Towards Data Gravity and Compliance Aware Distributed Deep Learning on Hybrid Clouds

Avinash Maurya, Jaiaid Mobin, M. Mustafa Rafique
High Performance Distributed Systems Laboratory (HPDSL), Rochester Institute of Technology, New York, USA
Email: {am6429, jm5071, mrafique}@cs.rit.edu

***Abstract*—To store large volumes of data concurrently from a diverse set of sources, data stores such as data silos, lakes, and warehouses, have been widely embraced by various organizations. Thanks to data fabric architectures, such scattered data (both structurally and geographically), can be accessed transparently at scale while adhering to various administrative regulations (e.g. governance, privacy, compliance, etc.). However, modern workload schedulers and distributed deep learning (DDL) runtimes are oblivious to the uneven data distribution across different storage services and compliance regulations, leading to sub-optimal resource utilization and training completion times. Although state-of-art workflow schedulers such as Apache Hadoop Yarn, Horovod, etc. exploit data locality, they require application developers to explicitly map data and resources available across various cloud services during job submission. These approaches are redundant and counterproductive for next-generation data fabric architectures that feature automated transparency and compliance abstractions for accessing disparate data sources with uneven data distribution. To this end, we propose an algorithm based on greedy programming that leverages the meta-data catalog of data fabric to efficiently determine training schedules based on data gravity, compliance, and resource availability. Our simulations based on synthetic data and resource distribution profiles demonstrate significant improvements in execution times and resource utilization compared to traditional DDL scheduling approaches in hybrid multi-cloud environments.**

***Index Terms*—Data Fabric, Heterogeneous Data Distribution, Workflow Scheduling, Data-intensive Workloads**

I. Introduction

Modern enterprises and research organizations have demonstrated rapid adoption of data-driven services, such as deep learning (DL) based recommendations, scientific exploration and experimentation, web and business-intelligence services, big-data analytics-based revenue projection and medical surveys, etc. The importance of DL can be characterized by the numerous breakthroughs it has achieved in a wide variety of domains, ranging from DL-based drug discovery and climate modeling, to stock volatility prediction and autonomous androids [1]. Such DL-based workloads use huge amounts of data, collected through various primary (e.g. IoT or edge devices, experimental equipment, etc.) and derived (e.g. transformed, pre-processed, extrapolated, etc.) data sources. The unprecedented pace and scale of data management required to fuel such DL workloads is a high-priority challenge for the systems research community.

Data stores (e.g. cloud-based file and object storage) have been extensively explored and optimized to manage large-volumes of data. However, efficient data management on such hybrid multi-cloud data store services entails challenges associated with data gravity, governance, compliance, discovery, provenance, and security [2]. The data fabric architecture aims to mitigate, and potentially eliminate these issues through a series of modular data management components. In this context, data virtualization is a key component for transparent and real-time access of data scattered across various data sources, i.e., different storage services (e.g. object stores, key-value, file storage, etc.) and cloud providers [3]. Apart from minimizing data engineering efforts, such as integration across different storage services, data movement, and enforcing compliance regulations, data fabric has led to faster, robust, and high-quality discovery services [3], [4].

Although in nascent stages of development, data fabric has powered large-scale enterprise and research explorations (e.g. Dow Jones, DreamWorks, AstraZeneca, etc.) [5], demonstrating a promising potential for rapid adoption. While the data virtualization in data fabric allows seamless processing of low-frequency small-sized data (e.g. distributed query processing), data-intensive workloads (e.g. scientific experiments, image classification based on DL, etc.), typically require access to large input datasets at sub-second intervals. In this context, distributed deep learning (DDL) workloads trained on datasets scattered across various storage services, consume data at high-frequency. Further exacerbating this issue are accelerators (e.g. Graphic Processing Units (GPUs)) optimized for DL execution and fast-track data consumption. Therefore, consuming data through the virtualization layer of data fabrics imposes application overheads, escalating the training time. Even if the data could be moved across datacenters transparently behind the virtualization layer, the compliance policies on data fabric may restrict such data movement.

A key observation in the context of data fabrics running on hybrid-cloud environments is that the amount of data and available computational resources (workers) on each cloud can be variable. For instance, an organization capturing the urbanization (using high-resolution drone imaging) of various geographies to determine strategic advertisement locations would have a higher number of images in metropolitan-dense continents. Similarly, scientific expeditions to find atmospheric oceans through DL-based image processing may feature high data accumulation in cloud-storage repositories that are closer to the regions of interest (e.g. hurricane-prone areas). Furthermore, the resource availability on each cloud may be limited either by (1) total compute machines available or (2) operational costs on each cloud service. A naive approach to address the contrasting objectives of data-consumption rate of DL workloads and compliance would be to migrate all data from various storage services to a central compliance-abiding processing location, e.g. in-house cloud center. While this approach respects the compliance policies, it suffers from two performance limitations that can potentially violate the service-level agreement (SLA) requirements of the workload: (1) the DDL training execution time would be inhibited by the limited number of processing units available at the central processing location, and (2) the I/O overhead (both in terms of time and cost ($)) to transfer data to a central location.

State-of-the-art data-locality-based workflow schedulers, such as Yarn [6] and Horovod [7], eliminate the transfer and compliance challenges by placing compute close to data sources. However, these approaches require specialized Development and Operations (DevOps) teams to map the workflow based on data size and resource availability on each cloud service. While currently used in practice, it obliterates the transparency and meta-data semantics of data virtualization provided by data-fabric, compelling application developers to deal with the complexity of data distribution, access control, and compliance. Therefore, existing DDL scheduling techniques are not optimized for transparent, compliance-aware, and resource-restricted data access through data fabric.

To address the limitations of existing DDL schedulers, we propose a set of design principles and algorithms for data fabric architectures based on hybrid multi-cloud environments. We summarize our contributions as follows:

1) We formulate the problem of compliance-aware, resource-constrained, data distribution-aware training of DDL workloads on data fabric architectures (§ II).
2) We propose a set of design principles and algorithms to optimize resource consumption and DDL training time which optimizes scheduling based on data fabric features such as transparent data access, data gravity, and compliance. Specifically, we propose a greedy-programming-based algorithm that leverages meta-data catalog to transparently and efficiently schedule DDL training on hybrid multi-cloud setups (§ IV).
3) We evaluate our approach using simulations on a series of synthetic traces and demonstrate performance and operational improvements introduced by our proposed approach as compared to the existing techniques (§ V).

II. Problem Formulation

We consider the case of a distributed deep learning (DDL) training workload, such that the training data is distributed in a hybrid multi-cloud setup across C different private and public cloud providers, and accessed concurrently during training. Each cloud consists of a variable number of worker resources. Using a series of transformations (e.g. image rotation, decimation, cropping, etc.), defined by the application developer, the data to be used during training is pre-processed during data ingestion, such that is available in readily consumable formats (e.g. tensor vector) on each cloud before the training starts. Furthermore, conforming to network and I/O SLAs provided by cloud vendors, we assume that the average latency and throughput across different clouds remain consistent throughout application execution, and is not shared by other workloads.

For the scope of this work, we limit the compliance characteristics only to *motility*, i.e., the data on a given cloud is either movable (*motile*) or non-movable (*non-motile*). During training, the DDL workload accesses data from both *motile* and *non-motile* cloud sources (e.g. data is *motile* on some geographic locations, whereas it is strictly *non-movable* on others). As opposed to micro-scale compliance management (e.g. selective redaction of personal information, homomorphic encryption based training, etc.) to make the dataset eligible for movement, we assume extreme inflexibility by marking entire data on given a cloud as either movable or non-movable. Such simplification of extensive compliance guidelines (e.g. GDPR [8]) to a single binary valued attribute, *motility*, allows us to study the performance implications on DDL workloads under exceptionally conservative compliance requirements.

To simplify the scope of this paper and emphasize the impact of our proposal, we make the following assumptions:

- All cloud providers consist of homogeneous worker nodes/resources, i.e. the compute, I/O, and memory capacity of all workers are identical.
- We assume that the DDL training time is significantly higher than the synchronization overhead after every epoch (e.g. using $\mathcal{O}(log(N))$ all-reduce for N workers [9])
- Compliance only corresponds to the motility of data, i.e. the data on a cloud is either motile (1) or non-motile (0).
- We assume that all clouds perform local storage-only shuffling, i.e., each cloud only shuffles the data that it had ingested and the data that was outsourced to it at the beginning of the execution from the overloaded peer clouds. Therefore, data is not shuffled across different clouds during training.
- Other applications do not inhibit the latency or throughput of any given cloud storage throughout execution.
- Each cloud service already contains the pre-processed data in ready-to-consume format on the fastest available storage service, such that the access latency of the local data across all clouds by each worker is identical.

Our goal is to efficiently schedule DDL training workloads on next-generation data fabric architectures comprising hybrid multi-cloud storage services. The key novelty of our proposal is the efficient and transparent scheduling of data-intensive DDL workloads in compliance-enforced, resource-constrained, and variable data distribution-based data fabric environments. Note that our proposed design principles can be easily extended to other data-intensive workloads running on hybrid-cloud setups.

III. Related Works

A. Data Fabric Architecture

Data fabric aims to address the challenges in hybrid multi-cloud data landscape [4]. Based on the reference data fabric architecture provided by IBM [2] (described by Forrester Research, Inc.), the data fabric consists of six major components working in tandem to perform efficient data management. The Open Data Fabric [3] implements a prototype for big-data processing. On an orthogonal trajectory, the M-Data-Fabric [4] optimizes knowledge map creation in the data fabric. However, none of these approaches study the impact of processing large-scale data and I/O-intensive workloads on data fabric architectures.

B. Data Compliance and Privacy

Various government, enterprises, and research agencies detail extensive guidelines for managing data life-cycle during storage, transfer, and processing. Compliance policies (e.g. GDPR [8]) impact a wide range of organizations operating world-wide across different application domains. This has compelled organizations to adopt smart data management solutions such as data fabric to provide self-managed universal compliance administration. However, current workflow schedulers and distributed runtime do not leverage the optimizations and abstractions provided by data fabric.

C. Workflow Scheduling on Distributed Resources

Workflow scheduling has been extensively studied across the systems research community [10]. Scheduling frameworks, such as Horovod [7], Yarn [6], etc. have been widely operated across organizations with multi-cloud setups [11], but lack automated data compliance and data gravity integrations. However, none of these frameworks leverage meta-data catalog of data fabric to efficiently schedule workflows across distributed resources.

To the best of our knowledge, we are the first to consider the problem of data gravity and compliance aware DDL training on resource constrained next-generation data fabric architectures running on hybrid multi-cloud environments.

IV. Proposed System Design

In this section, we propose the key design principles and algorithms to optimize DDL scheduling on hybrid multi-cloud setups constrained by compliance and resource limitations.

A. Dynamic Discovery of Data Sources using Meta-data

Existing DDL frameworks impose the burden of statically passing the data-sources during job execution on application-developers. To accomplish this, application-developers need to be aware of the underlying data distribution across various sources, and different data access mechanisms (e.g. cloud toolkit interface, authentication and authorization APIs, tokens, etc.). Further aggravating the developer burden are modification of data sources and access mechanisms, i.e. adding, removing, or modifying controls (change ownership, access method, etc.), which requires remapping of data sources and their corresponding control mechanisms before submitting the workload for execution. Although specialized development and operations (devops) team assist developers in solving these issues, the manual approach is still error-prone, complex and time-consuming. To solve these challenges, we propose a dynamic discovery abstraction that leverages the meta-data catalog of data fabric to adaptively map data source across hybrid multi-cloud setups. Our approach allows the application developer to provide a set of unique *labels* to describe the data required during training of their DDL workload. *Labels* are identifiers that get assigned to the data records during their ingestion process and are captured in the meta-data catalog. These labels can be queried during application runtime to discover the location and size of data residing across various cloud sources. In addition to aligning with transparent data access abstractions, this approach also allows dynamic modifications to data sources, resulting in simpler and faster application deployment times on data fabric.

B. Compliance and Gravity Aware Workload Distribution

Tightly coupled applications, such as DDL, that perform synchronization after every epoch, are constrained by the slowest worker process in a distributed training setup. For typical DDL training having uniform data distribution, determining and optimizing slow workers (stragglers) based on resource profiles (e.g. processing capability, memory, etc.) has been extensively studied [12]. However, on data fabric architectures, the uneven data distribution coupled with compliance regulations and limited resource availability necessitates a new workflow scheduling paradigm that leverages various components of the data fabric to optimize both job completion times and resource utilization. To this end, based on compliance requirements, resource profile, and data size, we determine the straggler worker using the metric *maximum batches processed per worker*. The clouds comprising of *non-motile* data compels the locally available workers to compute all available batches, i.e. the data on a given cloud is evenly distributed across the workers available on that cloud. On the other hand, *motile* data based cloud providers can move data to resource-rich clouds, i.e. clouds having spare workers, to complete training faster.

Once we compute the maximum data-load per worker, we attempt to employ limited resources, such that every worker processes exactly *maximum batches processed per worker* (last worker may be underutilized when data cannot be split into equal batches). Lastly, the data from overloaded *motile* cloud servers can be transferred to under-loaded *motile* cloud servers considering that DDL training time is higher than the data transfer time. We detail this approach in § IV-C.

C. Greedy-Programming Based Compliance Aware Workload Distribution Strategy

In this subsection, we zoom on our greedy-programming based algorithm that leverages data fabric meta-data to efficiently schedule workloads across hybrid multi-cloud setups.

Our algorithm is based on the intuition that all the workers distributed across different clouds should be equally engaged even if the data distribution is uneven. This ensures that all the workers perform synchronization at epoch boundaries simultaneously and underutilized workers do not wait for stragglers after every epoch. Using a consistent number of

Algorithm 1: Greedy-programming based compliance aware workload distribution strategy in resource constrained hybrid multi-cloud environments

```
Input  : list of cloud meta-data $C$ containing tuples
         $\langle b, m, w, T \rangle$, each representing total batches $b$,
         motility $m$ (0 or 1), max. available workers $w$, and
         list of transfer rates to remote clouds $T$; per batch
         execution time per worker $t_b$
Output: list of workload distribution plan $P$ with tuples
         $\langle src, dest, workers, train\_time, transfer\_time \rangle$
   // Helper to filter motile/non-motile from C
1  Function filter(motile):
2      return $c \in C \mid c.m == motile$
3  $non\_motile\_bpw \leftarrow \max_{\texttt{filter}(false)} (c.b/c.w)$
4  $motile\_batches \leftarrow \sum_{\texttt{filter}(true)} c.b$
5  $motile\_workers \leftarrow \sum_{\texttt{filter}(true)} c.w$
6  $motile\_bpw \leftarrow motile\_batches/motile\_workers$
   // compute maximum batches processed per worker
7  $bpw \leftarrow max(non\_motile\_bpw, motile\_bpw)$
   // assign maximum required local workers
8  for $c \in C$ do
9      $workers \leftarrow min(c.w, c.b/bpw)$
10     $c.w \leftarrow c.w - workers$
11     $c.b \leftarrow c.b - bpw \cdot workers$
12     $P \leftarrow P \cup \{(i, i, workers, bpw \cdot t_b, 0)\}$
   // greedily transfer and schedule from heavily
      overloaded cloud to fastest peer cloud
13 while $\exists i = argmax(C[i].b \mid C[i].b > 0)$ do
14     while $\exists j = argmax(C[j].w * C[i].T[j])$ and
         $C[i].b > 0$ do
15         $workers \leftarrow min(C[j].w, C[i].b/bpw)$
16         $C[i].b \leftarrow C[i].b - bpw \cdot workers$
17         $C[j].w \leftarrow C[j].w - workers$
18         $P \leftarrow P \cup \{(i, j, workers, bpw \cdot t_b, C[j].b/C[i].T[j])\}$
19 return $P$
```

batches per worker achieves better (§ V) resource utilization and minimum training time.

As described in the design principles, we exploit the data fabric meta-data to generate an optimal transfer and training schedule. To this end, during the workload submission, the application developer simply provides the training script (that can adapt to scheduled number of workers and inputs, e.g. using lambda functions), the data *labels*, the batch size bs, and total number of epochs $epochs$ as input to our scheduler. After querying the data sources through meta-data catalog, we compute the number of batches on each cloud by dividing the input data into b groups of bs pre-processed data items each (§ II assumes that data is pre-processing during ingestion). Based on the number of $epochs$, batch size bs, and the profile of available workers (§ II assumes homogeneous workers across all clouds), we estimate the execution time per batch per worker t_b. To estimate t_b we can make a short run of the workload in one worker, as all workers are homogeneous it should be good enough.

Next, for each cloud we derive the motility, number of workers available, and transfer throughput to peer clouds by using the meta-data, and/or real-time querying of workers/throughput across different clouds. We consider *batch* as the standard unit in our approach, i.e., the transfer rate and training rate are expressed in `batches/second`. Note that the application simply provides the application attributes as input, and the training schedule is dynamically determined by our meta-data discovery and compliance-aware techniques.

In our design, we consider two categories of clouds, based on the binary value of the *motility* attribute to enforce compliance (as described in § II). As outlined in Algorithm 1, we first determine the maximum *batches processed per worker* (`bpw`) to identify the workers tasked with processing the largest amount of data, i.e. straggler workers. Since the job completion time would be determined by these straggler workers, we employ the `bpw` heuristic to minimize resource utilization.

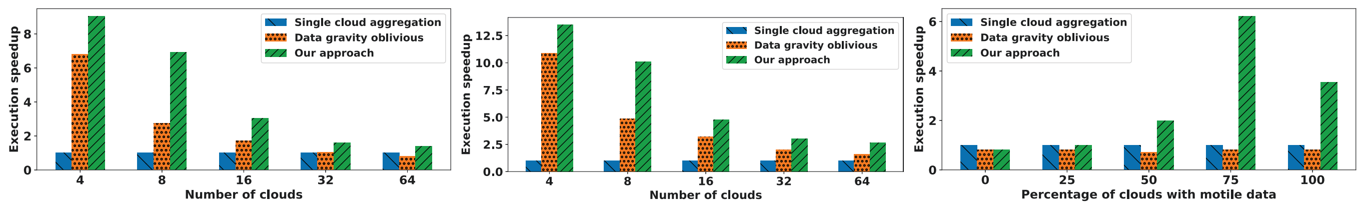
We first compute the `bpw` for *non-motile* cloud servers which cannot move their data to other clouds. The `non_motile_bpw` is dictated by the *non-motile* cloud which has the highest batches to worker proportion (Line 3 of Algorithm 1). However, since data can be moved in the case of *motile* clouds, the maximum load per worker `motile_bpw` is determined by evenly distributing the total data across motile clouds amongst the workers (Line 6). Finally, we determine the maximum `bpw` based on the maximum load across *motile* or *non-motile* cloud providers (Line 7).

Once the maximum batches processed per worker is determined, we assign each worker an identical number of batches to processes. Taking into consideration data locality, we first assign the *maximum amount of batches per worker* to cloud-local workers for both *motile* and *non-motile* categories (Lines-8-12 of Algorithm 1). Next, we adopt a greedy approach to distribute the data from overloaded *motile* clouds to underutilized peer clouds. We first find the *motile* clouds with highest remainder data (Line 13), i.e., workers with more than maximum `bpw` share, and transfer it to the peer cloud with highest processing rate. The peer cloud with highest processing rate is computed by a combination of two factors, namely, number of spare workers, and transfer throughput per worker (Line 14). Note that we consider that each worker from the underutilized *motile* cloud can transfer data from over-utilized cloud in parallel. Once the most suitable peer cloud has been determined, we allocate the maximum amount of workload to the peer *motile* clouds greedily until there is no residual data left on the clouds with excess of data (Line 15-18). Once the plan P is determined (Line 19), we compute the total training time using $max_{p \in P}(p.train\_time + p.trf\_time)$, i.e., the maximum time taken by any cloud to perform training and transfers.

V. Experimental Evaluations

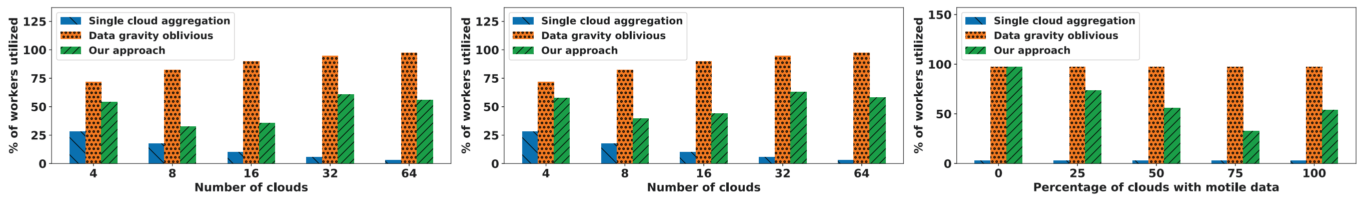
A. Evaluation Methodology

We implement our proposed data gravity and compliance aware workload scheduler based on the design principles and algorithm described in section IV in a simulation framework written in Python. The framework takes as input both the user and system level attributes of the DDL job, i.e., batch size and training time per batch per worker, and meta-data



(a) Random amount of data and workers (b) Homogeneous data sizes, random workers (c) Variable amount of motile clouds

Fig. 1: Execution speedup normalized to single client aggregation approach. Higher is better.



(a) Random amount of data and workers (b) Homogeneous data sizes, random workers (c) Variable amount of motile clouds

Fig. 2: Proportion of workers utilized by the three compared approaches for DDL training. Lower is better.

from data fabric which contains batches, available workers per cloud service, and transfer rate across different clouds. The output of the simulator is the workflow scheduling plan, total execution time, and proportion of resources consumed across hybrid multi-cloud data centers.

B. Compared Approaches

Across all our evaluations, we compare the following approaches to perform compliance aware data-intensive DDL training using our DDL schedule simulation framework.

1) *Single cloud aggregation (Baseline):* This approach implements a naive technique in which the pre-processed data from across all public clouds are pulled to a single private (on-premise) cloud, which is compliance aware. While this defies *data-locality* based scheduling, it is relatively simple to implement, deploy, and can be efficiently used for purposes such as debugging, real-time analytics, etc., and hence is considered as the baseline for our evaluations.
2) *Data gravity oblivious (Conventional):* Complementary to the previous approach, in this approach, we attempt to launch maximum number of resources available on each cloud, such that the cloud-local data is processed by the cloud-local workers. While this enables faster training by eliminating data transfers, it also suffers from sub-optimal resource utilization across various cloud due to variable batches processed per worker. However, for data gravity oblivious developers or DevOps experts, this is a straightforward and practical technique, especially when used with distributed training strategies such as TensorFlow's `MultiWorkerMirroredStrategy`. This approach utilizes all workers available across all clouds, and represents the current practices for scheduling DDL training on multi-cloud environments using state-of-the-art distributed workflow frameworks.
3) *Our approach:* This approach implements the workflow scheduling strategy based on greedy-programming, as described in Algorithm 1.

Note that the *data gravity oblivious* and *our approach* do not employ the workers of private cloud (as done by the *single cloud aggregation* approach, and assume that those can be used for supporting development workloads, or are reserved for mission-critical latency sensitive workloads.

C. Key Evaluation Metrics and Simulation Traces

For each of the aforementioned approaches compared in our evaluations, we represent two metrics for evaluation, namely *Execution speedup*, and *Proportion of resources utilized*. The *execution speedup* demonstrates the speedup achieved by *data gravity oblivious* and *our approach* as compared to the *single cloud aggregation* approach. This metric represents the total time taken for training and data transfers across all workers available in multiple hybrid clouds and is representative of performance gains achieved by our approach. The second metric *Proportion of resources utilized* is important from an operational perspective, both to maximize worker throughput and minimize training cost ($).

We generate synthetic cloud profiles C for hybrid multi-cloud setups using Python's pseudo-random number generator. For each cloud, the corresponding meta-data catalog is generated using uniform distribution on the following ranges: batches (per cloud) $[1000, 20000]$, workers (per cloud) $[10, 200]$, motility (per cloud) $[0, 1]$, and transfer rate (in batches transferred per second) to peer clouds $[1, 8]$. Note that each attribute is a non-negative integer across the inclusive data ranges. Due to limited space, we omit fine-grained trace generation details, our motivation for deriving various configurations, and corresponding ablation study of each attribute.

D. Results

1) Normalized Execution Speedup: Our first set of experiments evaluates the execution speedup of our approach as compared to the *single cloud aggregation* and *data gravity oblivious* approaches as a series of scalability analysis for an increasing number of clouds. As observed in Figure 1a and Figure 1b, our approach outperforms the two compared approaches even at scale for the scenarios corresponding to fully random configuration and homogeneous data distribution. More specifically, our approach is 12.5× to 2.4× faster than the baseline for an increasing number of clouds. This is because our approach employs more number of workers at scale, increasing the total amount of worker-hours invested, as opposed to single cloud aggregation based which performs expensive transfers from various clouds, but operates using limited on-premise workers, thereby having lower worker-hours spent on the given DDL workload. As expected, the *data gravity oblivious* approach is slower than our approach since it misses on the opportunity to share data from over-utilized to under-utilized workers across *motile* clouds.

2) Proportion of Workers Utilized: Our next set of experiments study the resource efficiency of our scheduling approach. As depicted in Figure 2a and Figure 2b our approach utilizes on average 55% of available resources across all clouds. Since the *single cloud aggregation* approach only utilizes on-premise workers, the number of resources used is consistently low for all configurations across varying number of clouds. This represents the scenario of resource under-utilization, since by consuming 20% lesser available resources as compared to our approach, it results in 12.5× slower execution (correlating with Figure 1a). On the other end, the *data gravity oblivious* approach utilizes all the available public cloud resources (85% average utilization), but still delivers about 1.8× slower execution speedup as compared to our greedy-programming based approach. Therefore, our approach performs efficient worker utilization while delivering significant execution speedup at scale.

3) Variable Fraction of Clouds with Motile Data: Lastly, we study the impact of varying proportion of clouds containing motile data. In this experiment we use a configuration consisting of 64 clouds and we incrementally designate 25% of the available clouds as motile, starting from 0% motile clouds, i.e., no cloud is eligible to move its data. As seen in Figure 1c and Figure 2c, with increasing motility our approach demonstrates an increasing execution speedup and decreasing worker consumption, respectively. For the case when a low proportion of clouds contain motile data, i.e. most of the data is immovable, we have fewer opportunities to perform load-balancing using greedy approach, and relatively more workers utilized as compared to the baseline approach due to which we see execution slowdown for 0% and 25% motile clouds. *Data gravity oblivious* approach proves ineffective throughout different motility configurations for both execution speedup and percentage of workers utilized due to sub-optimal workload scheduling.

VI. Conclusions

In this work, we study the problem associated with scheduling data-intensive workloads, in particular distributed deep learning (DDL) on data fabric architectures. These DDL training workloads require low-latency access to data residing across various hybrid multi-cloud environments, while following compliance regulations imposed by various agencies and organizations. While next-generation data fabric architectures address these limitations, existing DDL and workflow schedulers are not sufficiently optimized to leverage data fabric. To this end, we propose a greedy-programming based algorithm to efficiently determine optimal workflow execution plan on a limited set of resources. We exploit the well-defined pipeline and meta-data catalog of modern data fabrics to perform dynamic discovery of data and, compliance and data gravity aware workload scheduler. Our evaluations based on synthetic traces demonstrate up to 12.5× and 2.4× faster execution as compared to aggregation-based and data gravity oblivious scheduling approaches, respectively.

In the future, we plan to optimize various characteristics of our proposal, e.g. heterogeneous worker profiles, comprehensive criteria for compliance regulations, and optimization of our greedy-programming based algorithm. We will also extend our proposal to real-life workload schedulers and DDL frameworks to empirically determine the impact of our proposal on data fabric architecture.

Acknowledgment

This work is supported in part by the National Science Foundation (NSF) under Awards No. 2106634 and 2106635. Results presented in this paper are obtained using the Chameleon and CloudLab testbeds supported by the NSF.

References

[1] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, 2021.

[2] "What is a data fabric? — IBM." [Online]. Available: https://www.ibm.com/topics/data-fabric

[3] S. Mikhtoniuk and O. N. Yalcin, "Open data fabric: A decentralized data exchange and transformation protocol with complete reproducibility and provenance," *arXiv preprint arXiv:2111.06364*, 2021.

[4] K. Liu, M. Yang, X. Li, K. Zhang, X. Xia, and H. Yan, "M-data-fabric: A data fabric system based on metadata," in *International Conference on Big Data and Artificial Intelligence (BDAI)*, 2022.

[5] Netapp, "Data fabric - a unified cloud data management solution." [Online]. Available: https://www.netapp.com/data-fabric/

[6] 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 *Symposium on Cloud Computing (SoCC)*, 2013.

[7] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[8] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, Cham: Springer International Publishing*, 2017.

[9] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li, "Communication-efficient distributed deep learning: A comprehensive survey," *arXiv preprint arXiv:2003.06307*, 2020.

[10] M. Menaka and K. S. Kumar, "Workflow scheduling in cloud environment–challenges, tools, limitations & methodologies: A review," *Measurement: Sensors*, 2022.

[11] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR)*, 2019.

[12] M. Arif, M. M. Rafique, S.-H. Lim, and Z. Malik, "Infrastructure-aware tensorflow for heterogeneous datacenters," in *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2020.