OneDataShare: Data Transfer Optimization as a Service

Improvements and Support of Additional Features

Vamshi Krishna Kyatham
Department of Computer Science and Engineering
University at Buffalo
Buffalo, New York, United States
vkyatham@buffalo.edu

Tevfik Kosar
Department of Computer Science and Engineering
University at Buffalo
Buffalo, New York, United States
tkosar@buffalo.edu

Jacob Goldverg
Department of Computer Science and Engineering
University at Buffalo
Buffalo, New York, United States
jacobgol@buffalo.edu

***Abstract*— Fast, efficient, and reliable data transfer across wide-area networks remains a critical bottleneck for data-intensive cloud applications. OneDataShare (ODS) addresses this challenge by enabling high-speed, protocol-agnostic data transfers between heterogeneous endpoints. This paper presents how ODS is enhanced through advanced techniques such as Bayesian Optimization and Deep Deterministic Policy Gradient (DDPG), which intelligently tune transfer parameters to maximize throughput.**

I. Introduction

With the increasing demand for efficient, reliable, and high-throughput data transfers across heterogeneous systems, traditional tools often fall short in performance and usability. Cloud computing services and scientific collaborations require flexible and high-speed solutions that also guarantee security and user accessibility. OneDataShare aims to fulfil this demand by acting as a universal, protocol-independent data transfer optimization service hosted in the cloud.

The core goals of OneDataShare include:

- Optimization of end-to-end data transfers
- Interoperation across heterogeneous storage systems via protocol translation
- Prediction of data delivery times to support real-time decisions

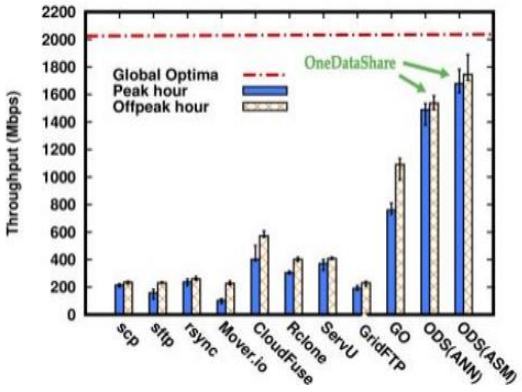

Figure 1: Performance comparison between previous work of ODS along with different data transfer services

This paper documents the enhancements made in optimization, authentication, and visualization aspects of ODS. We build on earlier optimization models such as HARP, ProMC, and ANN-based tuning, and introduce reinforcement learning and probabilistic modeling (Bayesian Optimization) to further improve parameter selection under dynamic conditions. Furthermore, the platform's security has been extended through the implementation of federated login options via Google, GitHub, and CILogon using OAuth2/OIDC.

II. Background and Related Work

OneDataShare (ODS) brings transformative improvements to the landscape of cloud-based data transfers through its emphasis on elasticity, sustainability, and user-centric design. Its architecture is engineered with key performance-critical features that not only enhance the system's efficiency but also ensure its practical adoption by a wide range of users.

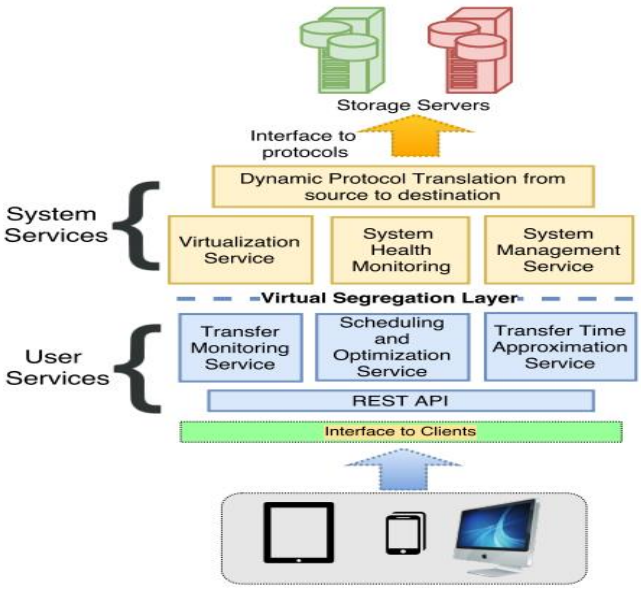

Figure 2: OneDataShare high-level overview

OneDataShare is structured to offer a seamless and optimized data-sharing experience through a modular yet cohesive design. As illustrated in Figure 2, it incorporates a variety of core services such as protocol translators, schedulers, provenance managers, and cloud managers. These components collectively operate within the backend infrastructure while remaining transparent to the user, appearing as a black-box interface. When a user submits a data transfer request via a RESTful API, OneDataShare processes the request through its internal engine.

The platform supports diverse client interfaces including web portals, mobile and tablet applications, command-line tools, and file system integrations allowing flexibility in how users interact with the service. A thin-client architecture ensures that all complex operations, including protocol translation, transfer optimization, and scheduling, occur in the cloud. This not only guarantees reliability and high availability but also provides cost-effective scalability, as users are charged only based on their specific resource usage.

System-level monitoring services oversee the performance and health of internal components, maintaining transparency and stability for end-users. These architectural choices reflect a strong alignment with both technical demands and usability expectations, positioning OneDataShare as a ubiquitous and sustainable solution for modern data transfer needs.

Several earlier models have informed the development of OneDataShare. HARP utilized historical data and real-time probing to predict optimal parameter values. ProMC extended this with data-centric adaptive tuning. ASM/ANN-based models employed supervised learning to predict parameter settings for transfer optimization by using offline adaptive sampling techniques (As shown in Figure 1).

However, these methods, while effective, are limited by their reliance on either static tuning or non-adaptive heuristics. Moreover, traditional systems often required complex user configurations and lacked support for federated authentication, reducing usability and accessibility.

OneDataShare overcomes these limitations by using intelligent sampling techniques and real-time learning, as well as integrating secure, modern authentication mechanisms.

III. METHODOLOGIES

INTELLIGENT TRANSFER PARAMETER OPTIMIZATION

Transfer performance in ODS is governed by three main parameters:

- **Parallelism**: Number of parallel threads per file
- **Concurrency**: Number of files transferred simultaneously
- **Pipelining**: Number of pipelined requests

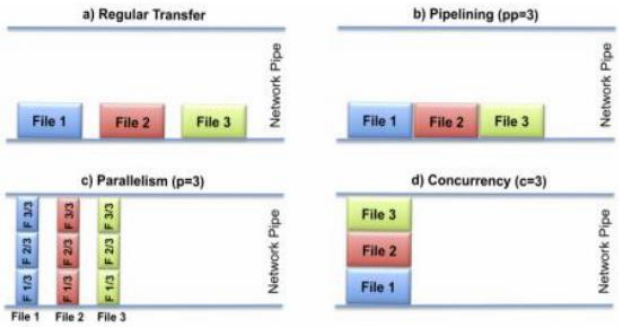

Figure 3: Protocol Parameters tuned in the data transfer

To find optimal combinations of these parameters dynamically, we employed two advanced methods:

3.1 **Bayesian Optimization**:

Bayesian Optimization (BO) is a powerful technique for optimizing black-box functions that are expensive to evaluate. In our case, the objective function represents the negative throughput of data transfers for a given set of parameters. Since we want to maximize throughput,

$f(x) = -Throughput(x)$.

Step-by-Step Process:

1. **Initialization**: Begin with a small number of randomly selected sample points $\{x_1, x_2, ..., x_n\}$ from the parameter space $\mathcal{X}$, and evaluate the objective function to get $\{y_i = f(x_i)\}$ for i = 1 to n.
2. **Surrogate Model Construction**: Use the sampled data $D_n = \{(x_i, y_i)\}$ to construct a surrogate model $\hat{f}(x)$, typically a Gaussian Process (GP):

   $\hat{f}(x) \sim GP(\mu(x), k(x, x'))$

   Where:

   $\mu(x)$ is the mean function (often assumed to be 0)

   $k(x, x')$ is the covariance or kernel function (e.g., RBF or Matern kernel)

   The GP gives a predictive distribution at any point x:

   $\hat{f}(x) \sim N(\mu_n(x), \sigma_n^2(x))$
3. **Acquisition Function Optimization**: Choose the next point x_{n+1} to evaluate by optimizing an acquisition function $\alpha(x; D_n)$ that balances exploration and exploitation.

   A common acquisition function is Expected Improvement (EI):

   $\alpha\_EI(x) = E[max(f\_best - \hat{f}(x), 0)]$

   This can be computed as:

   $\alpha\_EI(x) = (f\_best - \mu_n(x)) * \Phi((f\_best - \mu_n(x)) / \sigma_n(x)) + \sigma_n(x) * \varphi((f\_best - \mu_n(x)) / \sigma_n(x))$

   Where:

   f_best is the best observed value so far

   Φ is the cumulative distribution function (CDF) of the standard normal distribution

   ϕ is the probability density function (PDF) of the standard normal distribution
4. **Surrogate Model Update:**

   Evaluate $f(x_{n+1})$, update the dataset:

   $D_{n+1} = D_n \cup \{(x_{n+1}, f(x_{n+1}))\}$

   Then update the GP model using the new data.
5. **Termination:**

   Repeat steps 3 and 4 until one of the following criteria is met:

$max\ |\mu_{n+1}(x) - \mu_n(x)| < \varepsilon$ *(Convergence)*

$max\ \sigma_n(x) < \delta$ *(Uncertainty threshold)*

Number of function evaluations $\geqslant N\_max$ *(Budget limit)*

Implementation Note:

We use the skopt.gp_minimize function from scikit-optimize to implement this Bayesian Optimization process.

Application to ODS:

- The optimizer proposes values for concurrency and parallelism.
- These are used to execute data transfer jobs.
- The throughput values are smoothed using Exponential Moving Average (EMA), and the negative EMA throughput value is used as the objective. This smoothing approach ensures that short-term spikes do not mislead the optimization process and allows the model to focus on stable trends.
- The surrogate GP model predicts performance for untested configurations, and the acquisition function identifies the most promising next step.
- This iterative refinement enables BO to efficiently discover parameter combinations that maximize throughput with minimal overhead.

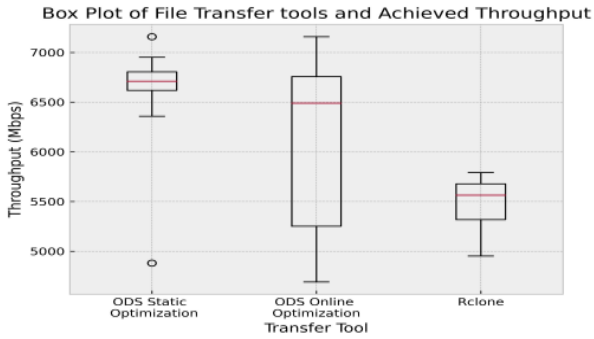

Figure 4: Throughput variation using Bayesian optimization compared to previous approaches.

As shown above, Bayesian Optimization helps in achieving high throughput.

3.2. Reinforcement Learning:

Deep Deterministic Policy Gradient (DDPG):

The Deep Deterministic Policy Gradient (DDPG) algorithm is a model-free, off-policy reinforcement learning method designed for continuous action spaces. In the context of OneDataShare, DDPG is used to dynamically tune transfer parameters such as concurrency and parallelism to maximize throughput based on observed network conditions.

DDPG maintains two neural networks:

- **Actor Network (*μ(s|θμ)*)**: Determines the best action (i.e., parameter values) to take given the current state (i.e., network conditions).
- **Critic Network (*Q(s, a|θQ)*)**: Evaluates the quality (expected reward) of a given action in a specific state.

To stabilize learning, DDPG uses **target networks** Q' and μ', which are delayed copies of the main networks and are updated slowly over time.

Training Loop:

1. Observe the current state s (e.g., network metrics like bandwidth and latency).
2. Use the actor network to select an action a = μ(s).
3. Execute the action (initiate a data transfer) and observe the reward r (measured throughput) and the next state s'.
4. Store the experience tuple (s, a, r, s') in a replay buffer.
5. Sample a mini-batch of experiences from the buffer to train both networks:
    - **Critic Update**: Minimize the loss: $L = (r + \gamma Q'(s', \mu'(s')) - Q(s, a))^2$
    - **Actor Update**: Maximize the objective: $J = E[Q(s, \mu(s))]$
6. Update the target networks:
    - $\theta Q' \leftarrow \tau\theta Q + (1 - \tau)\theta Q'$
    - $\theta\mu' \leftarrow \tau\theta\mu + (1 - \tau)\theta\mu'$ where τ is a small constant that controls the rate of update.

Application in ODS:

- **State Space:** Includes real-time network metrics such as RTT, available bandwidth, and packet loss rate.
- **Action Space:** Continuous values corresponding to concurrency and parallelism levels.
- **Reward Function:** Throughput achieved from the transfer, optionally smoothed using Exponential Moving Average (EMA).

DDPG empowers the ODS optimization engine to continually learn and adapt to changing network conditions, thereby improving the overall efficiency and reliability of data transfers.

IV. SYSTEM ARCHITECTURE

OneDataShare is composed of multiple Spring Boot-based microservices that interact to support secure, scalable, and optimized data transfers:

1. ODS API: This is the gateway service that exposes public APIs for user registration, login, transfer initiation, and metadata viewing. It acts as the main interface between users and the platform.
2. Optimization Service: This FastAPI based microservice hosts implementations of various optimization algorithms, including Bayesian Optimization and DDPG, to dynamically tune or schedule data transfers. It primarily focuses on maximizing throughput and is being extended to support carbon-aware data transfer strategies.

3. Endpoint Credential Service: Responsible for securely storing and managing credentials for user-specified servers and cloud storage endpoints using Hashicorp Vault. It ensures secure communication over TLS/SSL and keeps authentication tokens refreshed automatically.
4. Transfer Scheduler: Utilizes Hazelcast for distributed coordination of upcoming file transfers. This service enables real-time, decentralized scheduling decisions, supporting nodes that are behind firewalls or restrictive networks through encrypted communication.
5. Monitoring Service: Collects and maintains comprehensive telemetry on file transfers including throughput, latency, memory usage, CPU utilization, and energy consumption. Time series data is stored in InfluxDB, while CockroachDB holds transfer metadata and file operation logs.
6. Transfer-Service: A Spring Batch microservice responsible for performing the actual data transfers to and from cloud or server endpoints. It supports dynamic connections and configurable threading via API controls.
7. ODS Connector: A user-specific, containerized version of the file transfer service, typically deployed via Docker. It is isolated per user and provides a secure and private data transfer environment.
8. ODS CLI: A CLI with SDK that interacts with the ODS core to provide a replacement to the UI

This microservice architecture supports scalability, modular development, and robust performance for handling large-scale, optimized, and secure data transfers in distributed environments.

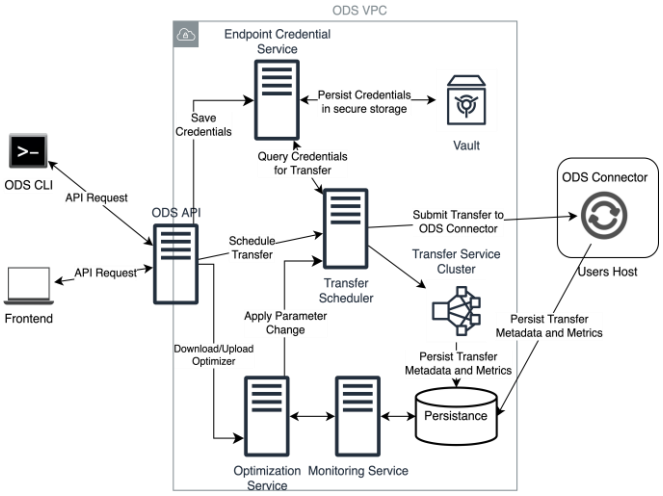

Figure 5: System Architecture of OneDataShare

V. FILE TRANSFER PROTOCOLS SUPPORTED

OneDataShare is designed to function as a protocol-agnostic data transfer platform, enabling interoperability across diverse storage systems and endpoints. It supports on-the-fly protocol translation and seamless data movement between heterogeneous systems. Currently supported protocols include:

FTP (File Transfer Protocol): Traditional and widely used for transferring files over TCP/IP.

SFTP (SSH File Transfer Protocol): Secure version of FTP, leveraging SSH for encrypted file transfer.

HTTP/HTTPS: Enables file transfers over standard and secure web protocols.

S3 (Amazon Simple Storage Service): Allows interaction with Amazon S3-compatible cloud storage services.

Dropbox: Supports transfers to and from Dropbox cloud storage.

Box: Integration for enterprise-grade cloud storage via Box.

VFS (Virtual File System): Abstracts over multiple backend storage types, providing a unified interface.

Google Drive: Direct integration with Google Drive for cloud-based file transfers.

This multi-protocol support, combined with dynamic translation capabilities, allows users to transfer data between any combination of the above systems with minimal configuration, significantly improving flexibility and usability in complex workflows.

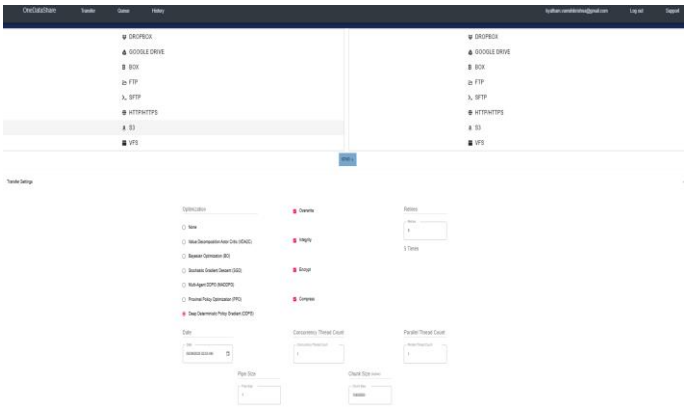

Figure 6: Various File transfer protocols supported by OneDataShare

VI. SECURITY ASPECTS OF ONEDATASHARE

To enhance the usability and security of ODS, we implemented federated authentication using social and organizational login services. The login system supports:

- **Google (OIDC)**
- **GitHub (OAuth2)**
- **CILogon (OIDC with Institutional login)**

4.1 Architecture and Implementation:

Using Spring Security, the backend was extended to support OAuth2/OIDC by configuring a custom SecurityFilterChain

bean. This method first added support for the OAuth2 login flow alongside the default form login method. It specified:

- The base URI /oauth2/authorization/{provider} for initiating the login
- The redirection endpoint /oauth2/callback/* to handle responses from providers
- A custom OAuth2AuthorizationRequestRepositoryCookie to store authorization requests in browser cookies for retrieval post-redirect

Both custom OAuth2/OIDC services extract user data using an abstract class and concrete implementations map provider-specific (Google, GitHub, CILogon) attribute keys.

Authentication success is handled by a custom success handler, which issues a JWT token and stores it in cookies for frontend access. Authentication failures are handled by custom failure handler, which redirects users with appropriate error messages.

A custom TokenAuthenticationFilter ensures that both ODS-native and third-party tokens (with ATOKEN and BEARER headers respectively) are handled appropriately before reaching the authentication context.

4.2 Security Considerations

- Authorization Code Grant was used for all OAuth2/OIDC flows to ensure tokens are never exposed in URLs
- JWT tokens are issued upon successful authentication and stored securely
- Clients are configured using InMemoryClientRegistrationRepository with sensitive information loaded from application-prod.properties
- Account conflict resolution logic prevents users from accidentally creating duplicate accounts across login providers

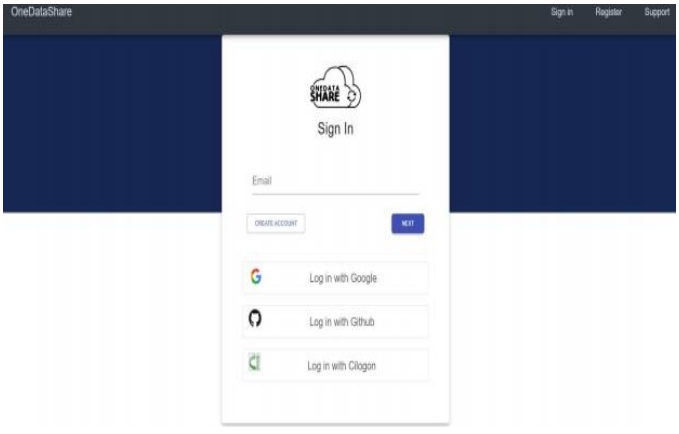

Figure 7: Landing page of OneDataShare with various secure login features

This security mechanism enhances both the accessibility and trustworthiness of OneDataShare, enabling seamless login experiences while ensuring robust protection of user data.

VII. OneDataShare Metrics Visualization Tool

To support in-depth analysis and monitoring of data transfer behavior, OneDataShare integrates a dynamic Metrics Visualization Tool. This web-based dashboard provides a user-friendly interface to track and interpret key performance indicators for each data transfer job submitted through the platform.

Key Features:

Job Selector Dropdown: Users can select from a list of transfer jobs they have triggered. Each entry corresponds to a unique job ID, allowing for targeted inspection and analysis.

Time-Series Visualization: Performance metrics are visualized per epoch, enabling detailed inspection of job performance over time. Each epoch represents a time interval during the transfer operation.

Metrics Tracked:

Parallelism: Displays the number of parallel threads used during each phase of the transfer, helping users understand how data was distributed across threads.

Concurrency: Shows how many files were transferred simultaneously, illustrating the system's ability to handle multi-file workloads.

Throughput: Real-time and historical throughput (e.g., in MBps), including smoothed trends using Exponential Moving Average (EMA), to highlight consistent performance and detect anomalies.

Loss: Model's prediction error during training (Reinforcement Learning algorithms).

This dashboard supports interactive visualization, allowing users to hover over data points for exact values and compare trends across epochs. It leverages File System/S3 for storing the metrics data.

By providing granular insights into transfer parameters and system behavior, the Metrics Visualization Tool empowers users to make informed optimization decisions, debug issues, and validate the impact of changes in configurations or environments.

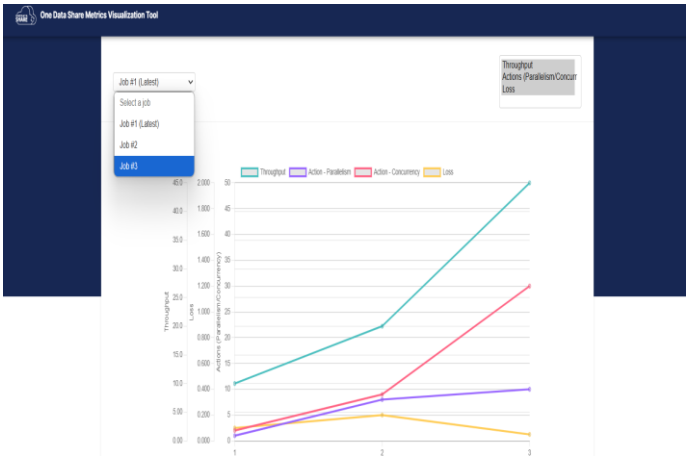

Figure 8: OneDataShare Metrics Visualization Tool

VIII. CONCLUSION AND FUTURE WORK

We have presented our vision for improved OneDataShare as a cloud-hosted data transfer scheduling and optimization service, designed to relieve users from the complexities of managing end-to-end data transfers. OneDataShare aims to provide a simple, intuitive platform that supports data sharing regardless of file size, type, or protocol differences between sender and receiver.

Through its modular microservices architecture, support for diverse transfer protocols, real-time optimization via Bayesian Optimization and Deep Deterministic Policy Gradient (DDPG), and seamless federated authentication, OneDataShare represents a robust and scalable solution for modern data-intensive workflows. The integration of a dynamic Metrics Visualization Tool further enhances transparency and operational insight, helping users fine-tune performance and diagnose transfer bottlenecks.

As part of our ongoing research, we are exploring additional reinforcement learning algorithms such as Proximal Policy Optimization (PPO) to further improve parameter tuning and adaptability. Building OneDataShare into a more reliable, scalable, and sustainable ecosystem that can benefit a broader range of users and scientific communities remains a core area of future development.

REFERENCES

[1] Engin Arslan and Tevfik Kosar. High-speed transfer optimization based on historical analysis and real-time tuning. IEEE Transactions on Parallel and Distributed Systems (TPDS), 29(6):1303–1316, 2018

[2] Engin Arslan, Bahadir A Pehlivan, and Tevfik Kosar. Big data transfer optimization through adaptive parameter tuning. Journal of Parallel and Distributed Computing

[3] MD SQ Zulkar Nine, Kemal Guner, Ziyun Huang, Xiangyu Wang, Jinhui Xu, and Tevfik Kosar. Big data transfer optimization based on offline knowledge discovery and adaptive sampling. In Proc. of IEEE Big Data 2017, IEEE International Conference on, pages 465–472. IEEE, 2017

[4] H. Jamil, L. Rodolph, J. Goldverg and T. Kosar, "Energy-Efficient Data Transfer Optimization via Decision-Tree Based Uncertainty Reduction," 2022 International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 2022, pp. 1-10, doi: 10.1109/ICCCN54977.2022.9868866.

[5] H. Jamil, E. Rodrigues, J. Goldverg and T. Kosar, "Learning to Maximize Network Bandwidth Utilization with Deep Reinforcement Learning," GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 2023, pp. 3711-3716, doi: 10.1109/GLOBECOM54140.2023.10437507.

[6] Asif Imran, Md S Q Zulkar Nine, Kemal Guner, and Tevfik Kosar, “OneDataShare: A Vision for Cloud-hosted Data Transfer Scheduling and Optimization as a Service”