The Tipping Point of Edge-Cloud Data Management

Faisal Nawab
nawabf@uci.edu
UC Irvine

Moshe Shadmon
moshe@anylog.co
AnyLog

ABSTRACT

Edge and Internet of Things (IoT) applications have attracted significant attention from both industry and academia due to their immense potential. As a result, the database community—through communications such as the recent database Seattle reports—has recognized the criticality of developing a new breed of data management systems specifically tailored for IoT and edge applications. These systems need to be distributed across edge locations to effectively handle the unique challenges posed by these environments. However, the development of such databases remains largely minimal in both industry and academia.

Over the past five years, our team has conducted extensive research and collaborated with industry partners to bring an edge-cloud database to market and to investigate the reasons behind the limited progress and adoption of distributed edge-cloud databases. As part of our efforts, we have developed a distributed edge-cloud database called AnyLog and deployed it with numerous industry partners in different IoT and smart city sectors. Our interactions with many technical teams from diverse industries have provided invaluable insights into the challenges and opportunities associated with distributed edge-cloud data management.

In this paper, we present a comprehensive summary of our findings, drawing from our five-year experience in the field. We highlight the real challenges faced in distributed edge-cloud data management and discuss the opportunities that lie within. Furthermore, we showcase the capabilities of edge-cloud databases using AnyLog, illustrating our insights about the challenges and opportunities of edge-cloud databases.

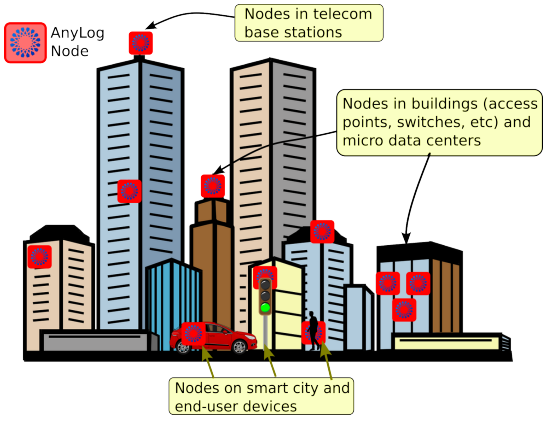


Figure 1: An example of an AnyLog Edge-Cloud deployment. AnyLog nodes are deployed on the Edge across buildings, end-user and smart city devices, and micro data centers. Each AnyLog node is decentralized, where it can be added and/or removed independently. AnyLog nodes coordinate through a decentralized protocol that orchestrates and synchronizes the process of the distributed nodes and the data that they store.

1 INTRODUCTION

IoT and edge applications encompass a diverse range of applications that heavily rely on IoT devices (including Internet-connected sensors, PLCs, appliances, and cameras) and edge devices (including wearables, virtual and augmented reality headsets/glasses, and mobile phones). These applications typically have a subset of these properties: low-latency requirement, sporadic connectivity, and generating vast amounts of data continuously. With their increasing popularity and predicted status as billion-dollar industries, they are anticipated to play a pivotal role in advancing and sustaining various sectors. Industries like smart cities and spaces, industrial and automation systems, personalized healthcare, immersive virtual and augmented reality, mobile gaming, and surveillance heavily depend on the successful deployment and data infrastructure of IoT and edge applications.

As IoT and edge applications continue to advance, it becomes increasingly apparent that efficient and scalable data management systems tailored specifically for these environments are essential. These systems should possess the ability to *reliably process and analyze huge amounts of data in real-time*, enabling timely decision-making and unlocking the full potential of IoT and edge applications. The database and storage communities have emphasized this in documents such as the most recent "Seattle Report on Database Research" [2] and the "Data Storage Research Vision 2025" Report on NSF Visioning [4].

While significant efforts have been made to develop databases and stream-processing systems for IoT and edge data [1, 5, 7, 17–19, 23, 24], most of these solutions still adhere to the traditional paradigm of centralization, where data is sent to centralized nodes either directly or through multiple processing levels between the edge and central location. Other solutions rely on centralization differently, where queries and control are orchestrated by a centralized node (e.g., a gateway orchestrating data processing across a sensor network) [11, 14]. Unfortunately, this centralized approach falls short in supporting edge-cloud databases as it lacks the crucial characteristic of being in close proximity to both the data sources

 14th Annual Conference on Innovative Data Systems Research (CIDR '24). TODO, Chaminade, USA

and users. Without such close proximity, properties such as low-latency requirements and tolerating sporadic connectivity cannot be supported. In addition, the functionalities needed at the edge span beyond a single product—at the edge, developers need to shoehorn multiple products (relational databases, object stores, stream processors, data brokers, and more). In addition, they need to independently manage security, high availability, user and data authentication, and integrate all of that with a platform that is able to orchestrate and monitor the distributed edge resources. This approach creates proprietary software stacks intermingled with domain knowledge and assumptions which leads to data silos and setups that are hard to deploy, manage, and scale.

What is needed is a distributed and decentralized edge-cloud database architecture—where nodes are distributed at the Edge in close proximity to users. These Edge nodes can operate independently of the rest of the network, are co-located with IoT/edge devices, or deployed on infrastructure in proximity to them (Figure 1). This edge platform needs to offer the needed edge functionalities as unified and integrated services, that are easily deployed, managed, and automatically scalable.

The absence of distributed edge-cloud databases in the industry and the limited number of recent academic efforts on this topic raise the question of why this crucial need remains largely unaddressed. Over the past five years, our team has actively pursued research and development in the field of edge-cloud databases from both industry and academic perspectives. It all began with a vision we outlined in a previous paper published in CIDR 2020 [3], setting the stage for our comprehensive exploration of this area. Taking the industry standpoint, we embarked on building AnyLog, a distributed and decentralized edge-cloud database. This solution has been deployed across numerous industry partners representing diverse sectors, including smart spaces, IoT, and industrial automation applications. To gain deeper insights into the challenges and relevance of edge-cloud databases within these industries, we engaged with many technical teams in the IoT/edge domain, fostering fruitful discussions about their specific technical goals and requirements in relation to edge-cloud databases.

Simultaneously, from an academic standpoint, we dedicated our efforts to investigating various crucial aspects of data management in the context of edge-cloud databases. This encompassed exploring topics such as indexing and storage [13], transaction processing [9], distributed coordination and consensus [15], efficient analytics [20], as well as trust and security [10, 16, 21]. Through presentations and discussions with members of the research community, we sought to understand the position of edge-cloud research within the broader goals of the database and storage communities.

Our activities over the past five years have yielded insights and deepened our understanding of the real challenges and opportunities that lie ahead in the realm of building distributed edge-cloud databases. In this paper, we aim to summarize these experiences and insights with the goal of inspiring future research and industry products that can help in advancing and realizing edge-cloud databases.

In the remainder of this paper, we first provide an overview of AnyLog and how its distributed edge-cloud nature makes it different from existing databases in Sections 2 and 3. Then, we describe the prototype and scenarios of AnyLog in Section 4. The paper concludes in Section 5.

2 ANYLOG: AN EDGE-CLOUD DATABASE

AnyLog aims to transform the Edge[1] to a data infrastructure that is simple to scale and manage. Another way to view this is that AnyLog aims to extend cloud computing and data management functionalities to resources on the Edge.

At its core, AnyLog is implemented in 2 layers (Figure 2): The first is a *data layer* that brings data management functionalities to Edge nodes. These functionalities include a local (and pluggable) database that hosts data locally and is integrated with upper layers (such as a rule engine). AnyLog has connectors to several widely-used databases such as PostgreSQL, SQLite [8], and MongoDB (the connectors are extensible to integrate with the user's DBMS of choice). The functionalities are offered as services on each AnyLog node and are enabled by native commands. This approach makes deployment simple; by selecting services from an integrated and unified stack, a user configures the profile of each node, rather than developing proprietary solutions or integrating functionalities from diverse products. For example, when AnyLog is deployed, the user enables services that will make the AnyLog Node appear to data sources as a data broker, ingest data to the local databases, create schema based on ingested data, make the data secure, authenticate users, offer a rule engine that can act on data, events and machine state, make the data Highly Available, and more.

The second layer is the *coordination layer*. The coordination layer maintains a decentralized ledger of meta-information and system configurations and makes the information available to all the member nodes. This includes node memberships (i.e., what nodes are part of the AnyLog network), data maps (i.e., where data is stored, how it is replicated and distributed, the list of tables and their schema), and information that governs the access to data and the propagation of data from data producers to AnyLog nodes. The decentralized ledger is a separate module that can be implemented in various ways as long as it provides an interface that maintains a consistent log of updates to the various information maintained by the ledger. AnyLog currently has different ledger implementations that a user can choose from including ones based on permissionless and permissioned blockchain as well as a gossip-based log replication protocol.

The data and coordination layers achieve two important design concepts in AnyLog:

1. **Data virtualization through virtual tables:** AnyLog makes the distributed data appear as a unified collection of data. More specifically, using data virtualization, users and applications interact with a set of *virtual tables*. The data itself remains in-place (distributed and hosted in local databases of the Edge nodes). A query to a virtual table is satisfied by the nodes that contain data of that virtual table.
2. **Resource Virtualization through fluid virtual nodes:** AnyLog makes the distributed resources (all the nodes where AnyLog is deployed such as switches, gateways, and servers)

[1]Throughout this paper, we use the term Edge to represent the edge of the network where the users and devices are located. This includes end-user devices such as wearables and access points, base stations, and edge data centers which are clusters of machines deployed on-premises in university campuses, companies, smart cities, etc.

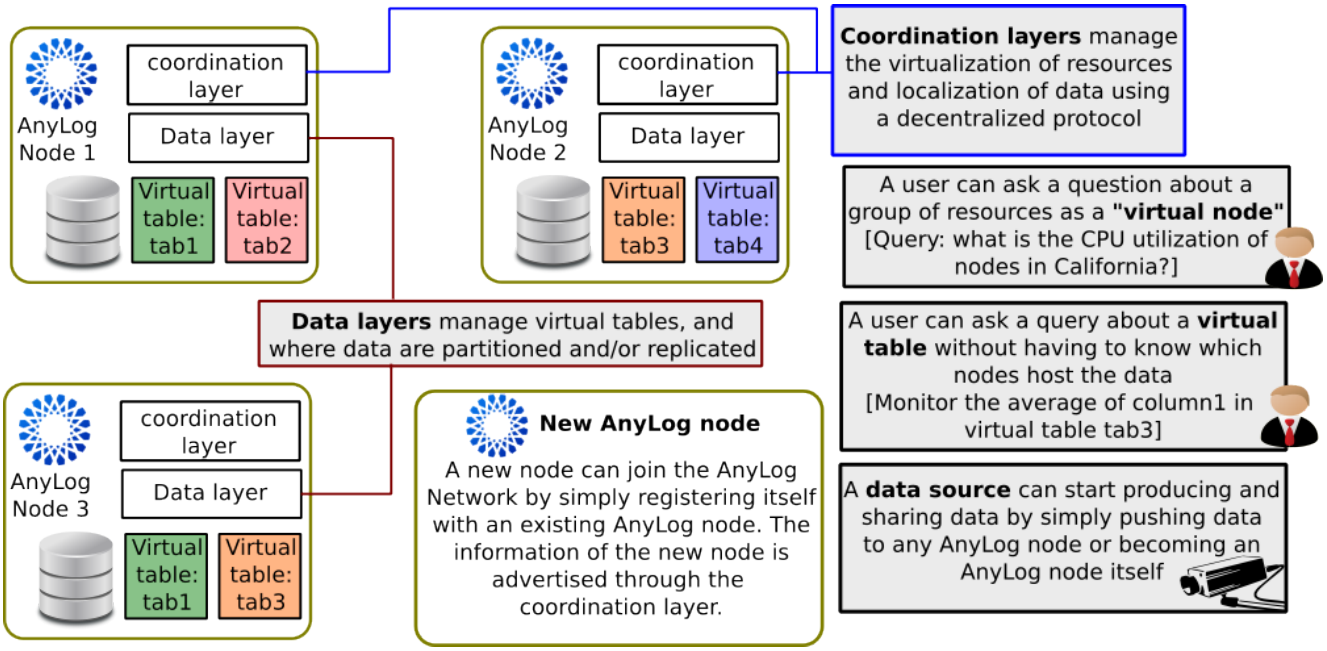


Figure 2: System overview of an Edge-Cloud AnyLog deployment.

managed as a single "virtual" node. A virtual node can represent one or more physical nodes and two virtual nodes may intersect in some of the virtual resources that they represent. We call this a *fluid virtual node*. With this setup, a user can query and monitor the status of a particular machine, or a group of machines using a single call. For example, a user can request—in a single call—the CPU usage of all the edge nodes deployed in California.

By allowing the management of distributed data from a single point, and by allowing the monitoring and management of edge resources from a single point, the Edge becomes as simple and intuitive environment as the Cloud: Applications query for the list of (virtual) databases, then select a table, then request the list of columns in the table, and issue a query. This is no different than querying a centralized (or a cloud-based) database. In the same manner, users can query the shared metadata dynamically for a profile of nodes (e.g. nodes in California, nodes associated with a data owner, nodes assigned with the data generated from a particular machine, etc.) and join the metadata response with a query for a specific state (e.g. disk usage, configuration setting, etc.) to an aggregate response from the identified nodes.

3 KEY DIFFERENCES TO EXISTING TECHNOLOGIES

AnyLog edge-cloud design represents many architectural differences from existing centralized, distributed, and cloud-based platforms. We list the main differences that we aim to showcase in the demonstration below:

Data decentralization rather than data partitioning. AnyLog is different from distributed databases (such as CockroachDB [22], Spanner [6], and CosmosDB [12]) in that rather than partitioning the data by a key, with AnyLog, each node is a candidate to host any data item. This approach is important for Edge data as we aim to keep data in place, where it is generated whereas a partitioned key will force the data to move to a target node (determined by the key) unrelated to the need in data locality. In addition, the nature of edge data management is that new nodes continuously join (or drop from) the network. This dynamic nature of the Edge will not be served well when the data is partitioned by a key as data will need to be continuously redistributed. With AnyLog, when a node receives data assigned to a (virtual) table, the node will register itself as being one of the nodes managing the table's data (once) on the shared metadata. From thereon, the node will participate in queries that consider the table's data. This approach allows nodes to immediately join the network and there is no need for synchronization or redistribution of data.

Decentralization of resources. The traditional distributed databases aim at a relatively small number of nodes whereas AnyLog scales to a large number of nodes. The reason is that for a given query, only a subset of nodes participate in the query process and the remaining nodes are not impacted (i.e., a query to provide the electricity usage in San Francisco in the last 2 hours would be processed only on the subset of nodes with electricity usage in San Francisco, whereas the entire network may serve data related to all cities in the world as well as data of other use cases, and these nodes are not involved in the query for electricity usage in SF). By distributing the data to multiple nodes, a query is processed on multiple nodes concurrently with a high degree of parallelism. Whenever possible, functions are pushed to the edge nodes minimizing data transfer. Regardless if functions are pushed, this setup is efficient as it avoids the overhead of centralizing the data. Also, this approach scales horizontally. Companies can adjust

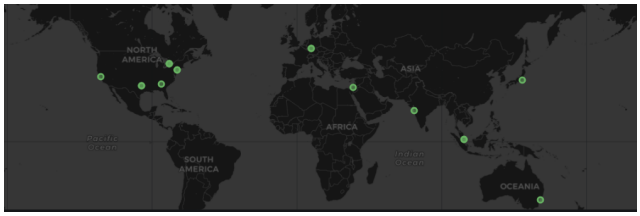


Figure 3: AnyLog Test Network (TestNet). Each green dot represents a location that hosts one or more AnyLog nodes.

to the needed performance by dynamically adding nodes. Each added node increases the distribution of the data and therefore the degree of parallelism.

Orchestration across independent local databases. AnyLog is a platform that is layered on top of local databases; it makes the group of all local databases at the Edge appear as a single machine, and it provides functionality not only to manage distributed data, but also to manage the edge resources. For example, the network will satisfy lookups on access statistics, disk, and networking usage. The shared metadata serves as a single point to control all the distributed nodes. If a policy is added (to the metadata) or changed, it becomes available to all the nodes, and nodes are designed such that their functionality and operations are derived from the policies (and configurations). Each AnyLog node maintains a rule engine that can act on data and resource state and offers APIs (like REST and Pub-Sub) to connect to data sources and applications.

4 ANYLOG FUNCTIONALITY SHOWCASE

We present a demonstration of AnyLog's main features such as data and resource virtualization (Section 2) as well as its key differences compared to traditional distributed databases (Section 3). In this section, we describe our experimental setup—which we have utilized in our demonstrations with many industry and academic teams—that we will utilize in this section. Then, we will present a mock-up of how the functionality overview will be structured (given space limitation, we only discuss this part for a representative subset of the demonstration).

4.1 AnyLog Test Network

The AnyLog Test Network (TestNet) is a decentralized network of nodes with AnyLog deployed and configured on each node (Figure 3). 20 Nodes participate in the TestNet and these nodes are deployed all over the world.

The nodes in the network coordinate using a shared metadata layer that is hosted in a decentralized ledger (blockchain). The shared metadata is a collection of JSON scripts (we call each script a policy), and the blockchain serves as a platform that makes the policies available to all the participating nodes consistently (regardless of the nodes or users that created the policies). For example, when data is streamed from a device to a node, the node determines if a schema that satisfies the data exists (on the shared metadata). If a schema exists, the node will use the existing schema to host the data in a local database. If the schema does not exist, the node will create the schema, represent the schema as a policy, and publish

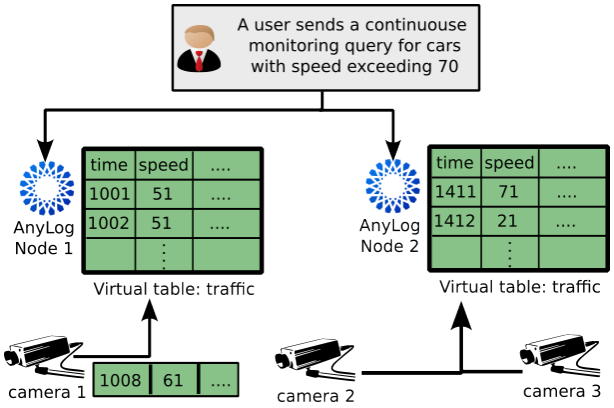


Figure 4: An overview of the demo setup of cameras monitoring traffic and sending data to AnyLog nodes with information about car speeds.

the schema to be part of the shared metadata (so it will be available to a different node that ingests similar data).

Unlike the metadata, the data is hosted in local databases on the participating nodes. The architecture is such that users can determine the physical database (which can be different on each node). This is done as different setups may have different preferences for the database to use. In the TestNet, we use PostgreSQL for larger nodes, SQLite for switches and gateways, and MongoDB for unstructured data (to host video and images). By default, the created schemas are based on the self-describing nature of the device data (there is a variety of platforms that satisfy the device API and deliver self-describing data, for example, EdgeX Foundry[2]). In addition, AnyLog offers a high-level scripting language that can map the structure of ingested data to a target structure or enforce a target schema whenever the default mapping (to create the schema or satisfy the schema) is not sufficient.

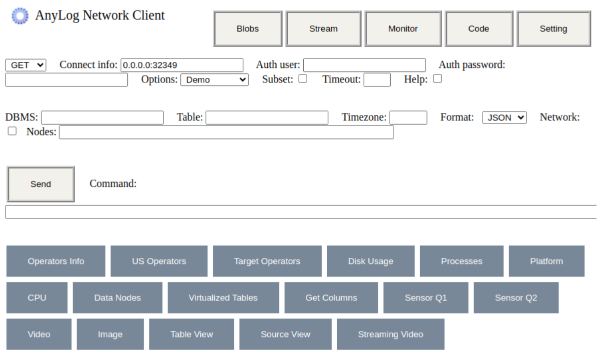
On each node, the data is organized by time, and as most queries to sensor data provide a time interval, lookups are directed to the relevant rows (rather than full table scans). The partitioning by time allows to remove and archive old data without downtime, and in most deployments users place a rule (on the rule engine) that removes old partitions (i.e., keep the most recent partitions or remove old partitions if disk space is under a threshold).

Queries, commands, and user requests are sent to AnyLog nodes using a REST interface. A request can be issued to any one of the AnyLog nodes in the network, and the AnyLog node receiving the request (or query or command) acts as an Orchestrator by routing the request to the peer nodes that need to process the request or with the relevant data. The participating node processes the request locally and replies to the Orchestrator that organizes the replies and returns a unified reply to the user or application.

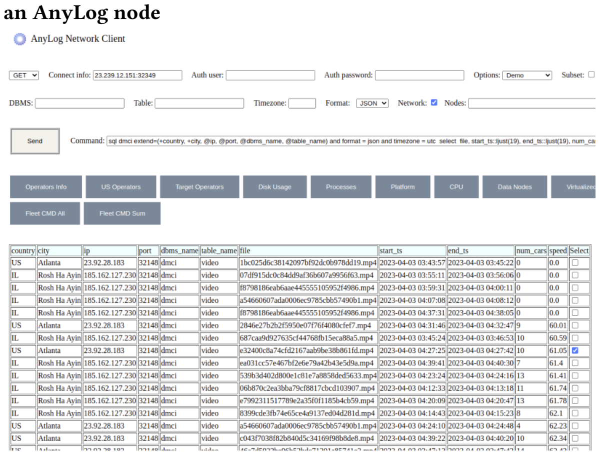
4.2 Demo Structure

The proposed demo showcases the unique challenges, features, and opportunities of edge-cloud data management with AnyLog. The following is a representative set of the demonstration parts.

[2]https://www.edgexfoundry.org/



(a) The client interface to write SQL queries and send them to an AnyLog node



(b) An example of the SQL response to a query asking about cars and their speed

(c) An example showing streaming the video that corresponds to a data entry in the response

Figure 5: Screenshots of the client interface to be used in the demo of AnyLog

The demo will utilize a video analytics application. The application is for traffic management, where a group of cameras streams images from roads and intersections to AnyLog nodes (Figure 4). Three cameras with integrated AI continuously process images and extract information about detected cars and their speeds. Then, the AI generates a data record for the time intervals (that contains the number of cars identified, the speed, location, and other information) and sends it to the closest AnyLog node. The AnyLog node that receives the data, stores the records in a table that corresponds to the schema of the records (called the *traffic table*), and the images are stored, on the same local node using a blob database.

Virtual tables. The first feature that is presented in the demo is the concept of virtual tables. In the demo, we send a command to the AnyLog network (specifically, to a node in the network) to display the virtual tables (i.e., the command is *get virtual tables*, which serves as exploring the list of tables). The output of this command is the list of all databases, and for each database, the list of all tables. One of these tables is the *traffic* table which contains the information about the traffic video generated by the AI process.

User queries. The next part of the demo is to show how the user issues queries to retrieve data from the traffic table without the need to know which are the nodes that host the table's data (to support an administrator, users can query the network to determine, for each virtual table, which are the nodes that host the data). To make the demo more intuitive, we have built a client interface shown in Figure 5. The client interface enables users to send AnyLog-specific commands (e.g., *get virtual tables;*) and SQL queries about the data in the virtual tables that are managed by the network. The screenshots in the figure show the initial interface (Figure 5(a)) with the different options that the user can choose and a text space to write commands and SQL queries. When the user sends a request, it is sent to the AnyLog node specified by the chosen IP address. This node will either process the request or act as an Orchestrator to determine the target peer nodes, deliver the request, and aggregate and unify the reply from the peers that are involved in the process.

Figure 5(b) shows the response to a query to list all cars in the traffic table that were recorded in the previous hour ordered by their speed (the following SQL query is shortened for ease of exposition):

```sql
SELECT start_ts, end_ts, num_cars, speed
FROM traffic
WHERE start_ts >= NOW() - 1hour and end_ts <= NOW()
ORDER BY speed;
```

The response to the query is collected from all AnyLog nodes that contain data relevant to the traffic virtual table (2 nodes support the traffic table in the demo), without specifying in the query which are the nodes that host the data. These target nodes are determined transparently by the Orchestrator by considering the information contained in the query text and the information in the metadata.

In the next part, we show a feature of AnyLog for multimedia data. Each data item in the traffic table represents the AI derivatives (like speed) associated with the video segment. These video segments are maintained in-place using an object database in the local AnyLog node and are not centralized. If the user is interested in streaming the video assigned to an AI insight, (e.g., the video of a speeding car), then the user can click on that data item. The AnyLog client interface then sends a request to stream the corresponding

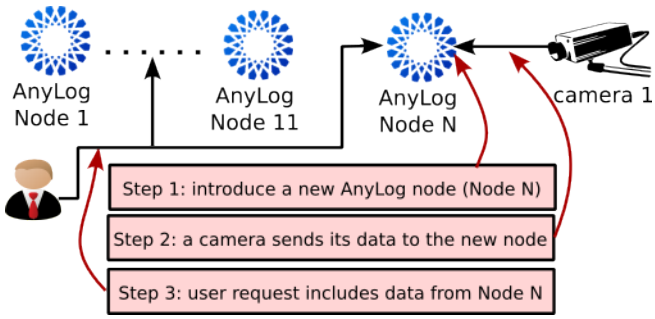


Figure 6: The steps in the demo scenario to show the seamless plug-and-play deployment of AnyLog nodes and the decentralization of data and resources.

video clip from the AnyLog node at the Edge. Figure 5(c) shows the outcome of such a request.

Decentralization. The next part of the demo (Figure 6) will showcase the decentralization nature of the AnyLog Network. In this part, during the normal operation of AnyLog, we will add a new AnyLog node. We will show how the process of joining and participating in the AnyLog network is seamless. First, the new AnyLog node N sends its information to the rest of the network as a new policy published in the decentralized ledger. Once the policy is added, AnyLog nodes can coordinate with node N. Then, we make one of the cameras send frames to the new node N. When the node N receives the data for the traffic virtual table, it discovers that data with the same schema and table name exist by querying the policies in the ledger. Therefore, it only needs to register itself as a node that hosts data for the traffic table (this process only needs to be done once). This makes node N a node that will be considered (by the Orchestraor node) when queries to the traffic table are issued. Note that this process is efficient as there is no redistribution of data and as the new node N joins the process without any dependencies on peer members or external (to the node) processes.

Other features. Due to space limits, we do not describe in detail the other features and properties of AnyLog that we will include in the demo. However, we list them here briefly. These features and scenarios include sending commands that inquire about nodes' statistics and hardware utilization, tolerating failures and nodes disappearing, complex queries that manipulate data from various virtual tables, and consistency issues that may arise due to the decentralized nature of AnyLog coordination.

5 CONCLUSION

AnyLog is bridging this gap by delivering a single and unified stack that replaces engineering efforts with services that are enabled on edge nodes and delivers a complete and integrated solution that: (a) allows a plug-and-play approach to deploying edge instances, (b) has a protocol that treats and serves all the decentralized data as a single unified collection of data, (c) treats and monitors all the edge resources as a single unified machine, and (d) scales horizontally to address the growth of the data at the edge to services the edge data with low latency to AI processes and edge and cloud applications. All of that is without the performance and monetary overheads of centralizing the data.

6 ACKNOWLEDGMENTS

This research is supported in part by the NSF under grants CNS-1815212 and SaTC-2245372.

REFERENCES

[1] [n. d.]. InfluxDB. https://www.influxdata.com/.
[2] Daniel Abadi, Anastasia Ailamaki, David Andersen, Peter Bailis, Magdalena Balazinska, Philip A Bernstein, Peter Boncz, Surajit Chaudhuri, Alvin Cheung, Anhai Doan, et al. 2022. The Seattle report on database research. *Commun. ACM* 65, 8 (2022), 72–79.
[3] Daniel Abadi, Owen Arden, Faisal Nawab, and Moshe Shadmon. 2020. Anylog: a grand unification of the internet of things. In *Conference on Innovative Data Systems Research (CIDR '20)*.
[4] George Amvrosiadis, Ali R Butt, Vasily Tarasov, Erez Zadok, Ming Zhao, Irfan Ahmad, Remzi H Arpaci-Dusseau, Feng Chen, Yiran Chen, Yong Chen, et al. 2018. Data Storage Research Vision 2025: Report on NSF Visioning Workshop held May 30–June 1, 2018.
[5] Ronald Barber, Christian Garcia-Arellano, Ronen Grosman, Guy Lohman, C Mohan, Rene Muller, Hamid Pirahesh, Vijayshankar Raman, Richard Sidle, Adam Storm, et al. 2019. Wiser: A highly available HTAP DBMS for iot applications. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 268–277.
[6] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. 2013. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)* 31, 3 (2013), 1–22.
[7] Michael J Franklin, Shawn R Jeffery, Sailesh Krishnamurthy, Frederick Reiss, Shariq Rizvi, Eugene Wu, Owen Cooper, Anil Edakkunni, and Wei Hong. 2020. Design Considerations for High Fan-in Systems: The HiFi Approach. In *Conference on Innovative Data Systems Research (CIDR)*.
[8] Kevin P Gaffney, Martin Prammer, Larry Brasfield, D Richard Hipp, Dan Kennedy, and Jignesh M Patel. 2022. Sqlite: past, present, and future. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3535–3547.
[9] Samaa Gazzaz, Vishal Chakraborty, and Faisal Nawab. 2022. Croesus: Multi-stage processing and transactions for video-analytics in edge-cloud systems. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1463–1476.
[10] Suyash Gupta, Sajjad Rahnama, Erik Linsenmayer, Faisal Nawab, and Mohammad Sadoghi. 2023. Reliable transactions in serverless-edge architecture. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 301–314.
[11] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. 2005. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)* 30, 1 (2005), 122–173.
[12] Microsoft. [n. d.]. Introduction to Azure Cosmos DB. https://learn.microsoft.com/en-us/azure/cosmos-db/introduction.
[13] Natasha Mittal and Faisal Nawab. 2021. Coolsm: Distributed and cooperative indexing across edge and cloud machines. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 420–431.
[14] Rene Mueller, Gustavo Alonso, and Donald Kossmann. 2007. SwissQM: Next generation data processing in sensor networks. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*. CIDR.
[15] Faisal Nawab, Divyakant Agrawal, and Amr El Abbadi. 2018. Dpaxos: Managing data closer to users for low-latency and mobile applications. In *Proceedings of the 2018 International Conference on Management of Data*. 1221–1236.
[16] Faisal Nawab and Mohammad Sadoghi. 2019. Blockplane: A global-scale byzantizing middleware. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 124–135.
[17] Dan O'Keeffe, Theodoros Salonidis, and Peter Pietzuch. 2018. Frontier: Resilient edge processing for the internet of things. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1178–1191.
[18] John Paparrizos, Chunwei Liu, Bruno Barbarioli, Johnny Hwang, Ikraduya Edian, Aaron J Elmore, Michael J Franklin, and Sanjay Krishnan. 2021. VergeDB: A Database for IoT Analytics on Edge Devices.. In *CIDR*.
[19] Yuya Sasaki. 2021. A survey on IoT big data analytic systems: Current and future. *IEEE Internet of Things Journal* 9, 2 (2021), 1024–1036.
[20] Abhishek A. Singh, Aasim Khan, Sharad Mehrotra, and Faisal Nawab. 2023. TransEdge: Supporting Efficient Read Queries Across Untrusted Edge Nodes. In *Proceedings 26th International Conference on Extending Database Technology, EDBT*. 684–696. https://doi.org/10.48786/EDBT.2023.57
[21] Abhishek A. Singh, Yinan Zhou, Mohammad Sadoghi, Sharad Mehrotra, Shantanu Sharma, and Faisal Nawab. 2023. WedgeBlock: An Off-Chain Secure Logging Platform for Blockchain Applications. In *Proceedings 26th International Conference*

on Extending Database Technology, EDBT. 526–539. https://doi.org/10.48786/EDBT.2023.45

[22] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, et al. 2020. Cockroachdb: The resilient geo-distributed sql database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1493–1509.

[23] Chen Wang, Xiangdong Huang, Jialin Qiao, Tian Jiang, Lei Rui, Jinrui Zhang, Rong Kang, Julian Feinauer, Kevin A McGrail, Peng Wang, et al. 2020. Apache IoTDB: Time-series database for internet of things. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2901–2904.

[24] Steffen Zeuch, Ankit Chaudhary, Bonaventura Del Monte, Haralampos Gavriilidis, Dimitrios Giouroukis, Philipp M Grulich, Sebastian Bress, Jonas Traub, and Volker Markl. 2020. The NebulaStream Platform: Data and Application Management for the Internet of Things. In *Conference on Innovative Data Systems Research (CIDR)*.