Sharing and Caring of Data at the Edge

Animesh Trivedi* [1], Lin Wang* [1,2], Henri Bal[1], and Alexandru Iosup[1]
[1] *VU Amsterdam* [2] *TU Darmstadt*

Abstract

Edge computing is an emerging computing paradigm where data is generated and processed in the field using distributed computing devices. Many applications such as real-time video processing, augmented/virtual reality gaming, environment sensing, benefit from such decentralized, close-to-user deployments where low-latency, real-time results are expected. As with any distributed application, one of the key challenges in the development of collaborative applications is how to efficiently share data and state among multiple edge clients. The dynamic and heterogeneous environment together with diverse application's requirements make data sharing at the edge a challenging problem. Although there have been prior efforts, a systematic understanding of the area is missing. In this paper, we conduct a methodological study of different edge applications, their data sharing needs, and designs of state-of-the-art systems. In the process, we identify design options, under-explored opportunities, and associated challenges. We then present Griffin, our edge data sharing service, and seek feedback on its design.

1 Introduction

In recent years, we have witnessed a computing shift outwards from a data center to the network "edge", incubating a new paradigm called edge computing. In edge computing, a large amount of data is generated and consumed in the field (i.e., data locality) outside the data center by various edge applications. While cloud computing infrastructure services within a data center enabled flexible and economic pay-as-you-go solutions with automatic resource management, edge computing is growing due to its high potential to have immediate real-world deployments and impact in everyday situations [94], such as driving, farming, manufacturing, logistics, and monitoring. As a result, many frameworks are developed that propose new edge-friendly abstractions and APIs for applications [4, 43, 76, 81, 82, 95, 96].

*Equal contributions

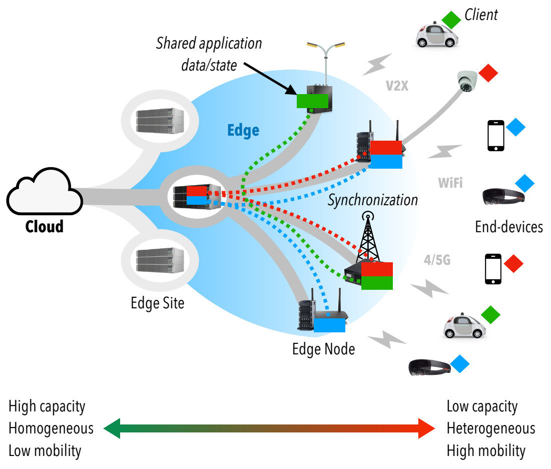

Figure 1: Data sharing at the edge.

One of the key challenges in the development of applications at the edge is how to efficiently share data[1] among clients. This data sharing is necessary to build applications such as collaborative machine learning (ML) [87, 100, 101], AR/VR gaming [23, 64, 117, 119], autonomous driving [18, 55, 59, 67, 118], video analytics [44, 51, 52, 70], and distributed environment sensing [9, 38, 105, 113]. A hallmark of these applications is that multiple clients work on a shared dataset and coorporate to achieve a common objective [62]. Data sharing can be managed within individual application frameworks or through an external storage service. The latter is a more popular choice at the edge because it allows extending the familiar RESTful/serverless abstraction from the cloud to the edge [48, 75, 90]. The use of an external, shared data store cleanly decouples compute from state management, leading to a stateless, fault-tolerant computation framework [14, 104].

The separation of state and compute is not a new idea. Within data centers there has already been a push to maintain ephemeral data and application-specific state in first-class storage services [57, 89, 99]. There exists more than half a dozen key-value storage, file systems, and database services

[1]Here, sharing refers to both *data* and application-specific *state*.

that cater to various shared state and data storage needs [3, 5, 41, 42, 77]. In comparison, platforms for edge storage services are still in their infancy. Furthermore, edge infrastructure presents new challenges (§1.1) for efficient data sharing.

In this paper, we present a first-of-its-kind systematic study of data sharing requirements at the edge (§2) and how proposed systems (fail to) fulfill them (§3). We then identify under-addressed concerns (e.g., heterogeneity, mobility), analyze trade-offs, present the design of Griffin, our edge data sharing service (§4), and conclude with discussion points (§5).

1.1 Edge Environment and Challenges

We adopt a definition of "edge" similar to the one from ETSI [36], where an edge *platform* consists of edge *sites* which can be gateway node(s) [24, 28] and/or local server machines (as done in Farmbeats [105]) that connect to the Internet, or edge station(s) such as a cell tower, a base station, or a roadside units (RSUs). An edge site can contain multiple edge nodes. *End-devices*, such as mobile devices, wearable sensors, IoT nodes, and smart cars, will connect to the edge nodes for storage/compute services using 4/5G, WAN, and WiFi (including V2X and IEEE 802.11p standards) technologies [88] as shown in Figure 1. We expect one or more edge *nodes* will be involved between an end-device and the nearest data center. We assume fixed edge sites, but with limited resources that decrease as we move closer to end-devices: small few-core CPUs, limited DRAM, and little permanent storage [88]. In this environment, a data center state or storage service (such as geo-distributed storage services) faces multiple challenges because of the following properties [25]:

- **Distributed.** Edge sites are dispersed geographically and do not have a centralized, equi-distant node to run the server service (such as the namenode in HDFS). For more decentralized approaches, such as Cassandra [61], a previous study has shown that the lack of a centralized administrative node makes consistent metadata management (membership, keyspace) in the presence of client mobility a challenging task [25]. Furthermore, typical multi-RTT application-level protocols (e.g., MySQL protocols) can diminish many edge-related latency gains for applications [86].
- **Heterogeneous.** Edge sites/nodes can be heterogeneous in terms of their storage and network capabilities. For example, an embedded edge node (e.g., colocated at a WiFi access point) with a Jetson board is typically equipped with a 16/32 GB eMMC, while an edge site can host 10-100s of servers with few TB storage capacities each. Edge network links might be bandwidth constrained and not symmetric, thus presenting challenges for cross-site traffic for achieving consistency. A large quantity of metadata to track resource heterogeneity is required, which needs to be handled efficiently and in a scalable manner.
- **Dynamic.** The edge environment can be unpredictable (interferences) and dynamic (changing conditions due to mobility, diurnal patterns) with multiple administrative domains. Thus, keeping up-to-date information regarding the infrastructure, its usage, and monitoring can be challenging.

Domain	**Shared Data Type**
Edge ML/DL [87, 100, 101]	model parameters, training data
Gaming (AR/VR) [23, 64, 117, 119]	user profiles, game world and state
Autonomous cars [18, 55, 59, 67, 118]	LiDAR data, maps, ML models
Edge analytics [6, 10, 31, 34]	aggregation states, ephemeral datasets
Video analytics [44, 51, 52, 70]	frames, execution state, objects
Sensing [9, 38, 105, 113]	environment, health, sensor data

Table 1: Overview of shared data types in edge applications.

2 What Do Edge Applications Want

Edge applications are diverse, and so are their data sharing needs. In this section we focus on *distributed* and *collaborative* edge applications and synthesize their shared data types, sizes, mobility patterns, consistency and performance requirements. Table 1 provides an overview.

Edge machine learning. Machine learning represents one of the most popular applications between the cloud and edge [87, 100, 101]. Here, the model parameters represent the distributed shared state that is read and updated by multiple participants (typically) using the distributed parameter server approach [19, 60, 63]. Further research has also demonstrated the additional value of doing collaborative, federated learning and transfer of knowledge between multiple nodes [69, 100, 110]. Typically, a model size would be 10-100 MBs and would require 1-100 ms (depending upon the network technology) of access and update latencies [50, 54, 87, 100]. There is also a large body of work in model compression, splitting, and network-efficient communication [17], which is orthogonal to this work. The expected state mobility is low to medium based on the client mobility.

Real-time massively multiplayer online game (MMOG). Collaborative, real-time MMOGs have multiple geo-dispersed players that share and interact within a common game world, potentially AR/VR augmented, e.g., Pokémon Go. An edge-conscious deployment can help deliver a smooth gameplay by meeting the strict latency and bandwidth requirements for shared world accesses from players [23, 32, 64, 117, 119]. In a streaming based setup, the interaction events can be a few kilobytes, with recommended streaming bandwidth in 1-10 Mbps and access delay tolerance of tens of millisecond [66]. Furthermore, different shared state types in a game environment can tolerate different consistency levels (strong, causal, read-your-writes, or eventual) to deliver the best gaming experience [29, 30, 109]. For example, a player-local view data needs to be refreshed very quickly with a strong consistency (within 20 ms [12]) in response to events, whereas shared game/world state updates can tolerate higher latencies (~100 ms) [117]. Based on the nature of the game (client mobility, co-locating, load balancing) medium-to-high dynamic

	Abstraction/API	Locality	Heterogeneity	Mobility	Failover	Scalability	App. Semantics
Generic edge storage							
PathStore [80, 81]	relational/CQL [7]	✓	✗	◖	◖	✓	session/eventual
FogStore [46]	key-value	✓	✗	✗	✓	✓	context-aware
DataFog [47]	key-value	✓	✓	✗	◖	✓	eventual
RedWedding [75]	CRDT [97]	✓	✗	✗	○	✓	conflict-free
DPaxos [84]	transactions	✓	✗	◖	✓	✓	quorum-based
EdgeCons [49]	events	✓	✗	✗	✓	✓	quorum-based
Timeseries DBs [2, 115]	timeseries	✓	✗	✗	◖	✓	range, aggregate
App-specific edge storage							
Cachier [34]	objects, content	✓	✗	✗	○	○	N/A
Vision-specific [91]	key-frames, feature vectors	✓	✗	✗	○	○	N/A

✓: full support ◖ : partial support ✗: no support ○: unknown

Table 2: Requirements analysis of existing edge storage systems.

state migration is needed. Mobile to cloud offloading of games already exists [53]. Within the edge, the EC+ gaming architecture models client mobility using a Markov decision process and proactively moves the game state [117].

Cooperative autonomous driving. Autonomous driving applications can be deployed in a collaborative manner where federated learning between multiple cars can be possible to mark hazards, predict traffic in real-time, share models, and transfer learning about a driver's profile [18,55,59,118]. High-precision, environment condition maps and LiDAR data are other datasets that can be maintained in a cooperative manner between multiple vehicles [22, 114]. Depending upon the volatility in the environment conditions, the dataset will be frequently updated, shared, and needs strong consistency. As autonomous vehicles have a strict time budget to make decisions (100 ms@10 Hz [65]), it is important that datasets are accessed in a predictable, low-latency manner. Depending on the type of the dataset, we expect none (e.g., neighborhood condition map) to high (e.g., a driver's profile) mobility.

Others. Beyond these key domains, there is a large body of work in analytics [6], stream processing [31], video analytics [44, 51, 52, 70], caching/CDN [33, 34, 108], etc. Stream applications are developed to prevent DDoS attacks that utilize malfunctioning/hacked IoT devices by sharing the monitoring state [10]. Here, the edge represents the first line of defense, but not a single edge site can observe all the traffic, and deduce that there is a DDoS attack underway. Hence, it is important that edge sites collectively maintain a shared network patterns state that can be used to identify an attack. For analytics, serverless has emerged as a popular choice (e.g., AWS IoT Greengrass, and others [48, 75, 90]). In the stateless serverless functions, data is shared though an external data store [57], which has unique properties [56]. Such a model is also being proposed for stream processing systems [71, 73].

Summary. Edge applications are diverse, and have a wide set of requirements for their state management. Instead of letting each application have to reinvent the wheel, there is an opportunity to build a common, unified shared data management service for the edge.

3 Wishes vs. Reality

We synthesize eight storage requirements for the aforementioned edge applications and conduct a comprehensive analysis (see Table 2) of the existing storage systems by focusing on how well they fulfill these requirements.

RQ1: Abstractions and APIs. Edge applications deal with different data types, e.g., text, images, and videos, with associated access and consistency semantics. Apart from the traditional RDBMS with SQL interfaces [8,27,98], the simple key-value interface (`get/put`) has been the most popular and versatile interface used by many data center stores [11,46,61]. Some edge stores also use the key-value interface for its simplicity [46,47]. There are others that adopt heavy transactional semantics [49, 84]. However, works in other domains such as ML have shown that having an expressive storage API is beneficial to run common operations, like data reduce, close to the state instead of always pulling/pushing raw values [13,68]. Such domain-specific data abstractions and APIs could make edge applications more efficient by relieving both the computation and the network traffic burden. Time series data is also popular in edge applications like environment monitoring/sensing. Cloud timeseries databases (TSDBs) such as OpenTSDB [2] need support from backend storage services such as HDFS. For edge-ready TSDBs [111, 115] the focus has been on efficient index building, stream merging, and range queries computations.

RQ2: Data locality. The proximity to data in edge computing helps reduce network latency, save network bandwidth via data filtering, and protect privacy. Many existing edge stores [75, 84] borrow ideas from geo-distributed storage systems such as Cassandra [61] and Spanner [27] that typically achieve data locality by careful replica placement and request dispatching across the distributed infrastructure [11, 112]. PathStore [81] adopts a hierarchical architecture that achieves data locality by serving requests from the nearest edge node by partially replicating data sets there. FogStore [46] and DataFog [47] perform location-aware replica placement by leveraging spatio-temporal encoding (e.g., ST-Hash [45]) to partition the data. In essence, data partitioning and replica placement on edge nodes/sites should be carefully chosen and continuously adapted following client mobility.

RQ3: Heterogeneity. Edge nodes can have very diverse hardware specifications, as do edge sites hosting different numbers of edge nodes. Unfortunately, existing edge stores mostly run under the infinite-homogeneous-resource assumption and only rarely draw attention to the implications of heterogeneity [46]. One exception is DataFog [47], which explicitly addresses resource scarcity at the edge through TTL-based data eviction and data aggregation and compression. The design of PathStore inherently handles heterogeneity by caching only partial replicas on edge nodes at the expense of high fetching cost in case of cache misses [81]. However, none of these systems explicitly consider the load, capacity, and processing power available at the edge in their replica placement decision. Such constraints can be very dynamic due to frequent capacity reclamation or client movements.

RQ4: Mobility. Similar to heterogeneity, mobility is a unique feature of the edge environment, which mainly comes from the movement of end-devices such as smartphones and autonomous vehicles. Existing edge storage systems are mostly mobility-agnostic [46, 47, 49, 75, 81, 84]. DPaxos treats client mobility as a non-imminent issue and relies on reactive data migration [84]. PathStore partially supports client mobility by tracking state changes (logging CQL queries) and reactively updating the replica when a client reestablishes the network connection to a new replica with session consistency guarantees [80]. There are generally two ways of addressing mobility: (1) Reactive methods constantly monitor the client's movement through the wireless network connection by tracking the real-time signal strength (e.g., RSSI) [83, 120]. (2) Proactive methods try to build a model for the client mobility pattern and then take precautionary actions according to model-generated predictions [16, 117].

RQ5: Failure, partitions, and limited connectivity. We expect limited and high cost of communication between edge nodes, hence building a conventional peer-to-peer system would not be useful as (1) all edge peers do not have equal storage, network and processing capabilities and (2) peer pairs may have limited connectivity. Such a setup also necessitates a careful placement of centralized components of data management protocols (e.g., Paxos leaders, primary backup) where one node has more responsibilities than others. Existing edge stores mostly rely on the default data replication feature for fault tolerance and generally follow the CAP theorem. PathStore handles temporary node failures by retrying to propagate timestamped writes to node ancestors for merging [80, 81]. Under network partitions, PathStore keeps serving write requests with local replicas and also read requests if the data is replicated in the local cache. FogStore handles geographically correlated failures by placing the replicas out of the context of interest (where end-devices are) far away from the others [46]. DPaxos handles client-specified failure rates at both the edge site level and the zone (a collection of edge sites) level [84]. Konwar et al. explore the use of erasure codes between edge nodes and cloud for a two-layered distributed edge store [58]. RedWedding leverages CRDT data types to allow concurrent accesses in a partitioned state, and then eventually reaches a common final state [97].

RQ6: Scalability and load balancing. A single edge store can be deployed across hundreds of edge sites (e.g., cell towers within a large city) with small capacities, several thousands of clients, with millions of objects. So far, scalability has been considered as a first-class citizen in most edge store designs. FogStore [46] and DataFog [47] adopt the spatio-temporal hashing which balances requests across edge nodes. In addition, DataFog uses neighbor edge nodes for offloading when hotspots are found in the system. PathStore [80, 81] scales well with the hierarchical architecture, but it does not address request load balancing issues. RedWedding store proposes to create a storage instance along with every serverless function invocation to achieve elastic concurrency [75]. An ideal edge storage system should be able not only to scale to enormous edge nodes/sites (needs efficient metadata management), but also to handle the read/write requests imbalance.

RQ7: Application-specific semantics. As we saw before, different edge applications have different consistency semantics. These requirements might be contextual (game vs. player data in online gaming), location-based (distance from a data source), or state dependent (reading a configuration vs. training data). Most existing edge stores adopt a single semantic, i.e., a single consistency model such as strong [49, 84] or eventual consistency [47], for all target applications. PathStore provides session consistency on top of eventual consistency [80]. FogStore allows applications to specify contexts of interest and apply differential consistency based on the context [46]. Ideally, an edge store would allow applications to easily attach multiple consistency semantics on stored data sets, thus dynamically trading performance with consistency to match the dynamic nature of edge deployments.

RQ8: Monitoring infrastructure. Considering an edge store is expected to operate in a highly dynamic environment, we expect the store to either leverage or provide good monitoring capabilities to identify environment changes, node mobility, or failures. So far, monitoring an edge store has been overlooked in the literature. CloudPath provides a monitoring module to collect statistics about CPU and memory metrics from the edge system [81], but it is not clear how the system leverages this information for data store optimization.

4 Griffin: Quest for an Ideal Storage Service

Having discussed requirements and various state-of-the-art stores proposed in the literature, we now present our initial design sketch for Griffin, a shared storage service for the edge, and seek feedback on further work in §5.

Overview. Griffin is a multi-layered hierarchical distributed storage service. It uses a client-server model where every edge site and node runs a data storage daemon. Data creation and placement decisions are taken in the cloud (centralized resource provisioning), and then read/write accesses are done by the client. A client bootstraps by first contacting a well-known service address in the cloud. Similar to DataFog [47], Griffin ensures data locality (RQ2) and scalability (RQ6) by including space-time labels to data items and employing spatio-temporal encoding (e.g., the Hilbert's curve [93]) for data partitioning and load balancing. Griffin relies on data replicas and careful replica placement for node and geographically correlated failures, respectively (RQ5). In addition, Griffin's design aims to answer the following questions:

What are the right data storage abstraction (RQ1) and consistency model (RQ7)? Griffin is a multi-consistency storage system. However, defining the consistency and latency trade-off can be a tedious job. We take inspiration from the Pileus storage service [102] which supports multiple consistency models with a *declarative* policy interface. Instead of having to choose one consistency model at the development time, a developer can specify their expected SLA latencies, desired consistency models (multiple are possible, e.g., eventual consistency can also be satisfied with a monotonic read), and their *utilities* in a declarative manner. The system dynamically optimizes to deliver the best utility for every data access. Building such a system at the edge presents challenges associated with: (1) How to reliably predict loads and latencies in the edge environment? Such information is necessary for a client to determine which consistency model can be satisfied under the current operational environment. There has been a large body of work in data center sensing, monitoring, and modeling congestion and RTTs [40, 78]. Our on-going work is looking into validating and extending the work to the edge. (2) Can we generate accurate timestamps for providing ordering guarantees between concurrent accesses? Previous research has shown that it is possible to build a bounded clock abstraction in data centers [27,39]. Edge stores like PathStore expect such abstractions to be available at the edge. D'souza et al. have proposed Time-as-a-Service (TaaS) middleware for sub-millisecond clock synchronization (using a mix of NTP, PTP, and Huygens [39] protocols) at the geo-scale infrastructure [35]. However, it expects certain underlying infrastrucutre support (e.g., hardware timestamping, Zookeeper, and pub/sub services). Mani et al. have shown that clock drifts, stability, and errors at the IoT/Edge environment exhibit very a different behavior than the server clock, thus needing a new clock synchronization mechanism [72]. Their algorithm, SPoT, can synchronize the clock of many heterogenous IoT devices with a cloud server within an accuracy of ~15 ms. We are investigating the use of SPoT in Griffin to generate timestamps. (3) How to reduce high (de)serialization costs associated with the key-value abstraction? Here we propose to use lightweight in-memory data formats as the storage format (standardized Apache Arrow [1], or high-performance formats like Albis [103]). This helps to reduce the cost of conversion between multiple languages and frameworks [79].

How to handle heterogeneity at the edge (RQ3)? Griffin incorporates a graph-based optimization engine for data replication and placement, taking into account the capability constraints of edge nodes/sites. Similar approaches have been used for managing edge compute resources [20, 85, 106, 113]. Griffin assumes that edge sites are fixed and builds an annotated graph of the edge infrastructure to capture the system status. A vertex in the graph represents an edge site consisting of a set of edge nodes, annotated with properties including resource specification and utilization. A link in the graph represents a connection hop between edge sites, annotated with properties such as per-hop network latency and bandwidth. Note that some of the properties on the graph, such as the resource (CPU, DRAM, and storage capacity) specification, are static, while others including real-time load, utilization, and the number of connected clients are dynamic. Applications will also be modeled through annotated graphs of data flow with performance constraints (e.g., required storage capacity, SLA latency) specified on both vertices and links. Essentially, the task of data replication and placement can be transformed into mapping the data flow graph of applications to the infrastructure graph. This problem is similar to the conventional virtual network embedding problem [37] and we will leverage its recent advancements [92] to solve this problem. Through the graph-based models and optimization, the heterogeneity of edge sites/nodes will be addressed inherently.

How to support system dynamics imposed by client mobility (RQ4, RQ8)? Griffin includes an in-band monitoring system which continuously collects and reports system statistics for the optimization engine to build the graphs and then adapts system configurations in response to system dynamics. For the graph-based optimization engine to work properly, at least the following system status information should be gathered at runtime: (1) the current compute/storage utilization on edge nodes, (2) the network bandwidth and latency between edge nodes/sites, (3) the quality of the (wireless) connection to the end-devices of users. The first two are to be used directly in the graph-based optimization engine and need to be measured constantly. However, achieving real-time monitoring, especially when the system dynamic is very high in application scenarios such as autonomous driving, is non-trivial and our monitoring system will have to make sure that both the time and traffic overhead are negligible [15]. The last is used for proactively taking data management actions for upcoming client movements and for delivering SLA guarantees for data accesses. Existing approaches, like those based on directly measuring the wireless channel RSSI, are typically end-device centric. However, end-devices are out of the reach of the storage system and thus, such approaches will not be applicable. Griffin proposes to leverage indirect statistics information, such as the packet retransmission rate and queue sizes, that are available at the cellular base stations for example to derive the desired connection quality information [21].

5 Discussion Topics

Expected feedback and discussion points.

- Is it realistic to assume that developers can identify, often manually, consistency and performance requirements of different application datasets? We hope to hear opinions from developers about what kind of abstractions are needed to facilitate an easy (or automatic) state externalization.
- How should we benchmark a data sharing service at the edge? Are there standard, open-source storage benchmarks and data sets (similar to Cloud YCSB [26], DeFog [74], CAVBench [107]) we can use to evaluate edge stores?
- We expect that a storage service will not be the only service to be deployed on the shared edge infrastructure. How should resources at the edge be provisioned and (de)allocated to the storage service?
- More broadly, we seek to instill a discussion regarding what it would take to completely automate the edge data management with the help of runtime, compiler, new storage abstraction, and the deployment framework.

Controversial topics and open challenges.

- Is another data store needed at the edge right now?
- Modeling and monitoring any distributed system is a non-trivial pursuit, especially at the edge. Is it realistic to model the edge environment with all its complexities? There are some precedents inside a data center[2]. Can we learn and extend those ideas in an edge environment?
- We have not investigated edge client-side concerns related to processing power, memory, and energy requirements to access data. For example, we expect client-heavy protocols where heavy states (network, storage, consistency) are maintained at the client side to face challenges. Ideally, a stateless thin-client protocol would be preferred where most of the logic resides on the server-side. A systematic exploration of client's needs would be an interesting study.

When does the idea fall apart?

- Contrary to data centers, no single operator runs the whole edge infrastructure. An edge operator/company may not be willing to disclose topology, resources, and monitoring information. So there has to be a clear value-addition to their business cases for providers to share information and build such a service.
- Applications may not externalize state due to security and privacy concerns, e.g., health data stored on smart watches. Can we integrate "end-devices" in the edge storage service as well? Such a design will drastically change the current proposal where we must then deal with unreliable connections, high churn rates, without having to deal with data sharing for specific data sets (all accesses must be local). We can envision storage solutions integrated with a lightweight web-assembly function shipping to end-nodes to support secure computation [116].

[2]Strymon - a platform for online modeling of enterprise data center behavior, http://strymon.systems.ethz.ch/

Acknowledgments

Lin Wang was funded partially by the German Research Foundation (DFG) under the joint Sino-German grant No. 392046569 (NSFC: 61761136014) and by the DFG Collaborative Research Center (CRC) 1053 – MAKI.

References

[1] Apache Arrow: A cross-language development platform for in-memory data. https://arrow.apache.org/. Accessed: 2020-02-15.

[2] The Scalable Time Series Database. http://opentsdb.net. Accessed: 2020-05-05.

[3] Amazon. Databases on AWS. https://aws.amazon.com/products/databases/?nc2=h_ql_prod_db. Accessed: 2020-02-15.

[4] Amazon. AWS IoT for the Edge. https://aws.amazon.com/iot/solutions/iot-edge/. Accessed: 2020-02-15.

[5] Amazon. Overview of Amazon Web Services. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/storage-services.html. Accessed: 2020-02-15.

[6] Muhammad Anwar, Shangguang Wang, Muhammad Zia, Ahmer Jadoon, Umair Akram, and Syed Salman Raza Naqvi. Fog computing: An overview of big iot data analytics. *Wireless Communications and Mobile Computing*, 2018:1–22, 05 2018.

[7] Apache. The Cassandra Query Language (CQL). http://cassandra.apache.org/doc/latest/cql/. Accessed: 2020-02-15.

[8] Jason Baker, Chris Bond, James C. Corbett, J. J. Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, and Vadim Yushprakh. Megastore: Providing scalable, highly available storage for interactive services. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 223–234, 2011.

[9] Roshan Bharath Das, Marc Makkes, Alexandru Uta, Lin Wang, and Henri Bal. Aves: A framework for energy-efficient stream analytics across low-power devices. In *IEEE Big Data*, 12 2019.

[10] Ketan Bhardwaj, Joaquin Chung Miranda, and Ada Gavrilovska. Towards iot-ddos prevention using edge computing. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[11] Kirill Bogdanov, Waleed Reda, Gerald Q. Maguire Jr., Dejan Kostic, and Marco Canini. Fast and accurate load balancing for geo-distributed storage systems. In *ACM Symposium on Cloud Computing (SoCC)*, pages 386–400. ACM, 2018.

[12] John Carmack. Latency mitigation strategies. https://danluu.com/latency-mitigation/ (mirror). Accessed: 2020-02-15.

[13] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. A case for serverless machine learning. In *ACM Workshop on Systems for ML and Open Source Software (MLSys)*, 2018.

[14] Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. The rise of serverless computing. *Communications of the ACM*, 62(12):44–54, 2019.

[15] Roger Pueyo Centelles, Mennan Selimi, Felix Freitag, and Leandro Navarro. A monitoring system for distributed edge infrastructures with decentralized coordination. *ALGOCLOUD*, 2019.

[16] Addison Chan and Frederick W. B. Li. Utilizing massive spatiotemporal samples for efficient and accurate trajectory prediction. *IEEE Transactions on Mobible Computing*, 12(12):2346–2359, 2013.

[17] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[18] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *ACM/IEEE Symposium on Edge Computing (SEC)*, page 88–100, 2019.

[19] Yitao Chen, Kaiqi Zhao, Baoxin Li, and Ming Zhao. Exploring the use of synthetic gradients for distributed deep learning across cloud and edge resources. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.

[20] Bin Cheng, Apostolos Papageorgiou, Flavio Cirillo, and Ernoe Kovacs. Geelytics: Geo-distributed edge analytics for large scale iot systems based on dynamic topology. In *IEEE World Forum on Internet of Things (WF-IoT)*, pages 565–570, 2015.

[21] Sandeep Chinchali, Pan Hu, Tianshu Chu, Manu Sharma, Manu Bansal, Rakesh Misra, Marco Pavone, and Sachin Katti. Cellular network traffic scheduling with deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 766–774, 2018.

[22] Hyunggi Cho, Young-Woo Seo, B.V.K. Vijaya Kumar, and Ragunathan (Raj) Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843, 2014.

[23] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Systems*, 20(5):503–519, 2014.

[24] Cisco. Kinetic Edge and Fog Processing Module (EFM). https://www.cisco.com/c/dam/en/us/solutions/collateral/internet-of-things/kinetic-datasheet-efm.pdf. Accessed: 2020-02-15.

[25] Bastien Confais, Adrien Lebre, and Benoît Parrein. *Performance Analysis of Object Store Systems in a Fog and Edge Computing Infrastructure*, pages 40–79. 2017.

[26] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *ACM Symposium on Cloud Computing (SoCC)*, page 143–154, 2010.

[27] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson C. Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. Spanner: Google's globally distributed database. *ACM Transactions Computer Systems*, 31(3):8:1–8:22, 2013.

[28] Dell. Edge Gateway 3003 Specification. https://topics-cdn.dell.com/pdf/dell-edge-gateway-3000-series_Specifications3_en-us.pdf. Accessed: 2020-02-15.

[29] Ziqiang Diao. Consistency models for cloud-based online games: The storage system's perspective. In *GI-Workshop on Foundations of Databases (Grundlagen von Daten-banken)*, volume 1020, pages 16–21, May 2013.

[30] Ziqiang Diao, S. Wang, E. Schallehn, and Gunter Saake. Cloudcraft: Cloud-based data management for mmorpgs. In *Databases and Information Systems VIII*, volume 270, pages 71–84, January 2014.

[31] Marcos Dias de Assuno, Alexandre da Silva Veith, and Rajkumar Buyya. Distributed data stream processing and edge computing. *Journal of Network and Computer Applications*, 103(C):1–17, 2018.

[32] Jesse Donkervliet, Animesh Trivedi, and Alexandru Iosup. Towards supporting millions of users in modifiable virtual environments by redesigning minecraft-like games as serverless systems. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2020.

[33] Utsav Drolia, Katherine Guo, and Priya Narasimhan. Precog: refetching for image recognition applications at the edge. In *ACM/IEEE Symposium on Edge Computing (SEC)*, 2017.

[34] Utsav Drolia, Katherine Guo, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. Cachier: Edge-caching for recognition applications. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 276–286, 2017.

[35] Sandeep D'souza, Heiko Koehler, Akhilesh Joshi, Satyam Vaghani, and Ragunathan (Raj) Rajkumar. Quartz: Time-as-a-service for coordination in geo-distributed systems. In *ACM/IEEE Symposium on Edge Computing (SEC)*, page 264–279, 2019.

[36] ETSI. Multi-access Edge Computing (MEC): Framework and Reference Architecture. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf, 2019. Accessed: 2020-02-15.

[37] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys and Tutorials*, 15(4):1888–1906, 2013.

[38] Julien Gedeon, Michael Stein, Jeff Krisztinkovics, Patrick Felka, Katharina Keller, Christian Meurisch, Lin Wang, and Max Mühlhäuser. From cell towers to smart street lamps: Placing cloudlets on existing urban infrastructures. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 187–202, 2018.

[39] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosenblum, and Amin Vahdat. Exploiting a natural network effect for scalable, fine-grained clock synchronization. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 81–94, 2018.

[40] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosenblum, and Amin Vahdat. SIMON: A simple and scalable method for sensing, inference and measurement in data center networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 549–564, 2019.

[41] Google. Cloud storage products. https://cloud.google.com/products/storage. Accessed: 2020-02-15.

[42] Google. Google Cloud Databases. https://cloud.google.com/products/databases. Accessed: 2020-02-15.

[43] Google. Google Cloud IoT. https://cloud.google.com/solutions/iot. Accessed: 2020-02-15.

[44] Giulio Grassi, Kyle Jamieson, Paramvir Bahl, and Giovanni Pau. Parkmaster: an in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. In *ACM/IEEE Symposium on Edge Computing (SEC)*, pages 16:1–16:14, 2017.

[45] Xuefeng Guan, Cheng Bo, Zhenqiang Li, and Yaojin Yu. St-hash: An efficient spatiotemporal index for massive trajectory data in a nosql database. In *International Conference on Geoinformatics, Geoinformatics*, pages 1–7, 2017.

[46] Harshit Gupta and Umakishore Ramachandran. Fogstore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access. In *ACM International Conference on Distributed and Event-based Systems (DEBS)*, pages 148–159, 2018.

[47] Harshit Gupta, Zhuangdi Xu, and Umakishore Ramachandran. Datafog: Towards a holistic data management platform for the iot age at the network edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[48] Adam Hall and Umakishore Ramachandran. An execution model for serverless functions at the edge. In *ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI)*, page 225–236, 2019.

[49] Zijiang Hao, Shanhe Yi, and Qun Li. Edgecons: Achieving efficient consensus in edge computing networks. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[50] Ke-Jou Hsu, Ketan Bhardwaj, and Ada Gavrilovska. Couper: Dnn model slicing for visual analytics containers at the edge. In *ACM/IEEE Symposium on Edge Computing (SEC)*, page 179–194, 2019.

[51] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodík, Leana Golubchik, Minlan Yu, Victor Bahl, and Matthai Philipose. Videoedge: Processing camera

streams using hierarchical clusters. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 115–131, 2018.

[52] Si Young Jang, Yoonhyung Lee, Byoungheon Shin, and Dongman Lee. Application-aware iot camera virtualization for video analytics edge computing. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 132–144, 2018.

[53] M. H. Jiang, Otto W. Visser, I. S. W. B. Prasetya, and Alexandru Iosup. A mirroring architecture for sophisticated mobile games using computation-offloading. *Concurrency and Computation Practice and Experience*, 30(17), 2018.

[54] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, page 615–629, 2017.

[55] Gorkem Kar, Shubham Jain, Marco Gruteser, Fan Bai, and Ramesh Govindan. Real-time traffic estimation at vehicular edge nodes. In *ACM/IEEE Symposium on Edge Computing (SEC)*, 2017.

[56] Ana Klimovic, Yawen Wang, Christos Kozyrakis, Patrick Stuedi, Jonas Pfefferle, and Animesh Trivedi. Understanding ephemeral storage for serverless analytics. In *USENIX Conference on Usenix Annual Technical Conference (ATC)*, page 789–794, 2018.

[57] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. Pocket: Elastic ephemeral storage for serverless analytics. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*, page 427–444, 2018.

[58] Kishori M. Konwar, N. Prakash, Nancy Lynch, and Muriel Médard. A layered architecture for erasure-coded consistent distributed storage. In *ACM Symposium on Principles of Distributed Computing (PODC)*, page 63–72, 2017.

[59] M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646, 2015.

[60] Dhruv Kumar, Aravind Alagiri Ramkumar, Rohit Sindhu, and Abhishek Chandra. Decaf: Iterative collaborative processing over the edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.

[61] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[62] Zach Leidall, Abhishek Chandra, and Jon Weissman. An edge-based framework for cooperation in internet of things applications. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.

[63] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 583–598, 2014.

[64] Y. Li and W. Gao. Muvr: Supporting multi-user mobile virtual reality with resource constrained edge cloud. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 1–16, 2018.

[65] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E. Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, page 751–766, 2018.

[66] Yuhua Lin and Haiying Shen. Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):431–445, 2017.

[67] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[68] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Very Large Data Bases (VLDB)*, 5(8):716–727, 2012.

[69] Sidi Lu, Yongtao Yao, and Weisong Shi. Collaborative learning on the edges: A case study on connected vehicles. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.

[70] Bing Luo, Sheng Tan, Zhifeng Yu, and Weisong Shi. Edgebox: Live edge video analytics for near real-time event detection. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 347–348, 2018.

[71] Manisha Luthra, Sebastian Hennig, Pratyush Agnihotri, Lin Wang, and Boris Koldehofe. Highly flexible server agnostic complex event processing operators. In *ACM*

International Middleware Conference (Middleware), pages 11–12, 2019.

[72] Sathiya Kumaran Mani, Ramakrishnan Durairajan, Paul Barford, and Joel Sommers. An architecture for iot clock synchronization. In *International Conference on the Internet of Things (IOT)*, 2018.

[73] Ovidiu-Cristian Marcu, Radu Tudoran, Bogdan Nicolae, Alexandru Costan, Gabriel Antoniu, and María S. Pérez-Hernández. Exploring shared state in key-value store for window-based multi-pattern streaming analytics. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, page 1044–1052, 2017.

[74] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal de Lara, and Blesson Varghese. Defog: Fog computing benchmarks. In *ACM/IEEE Symposium on Edge Computing (SEC)*, page 47–58, 2019.

[75] Christopher Meiklejohn, Heather Miller, and Zeeshan Lakhani. Towards a solution to the red wedding problem. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[76] Microsoft. Azure IoT Edge. https://azure.microsoft.com/en-us/services/iot-edge/. Accessed: 2020-02-15.

[77] Microsoft. Azure products : Datbases and Storage categories. https://azure.microsoft.com/en-us/services/. Accessed: 2020-02-15.

[78] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. Timely: Rtt-based congestion control for the datacenter. *ACM SIGCOMM Computer Communication Review*, 45(4):537–550, 2015.

[79] Philipp Moritz and Robert Nishihara. Plasma: A High-Performance Shared-Memory Object Store. https://arrow.apache.org/blog/2017/08/08/plasma-in-memory-object-store/. Accessed: 2020-02-15.

[80] Seyed Hossein Mortazavi, Bharath Balasubramanian, Eyal de Lara, and Shankaranarayanan Puzhavakath Narayanan. Toward session consistency for the edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[81] Seyed Hossein Mortazavi, Mohammad Salehe, Carolina Simoes Gomes, Caleb Phillips, and Eyal de Lara. Cloudpath: A multi-tier cloud computing framework. In *ACM/IEEE Symposium on Edge Computing (SEC)*, 2017.

[82] Richard Mortier, Jianxin R. Zhao, Jon Crowcroft, Liang Wang, Qi Li, Hamed Haddadi, Yousef Amar, Andy Crabtree, James A. Colley, Tom Lodge, Tosh Brown, Derek McAuley, and Chris Greenhalgh. Personal data management with the databox: What's inside the box? In *ACM Workshop on Cloud-Assisted Networking (CAN)*, pages 49–54, 2016.

[83] A. B. M. Musa and Jakob Eriksson. Tracking unmodified smartphones using wi-fi monitors. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 281–294, 2012.

[84] Faisal Nawab, Divyakant Agrawal, and Amr El Abbadi. Dpaxos: Managing data closer to users for low-latency and mobile applications. In *ACM International Conference on Management of Data (SIGCOMM)*, pages 1221–1236, 2018.

[85] José Leal D. Neto, Se-Young Yu, Daniel F. Macedo, José Marcos S. Nogueira, Rami Langar, and Stefano Secci. Uloof: A user level online offloading framework for mobile edge computing. *IEEE Transactions on Mobile Computing*, 17(11):2660–2674, 2018.

[86] Chanh Nguyen, Amardeep Mehta, Cristian Klein, and Erik Elmroth. Why cloud applications are not ready for the edge (yet). In *ACM/IEEE Symposium on Edge Computing (SEC)*, page 250–263, 2019.

[87] Samuel S. Ogden and Tian Guo. MODI: Mobile deep inference made efficient by edge computing. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[88] Quoc-Viet Pham, Fang Fang, Ha-Nguyen Vu, Mai Le, Zhiguo Ding, Long Bao Le, and Won-Joo Hwang. A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art. *CoRR*, abs/1906.08452, 2019.

[89] Russell Power and Jinyang Li. Piccolo: Building fast, distributed programs with partitioned tables. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*, page 293–306, 2010.

[90] Thomas Rausch, Waldemar Hummer, Vinod Muthusamy, Alexander Rashed, and Schahram Dustdar. Towards a serverless platform for edge AI. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.

[91] Arun Ravindran and Anjus George. An edge datastore architecture for latency-critical distributed machine vision applications. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[92] Matthias Rost and Stefan Schmid. Virtual network embedding approximations: Leveraging randomized rounding. *IEEE/ACM Transactions on Networking*, 27(5):2071–2084, 2019.

[93] Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

[94] Mahadev Satyanarayanan. The emergence of edge computing. 50(1):30–39, 2017.

[95] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[96] Enrique Saurez, Kirak Hong, Dave Lillethun, Umakishore Ramachandran, and Beate Ottenwälder. Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In *ACM International Conference on Distributed and Event-based Systems (DEBS)*, pages 258–269, 2016.

[97] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *International Conference on Stabilization, Safety, and Security of Distributed Systems (SSS)*, page 386–400, 2011.

[98] Jeff Shute, Radek Vingralek, Bart Samwel, Ben Handy, Chad Whipkey, Eric Rollins, Mircea Oancea, Kyle Littlefield, David Menestrina, Stephan Ellner, John Cieslewicz, Ian Rae, Traian Stancescu, and Himani Apte. F1: A distributed SQL database that scales. *Very Large Data Bases (VLDB)*, 6(11):1068–1079, 2013.

[99] Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, Ana Klimovic, Adrian Schuepbach, and Bernard Metzler. Unification of temporary storage in the nodekernel architecture. In *USENIX Conference on Usenix Annual Technical Conference (ATC)*, page 767–781, 2019.

[100] Nisha Talagala, Swaminathan Sundararaman, Vinay Sridhar, Dulcardo Arteaga, Qianmei Luo, Sriram Subramanian, Sindhu Ghanta, Lior Khermosh, and Drew Roselli. ECO: Harmonizing edge and cloud with ml/dl orchestration. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[101] Zeyi Tao and Qun Li. esgd: Communication efficient distributed deep learning on the edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[102] Douglas B. Terry, Vijayan Prabhakaran, Ramakrishna Kotla, Mahesh Balakrishnan, Marcos K. Aguilera, and Hussam Abu-Libdeh. Consistency-based service level agreements for cloud storage. In *ACM Symposium on Operating Systems Principles (SOSP)*, page 309–324, 2013.

[103] Animesh Trivedi, Patrick Stuedi, Jonas Pfefferle, Adrian Schuepbach, and Bernard Metzler. Albis: High-performance file format for big data systems. In *USENIX Annual Technical Conference (ATC)*, pages 615–630, 2018.

[104] Erwin van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uță, and Alexandru Iosup. Serverless is more: From paas to present cloud computing. *IEEE Internet Computing*, 22(5):8–17, 2018.

[105] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. Farmbeats: An iot platform for data-driven agriculture. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 515–529, 2017.

[106] Lin Wang, Lei Jiao, Ting He, Jun Li, and Max Mühlhäuser. Service entity placement for social virtual reality applications in edge computing. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 468–476, 2018.

[107] Yifan Wang, Shaoshan Liu, Xiaopei Wu, and Weisong Shi. Cavbench: A benchmark suite for connected and autonomous vehicles. In *IEEE/ACM Symposium on Edge Computing (SEC)*, pages 30–42, 2018.

[108] Gala Yadgar, Oleg Kolosov, Mehmet Fatih Aktas, and Emina Soljanin. Modeling the edge: Peer-to-peer reincarnated. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.

[109] Amir Yahyavi and Bettina Kemme. Peer-to-peer architectures for massively multiplayer online games: A survey. *ACM Computing Surveys*, 46(1), 2013.

[110] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2), 2019.

[111] Yang Yang, Qiang Cao, and Hong Jiang. Edgedb: An efficient time-series database for edge computing. *IEEE Access*, 7:142295–142307, 2019.

[112] Victor Zakhary, Faisal Nawab, Divy Agrawal, and Amr El Abbadi. Global-scale placement of transactional data stores. In *International Conference on Extending Database Technology (EDBT)*, pages 385–396, 2018.

[113] Daniel (Yue) Zhang, Tahmid Rashid, Xukun Li, Nathan Vance, and Dong Wang. Heteroedge: Taming the heterogeneity of edge computing system in social sensing.

In *ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI)*, page 37–48, 2019.

[114] Jun Zhang and Khaled B. Letaief. Mobile edge intelligence and computing for the internet of vehicles. *Proceedings of the IEEE*, 108(2):246–261, 2020.

[115] Shuai Zhang, Wenxi Zeng, I-Ling Yen, and Farokh B. Bastani. Semantically enhanced time series databases in iot-edge-cloud infrastructure. In *IEEE International Symposium on High Assurance Systems Engineering (HASE)*, pages 25–32, 2019.

[116] Tian Zhang, Dong Xie, Feifei Li, and Ryan Stutsman. Narrowing the gap between serverless and its state with storage functions. In *ACM Symposium on Cloud Computing (SoCC)*, page 1–12, 2019.

[117] Wuyang Zhang, Jiachen Chen, Yanyong Zhang, and Dipankar Raychaudhuri. Towards efficient edge cloud augmentation for virtual reality mmogs. In *ACM/IEEE Symposium on Edge Computing (SEC)*, 2017.

[118] Xingzhou Zhang, Mu Qiao, Liangkai Liu, Yunfei Xu, and Weisong Shi. Collaborative cloud-edge computation for personalized driving behavior modeling. In *ACM/IEEE Symposium on Edge Computing (SEC)*, page 209–221, 2019.

[119] Xu Zhang, Hao Chen, Zhao, Zhan Ma, Yiling Xu, Haojun Huang, Hao Yin, and Dapeng Oliver Wu. Improving cloud gaming experience through mobile edge computing. *IEEE Wireless Communications*, 26(4):178–183, 2019.

[120] Ying Zhang. User mobility from the view of cellular data networks. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1348–1356, 2014.