Unicorn: Unified Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics

Qiao Xiang$^{+\ddagger}$, Shenshen Chen^{+}, Kai Gao*‡, Harvey Newman$^{\diamond}$,
Ian Taylor$^{\circ\dagger}$, Jingxuan Zhang^{+}, Yang Richard Yang$^{+\ddagger}$
$^{+}$Tongji University, ‡Yale University, *Tsinghua University,
$^{\diamond}$California Institute of Technology, $^{\circ}$Cardiff University, †University of Notre Dame
{qiao.xiang, kai.gao, yang.r.yang}@yale.edu, {jingxuan.zhang, cs9091}@tongji.edu.cn,
newman@hep.caltech.edu, taylorij1@cardiff.ac.uk

***Abstract*—Data-intensive analytics is entering the era of multi-organizational, geographically-distributed, collaborative computing, where different organizations contribute various resources, *e.g.*, sensing, computation, storage and networking resources, to collaboratively collect, share and analyze extremely large amounts of data. This new paradigm calls for a framework to manage a large set of distributively owned heterogeneous resources, with the fundamental objective of efficient resource utilization, following the autonomy and privacy of resource owners. In this paper, we design Unicorn, the first unified framework that accomplishes this goal. The foundation of Unicorn is RSDP, an autonomous, privacy-preserving resource discovery and representation system to provide accurate resource availability information. Its core is a novel abstraction called resource vector abstraction which describes the resource availability in a set of linear constraints. In addition, Unicorn also provides a series of advanced solutions to support automatic, efficient management of resource dynamics on both supply and demand sides, including an automatic workflow transformer, an intelligent resource demand estimator and an efficient, scalable multi-resource orchestrator. Being the first unified framework for this new paradigm, Unicorn plays a fundamental role in next-generation data-intensive collaborative computing systems.**

***Keywords*—multi-domain, data analytics, resource allocation.**

I. 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. Examples of this paradigm include the Compact Muon Solenoid (CMS) experiment at CERN [1], coalitions between different combating units, etc. Figure 1 summarizes the general settings of collaborative computing: data-intensive analytics workflows consume resources supplied by participating sites/resource owners, coordinated by a logically centralized orchestrator. Collaborative computing calls for a framework to manage a large set of distributively owned heterogeneous resources, with the fundamental objective of *efficient resource utilization, following the autonomy and privacy of resource owners*. To achieve this goal, this framework must be capable of the following functionalities.

- **Managing resource supply dynamic.** This requires a system for joint computation, storage and networking resource discovery and representation, which allows resource owners to make and practice their own resource supply strategies without exposing private information. Without such a component, it is infeasible to manage a large set of distributively owned, heterogeneous, dynamic resources.
- **Managing resource demand dynamic.** This requires a system for automatic, effective resource demand estimation, which automatically transforms high-level data analytics workflows (*e.g.*, Spark) to low-level task workflows (*e.g.*, HTCondor ClassAds) and finds the optimal configuration, *i.e.*, resource demand, for each task. Without such a component, users have to manually configure the low-level workflows for data analytics. On one hand, users might request more than necessary resources for workflows, resulting in inefficient use of resources. On the other hand, if users request insufficient resources for workflows, the system may need more resource distribution transactions, resulting in larger overhead.
- **Matching demand with supply.** This requires a system for efficient, scalable multi-resource orchestration, which makes efficient resource allocation decisions for analytics workflows based on resource supply and demand. This component is essential for achieving efficient resource utilization.

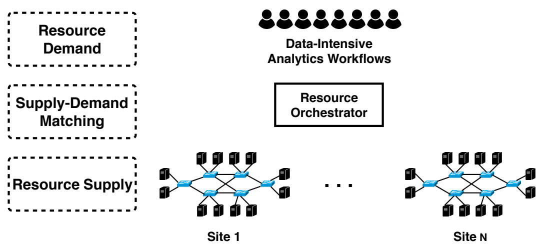


Fig. 1. General settings of multi-organization, geo-distributed, data-intensive collaborative computing: (1) users submit analytics workflows to produce resource demand; (2) different sites/resource owners provide resource supply; (3) a logically centralized orchestrator matches demand with supply, *i.e.*, allocating resources to analytics workflows.

In this paper, we propose Unicorn, the first unified framework providing all these functionalities for multi-organizational, geographically-distributed collaborative computing.

The fundamental design challenge for Unicorn is: *how to accurately discover and represent resource availability in a large set of distributively owned heterogeneous resources*? Although there is much related work on resource management of cluster computing [2]–[11], current systems are mostly designed for a single administrative domain and focus on computation and storage resources by assuming the networking resource is not a bottleneck. Such settings usually lead to easier designs. In particular, current systems typically adopt a graph representation to describe resource availability, where each node is a physical node representing computation or storage resources and each edge between a pair of nodes denotes the networking resource. In the multi-domain collaborative computing, where resources are distributively owned and all resources (*i.e.*, computation, storage and networking) have the same probability of becoming the bottleneck of analytics [12], this graph representation has quite a few drawbacks. First, it could lead to race conditions between resource suppliers and consumers. Secondly, multi-resource interferences would lead to inefficient use of allocated resources. Thirdly, this representation hides the underlying resource sharing between nodes or edges in the physical topology, which leads to the over-provisioning of resources.

To address these drawbacks and the design challenge, we developed RSDP, an autonomous, privacy-preserving resource discovery and representation system, to accurately represent available resources in collaborative computing systems. This is achieved through a novel abstraction called *resource vector abstraction*, which describes the resource availability using a set of linear constraints.

With RSDP managing the resource supply dynamic, the Unicorn framework also provides the Handyman system to manage the resource demand dynamic, which automatically transforms high-level analytics workflows to low-level task workflows and finds them the optimal configurations. Between supply and demand, Unicorn designs an efficient, scalable multi-resource orchestrator called Miro to achieve efficient resource utilizations.

The rest of the paper is organized as follows. We first give an overview of the Unicorn framework and introduce its core components in Section II. We then present the details of RSDP, the core resource discovery and representation system of the Unicorn framework in Section III and its preliminary evaluation results in Section IV. We briefly discuss related work in Section V and conclude the paper in Section VI.

II. System Architecture of Unicorn

Unicorn aims to achieve two design goals simultaneously for data-intensive collaborative computing: (1) achieve efficient utilization of a large set of distributively owned heterogeneous resource, and (2) allow each participating site to practice policies and protocols at its own choices without revealing private information. The first goal ensures that the available resource supply is efficiently matched to the resource demand of data-intensive analytics workflows. The second goal ensures the autonomy, privacy and security of each participating site.

Figure 2 gives the overall architecture of the Unicorn framework, which consists of three components. The foundation of Unicorn is RSDP, an autonomous, privacy-preserving resource discovery and representation system to accurately represent availability information of a large set of distributively owned resources. On the resource demand side, Unicorn provides Handyman, an analytics demand automation system, which automatically converts high-level analytics workflows into low-level task workflows and finds the optimal configuration (*i.e.*, resource demand) for each task. Between resource demand and supply, an efficient, scalable multi-resource orchestrator called

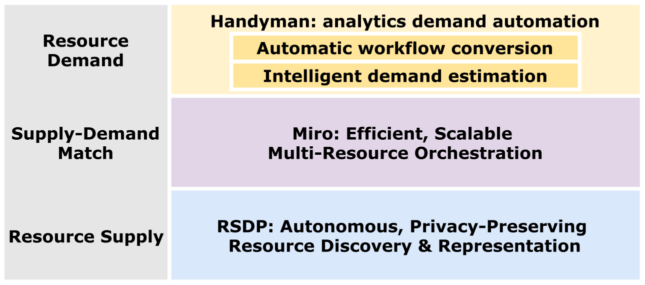

Resource Demand
Handyman: analytics demand automation
Automatic workflow conversion
Intelligent demand estimation
Supply-Demand Match
Miro: Efficient, Scalable Multi-Resource Orchestration
Resource Supply
RSDP: Autonomous, Privacy-Preserving Resource Discovery & Representation

Fig. 2. The Architecture of Unicorn framework.

Miro is provided in Unicorn to efficiently utilize resources in the system for data-intensive analytics workflows.

RSDP: an automatic, privacy-preserving resource discovery and representation system. RSDP provides an accurate view on resource supply dynamic and is the foundation of the Unicorn framework. This is achieved through a novel abstraction called *resource vector abstraction*. Given a set of tasks T, a set of resources R and a set of resource attributes P, resource vector abstraction uses a set of linear constraints to represent the feasibility and constraints of resource availability and sharing. When a set of original linear constraints C is obtained, RSDP does not directly return the whole set as the resource availability information. Instead, it uses a lightweight, optimal algorithm to compress C into a minimal, equivalent set of constraints C' where the feasible regions represented by C and C' are the same. Through this compression, RSDP (1) avoids the high communication overhead of transmitting resource availability information between the orchestrator and sites, and (2) minimizes the risk of each site exposing private information about its computing facilities, *e.g.*, topology, policies, etc.

Handyman: an analytics demand automation system. Handyman automatically (1) converts high-level analytics workflows into low-level task workflows, *i.e.*, a set of tasks with precedence encoded in a directed acyclic graph (DAG), and (2) finds the optimal configuration for each task. This component saves users from the trouble of manually specifying low-level task workflows. During the conversion, a critical challenge must be addressed is that high-level analytics workflows are sometimes stateful while low-level workflows are not. To guarantee that the workflows are equivalent before and after conversion, the current Handyman design performs conversion against the state, and schedules automatic conversion re-execution when state changes happen. Another solution under investigation involves changing the programming models of low-level task workflows by allowing them to be stateful.

After the conversion, Handyman automatically estimates the optimal configuration (resource demand, *e.g.*, the number of CPUs, the size of memory and disk, I/O bandwidth, etc.) for each task. It leverages the fact that low-level tasks are typically repetitive with strong similarities and applies reinforcement learning to estimate optimal configurations for similar tasks. In articular, it records the utilization and configurations of each task executed and predicts the resource utilization for different configurations. When a task comes, Handyman chooses an unrecorded configuration with the best prediction, and records the resource utilization of the actual execution for next round of reinforcement learning.

Miro: an efficient, scalable multi-resource orchestrator. Miro makes resource allocation decisions based on resource demand and supply dynamic. It receives the resource demand information, *i.e.*, a set of low-level task workflows and their configurations, from Handyman. Then it sends queries to the RSDP system at each site with a *what-if question*: what resource would be provided if the requested tasks were to be executed here? After receiving the responses encoded in resource vector abstraction, Miro looks up different resource provisions and makes task placement decisions to guarantee that the available resources are efficiently matched with the resource demand of each task. These decisions are then sent to the task execution agents at each site, who execute the tasks, monitor the progress and resource usage and report back to Miro. Miro supports different scheduling modes, including FIFO, global mixed binary programming, least-demand-first (LDF), etc. Comparing and understanding the performance of different scheduling modes is the next step of Miro.

The most fundamental design challenge of Unicorn is how to accurately discover and represent the resource availability over a large set of distributively owned heterogeneous resources. In the next two sections, we focus on addressing this challenge and present our solution RSDP, the foundation of Unicorn. Details of Handyman and Miro are omitted for the interests of brevity.

III. RSDP: Autonomous, Privacy-Preserving Resource Discovery and Representation

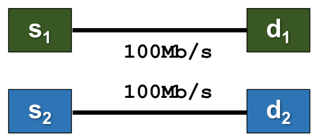
In this section, we first review the limitation of resource availability graph, a common resource discovery design adopted in most current resource management systems (Section III-A). We then propose RSDP, an autonomous, privacy-preserving resource discovery and representation framework to accurately represent available resources in collaborative computing systems, in which the resource availability is encoded in a novel abstraction called *resource vector abstraction* as a set of linear constraints (Section III-B). Next, we discuss the challenges of generalizing the resource vector abstraction design and some practical concerns for production deployment (Section III-C).

A. Resource Availability Graph: Incomplete State-of-the-Art

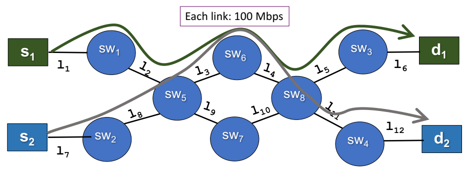
Basic idea. Computer systems typically consist of three types of resources: computation, storage, and networking. A typical design of resource discovery adopted by most current systems is a *resource availability graph*, which is built on top of the physical topology. Each node in the graph is a physical node, with computation/storage resources annotated, and each edge between a pair of nodes is annotated with available networking resources.

Drawbacks. The main assumption of a resource availability graph is that node and link attributes are described as independent variables. However, it has several drawbacks when used for resource discovery of a large set of distributively owned resources. First, the multi-domain nature of collaborative computing has determined that race conditions could happen frequently using resource availability graphs. Suppose a resource owner announces the availability of a set of resources, and two different tasks both want to use these resources. Without careful synchronizations, both tasks may try to use this set of resources, leading to conflicts.

Secondly, current systems mainly focus on discovering computation and storage resources (*e.g.*, CPU, memory and disk) and assume networking resource is not a bottleneck. However, recently during a production trace of cluster computing, it was shown that all three types of resources have approximately the same probability of becoming a bottleneck in affecting the performance of data-intensive analytics [12]. With this fact, the graph representation could lead to inefficient use of allocated resources. Consider an example where nodes A, B and edge $\{A, B\}$ are allocated to a task. Nodes A and B both have a local I/O bandwidth of $1Gb/s$ while edge $\{A, B\}$ is annotated with a bandwidth of $10Gb/s$. We see that the computation and storage resources become the bottleneck of the task (*i.e.*, $1Gb/s$), and the communication bandwidth (*i.e.*, $10Gb/s$) will be at most utilized by $1/10 = 10\%$.



(a) The resource graph representation shows that $\{bw(s_1, d_1) \leq 100Mb/s, bw(s_2, d_2) \leq 100Mb/s\}$.



(b) The physical topology shows that the paths of (s_1, d_1) and (s_2, d_2) share bottleneck links, *i.e.*, $bw(s_1, d_1) + bw(s_2, d_2) \leq 100Mb/s$.

Fig. 3. An example to demonstrate the inefficiency of resource graph representation.

Thirdly, the graph representation abstracts each resource into a single node or edge regardless of the shared resource between nodes or edges in the physical topology. This would lead to over-provisioning of resource and jeopardize the performance of analytic workflows. Consider the example in Figure 3. The resource graph in Figure 3a) contains two sets of resources: $\{s_1, (s_1, d_1), d_1\}$ and $\{s_2, (s_2, d_2), d_2\}$. The available bandwidth on edges (s_1, d_1) and (s_2, d_2) are both $100Mb/s$ and for simplicity, we assume the I/O bandwidths of all end hosts are larger than $100Mb/s$. However, in the physical topology shown in Figure 3b), the communication between two sets of nodes shares resources on links l_3 and l_4. Hence, the actual networking resource for the two sets of physical nodes must satisfy the constraint $bw(s_1, d_1) + bw(s_2, d_2) \leq 100Mb/s$. This key constraint cannot be expressed using the variable annotation in a resource availability graph.

The fundamental cause to these drawbacks is that resource availability graph lacks the ability to represent the feasibility and constraints related to resource sharing. To address this problem, we propose a new resource discovery and representation system called RSDP.

B. RSDP: Autonomous, Privacy-Preserving Resource Discovery and Representation

To address the aforementioned limitations of resource availability graph, we design a new system called RSDP for resource discovery and representation over a large set of heterogeneous distributively owned resources. In particular, we provide a complete view of resource availability at each site using a novel abstraction called resource vector abstraction.

Basic idea. The design principle of resource vector abstraction is simple yet powerful: instead of abstracting resources into a single node or edge out of the physical topology, we *keep the physical topology and use a set of linear constraints to represent the feasibility and constraints of resource availability and sharing*.

Resource vector abstraction. Given a set of data-intensive analytic tasks T, a set of physical resources R (*i.e.*, computation, storage and networking) and a series of attributes $r.P$ for each resource $r \in R$, we use $C(r,p)$ to denote the capacity of resource r in providing attribute p and use $c(t,r,p)$ to denote the usage of the attribute $p \in r.P$ of resource $r \in R$ by task $t \in T$, the resource availability of this set of physical resources R over the set of tasks T can be expressed as:

$$\begin{array}{ll} \sum_{t \in T} c(t,r,p) \leq C(r,p), & \forall r \in R, p \in r.P. \\ c(t,r_i,p) = c(t,r_j,p) & \forall r_i, r_j \in R, \text{ given } (t,p) \end{array} \quad (1)$$

In addition to the generic representation, another benefit of the resource vector abstraction is that the site-dependent policies can be naturally mapped into additional linear constraints. This is because a site-dependent policy p_s is typically represented as a set of tasks/flows that (1) can or cannot use a certain resource; (2) cannot exceed an upper bound of a certain resource; or (3) will be guaranteed a lower bound of a certain resource. All such policies can be expressed in the form of linear constraints.

Example. To illustrate how resource vector abstraction represents the resource availability, we revisit the physical topology in Figure 3b). Assume there are 2 tasks t_1, t_2 and we focus on the bandwidth attribute of networking resource in the set of links L, consisting of l_1 to l_{12}. We first follow the definition in Equation (1) and represent the resource availability of link bandwidth as:

$$\begin{array}{ll} c(t_1,l_i,bw) + c((t_2,l_i,bw) \leq 100Mb/s, & \forall l_i \in L. \\ c(t_1,l_i,bw) = c(t_1,l_j,bw), & \forall l_i,l_j \in L. \\ c(t_2,l_i,bw) = c(t_2,l_j,bw), & \forall l_i,l_j \in L. \end{array} \quad (2)$$

Assume the policies in this site enforces that (1) t_1 must use computation/storage resources on s_1 and d_1; (2) t_2 must use computation/storage resources on s_2 and d_2; and (3) the traffic of (s_1,d_1) and (s_2,d_2) must follow the predefined routes. After combining these policies with the constraints in Equation (2), we get:

$$\begin{array}{ll} c(t_1,bw) \leq 100Mb/s & \forall l_i \in \{l_1,l_2,l_5,l_6\}, \\ c(t_2,bw) \leq 100Mb/s & \forall l_i \in \{l_7,l_8,l_{11},l_{12}\}, \\ c(t_1,bw) + c(t_2,bw) \leq 100Mb/s & \forall l_i \in \{l_3,l_4\}, \\ c(t_1,bw) + c(t_2,bw) = 0 & \forall l_i \in \{l_9,l_{10}\}, \end{array} \quad (3)$$

which is a set of linear constraints that provides accurate, complete information about resource availability in this site.

Computing minimal, equivalent resource state abstraction. The representation of resource availability specified in Equation (1) and site policies is accurate and complete, but may result in a large set of linear constraints with redundant information. Directly sending them back to the querying party would introduce a large communication overhead and expose private information about each site, *e.g.*, site policies and topology. To address the efficiency, privacy and security concerns, we develop a lightweight, optimal algorithm in RSDP to compress the original large set of linear constraints into a minimal, equivalent set of linear constraints, which has the same feasible region as the original set but with a much smaller number of constraints. The basis of this compression algorithm is simple: given an original set of linear constraints $C : \mathbf{A}\mathbf{x} \leq \mathbf{b}$, we iteratively select one constraint $c \in C : \mathbf{a}^T\mathbf{x} \leq b$ and calculate the optimal solution of problem $y \leftarrow \max \mathbf{a}^T\mathbf{x},$ subject to, $C - \{c\}$. If b is smaller than the resulting y, c is an indispensable constraint in determining the feasible region and will be put into the minimal, equivalent constraint set C'. Otherwise, c is a redundant constraint. We prove the optimality of this algorithm via contradiction.

Applying the algorithm above on the original set of linear constraints in Equation (3) with 12 constraints, RSDP can compute and get the minimal, equivalent set of constraints C' with only 1 constraint: $\{(t1,bw) + bw(t2,bw) \leq 100Mb/s\}$, achieving a compression ratio of $\frac{1}{12}$.

Schedulability. RSDP provides an accurate view of resource availability while allowing resource owners to make and practice their own policies with minimal exposure of private information. One important remaining question is whether RSDP provides full a schedulability of resources for a logically centralized orchestrator. We answer this question with the following theorem.

Theorem 1: When the resources represented by the resource vector abstraction satisfies one of the following conditions:

1) resources represented in the original set of constraints C can be fully controlled on the edge, *i.e.*, all the attributes of each resource can be controlled at end host;
2) all the attributes computed in C', the minimal, equivalent resource vector abstraction, can be fully controlled on the edge;

RSDP provides a full schedulability of resources to a logically centralized resource orchestrator.

Proof: The proof of this theorem is straightforward. Condition 1 requires that all the resources and their attributes can be controlled on the edge for orchestration purposes. Because resource vector abstraction encodes all resource attributes in the original set of constraints C, it provides a complete view of resource availability so that the orchestrator can control them on the edge. For instance, if the sending rate of each end host can be controlled by end host rate limiting, the bandwidth usage of each end host, therefore, can be controlled by the orchestrator to achieve efficient bandwidth utilization. On the contrary, if TCP is used to perform window-based congestion control, the control functionality of bandwidth allocation is given to TCP and the orchestrator cannot allocate bandwidth via the representation provided by resource vector abstraction. Condition 2 relaxes condition 1 by only requiring the attributes left in the minimal, equivalent resource vector abstraction C' to be controllable. Because of the equivalence between C and C', satisfying condition 2 ensures that the orchestrator can achieve a full schedulability of resources through the resource vector abstraction representation. ■

C. Generalization of Resource Vector Abstraction

resource vector abstraction is a powerful abstraction for resource availability. However, several issues must be addressed to apply it in the general case. We describe these issues, propose potential solutions, and discuss practical concerns for production deployment of RSDP.

Inter-attribute Correlation. The first limitation of resource vector abstraction is that the availability of different resources is only coupled through common resource attributes, *e.g.*, bandwidth of storage and networking resources. But in practice, different resource attributes can have impacts on others too. For example, the block size of storage resources will affect the data transferring time between computation resources and storage resources. Such correlation between different resource attributes is not encoded in the current design of resource vector abstraction.

To address this issue, we can add the following constraints in the definition in Equation 1.

$$f(T, R, P) \leq 0, \tag{4}$$

which are a set of functions modeling the impact between different resource attributes. There are two challenging problems: (1) formulations of f depend on specific resource attributes and are often unknown; (2) whether it is possible to eliminate the redundancy between different f to get a minimal, equivalent resource vector abstraction depends on certain properties of f (*e.g.*, convex, linear or concave). Learning techniques can be applied to cope with these challenges and is part of the ongoing efforts in the Unicorn framework.

Coexistence of unschedulable resources. As Theorem 1 states, the orchestrator on top of RSDP cannot achieve full schedulability of resources via the current resource vector abstraction design, when some resources are not controlled at the edge. For example, if the site uses TCP for congestion control instead of end-host rate limiting, bandwidth of each flow is decided by TCP via congestion signaling, *e.g.*, packet loss. The resulted packet-level rate fluctuation would prevent an accurate prediction on the per-flow achievable bandwidth in a network containing many TCP flows.

We use a black-box approach to predict the converged resource availability of unschedulable resources. For instance, by applying the Newton-Exact-Diagonal (NED) method in network utilization maximization [13], it is possible to quickly compute the converged rate of flows. This approach adds the following constraint to resource vector abstraction:

$$g(t, R, P) \leq a, \tag{5}$$

where the resource availability of unschedulable resources is predicted as a scalar a. Again, it also requires certain learning techniques to accurately predict the performance of unschedulable resources and is part of the ongoing efforts of the Unicorn framework.

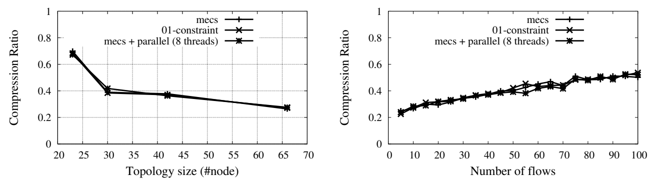
In addition, the generalization of resource vector abstraction also requires consideration on the correlation between task workflows and resource availability, the trade-off between maximizing short-term resource utilization and the long-term stability of resource availability, etc. We leave such topics as future work.

Practical issues for production deployment of RSDP. Other issues also arise when RSDP is deployed in production. First, the current design of RSDP requires a logically centralized controller to query resource availability at each site for a given set of analytics tasks. This design may not scale if the whole computing system involves many globally distributed sites and have heavy analytics workflows. Potential solutions to improve the scalability of RSDP include (1) using hierarchical organized controllers; (2) leveraging the repetitive property of analytics jobs to selectively query resource availability for only a small set of tasks instead of all of them; and (3) applying machine learning techniques to predict resource availability.

The second issue is whether site/resource owners should have total autonomy. This is related to the specific form of contracts between sites participating in collaborative computing. If all sites agree on partial autonomy, a resource enforcer module needs to be deployed together with RSDP to ensure that each site provides the amount of resources specified in contracts. The third issue is how to enforce global scheduling policies, *e.g.*, user/group/virtual organization priorities, etc., when querying for resource availability information. Solutions to this issue are overlapping with those of the previous two issues and we leave them as future work.

IV. Performance Evaluation

A prototype of RSDP has been implemented and we present key evaluation results to demonstrate its efficiency and efficacy in providing an autonomous, privacy-preserving resource representation. Without loss of generality, we focus on the bandwidth attribute of networking resources in our evaluation. We select 10 physical topologies from the topology zoo [14] with the number of nodes ranging from 13 to 117. We vary the number of tasks in the evaluation to range from 5 to 100.



(a) Compression ratio under different sizes of physical topology.

(b) Compression ratio under different numbers of tasks.

Fig. 4. Constraint compression ratio of RSDP.

We run different versions of the constraint compression algorithm proposed in Section III-B, including the regular version, a parallelized version and a modified version leveraging the fact that $c(t, r, p)$ always has a coefficient of 0 or 1 to speed up the compression. We present the compression ratio of RSDP in Figure 4. It can be observed that all three versions of the constraint compression algorithm produce the same compression ratio under all settings. Figure 4a) shows that the compression ratio of RSDP decreases as the topology size grows. This is because with more networking resource entities (*i.e.*, links) available, the chance for different tasks to share the same resource entities decreases. Figure 4b) shows that when the topology size is fixed, the compression ratio grows as the number of tasks grows. Even so, we observe that RSDP can still compress more than 40% of the original resource state, achieving a compression ratio less than 0.6. These results demonstrate the efficiency and scalability of RSDP in providing autonomous, privacy-preserving resource representation. We also measure the computation overhead of RSDP and observe that RSDP using a parallel compression algorithm has a very low computation delay, *e.g.*, <100ms, even for a combination of a large topology and a large set of tasks. We omit other evaluation results due to the page limit.

V. Related Work

The fundamental challenge of resource management for multi-organizational, geo-distributed, data-intensive collaborative computing is to accurately discover and represent resource availability in a large set of distributively owned heterogeneous resources. There exists a rich literature in the field of resource management of cluster computing [2]–[11]. Most of these studies focus on managing resources of a single cluster/data center. YARN [4] is the core resource management framework of Hadoop. Mesos [3] is a platform designed to share resources among multiple cluster computing frameworks, *e.g.*, MapReduce [15], Spark [16], MPI and etc. Google designs a system called Borg [5] to orchestrate the cluster resources for its proprietary data analytics frameworks. Microsoft (*i.e.*, Apollo [6]) and Facebook (*i.e.*, Corona [7]) also develop similar systems tailored to their data analytics needs. HTCondor [2] proposes a ClassAds programming model, which allows different resource owners to advertise their resource supply and the job owners to advertise the resource demand. The CMS [1] experiment at CERN uses HTCondor and glideinWMS [8] to manage a set of distributively owned computing resources in a globally distributed system.

The common settings of these systems, *i.e.*single-domain (except for HTCondor) and no networking bottleneck, usually lead to easier designs for resource discovery and representation. In particular, a graph representation is adopted by current systems to represent available resources. However, in multi-organizational, data-intensive collaborative computing, this design suffers from race conditions between resource suppliers consumers. What is worse, with the recent observation that computation, storage and networking resources have approximately the same probability to become the bottleneck affecting the performance of data-intensive analytics jobs [12], such a graph representation would lead to inefficient use of resources and resource over-provision. On the contrary, the RSDP system in Unicorn addresses these drawbacks by using a set of linear constraints to represent the feasibility and constraints of resource availability and sharing.

Another line of work called geo-distributed data analytics is also related to Unicorn. Solutions in this field include (1) moving the input dataset to a single data center before the computation [17], [18] and (2) placing different amounts of tasks at different sites depending on dataset availability to achieve a better parallelization and hence a lower latency [9]–[11]. The main focus of these solutions is to optimize the usage of a set of dedicated networking resources. This simplified setting is different from that of Unicorn, where different types of resources owned by different owners need to be orchestrated for data-intensive collaborative computing.

VI. Conclusion and Future Work

Multi-organizational, geo-distributed, data-intensive collaborative computing calls for a framework to manage a large set of distributively owned heterogeneous resources, with the fundamental objective of efficient resource utilization, following the autonomy and privacy of resource owners. In this paper, we propose Unicorn, the first unified framework that accomplishes this goal. The foundation of Unicorn is RSDP, an automatic, privacy-preserving resource discovery and representation system which describes the resource availability using a set of linear constraints. In addition, Unicorn also provides an analytics demand automation system, *i.e.*, Handyman, and an efficient, scalable multi-resource orchestrator, *i.e.*, Miro. We have implemented a prototype of Unicorn and performed a preliminary evaluation. For future work, we plan to evaluate the performance and scalability of Unicorn more extensively before moving to production deployment.

Acknowledgement

We thank Justas Balcas, Shiwei Chen, Lili Liu, Maria Spiropulu and Jean-Roch Vlimant for helpful discussion during the work. The Yale team was supported in part by NSF grant #1440745, CC*IIE Integration: Dynamically Optimizing Research Data Workflow with a Software Defined Science Network; International Technology Alliance Agreement No W911NF-16-3-0002; Google Research Award, SDN Programming Using Just Minimal Abstractions; NSFC #61672385, FAST Magellan. The Tongji team was supported by China Postdoctoral Science Foundation #2017M611618. The Caltech team was supported in part by DOE/ASCR project #000219898, SDN NGenIA; DOE award #DE-AC02-07CH11359, SENSE, FNAL PO #626507; NSF award #1246133, ANES; NSF award #1341024, CHOPIN.

References

[1] T. C. Collaboration, "The CMS experiment at the CERN LHC," *Journal of Instrumentation*, vol. 3, no. 08, 2008.

[2] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[3] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *NSDI*, 2011.

[4] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache Hadoop YARN: Yet another resource negotiator," in *SoCC*. ACM, 2013, p. 5.

[5] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *EuroSys*. ACM, 2015, p. 18.

[6] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: Scalable and coordinated scheduling for cloud-scale computing," in *OSDI*, 2014, pp. 285–300.

[7] "Under the hood: Scheduling MapReduce jobs more efficiently with Corona," http://on.fb.me/TxUsYN, [Online; accessed: 09-May-2017].

[8] I. Sfiligoi, D. C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, and F. Wurthwein, "The pilot way to grid resources using glideinWMS," in *CSIE*. IEEE, 2009, pp. 428–432.

[9] A. Vulimiri, C. Curino, B. Godfrey, K. Karanasos, and G. Varghese, "WANalytics: Analytics for a geo-distributed data-intensive world," in *CIDR*, 2015.

[10] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low Latency Geo-distributed Data Analytics," in *SIGCOMM*. ACM, 2015, pp. 421–434.

[11] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *SoCC*. ACM, 2015, pp. 111–124.

[12] K. Ousterhout, R. Rasti, S. Ratnasamy, S. Shenker, B.-G. Chun, and V. ICSI, "Making sense of performance in data analytics frameworks," in *NSDI*, 2015, pp. 293–307.

[13] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: Flowlet control for datacenter networks," in *NSDI*. USENIX Association, 2017.

[14] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," vol. 29, no. 9, pp. 1765–1775, 00249.

[15] D. Jeffrey and G. Sanjay, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, 2008.

[16] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *HotCloud'10*.

[17] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *SIGCOMM'13*.

[18] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *SIGCOMM*. ACM, 2013.