ALTO WG
Internet-Draft
Intended status: Informational
Expires: 13 January 2022

Q. Xiang
Xiamen University
J. Zhang
Tongji/Yale University
F. Le
IBM
Y. Yang
Yale University
H. Newman
California Institute of Technology

12 July 2021

Resource Orchestration for Multi-Domain, Exascale, Geo-Distributed Data Analytics

draft-xiang-alto-multidomain-analytics-06.txt

Abstract

As the data volume increases exponentially over time, data analytics is transiting from a single-domain network to a multi-domain, geo-distributed network, where different member networks contribute various resources, e.g., computation, storage and networking resources, to collaboratively collect, share and analyze extremely large amounts of data. Such a network calls for a resource orchestration framework that emphasizes the performance predictability of data analytics jobs, the high utilization of resources, and the autonomy and privacy of member networks.

This document presents the design of Unicorn, a unified resource orchestration framework for multi-domain, geo-distributed data analytics, which uses the Application-Layer Traffic Optimization (ALTO) protocol as the key component for (1) allows member networks to provide accurate information on different types of resources; (2) keeps the private information of member networks; and (3) allows data analytics jobs to accurately describe their requirements of different types of resources. As a part of Unicorn, an ALTO extension for privacy-preserving interdomain information aggregation is also presented.

Status of This Memo

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

This Internet-Draft will expire on 13 January 2022.

Table of Contents

1. Introduction

This document describes the design of Unicorn, a unified resource orchestration framework for large-scale data analytics in multi-domain, geo-distributed networks. An important use case for such settings is the Large Hadron Collider (LHC) network, which consists of over 180 member networks all over the world, to support scientists to access multiple resources, e.g., computing, storage and networking resources, distributed in the member networks to conduct large-scale data analytics. With more and more data being generated and stored in different geo-distributed member networks, network architects and administrators are exploring different designs for efficient resource orchestration in multi-domain, geo-distributed networks.

The design presented in this document is based on the development and deployment experience of Unicorn in the CMS network, one of the largest scientific experiments in the LHC network. The primary requirements of resource orchestration in such a multi-domain, geo-distributed environment are the performance predictability of various data analytics jobs, the high utilization of different types of resources, and the autonomy and privacy of resource owners, i.e., member networks.

Pre-production development and extensive testing have shown that the Application-Layer Traffic Optimization Protocol [RFC7285] is well suited as a fundamental component in Unicorn for providing a generic

representation that (1) allows different types of data analytics jobs to accurately describe their resource requirements and (2) allows member networks to provide accurate information on different types of resources they own and at the same time maintain their privacies. This is in contrast with the state-of-the-art resource orchestration frameworks, such as HTCondor and Mesos, which either do not provide accurate networking information or expose all the private details of member networks. This document elaborates on the design requirements of resource orchestration in multi-domain, geo-distributed networks that lead to this design choice and presents the details of Unicorn, including an ALTO extension for privacy-preserving, interdomain information aggregation.

This document first gives an overview of the characteristics of multi-domain, geo-distributed data analytics. Then, the design requirements for resource orchestration under such settings are summarized. After reviewing existing designs and their limitations, this document gives the arguments for using ALTO as the generic representation for describing both resource requirements and the resource information and describes the design details of Unicorn. Finally, a privacy-preserving, interdomain extension of ALTO is presented.

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 -05

* Add Section 10.4 to discuss how Ruzio, the latest scientific data management framework for the ATLAS experiment at LHC, can benefit from integrating with ALTO and Unicorn.

4. Characteristics of Multi-Domain, Geo-Distributed Data Analytics

This section describes the characteristics of multi-domain, geo-distributed data analytics.

4.1. Dynamic Data Analytics Workload

In multi-domain, geo-distributed data analytics, extremely large amounts of data are generated and stored across different member networks. Authorized users from different organizations can access data and resources in member networks to conduct various data analytics jobs using various data analytics applications.

An data analytics application usually provides an automated process that decomposes a large data analytics job into a set of smaller tasks, whose dependencies are expressed as a directed acyclic graph (DAG). Tasks without any dependency can be executed in parallel to improve the efficiency of the data analytics job they belong to. This decomposition is highly user- and application-dependent.

Each task may have different requirements on different resources. For instance, task T1 may require dataset A in storage node X as input and 1 CPU as the computing resource, while task T2 may require dataset B in storage node Y as input and 2 CPUs as the computing resource. Furthermore, each task may require resources from different member networks. In the previous example, T1 may require its output to be stored in a storage node in another member network for the purpose of secure storage. The resource requirements of tasks are highly user- and application-dependent.

From the above description, it is observed that the workload of multi-domain, geo-distributed data analytics is highly dynamic, in terms of the number of users, the types of applications, the number of jobs, the decomposition of jobs and the resource requirements of tasks.

Though with such dynamism, it is the general consensus of users to expect performance predictability of their analytics jobs (TODO: add Mogul citation). Hence the resource orchestration for multi-domain, geo-distributed data analytics must be able to achieve efficient resource sharing among different data analytics jobs of different applications from different users. To this end, a generic representation of resource requirements for different tasks from different analytics applications must be chosen. Furthermore, to ensure maximal deployment, the resource orchestration framework must be independent of and compatible with data analytics applications.

4.2. Dynamic Resource Availability

In the multi-domain, geo-distributed data analytics network, different member networks belong to different administrative domains. Each member network has its own resource management policies and can choose to use different management software, such as HTCondor and Mesos.

Each member network provides different types of resources with different amounts. For example, transit networks such as ESNet and Internet2 provide high-bandwidth networking resources. In contrast, campus science networks provide abundant computation and storage resources, but may provide limited networking bandwidths. And some smaller science networks only provide limited computation and storage resources. The availability of the resources in each member network is subject to the autonomous control of the member network.

Furthermore, member networks are interconnected with high bandwidth-delay-product links, where state-of-the-art networking resource allocation mechanisms, such as TCP, become inefficient [XCP].

From the above description, it is observed that the resource availability of the multi-domain, geo-distributed data analytics network is also highly dynamic, subject to the types of member networks, the resources provided by member networks and the resource management policies and management software used by member networks.

Though with such dynamism, it is the general consensus of member networks that the resource orchestration for multi-domain, geo-distributed data analytics must achieve high utilization of different types of resources, following the autonomy and privacy of each member network. To this end, a generic representation of resource availabilities for different types of resources must be chosen. Such a representation must be accurate and at the same time maintain the privacy of member networks. Furthermore, to ensure maximal deployment, the resource orchestration framework must be independent of and compatible with the resource management systems used by member networks.

5. Design Requirements

This section summarizes the design requirements for resource orchestration for multi-domain, geo-distributed data analytics from the previous section.

* REQ1: Provide performance predictability for data analytics jobs.

* REQ2: Achieve the efficient resource sharing among data analytics jobs.

* REQ3: Achieve the high utilization of different types of resources in member networks.

* REQ4: Maintain the autonomy and privacy of member networks.

* REQ5: Provide compatibility with different data analytics applications and resource management systems to maximize the deployment.

6. Review of Resource Orchestration Designs for Data Analytics

This section provides an overview of three general types of resource orchestration designs for data analytics -- the centralized resource-graph-based orchestration, the centralized ClassAds-based orchestration and the distributed opportunistic orchestration. Then, the key reason why these designs are inadequate for multi-domain, geo-distributed data analytics is provided.

6.1. Centralized resource-graph-based orchestration

Systems such as Mesos [Mesos] and Borg [Borg] adopt a graph-based abstraction to represent the resource availability of computing clusters. Each node in the graph is a physical node representing computation or storage resources and each edge between a pair of nodes denotes the networking resource connecting two physical nodes. This design is inadequate for multi-domain, geo-distributed data analytics system because (1) it compromises the privacy of different member networks by revealing all the details of resources; and (2) the overhead to keep the resource availability graph up to date is too expensive due to the heterogeneity and dynamicity of resources from different member networks.

6.2. Centralized ClassAds-based orchestration

HTCondor [HTCondor] proposes a ClassAds programming model, which allows different resource owners to advertise their resource supply and the job owners to advertise the resource demand. However, this programming model does not support the accurate discovery of networking resources, but leave the orchestration of networking resources completely to TCP, which has been known to behave poorly in networks with high bandwidth-delay products [XCP].

6.3. Distributed opportunistic orchestration

Some systems, such as Apollo [Apollo] and Sparrow [Sparrow], use a distributed design. In this design, given a data analytics job, a small number of computing and storage nodes are randomly selected as candidates. Then a scheduling algorithm makes the decision to select the best pair of computing and storage nodes within this small set of candidates. Though it is shown in production that this design achieves a performance very close to the theoretical optimal resource allocation scheme, this design cannot be applied to multi-domain, geo-distributed data analytics because (1) the pool of computing and

storage resources is much larger, and is distributed across the world, and (2) it is hard to distributively orchestrate networking resources in such a high bandwidth-delay product scenario.

6.4. Inadequacy of Existing Designs for Multi-Domain, Geo-Distributed Data Analytics

Applying the designs reviewed in the preceding subsections for multi-domain, geo-distributed data analytics only satisfies the design requirement of compatibility (REQ5), but leaves all the other requirements unfulfilled. The key reason is that they do not have an information model that simultaneously

* allows member networks to provide accurate information on different types of resources, e.g., the computing, storage and networking resources, they own;

* keeps the private information of member networks, such as physical topologies and policies, from the data analytics applications; and

* allows data analytics jobs to accurately describe their requirements of different types of resources.

7. Unicorn Design

This section presents the design of the Unicorn framework. First, the motivations of using ALTO as the information model of resource orchestration for multi-domain, geo-distributed data analytics are reviewed. Then the architecture of Unicorn is provided.

7.1. Choosing ALTO as the Resource Information Model

As reviewed in the preceding section, the commonly used resource-graph-based information model and the ClassAds information model do not support the accurate, yet privacy-preserving resource discovery across different member networks. In contrast, the ALTO protocol uses abstract maps of networks to provide network information with the goal of modifying network resource consumption patterns while maintaining or improving application performance [RFC7285]. This document proposes the use of ALTO for providing information of different types of resources, e.g., computing, storage and networking resources. This design has the following advantages:

* ALTO provides the network information based on abstract maps of a network. Additional services are built on top of the ALTO abstract maps to provide information of other types of resources, e.g., the computing and storage resources. These maps provide accurate information of different types of resources for the

resource orchestration system to effectively utilize them for data analytics applications. For example, the ALTO Endpoint Property Service can provide information of computing nodes and storage nodes.

* The ALTO abstract maps provide a simplified view of resources of member networks, instead of the full details of their resource availability. Thus ALTO allows member networks to keep their private information, such as physical topologies and policies, from the applications. For example, the ALTO Network Map service provides a "one-big-switch" view that defines a grouping of network endpoints. This view hides the details of the underlying physical topology of the network and a network deploying the ALTO server has the autonomy to adopt any endpoint grouping algorithm.

* ALTO uses a client-server model, in which applications can use ALTO clients to accurately describe their requirements of different types of resources and send these requirements to the ALTO servers to retrieve the accurate information of resources that suit their requirements. For example, the ALTO Multi-Cost service [RFC8189] allows an ALTO client to specify a logic set of tests in a query. Such tests are used by ALTO servers to filter out the information of unqualified resources from the response sent back to the ALTO client.

7.2. Architecture of Unicorn

This section describes the design details of Unicorn. Figure 1 presents the architecture of Unicorn for a multi-domain, geo-distributed data analytics system with N member networks. In particular, Unicorn consists of the following key components:

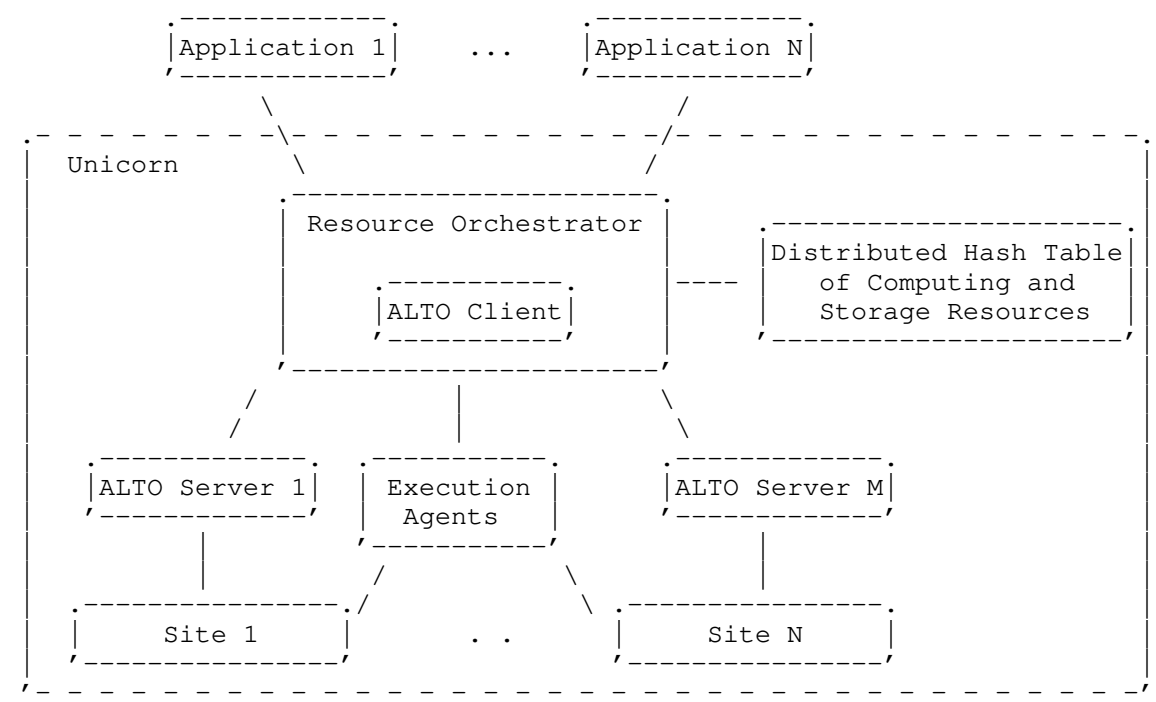

Figure 1: Architecture of Unicorn.

* ALTO Server: for each member network, one or more ALTO servers are deployed to provide accurate, yet privacy-preserving information of different types of resources owned by the corresponding network. Examples of such information include the link bandwidth between endpoints, the memory I/O bandwidth and the CPU utilization at computing endpoints and the storage space at storage endpoints. In addition to the basic ALTO services defined in [RFC7285], The ALTO servers in Unicorn also provide ALTO extension services such as the ALTO Multi-Cost Service [RFC8189], the ALTO Server-Sent Event Service [DRAFT-SSE] and the ALTO Multipart Cost Property Service [DRAFT-PV] to provide fine-grained resource information.

* Distributed Hash Table (DHT) of Computing and Storage Resources: A DHT system is deployed across member networks to lookup the location of computing and storage resources. Compared with the current centralized lookup services in the CMS network, i.e., PhEDEx and HTCondor, a DHT system provides a significant performance improvement for discovering the locations of computing and storage resources in multi-domain, geo-distributed data analytics systems.

* Resource Orchestrator: The orchestrator is a shim layer between the data analytics jobs from different applications and the member networks. It contains an ALTO client that communicates with the ALTO servers at member networks to retrieve resource information. Given a set of data analytics jobs, the orchestrator adopts a three-phase discovery process, which will be elaborated in the next section, to find the accurate information of all the resources that can be used to execute these jobs. Then the orchestrator runs a customized resource allocation algorithm to compute the resource allocation decisions for these jobs, and send the decisions to the execution agents at corresponding member networks.

* Execution Agent: One or more execution agents are deployed at each member network. They take the resource allocation decisions from the resource orchestrator, and communicate with the underlying resource management system deployed at the corresponding member network to reserve the resources for the data analytics jobs and execute them.

7.2.1. Three-Phase Resource Discovery

The preceding subsection describes the architecture and the key components of Unicorn. One missing component is how to accurately discover the information of different types of resources for a set of data analytics jobs with the assistance of ALTO. This section presents the three-phase resource discovery design in Unicorn.

7.2.1.1. Phase 1: Endpoint Property Discovery

Figure 2 shows the procedure of the endpoint property discovery phase. Given a set of data analytics jobs, the resource orchestrator communicates with the DHT lookup system to find the locations, i.e., the endpoint addresses, of all candidate computing and storage resources. With such information, the ALTO client then issues Endpoint Property Service (EPS) queries to the ALTO servers deployed at member networks to discover the information of all candidate endpoints.

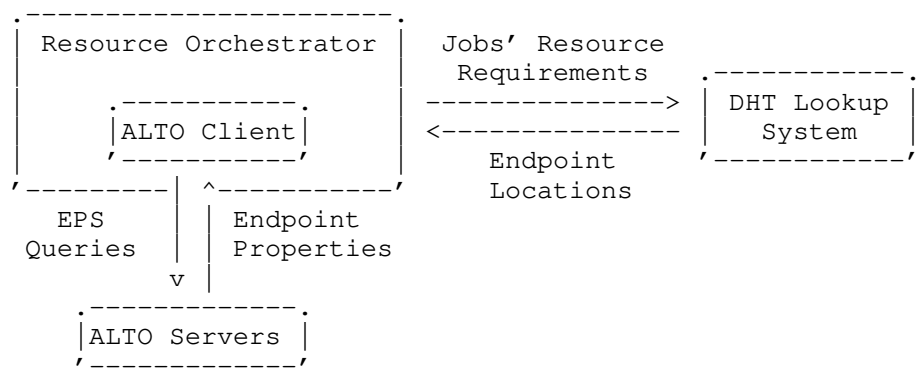

Figure 2: The Endpoint Property Discovery Phase.

7.2.1.2. Phase 2: Endpoint Path Discovery

Candidate computing and storage endpoints need to move data between them before, during and after the execution of a data analytics job. In multi-domain, geo-distributed data analytics, a pair of candidate endpoints may not be in the same member network. In this case, the orchestrator needs to find out the connectivity information between such a pair of candidate endpoints.

Figure 3 shows the procedure of the endpoint path discovery phase. Given a pair of candidate endpoints that are not in the same member network, the ALTO client in the orchestrator adopts an iterative process to find the interdomain connectivity information for this pair. It starts by issuing an ALTO Endpoint Cost Service query or an ALTO Flow-based Endpoint Cost Service [DRAFT-FCS] to the ALTO server of the member network where the source endpoint locates. The cost type of this query is a customized type called next-hop, with a customized cost mode tuple and a customized cost metric next-network.

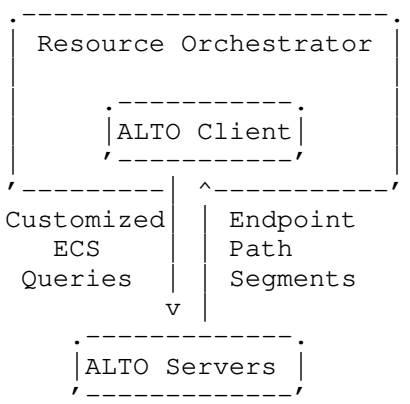

Figure 3: The Endpoint Path Discovery Phase.

The ALTO server returns a 2-tuple, where the first element is the autonomous number (AS) of the next member network along the AS-path from the source endpoint to the destination endpoint, and the second element is the ingress of this next member network. In a member network, the ALTO server can get such information from the underlying interdomain routing protocol, e.g., BGP. Based on the received response, the ALTO client then issues a similar query to the ALTO server of the next member network. The process stops when the ALTO server of the member network where the destination endpoint locates receives such a query, who will return a null 2-tuple in response to notify the ALTO client. By the end of this process, the ALTO client can assemble a domain-path, in the form of a path vector of (ingress, AS), of this pair of candidate endpoints.

7.2.1.3. Phase 3: Resource State Abstraction Discovery

After the second phase, the resource orchestrator has the connectivity information of each candidate endpoint pair, i.e., the domain-path. Equivalently, for each member network, it knows the set of all candidate endpoint pairs that will enter this network. With this information, the resource orchestrator can communicate with the ALTO servers at member networks to discover the resource sharing between all the candidate endpoint pairs. In particular, Unicorn extends the routing state abstraction [DRAFT-RSA] to the more generic resource state abstraction to represent such resource sharing.

Figure 4 shows the procedure of the resource state abstraction discovery phase. For each member network, the ALTO client in the orchestrator sends an ALTO Multipart Cost Property Service query defined in [DRAFT-PV] by providing the set of candidate endpoint pairs as input. The cost type of this query is path vector. Upon

receiving the query, the ALTO server in each member network computes an ALTO cost map and an ATLO property map to the ALTO client. These two maps represent a set of linear inequalities revealing the resource sharing among the set of candidate endpoint pairs in the member network.

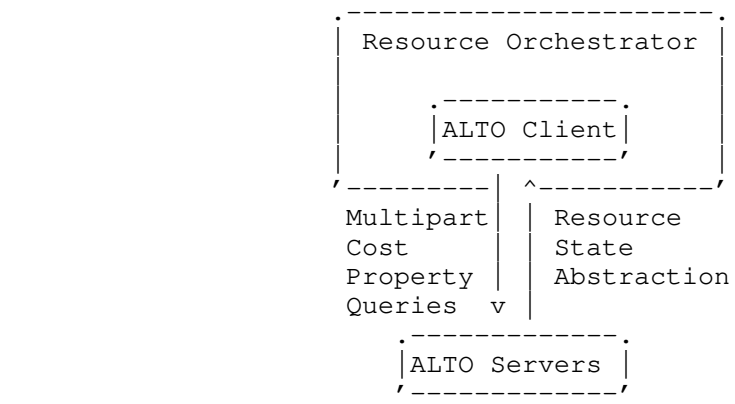

Figure 4: The Resource State Abstraction Discovery Phase.

Unicorn provides two mechanisms for the ALTO servers to return the computed cost maps and property maps to the ALTO client. The first mechanism is to let each ALTO server independently sends its response to the ALTO client. The second mechanism is a privacy-preserving interdomain information aggregation process, in which the ALTO servers in all member networks use a secure multi-party computation (SMPC) protocol to collectively send the responses to the ALTO client without revealing the source of any entry, i.e., the linear in equality, in the cost maps and property maps.

The first mechanism has a higher security risk in that it exposes the bottleneck resource information of each member network. In contrast, the second mechanism provides a better protection of the private information of each member network. The details of the privacy-preserving interdomain information aggregation process will be presented in the next section.

After receiving the responses sent back from the ALTO servers from all the member networks, the orchestrator finishes the whole resource discovery process and collects the accurate information of different types of resources for data analytics jobs.

7.2.2. Proactive Full-Mesh Resource Discovery

To ensure the resource discovery process scales, a proactive full-mesh resource discovery component is developed. The main idea of this component consists in having the ALTO client periodically query ALTO servers at all sites to discover the resource state abstraction between every pair of source and destination sites. As such, when an application submits a resource discovery request, the ALTO client does not need to send any query to the ALTO servers. Instead, using the site-level bandwidth sharing information, the ALTO client can immediately perform projection operations to get the resource information for the request. This mechanism substantially improves the scalability of Unicorn.

7.3. Example

This subsection gives an example to illustrate the workflow of Unicorn. Figure 5 gives a topology of three member networks, where s1 and s2 are storage endpoints and d1 and d2 are computation endpoints. Assume a data analytics job is composed of two parallel tasks T1 and T2. T1 needs dataset X as input and T2 needs dataset Y as input.

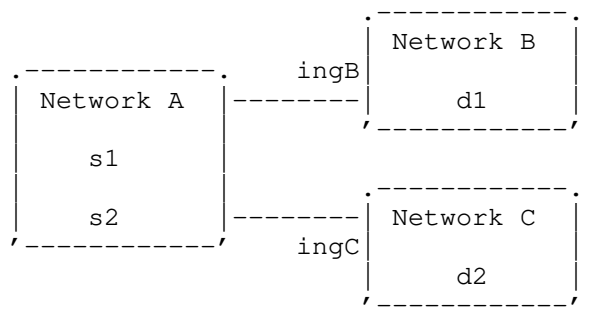

Figure 5: An Illustrating Example for Unicorn.

In the endpoint property discovery phase, the Unicorn resource orchestrator finds that s1 stores X and s2 stores Y, and that the locations of s1, s2, d1 and d2, from the DHT lookup system. It then issues EPS queries to network A, B and C, respectively, to discover that d1 satisfies the computing requirements of T1 and d2 satisfies the computing requirements of T2. Hence there are only two candidate endpoint pairs: (s1, d1) and (s2, d2).

In the endpoint path discovery phase, the ALTO client in the orchestrator iteratively issues Endpoint Cost Service (ECS) query to the ALTO servers in member networks, and finds that the domain-path for pair (s1, d2) is [(null, A), (ingB, B)] and the domain-path for pair (s2, d2) is [(null, A), (ingB, B)]. Hence both pairs will use the networking resources of network A, while only (s1, d1) will use network B and only (s2, d2) will use network C.

In the resource state abstraction discovery phase, the ALTO client in the orchestrator issues Multipart Cost Property Service queries to network A, B and C, respectively. Denote the available bandwidth that can be assigned to T1 as x1 and that to T2 as x2. Assume the linear inequalities computed by the three networks are:

```
A: x1 + x2 <= 10Mbps
B: x1 <= 3Mbps
C: x2 <= 3Mbps
```

If the ALTO servers use the first mechanism to directly return their resource information to ALTO client, respectively, each of them will send a cost map and a property map response encoding its own linear inequality to the ALTO client. In this way, the orchestrator gets the accurate information about networking resource sharing between (s1, d1) and (s2, d2). It then can invoke a resource allocation algorithm to allocate the resources to tasks T1 and T2. For example, if the goal is to maximize the minimal bandwidth of two tasks, the allocation decision will be to assign endpoints s1 and d1 to T1, with a bandwidth of 3Mbps, and assign endpoints s2 and d2 to T2, with a bandwidth of 3Mbps as well.

8. ALTO Extension: Privacy-Preserving Interdomain Information Aggregation for Resource Discovery

This section describes a customized ALTO extension in Unicorn that supports the privacy-preserving discovery of networking resource sharing among a set of candidate endpoint pairs.

8.1. Extension Specification

Figure 6 presents the workflow of the proposed ALTO extension. Assume a set of N member networks denoted as AS_1, AS_2, ... AS_N and the number of all candidate endpoint pairs is F. The interdomain information aggregation process works as follows:

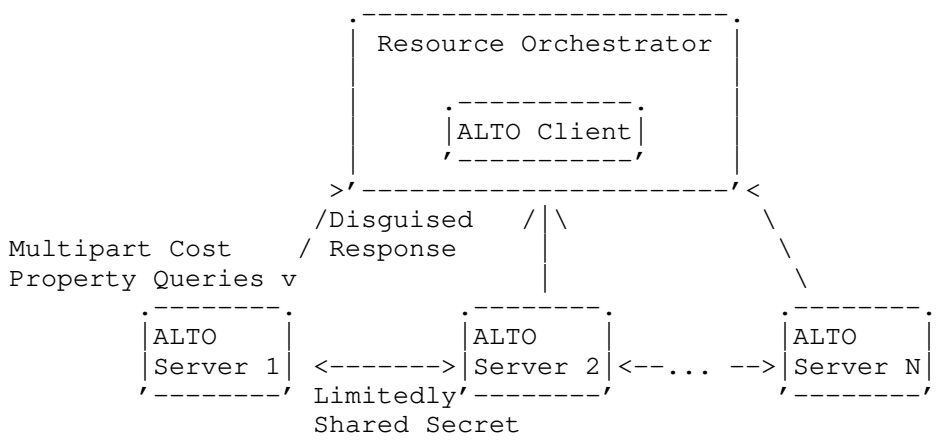

Figure 6: The Privacy-Preserving Interdomain Resource Information Aggregation.

* Step 1: The ALTO client sends the Multipart Cost Property Service request to and a homomorphic public key k_p to each member network.

* Step 2: The ALTO server of each network AS_i computes its own set of linear inequalities A_i x <= b_i. Denote the size of this set as m_i.

* Step 3: The ALTO server of each network AS_i introduces m_i non-negative slack variables to transform its set of linear inequalities into a set of linear equations.

* Step 4: The ALTO servers of all member networks use a private matrix SMPC summation protocol to collectively compute k=m_1 + m_2 + ... + m_N + 1. The value k is known to all the member networks.

* Step 5: The ALTO servers of each network AS_i selects a random k-by-m_i matrix P_i, and computes the matrix P_iA_i and P_ib_i.

* Step 6: The ALTO server of each network then uses a few matrices, which are only shared with a couple of other networks, to further obfuscate P_iA_i and P_ib_i, and sends the obfuscated matrices to the ALTO client via symmetric encryption.

* Step 7: the ALTO client decrypts the received responses from all ALTO servers, and sums up the decrypted response to get a set of linear equations sum P_iA_i x = sum P_ib_i.

This process ensures that the networking resource capacity region derived from sum P_iA_i x = sum P_ib_i is the same as that derived from A_1 x <= b_1, A_2 x <= b_2, ... A_N x <= b_N. More importantly, the ALTO client has no knowledge on the information of network resource sharing of a single member network.

8.2. Example

This subsection uses the same example in Figure 5 to illustrate the privacy-preserving information aggregation process. The set of linear inequalities computed by each network is as follows:

```
A: x1 + x2 <= 10
B: x1 <= 3
C: x2 <= 3
```

Then the networks collectively compute k=1+1+1+1=4. And then introduces slack variables to transform the linear inequalities into linear equations:

```
A: x1 + x2 + x3           <= 10
B: x1           + x4      <= 3
C:      x2           + x5 <= 3
```

For each network, the random matrix it chooses as follows:

```
P_A: [11, 49, 95, 34]
P_B: [58, 22, 75, 25]
P_C: [50, 69, 89, 95]
```

After the obfuscating process in Step 5 and Step 6 in the previous subsection, the decrypted set of linear equations the ALTO client gets is

```
69  x1 + 61  x2 + 11 x3 + 58 x4_ + 50 x5 = 434
71  x1 + 118 x2 + 49 x3 + 22 x4_ + 69 x5 = 763
170 x1 + 184 x2 + 95 x3 + 75 x4_ + 89 x5 = 1442
59  x1 + 129 x2 + 34 x3 + 25 x4_ + 95 x5 = 700
```

Assume the goal is still to maximize the minimal bandwidth of two tasks, the allocation decision made using this set of linear equations will still be x1=3 and x2=3, i.e., assigning endpoints s1 and d1 to T1, with a bandwidth of 3 and assigning endpoints s2 and d2 to T2, with a bandwidth of 3 as well.

9. Implementation and Demonstration Experience

The authors build an ALTO server on top of the OpenDaylight Software Defined Network controller. The ALTO server collects the network state information from the OpenDaylight controller, e.g., topology, policy and traffic statistics, processes the collected information into resource abstraction, and sends the abstraction back to the ALTO client at the resource orchestrator.

In the past few years, the Unicorn framework has been deployed and demonstrated in small federation networks connecting Dallas, Texas, Los Angles, California, and Denver, Colorado at different conventions. For example, in 2018, the federation in the demonstration is composed of three member networks. Network 1 is in Dallas, Texas, and Network 2 and network 3 are in Los Angeles, California. Network 1 is connected to network 2 through a layer-2 WAN circuit with a 100 Gbps bandwidth, provisioned by several providers such as SCinet, CenturyLink and CENIC. Network 1 is a temporal science network in the CMS experiment, while network 2 and 3 are long-running CMS Tier-2 sites. In this federation, users need to reserve network resources to transfer large-scale science datasets (e.g., with a size of hundreds of PB) between networks.

The authors evaluate the accuracy and latency of the framework for discovering network resources in this network. During the evaluation, the framework accurately discovers the network resource information for a large amount of circuits reservation requests with a very low discovery latency. Specifically, for all the reservation requests, Unicorn always provides the accurate information of available bandwidth sharing in the network (i.e., a 100% accuracy), with an average discovery latency of 100 milliseconds, and a worst latency of less than 1 second. With the discovered network resource information, users can transmit large-scale science datasets at a speed up to 100 Gbps, (i.e., the theoretical maximal throughput).

10. Discussion

10.1. Discovering the Domain-Paths Using a New Interdomain Routing Protocol

The current design of the endpoint path discovery process in Unicorn assumes that the underlying interdomain routing protocol is the standard BGP, which only provides the path vector of ASes instead of the path vector of (ingress, AS) tuples needed by Unicorn. If a multi-domain, geo-distributed data analytics system uses an interdomain routing protocol that provides the path vector of (ingress, AS) pairs, the endpoint path discovery process in Unicorn can be simplified to only send queries to the ALTO server of the network where the source candidate endpoint locates.

10.2. Comparison of the Efficiency to Achieve Optimal Resource Orchestration using ALTO and without using ALTO

The authors of this draft conduct a systematic investigation to understand the efficiency differences to achieve optimal resource orchestration between using ALTO and without using ALTO. Specifically, the authors study the problem of computing the optimal resource reservation without using ALTO, but only using the simple reservation interface deployed in PCE-based network resource reservation systems (e.g., OSCARS). Given a client's request, a novel algorithm is developed to compute the optimal resource reservation within O(n^3) times of queries on the simple reservation interface, where n is the number of flows in the request. This is the best known algorithm for this problem. Although the asymptotic complexity appears efficient, the authors' experience to test this design shows that it often takes tens of thousand queries on the simple reservation interface to compute the optimal resource orchestration, leading to substantial orchestration latencies. This result has been published at AAAI 2019.

In contrast, by leveraging the capability of ALTO to expose accurate network information to the client, the Unicorn framework can compute the optimal resource reservation by querying an ALTO server only once, resulting in orders of magnitude improvement on resource orchestration latencies.

10.3. Future Work 1: Unified Resource Representation as an ALTO extension

The authors of this draft are currently designing an ALTO extension which uses generic mathematic programming constraints as a unified resource representation of heterogeneous resources in the network. In particular, a SQL-style language is designed for an ALTO client to express its requirement on available resources. An SMT-based compiler is developed for the network, which translates a user's SQL

resource query into a constraint programming model, finds feasible resources in the network, and returns such resource information to user in the compact representation expressed as generic mathematic programming constraints. More details about this extension are reported in a different draft titled "ALTO Extension: Unified Resource Representation". A paper providing preliminary evaluation results of this extension has been accepted by ACM SIGCOMM NAI 2020 Workshop.

10.4. Future Work 2: Integrating Unicorn and ALTO into Rucio

Ruzio [RUCIO] is a scientific data management framework that supports scientific collaborations with the functionality to organize, manage, and access data at exascales. It is currently deployed at the ATLAS experiment of the LHC project, to manage detector data, simulation data and user data. Rucio interacts with the heterogeneous storage systems that are in use in ATLAS, and schedules the dataset transfers in the ATLAS science network using the network infrastructure provided by multiple national research and education networks, including ESnet and Internet2, as well as commercial cloud storage providers, including Google and Amazon.

However, the same as other scientific data management frameworks, e.g., PhEDEx, Rucio does not see the underlying network infrastructure. As such, it may suffer from underutilization of network resources, leading to a low data transfer efficiency. In contrast, ALTO and Unicorn can provide accurate, yet privacy-preserving network resource information. By integrating ALTO and Unicorn into the Rucio framework, Rucio will possess the capability to retrieve accurate resource information from the underlying network infrastructure, and use such information to schedule the exascale scientific data flows with better predictable performances (e.g., latency and throughput). As such, we plan to systematically investigate the integration of ALTO/Unicorn and Rucio in the new charter cycle of the ALTO working group.

11. Security Considerations

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

12. IANA Considerations

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

13. References

13.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

13.2. Informative References

[Apollo] Boutin, E., Ekanayake, J., Lin, W., Shi, B., Zhou, J., Qian, Z., Wu, M., and L. Zhou, "Apollo: Scalable and Coordinated Scheduling for Cloud-Scale Computing", 2014, <https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-boutin_0.pdf>.

[Borg] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and J. Wilkes, "Large-scale cluster management at Google with Borg", 2015, <https://dl.acm.org/citation.cfm?id=2741964>.

[DRAFT-FCS] Zhang, J., Gao, K., Wang, J., Xiang, Q., and Y. R. Yang, "ALTO Extension: Flow-based Cost Query", 2017, <https://datatracker.ietf.org/doc/draft-gao-alto-fcs/>.

[DRAFT-PV] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. R. 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., Xiang, Q., Gu, C., Yang, Y. R., and G. Chen, "A Recommendation for Compressing ALTO Path Vectors", 2017, <https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>.

[DRAFT-SSE] Roome, W. and Y. R. 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>.

[Mesos] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A., Katz, R., Shenker, S., and I. Stoica, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center", 2011, <http://static.usenix.org/events/nsdi11/tech/full_papers/Hindman_new.pdf>.

[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, <https://www.rfc-editor.org/info/rfc7285>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <https://www.rfc-editor.org/info/rfc8189>.

[RUCIO] Barisits, M., "Rucio: Scientific Data Management", 2019, <https://doi.org/10.1007/s41781-019-0026-3>.

[Sparrow] Ousterhout, K., Wendell, P., Zaharia, M., and I. Stoica, "Sparrow: Distributed, Low Latency Scheduling", 2013, <https://dl.acm.org/citation.cfm?id=2522716>.

[XCP] Katabi, D., Handley, M., and C. Rohrs, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments", 2002, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.4783>.

Authors' Addresses

Qiao Xiang
Xiamen University
School of Informatics
Xiamen
Fujian Province,
China

Email: xiangq27@gmail.com

J. Jensen Zhang
Tongji/Yale University
51 Prospect Street
New Haven, CT
United States of America

Email: jingxuan.zhang@yale.edu

Franck Le
IBM
Thomas J. Watson Research Center
Yorktown Heights, NY,
United States of America

Email: fle@us.ibm.com

Y. Richard Yang
Yale University
51 Prospect Street
New Haven, CT
United States of America

Email: yry@cs.yale.edu

Harvey Newman
California Institute of Technology
1200 California Blvd.
Pasadena, CA
United States of America

Email: newman@hep.caltech.edu