and data volumes to be transferred in each time slot. When the DTR scheduler is notified of events such as the cancellation of a DTR, the completion of a DTR or network state changes, the MFRA algorithm will also be invoked to make updated network path and bandwidth allocation decisions.

In each execution cycle, MFRA first marks all transfers as unsaturated. Then it solves a linear programming model to find the common minimum transfer satisfaction rate (i.e., the ratio of transferred data volume in a time interval over the whole data volume of this request) that is satisfied by all transfer requests. With this common rate found, MFRA then randomly selects an unsaturated request in each iteration, increases its transfer rate as much as

possible by finding residual paths available in the network, or by increasing the allocated bandwidth along an existing path, until it reaches its upper limit or can otherwise not be increased further, so it is saturated. At each iteration, newly saturated requests are removed from the subsequent process by fixing their corresponding rate value, and completed transfers are removed from further consideration. After all the data transfer rates are saturated in the given time slot, then a feasible set of data transfer volumes scheduled to be transferred in the slot across each link in the network can be derived.

The MFRA algorithm yields a full utilization of limited network resources such as bandwidth so that all DTR can be completed in a timely manner. It allocates network resources fairly so that no DTR suffers starvation. It also achieves load balance among the sites and the network paths crossing a complex network topology so that no site and no network link is oversubscribed. Moreover, MFRA can handle the case where particular routing constraints are specified, e.g., where all routes are fixed ahead of time, or where each transfer request only uses one single path in each time slot, by introducing an additional set of linear constraints.

8. Discussion

8.1. Deployment

The Unicorn framework is the first step towards a new class of intelligent, SDN-driven global systems for multi-domain, geo-distributed data analytics involving a worldwide ensemble of sites and networks, such as CMS and ATLAS. Unicorn relies heavily on the ALTO services for collecting and expressing abstract, real-time resource information from different sites, and the SDN centralized control capability to orchestrate data analytics workflows. It aims to provide a new operational paradigm in which science programs can use complex network and computing infrastructures with high throughput, while allowing for coexistence with other network traffic.

A prototype case study implementation of Unicorn has been demonstrated on the Caltech/StarLight/Michigan/Fermilab SDN development testbed. Because this testbed is a single-domain network, the current Unicorn prototype leverages the ALTO OpenDaylight controller, to collect topology information. The CMS experiment is currently exploring pre-production deployments of Unicorn, looking towards future widespread production use. To achieve this goal, it is imperative to collect sufficient resource information from the various sites in the multi-domain CMS network, without causing any privacy leak. To this end, the ALTO RSA service

[DRAFT-RSA] is under development. Furthermore, as will be discussed next, other ALTO topology extension services can also substantially improve the performance of Unicorn.

8.2. Benefiting From ALTO Extension Services

The current ALTO base protocol [RFC7285] exposes network topology and endpoint properties using the extreme "my-Internet-view" representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of end hosts called endpoints. Such an extreme abstraction leads to significant information loss on network topology [DRAFT-PV], which is the key information for Unicorn to make dynamic scheduling and resource allocation decisions. Though Unicorn can still allocate resource for data transfer and analytics requests on this abstract view, the resource allocation decisions are suboptimal. Alternatively, feeding the raw, complete network topology of each site to Unicorn is not desirable, either. First, this would violate privacy constraints of different sites. Secondly, a raw network topology would significantly increase the problem space and the solution space of the orchestrating algorithm, leading to a long computation time. Hence, Unicorn desires an ALTO topology service that is able to provide only enough fine-grained topology information.

Several ALTO topology extension services including Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA], [DRAFT-PM] are potential candidates for providing fine-grained abstract network formation to Unicorn. In addition, we propose that these services can also be used to provide information about computing and storage resources of different cluster/sites by viewing each computing node and storage node as a network element or abstract network element. For instance, the path vector service supports the capacity region query, which accepts multiple concurrent data flows as the input and returns the information of bottleneck resources, which could be a set of links, computing devices or storage devices, for the given set of concurrent flows. This information can be interpreted as a set of linear constraints for the multi-resource orchestrator, which can help data transfer and analytics requests better utilize multiple types of resources in different clusters.

8.3. Limitations of the MFRA Algorithm

The first limitation of the MFRA algorithm is computation overhead. The execution of MFRA involves solving linear programming problems repeatedly at every time slot. The overhead of computation time is acceptable for small sets of dataset transfer requests, but may increase significantly when handling large sets of requests, e.g.,

hundreds of transfer requests. Current efforts towards addressing this issue include exploring the feasibility of incremental computation of scheduling policies, and reducing the problem scale by finding the minimal equivalent set of constraints of the linear programming model. The latter approach can benefit substantially from the ALTO RSA service [DRAFT-RSA].

The second limitation is that the current version of MFRA does not involve dataset replica selection. Simply denoting the replica selection as a set of binary constraint will significantly increases the computation complexity of the scheduling process. Current efforts focus on finding efficient algorithms to make dataset replica selection.

9. Security Considerations

This document does not introduce any privacy or security issue not already present in the ALTO protocol.

10. IANA Considerations

This document does not define any new media type or introduce any new IANA consideration.

11. Acknowledgments

The authors thank discussions with Shenshen Chen, Linghe Kong, Xiao Lin and Xin Wang.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

12.2. Informative References

[DRAFT-CC] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "ALTO Cost Calendar", 2017, <https://datatracker.ietf.org/doc/draft-ietf-alto-cost-calendar>.

[DRAFT-DC]
Lee, Y., Bernstein, G., Dhody, D., and T. Choi, "ALTO Extensions for Collecting Data Center Resource Information", 2014, <https://datatracker.ietf.org/doc/draft-lee-alto-ext-dc-resource/>.

[DRAFT-MC]
Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", 2017, <https://datatracker.ietf.org/doc/draft-ietf-alto-multi-cost/>.

[DRAFT-NETGRAPH]
Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extensions: Node-Link Graphs", 2015, <https://tools.ietf.org/html/draft-yang-alto-topology-06>.

[DRAFT-PM]
Roome, W. and Y. Yang, "Extensible Property Maps for the ALTO Protocol", 2015, <https://datatracker.ietf.org/doc/draft-roome-alto-unified-props-new/>.

[DRAFT-PV]
Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Abstract Path Vector as a Cost Mode", 2015, <https://tools.ietf.org/html/draft-yang-alto-path-vector-01>.

[DRAFT-RSA]
Gao, K., Wang, X., Yang, Y., and G. Chen, "ALTO Extension: A Routing State Abstraction Service Using Declarative Equivalence", 2015, <https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>.

[DRAFT-SSE]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", 2015, <https://datatracker.ietf.org/doc/draft-ietf-alto-incr-update-sse/>.

[HTCondor]
Thain, D., Tannenbaum, T., and M. Livny, "Distributed computing in practice: the Condor experience", 2005, <http://dl.acm.org/citation.cfm?id=1064336>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Qiao Xiang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

Harvey Newman
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: newman@hep.caltech.edu

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Haizhou Du
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: duhaizhou@gmail.com

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing
China

Email: gaok12@mails.tsinghua.edu.cn

Azher Mughal
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: azher@hep.caltech.edu

Justas Balcas
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: justas.balcas@cern.ch

Jingxuan Jensen Zhang
Tongji University
4800 Cao'an Hwy
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

Y. Richard Yang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: yry@cs.yale.edu