Derecho's Extensible, Intelligent Object Store

Weijia Song
wsong@cornell.edu
Cornell University

Matthew Milano
milano@cornell.edu
Cornell University

Sagar Jha
srj57@cornell.edu
Cornell University

Edward Tremel
edward@cs.cornell.edu
Cornell University

Xinzhe Yang
xy269@cornell.edu
Cornell University

Ken Birman
ken@cs.cornell.edu
Cornell University

CCS Concepts

• **Computer systems organization** → **Distributed architectures**.

Keywords

Machine learning, Zero-Copy, RDMA, Derecho

ACM Reference Format:
Weijia Song, Matthew Milano, Sagar Jha, Edward Tremel, Xinzhe Yang, and Ken Birman. 2019. Derecho's Extensible, Intelligent Object Store. In *SOSP '19: ACM Symposium on Operating System Principles, October 27–30, 2019, Huntsville, Ontario*. ACM, New York, NY, USA, 3 pages.

1 Introduction

The Internet of Things (IoT) is increasingly coupled to cloud computing and machine learning, enabling smarter power delivery, smart highways and homes, intelligent health care and even smart farms. Yet although cloud providers offer all sorts of AI systems (examples include Microsoft Azure IoT [2] and Amazon Kinesis [1]), none is optimized for low latency, high volume decision-making at scale [8, 11–13]. A core challenge is data volume: IoT data (sensor records, photos, videos, etc) is predicted to grow from 0.1 zettabytes to 4.4 zettabytes just by 2020 [14]. Images and videos can often be quickly checked for interesting content and discarded immediately, yet in today's cloud it is more common to just upload and store everything, then process data later in big batches. This will need to change for the IoT edge: it is wasteful to write files, read them once, and then discard them, and the edge often must react instantly on an event-by-event basis, precluding batching.

Here, we argue that the next generation of machine learning applications should be layered over a new form of object store that employs standard APIs but is enhanced with machine-learning components. The idea of layering functionality on an object store is well accepted: many file systems and databases gain scalability this way [6, 15]. Our work extends this model, blending ML functionality with a sharded object store to allow a developer to customize the service. Moreover, our solution is optimized to minimize unnecessary data movement and to leverage hardware accelerators. The prototype was built using Derecho, a fault-tolerant RDMA-based distributed computing toolkit [4, 10].

2 Intelligent Object Store

Our work is still in progress but we have a working prototype, illustrated in Figure 1. The system runs as a `group` of `nodes`. Each node is a process targetted to run on a dedicated server. Within an application, nodes are grouped into `subgroups` that can be **sharded** if desired; the figure shows 3 shards with 3 nodes in each.

The object store offers a key-value API that maps to the underlying Derecho functionality but is intended as the sole API visible to our machine-intelligence modules. These *intelligence modules* consist of developer-supplied ML algorithms coded using MXNet [7], a popular machine-learning infrastructure (we picked MXNet because it is implemented in C++, matching Derecho). In the prototype, these modules are compiled at the time the service is created, but ultimately we should be able to also dynamically load them as DLLs. The resulting service can then be managed by a standard cloud *App Service* and used like any other service from web page builders or "cloud functions" running in the function tier of the cloud infrastructure.

We obtain the structure seen in Figure 1. A set of robot clients explore the world, finding interesting objects that they classify with help from the cloud. The example has two subgroups: a function tier intended to run as a cloud layer (with lightweight tasks that run in response to events and

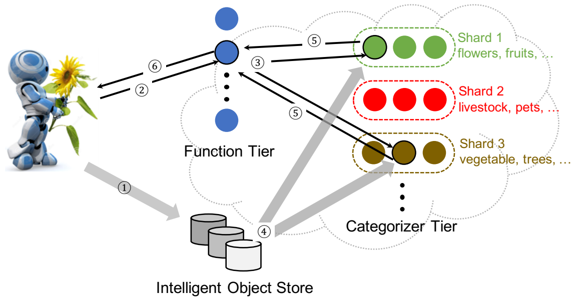

Figure 1: Model Serving by Derecho

terminate quickly), supported by a categorizer tier that uses intelligent object store. The functions in the function tier expose a load-balanced, RESTful API to clients, but all the communication internal to the service runs over RDMA.

Since functions shouldn't do heavy lifting, our function tier limits itself to a preliminary categorization: perhaps the client has found a flower, or a vegetable. Then it relays the request to the corresponding shards within a fine-grained categorizer tier. This categorizer holds specialized knowledge (CNN models), sharded by model category. Since these models are valuable, the shards holding them are replicated for fault-tolerance; if a new model is learned or an existing one is updated, an atomic multicast or a durable Paxos update (Derecho supports both options) guarantees consistency.

It is helpful to walk the critical path. The client puts the data in the object store then sends a request (①②). Upon receiving a classification request, the function tier coarsely identifies it as a flower or vegetable (②). To request a finer-grained classification, it forwards the request to a node in shard 1 and another node in shard 3 since they contain corresponding models (③). As seen, the two classifiers pulls the photo from the object store (④) and we obtain a vector of possibilities (⑤). The function tier node replies to the robot with the most probable result(s) (⑥).

The exciting innovation is that whereas a normal object store lacks intelligence, our Derecho object store can be extended with AI/ML logic *that runs right inside the store*. Thus whereas a normal cloud would have an object store holding photos, our approach permits the creation of an object store that will process incoming photos even before storing them, using ML segmentation, classification, prediction, or other functionality. Whereas the standard approaches tend to break those steps up into a graph of loosely coupled tasks (as in Spark or Tensor Flow), Derecho allows the developer to shape key choices so that all of those steps will occur in a single place – eliminating costly storage and copying.

3 Hardware Acceleration

Perhaps the most interesting challenge we confront is to leverage hardware accelerators such as GPU efficiently, without forcing the service designer to implement the logic that moves objects from place to place. The puzzle is that the ML application simply expresses operations as MXNet computations over data identified in a key-value manner, independent of location. In our setting, data could be incoming over RDMA whereas the ML model may have been stored in the object store but saved in NVM (SSD or Optane). On the other hand, a copy of the model might be cached in GPU memory (and similarly for intermediate results from prior computational step). Thus, the runtime issue of object location and placement poses a rich set of questions.

We tested a client/server setup where the client upload photos to be identified with a ResNet-50 [9] on the server-side. Photo data is written directly to the input layer of the neural network model with one-sided RDMA. Transferring one photo takes only 81 microseconds with 100 Gbps InfiniBand RDMA, but this soars to 5.2 milliseconds with TCP over 100 Gbps Ethernet. A GPU can run the classifier step in just 2.1 milliseconds [3]. Thus, an RDMA transfer is inexpensive compared to the inference latency, but TCP would dominate the cost of inference (worse, TCP also wastes CPU cycles copying from user-space to kernel and back).

A further challenge has to been to avoid excessive copying and locking: today's operating system designs and APIs are relaxed about both actions, yet copying within a node can actually be far slower than moving data from one node to a different one. Serialization and temporary disk writes further slow the path, to a point that standard systems are many orders of magnitude slower than the theoretical ideal. We are not the first to encounter this issue (see, for example, Xue's study of unnecessary copying in TensorFlow [16]).

Moreover, GPU is not the only hardware accelerator of interest. For example, when uploading photos in jpg format to a service that will classify them using MXNet, one might wish to use a bump-in-the-wire hardware accelerator to convert from jpg format to the MXNet `ndarray` format is required. Our design therefore adopts a perspective in which control is separated from data movement, and in which hardware accelerators can be situated at each hop of the data pipeline.

In the work completed as of today, we have extended Derecho's RDMA data paths to permit GPUs to efficiently access data in Derecho's key-value object store, with support for caching if a GPU will access the same object repeatedly. A key idea is to have the object store store *immutable, temporally versioned* data. Such data can safely be cached without need for a complex cache-invalidation protocol, hence by having Derecho track GPU cache contents, we can dynamically identify an optimal data movement pattern. To further reduce data copying, we manage memory using a system-wide memory map in which each node has a distinct address range of RDMA-capable memory, pinned and registered at the outset. Thus object pointers are meaningful system-wide. If a computation references a remote object, a page-fault will occur, and we then can redirect the request to a cached copy, fetch the object and map it locally, or even ship the computation to the remote location. In effect, much as Spark/DataBricks schedules tasks in an RDD-aware manner, we should be able to make real-time scheduling choices aimed at efficiency, load-balancing, and effective use of accelerators.

4 Related Work

There are many efforts to improve the ease of machine learning on IoT data. Clipper uses caching and batching to improve the latency as well as the throughput of a model serving system [8]. Pretzel discourages containerization and suggests reuse of sharing operators among models [11]. Ray introduced a new scheduler to manage a large scale machine learning system [12]. Closest to our work, Lu reimplemented gRPC with RDMA to support fast data exchange in TensorFlow [5]. Xue removed the gRPC layer, performing tensor transfers over RDMA to significantly improve TensorFlow training throughput [16]. In contrast to these efforts, our work focuses on optimization of the data path with zero-copy messaging, and avoiding suprious copying and locking.

References

[1] Amazon kinesis:easily collect, process, and analyze video and data streams in real time. https://aws.amazon.com/kinesis/. Accessed: 2019-09-09.

[2] Azure iot: Empowering businesses and industries to shape the future with the internet of things (iot). make things happen. https://azure.microsoft.com/en-us/overview/iot/. Accessed: 2019-09-09.

[3] Nvidia ai inference platform performance study. https://www.nvidia.com/content/dam/en-zz/Solutions/data-center/gated-resources/inference-technical-overview.pdf. Accessed:2019-09-10.

[4] Behrens, J., Jha, S., Birman, K., and Tremel, E. Rdmc: A reliable rdma multicast for large objects. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2018), IEEE, pp. 71–82.

[5] Biswas, R., Lu, X., and Panda, D. K. Accelerating tensorflow with adaptive rdma-based grpc. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)* (2018), IEEE, pp. 2–11.

[6] Brantner, M., Florescu, D., Graf, D., Kossmann, D., and Kraska, T. Building a database on s3. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2008), SIGMOD '08, ACM, pp. 251–264.

[7] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274* (2015).

[8] Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., and Stoica, I. Clipper: A low-latency online prediction serving system. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)* (2017), pp. 613–627.

[9] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[10] Jha, S., Behrens, J., Gkountouvas, T., Milano, M., Song, W., Tremel, E., Renesse, R. V., Zink, S., and Birman, K. P. Derecho: Fast state machine replication for cloud services. *ACM Transactions on Computer Systems (TOCS) 36*, 2 (2019), 4.

[11] Lee, Y., Scolari, A., Chun, B.-G., Santambrogio, M. D., Weimer, M., and Interlandi, M. {PRETZEL}: Opening the black box of machine learning prediction serving systems. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)* (2018), pp. 611–626.

[12] Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., and Stoica, I. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)* (Carlsbad, CA, Oct. 2018), USENIX Association, pp. 561–577.

[13] Stoica, I., Song, D., Popa, R. A., Patterson, D. A., Mahoney, M. W., Katz, R. H., Joseph, A. D., Jordan, M. I., Hellerstein, J. M., Gonzalez, J. E., Goldberg, K., Ghodsi, A., Culler, D., and Abbeel, P. A berkeley view of systems challenges for AI. *CoRR abs/1712.05855* (2017).

[14] Studio, P. D. The iot data explosion: How big is the iot data market? https://priceonomics.com/the-iot-data-explosion-how-big-is-the-iot-data/. Accessed:2019-09-10.

[15] Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D., and Maltzahn, C. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation* (2006), USENIX Association, pp. 307–320.

[16] Xue, J., Miao, Y., Chen, C., Wu, M., Zhang, L., and Zhou, L. Fast distributed deep learning over rdma. In *Proceedings of the Fourteenth EuroSys Conference 2019* (2019), ACM, p. 44.