Turbocharging Dask Apps

Accelerating Data Flow with ProxyStore

Klaudiusz Rydzy
Loyola University Chicago

Greg Pauloski (Advisor)
University of Chicago

Kyle Chard (Advisor)
University of Chicago

SC24 ATLANTA NOV 17-22

globus labs

1 ABSTRACT

- Despite advancements in parallel computing libraries, performance challenges such as data serialization and transfer overheads still persist
- We focus on understanding data limitations within Dask, a versatile and popular Python library designed for distributed and parallel computing
- We investigate using the pass-by-proxy paradigm (where data is not directly transferred between nodes) implemented by ProxyStore to address these limits
- By integrating ProxyStore, we streamline data flow in Dask applications, reducing overheads associated with data serialization and scheduler overheads
- Our approach evaluates the impact of proxies on data transfer times and overall computational efficiency in synthetic experiments and machine learning applications, leading to a 5-6x reduction in task overheads

2 MOTIVATION

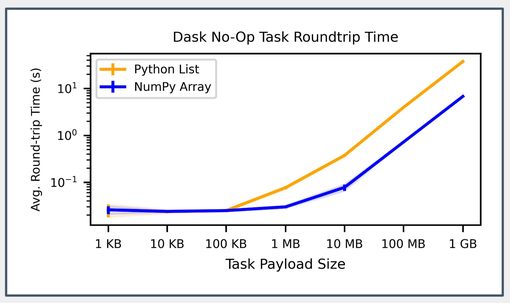
Why use Dask?

Pros:

- Popular for scientific computing
- Intuitive interface for parallel computing
- Integration with PyData stack
- Parallelize across CPUs and GPUs

Cons:

- MessagePack (used by Dask) causes performance bottlenecks
- MessagePack is slow to serialize bytestrings larger than 1 MB
- MessagePack cannot handle bytestrings over 4 GB



Increasing Data Size

Decreasing Performance

3 BACKGROUND

dask

- Dask Distributed extends Dask to efficiently manage and execute large scale computations across a distributed cluster
- Provides dynamic task scheduling, scalability, and real-time monitoring of tasks through the Dask client

ProxyStore

- Facilitates efficient data management in distributed Python applications
- Proxy object acts as reference to an object in a global store
- Supports pass-by-reference with just-in-time object resolution
- Allows dynamic changes to communication methods without altering application code

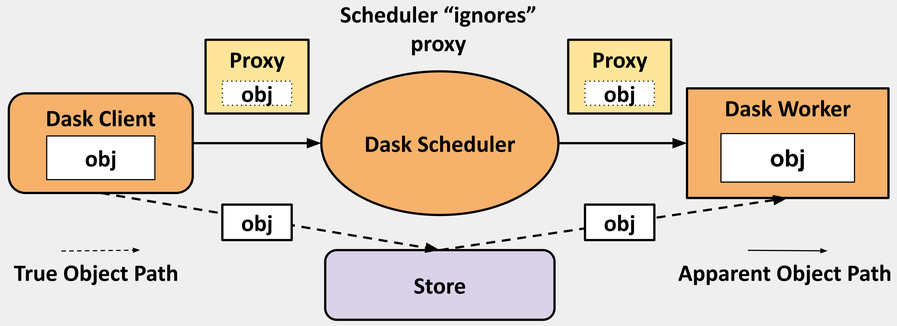
4 INTEGRATING DASK AND PROXYSTORE

Implementation

- Proxy objects that Dask struggles with
- Store the original object with ProxyStore
- Send its proxy to the Dask scheduler
- The scheduler treats the proxy as a small ByteString, ignoring its content
- The proxy is sent to the where, where it is resolves and reconstructs the original object from the store

Compatibility

- Custom Dask Client automatically proxies inputs/outputs and can be configured by object type or size
- Supports Peer-to-Peer (P2P) communications for direct node-to-node data transfer
- Provides support for connectors such as Redis, Kafka, ZeroMQ



5 IMPROVING PROXYSTORE PERFORMANCE

Goals

- Provide custom, optimized serialization for common data types for scientific computing
- Support the PyData stack (Pandas and NumPy) and Polars
- Added the ability to include custom serializers in addition to pickle fallback

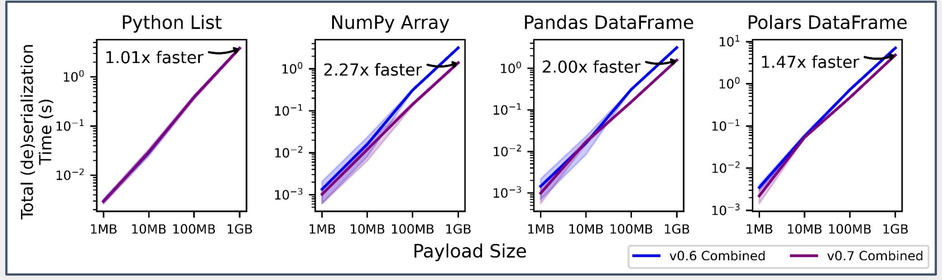
pandas NumPy

Table 1: Serialization time for 1 GB data structures.

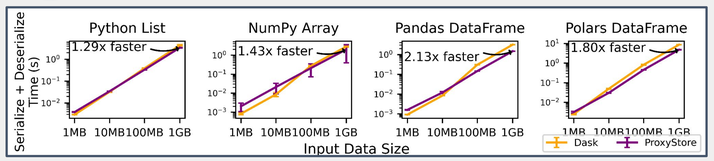
Data Structure	Serialization Time (s)		Improvement (%)
	Before	*After*	
Numpy Array	1.570	0.969	38.34%
Pandas DataFrame	1.573	0.788	49.88%
Polars DataFrame	3.744	2.436	34.94%

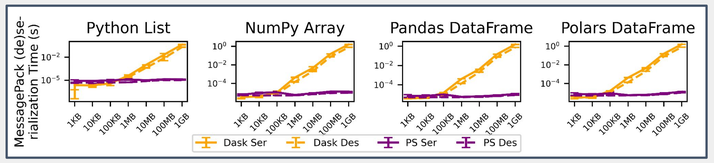
Speed-Up

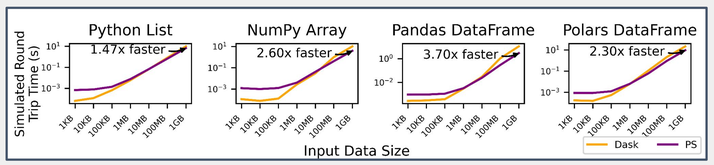
Serialization and deserialization times are 30% to 50% faster!



6 EVALUATION





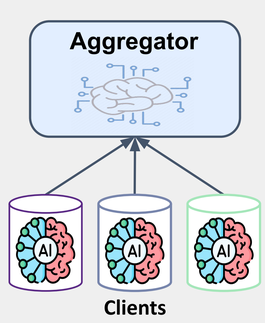


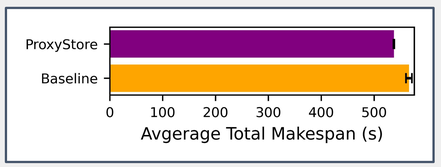
We tested the serialization and deserialization times of Dask and ProxyStore, comparing the entire serialization process, just MessagePack, and a simulated RTT (Round Trip Time), all of which outperformed Dask at high data sizes.

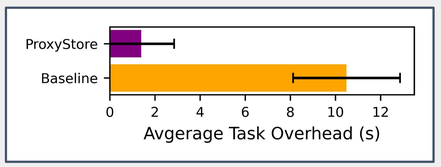
7 APPLICATIONS

Federated Learning

Federated learning is a collaborative machine learning approach where multiple clients train a shared model using their local data, without transferring the data to a central server. This implements a simple federated learning application, we run the application first using just Dask, then using Dask with ProxyStore.







References

[1] M. Dugré, V. Hayot-Sasson, and T. Glatard, "Performance comparison of dask and apache spark on hpc systems for neuroimaging," Concurrency and Computation: Practice and Experience, vol. 35, no. 21, p. e7635,2023
[2] Mathieu Dugré, Valérie Hayot-Sasson, and Tristan Glatard. 2023. Performance comparison of Dask and Apache Spark on HPC systems for neuroimaging.
Concurrency and Computation: Practice and Experience 35, 21 (2023), e7635
[3] J. Gregory Pauloski, Valerie Hayot-Sasson, Logan Ward, Alexander Brace, André Bauer, Kyle Chard, and Ian Foster. 2024. Object Proxy Patterns for Accelerating Distributed Applications. arXiv:2407.01764 [cs.DC]
https://arxiv.org/abs/2407.01764
[4] J. Gregory Pauloski, Valerie Hayot-Sasson, Logan Ward, Nathaniel Hudson, Charlie Sabino, Matt Baughman, Kyle Chard, and Ian Foster. 2023. Accelerating communications in federated applications with transparent object proxies. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 1–15.
[5] Matthew Rocklin et al. 2015. Dask: Parallel computation with blocked algorithms and task scheduling.. In SciPy . 126–132.
[6] TaPS: Task Performance Suite. https://github.com/proxystore/taps. Accessed July 2024.

Acknowledgements

This work is supported in part by the National Science Foundation grant NSF-1461260 (BigDataX REU).

GitHub →