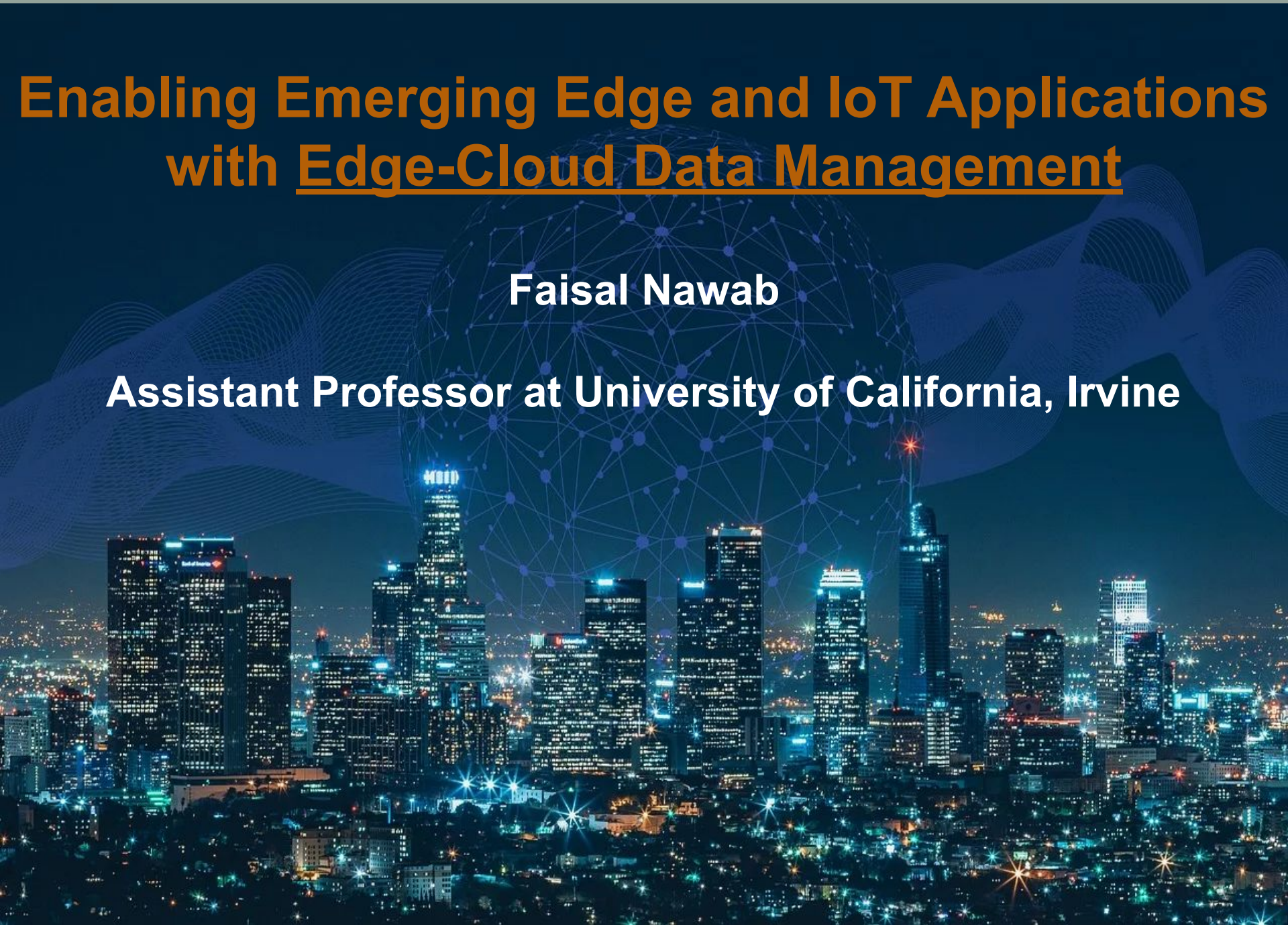
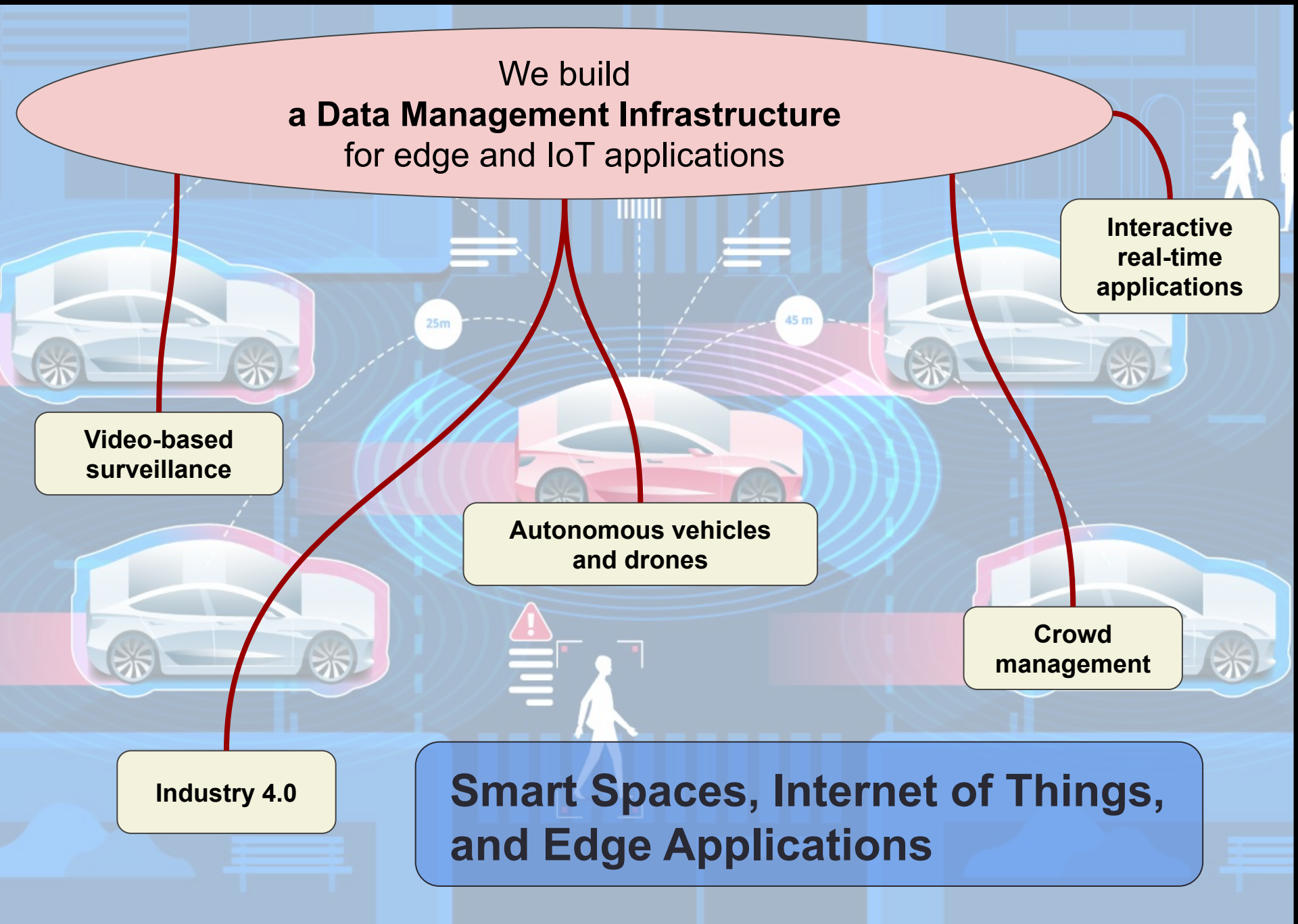
Enabling Emerging Edge and IoT Applications with Edge-Cloud Data Management

Faisal Nawab

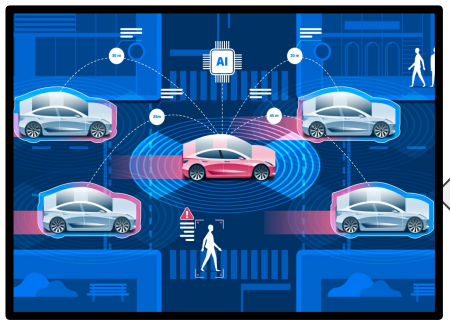
Assistant Professor at University of California, Irvine

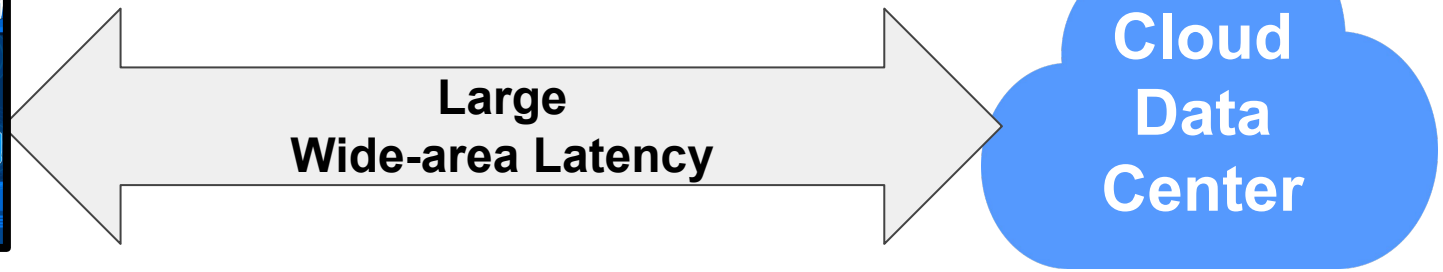



We build
a Data Management Infrastructure
for edge and IoT applications
Interactive real-time applications
Video-based surveillance
Autonomous vehicles and drones
Crowd management
Industry 4.0
Smart Spaces, Internet of Things, and Edge Applications
25m
45 m

What's missing?

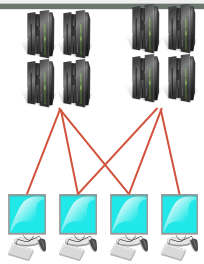
- Manage data in **real-time, close to users**
- The cloud (and centralization) got as far, but now is time for "the next step"
  - High wide-area latency
  - Communication throughput demands
  - Regulations about using the cloud
- **What's the next step?**
  - Let's reflect on the history of computing



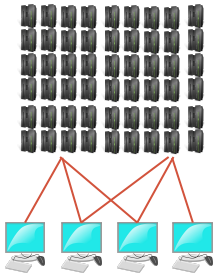


Consolidation/Centralization (higher throughput/resource utilization)

Mainframes
1960-70s



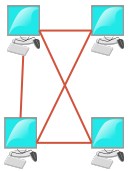
Client-Server
1990s



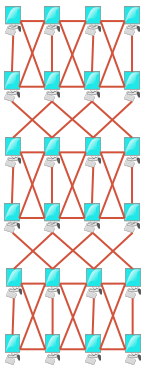
Cloud
2010s

Global Edge-Cloud

PCs and Networked Machines
1980s

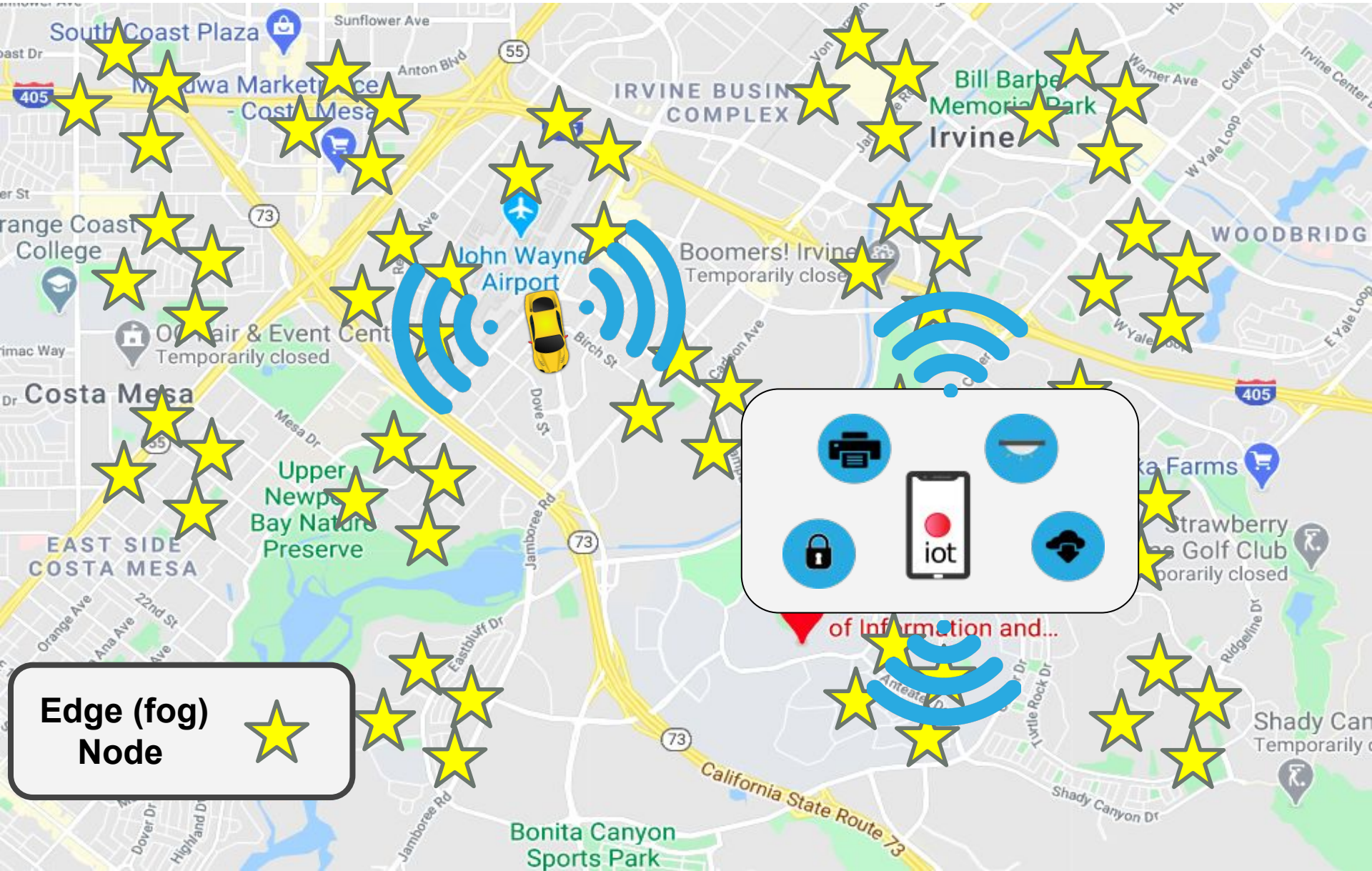


Edge computing & peer-to-peer
2000s



Distribution (lower latency/client resources)

Global Edge Cloud: the next phase



AnyLog

Data Management Infrastructure for the Global-Edge Cloud

CIDR 2020

AnyLog: a Grand Unification of the Internet of Things

Daniel J. Abadi
University of Maryland
abadi@cs.umd.edu

Owen Arden
University of California, Santa Cruz
owen@soe.ucsc.edu

Faisal Nawab
University of California, Santa Cruz
fnawab@ucsc.edu

Moshe Shadmon
AnyLog
moshe@anylog.co

CIDR 2024

The Tipping Point of Edge-Cloud Data Management

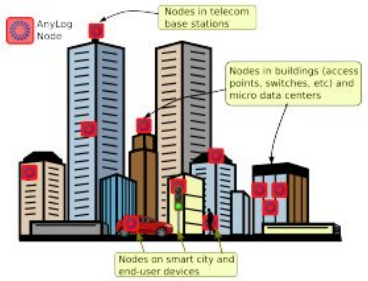
Faisal Nawab
nawabf@uci.edu
UC Irvine

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

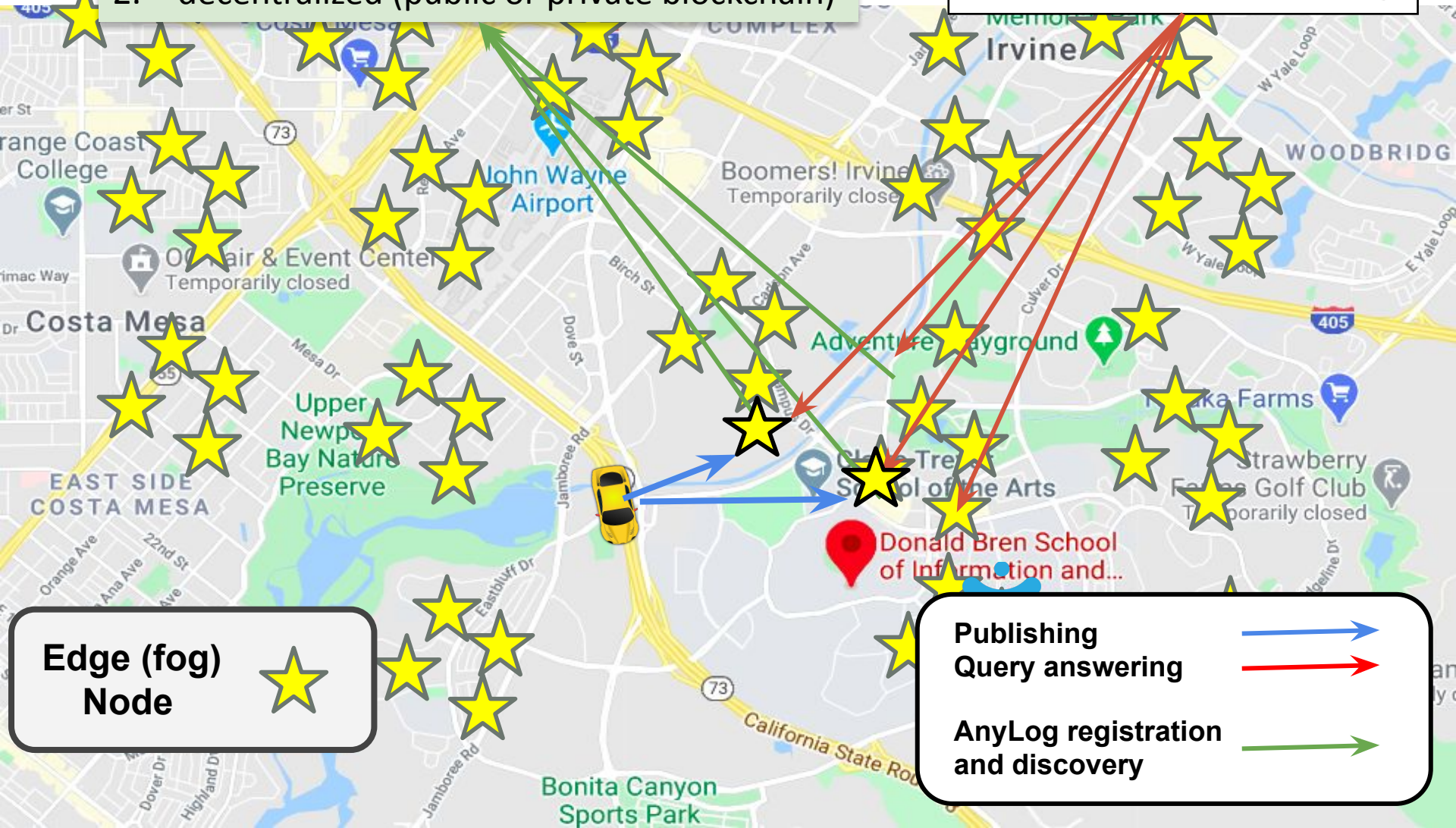


A Data Infrastructure for the Global Edge Cloud

Sharing and coordination metadata layer

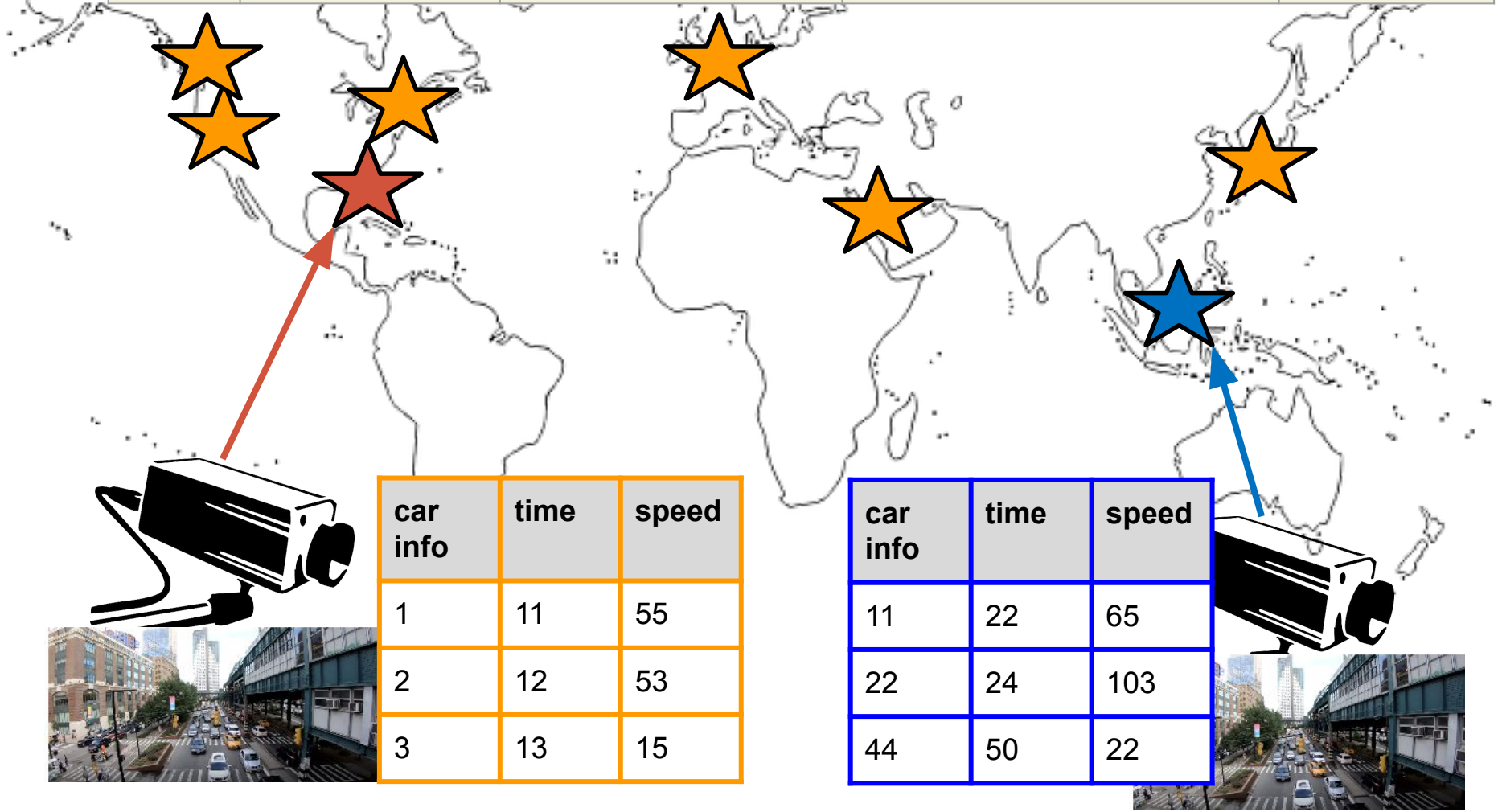
1. centralized (cloud), or
2. decentralized (public or private blockchain)

Query: List the cars that entered UCI this morning



AnyLog Deployment

Coordination and Metadata Layer				
…	Register Node Info. Policies	Node Atlanta contains data about Table Cars	Node Singapore replicates data to Node AAA and BBB	…

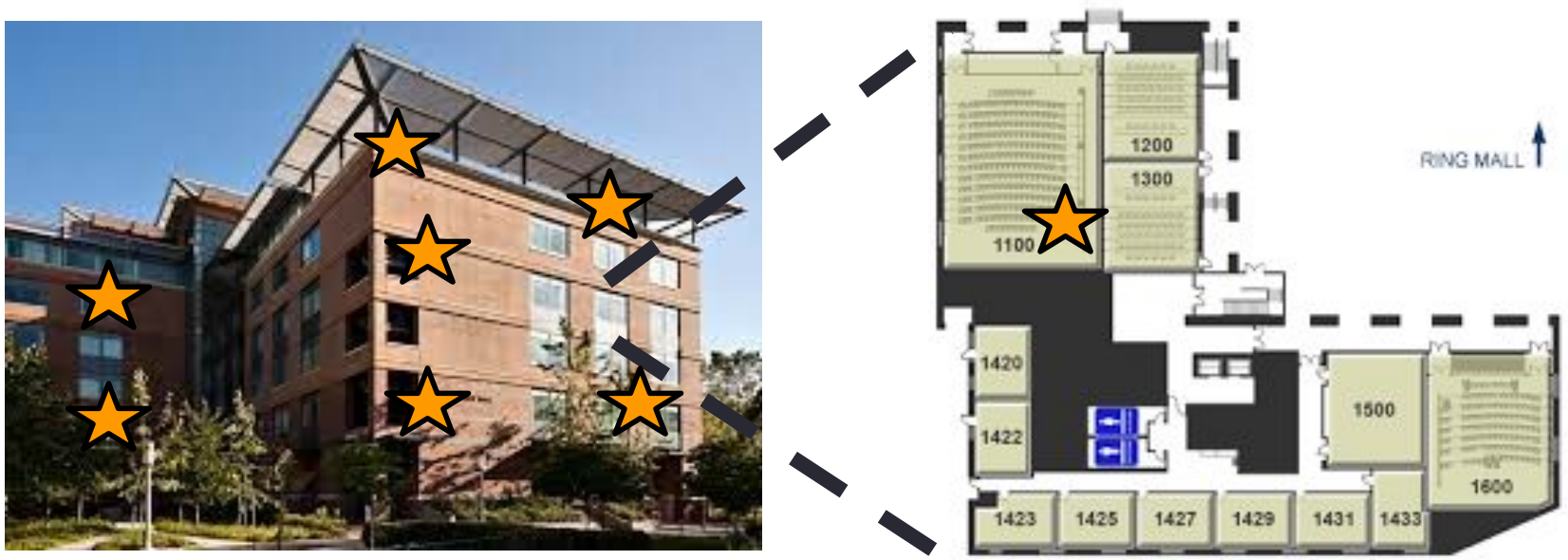


car info	time	speed
1	11	55
2	12	53
3	13	15

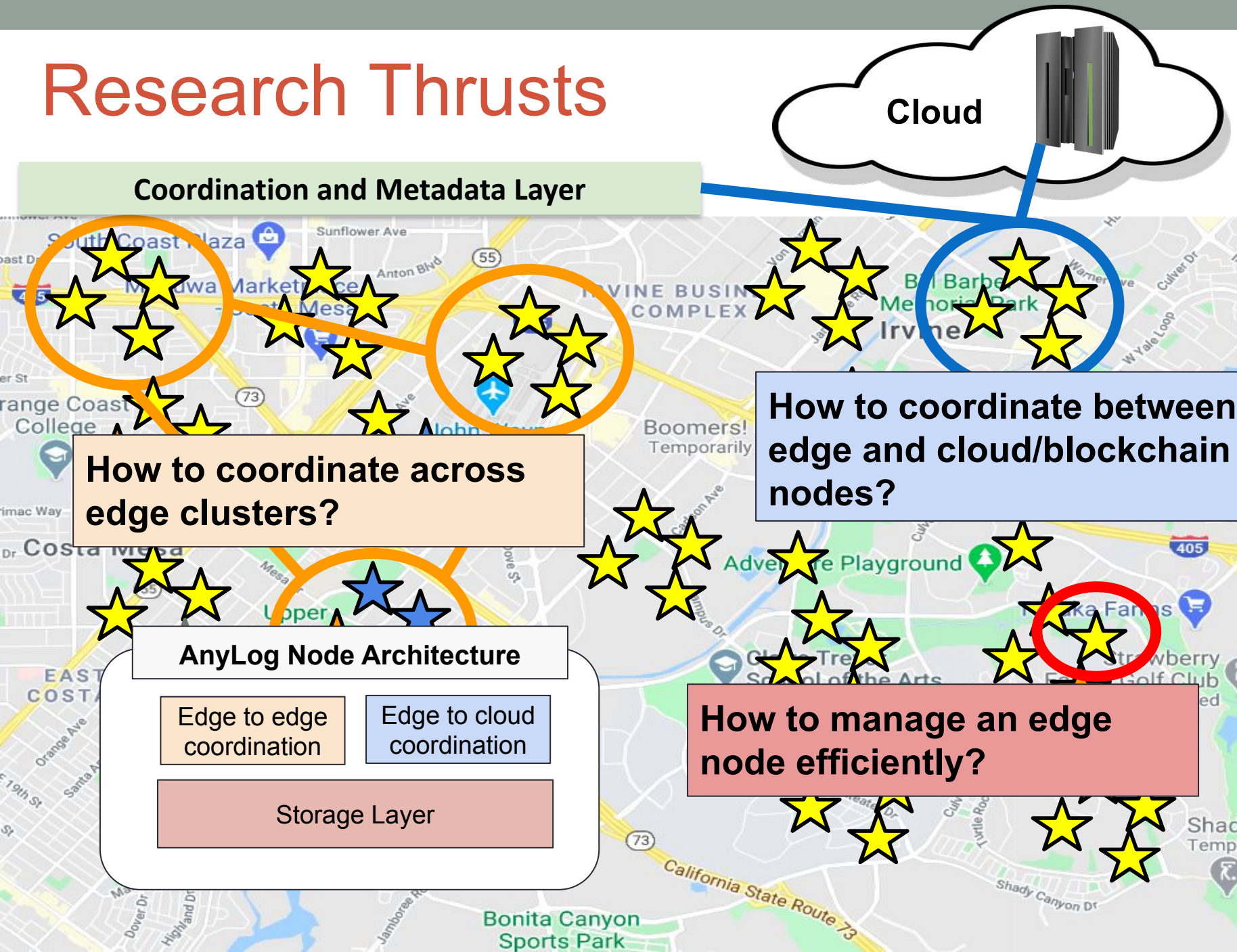
car info	time	speed
11	22	65
22	24	103
44	50	22

AnyLog deployments

- Now available to install
  - Pip package and virtual machines
- Partners from industry and academia
  - Smart city, edge, and IoT technology industry



Research Thrusts



1. Edge Node Efficiency (Energy and Endurance)

2018	2019	2020	2021	2022	2023
			Predict n' Write [ICDE'21] **NVM Lifetime** [VLDB'21Tutorial]		**Hamming Tree** [SIGMOD'23] **E2-NVM** [EDBT'23]

2. Edge-to-Edge Coordination

2018	2019	2020	2021	2022	2023
DPaxos [SIGMOD'18] **Nomadic** [EDBT'18]	**BlockPlane** [ICDE'19] **IoT Blockchain** [IEEE IoT'19]	**AnyLog** [CIDR'20]		**PeloPartition** [Blockchain'22]	**TransEdge** [EDBT'23]

3. Edge-to-Cloud (or blockchain) Coordination

2018	2019	2020	2021	2022	2023
Edge-Video [VLDB'18Demo]			**CooLSM** [ICDE'21] **WedgeChain** [ICDE'21]	**Croesus** [ICDE'22] **FiME** [Blockchain'22]	**ServerlessBFT** [ICDE'23] **WedgeBlock** [EDBT'23]

1. Edge Node Efficiency (Energy and Endurance)

2018	2019	2020	2021	2022	2023
			Predict n' Write [ICDE'21] **NVM Lifetime** [VLDB'21Tutorial]		**Hamming Tree** [SIGMOD'23] **E2-NVM** [EDBT'23]

2. Edge-to-Edge Coordination

2018	2019	2020	2021	2022	2023
DPaxos [SIGMOD'18] **Nomadic** [EDBT'18]	**BlockPlane** [ICDE'19] **IoT Blockchain** [IEEE IoT'19]	**AnyLog** [CIDR'20]		**PeloPartition** [Blockchain'22]	**TransEdge** [EDBT'23]

3. Edge-to-Cloud/blockchain Coordination

2018	2019	2020	2021	2022	2023
Edge-Video [VLDB'18Demo]			**CooLSM** [ICDE'21] **WedgeChain** [ICDE'21]	**Croesus** [ICDE'22] **FiME** [Blockchain'22]	**ServerlessBFT** [ICDE'23] **WedgeBlock** [EDBT'23]

WedgeChain and WedgeBlock

Lazy (Asynchronous) Trust

Nawab [ICDE'2021], and

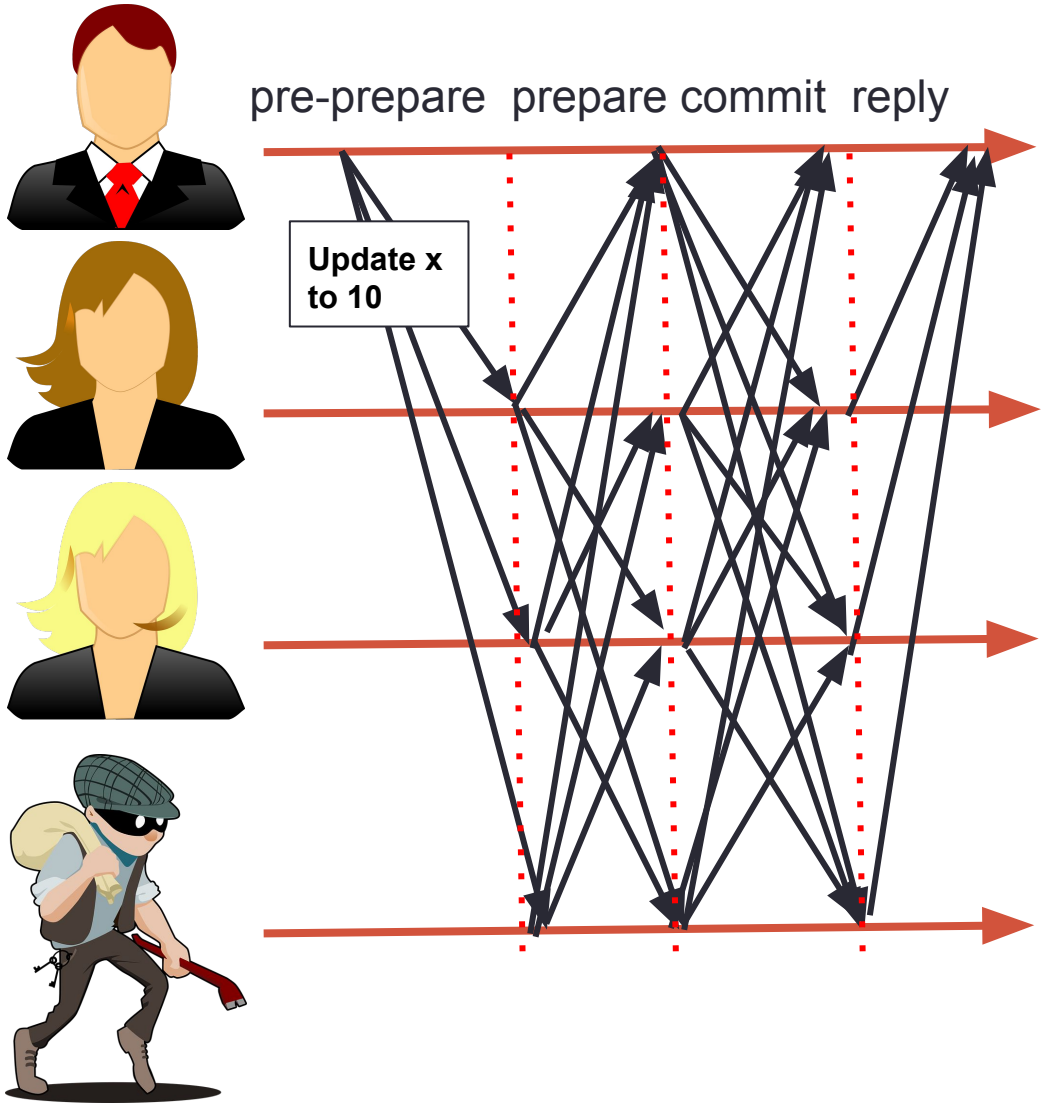
Singh, Zhou, Sadoghi, Mehrotra, Sharma, Nawab [EDBT'2023]

Tolerating Malicious Activity

- Edge nodes can be malicious
- The old way to tolerate malicious activities: control all operation to ensure no one can act maliciously
  - But it is very expensive!

Old way #1: Byzantine Fault Tolerance

Many rounds of communication to detect lies



- Byzantine FT protocols [LAMPORT, L., SHOSTAK, R., & PEASE, M 1982], e.g., PBFT [OSDI'99],
- Expensive communication rounds and message complexity
- Must make an assumption about the *maximum number of malicious nodes*
- not suitable for the edge!

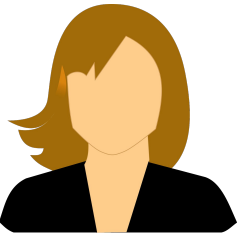
Old way #2: utilize a trusted entity

A detective "prevents" malicious activity

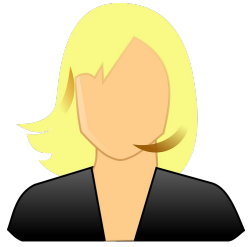
publishers
& Clients

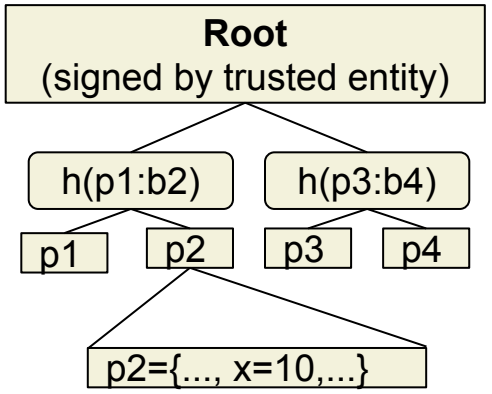
Update x to 10

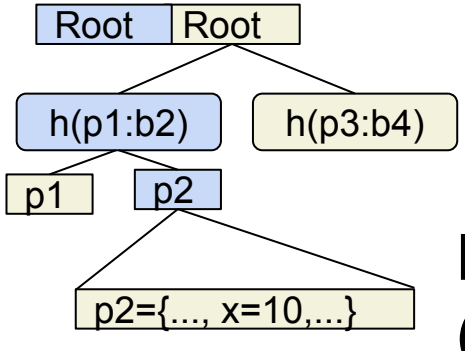
Trusted Node
(cloud or
blockchain node)



read(x)







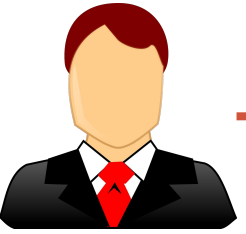
Send new Merkle Tree with x=10

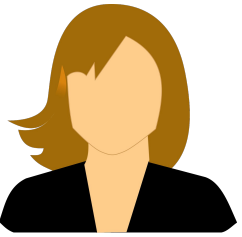
Edge Node
(untrusted)

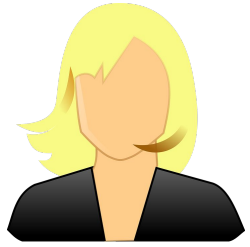
Old way #2: utilize a trusted entity

A detective "prevents" malicious activity

publishers & Clients







But…

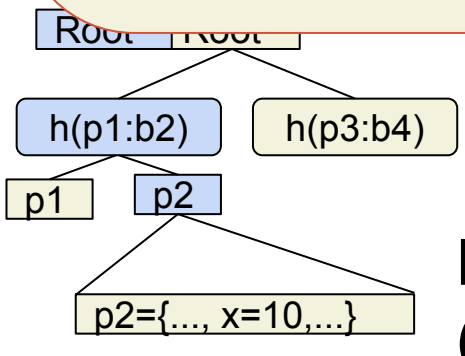
This leads to

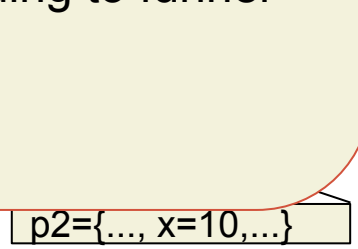
(1) **High latency.** Trusted entity is far (cloud) or slow (blockchain)

(2) **Low throughput.** Due to needing to funnel everything to a single entity

Trusted Node (cloud or blockchain node)

Edge Node (untrusted)



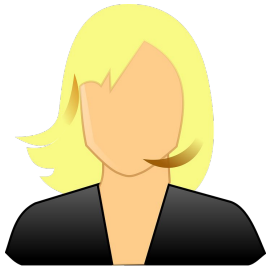


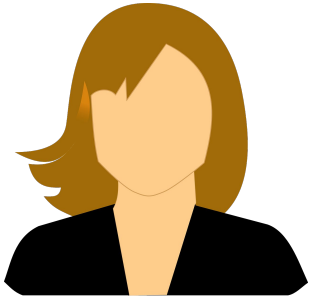
Send new Merkle Tree with x=10

A New Way to Do Trusted Computation!

Design Principle Alert!

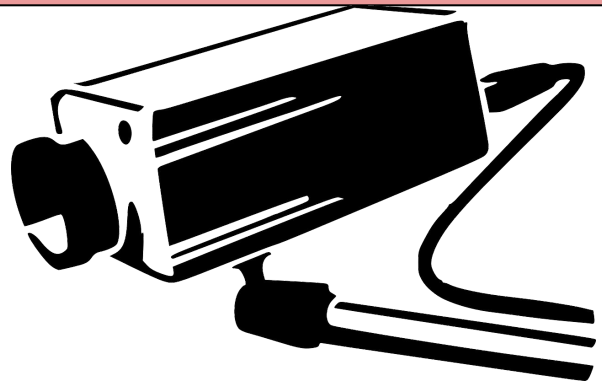
- Observation: We do not trust the edge nodes, but **we know who they are!**
  - Generally true for permissioned blockchain too
- Design: **Allow malicious activity**, but
  - **guarantee they are detected**… and punished!
- related to auditing in byzantine systems:
  - PeerReview [SOSP'07], Fides [ICDCS'20]





Proposal: Lazy Trust

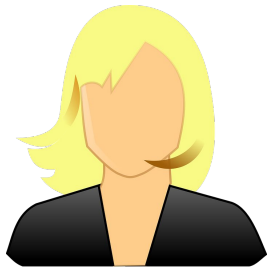
Allow malicious activity but *punish* it eventually

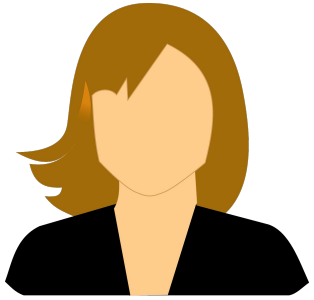


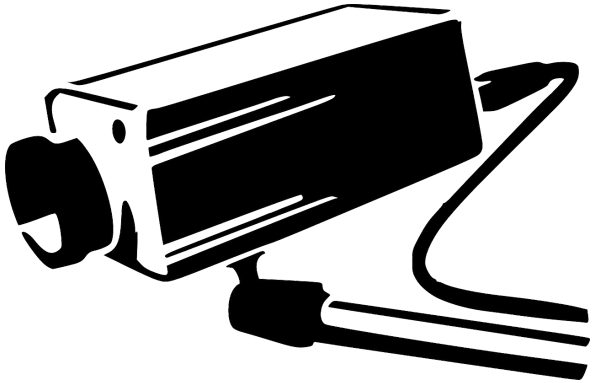
A New Way to Do Trusted Computation!

How to guarantee detecting and punishing malicious activity?

- **Utilize a trusted component:**
  - **(a) a trusted cloud node, or**
  - **(b) a blockchain smart contract**



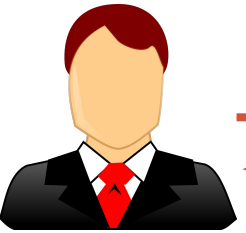




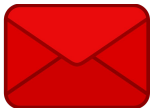
Proposal: Lazy Trust

Allow malicious activity but *punish* it eventually

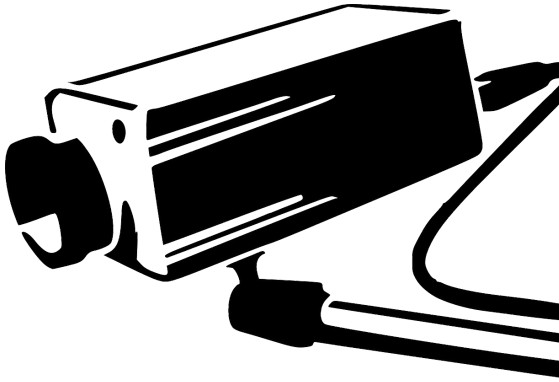
publishers & Clients



Update x to 10

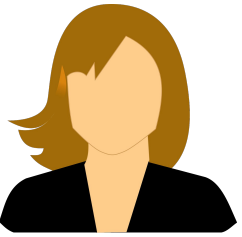


I added x=10 to log entry #1



Trusted Node

Log #1 {..x=10..}	Log #2 {..x=15..}



read(#5

signed(digest (Log #1 {..x=10..}))

TRU

Log # {..x=10..}

OR

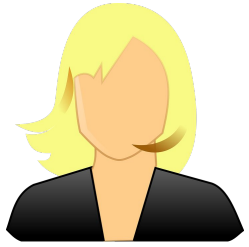
LIE

Log #1 {..x=6..}

Edge Node

Lazy-Certify digest (Log #1 {..x=10..})

signed (digest (Log #1 {..x=10..}))



Edge Processing (Phase I Commit)

Lazy Certification (Phase II Commit)

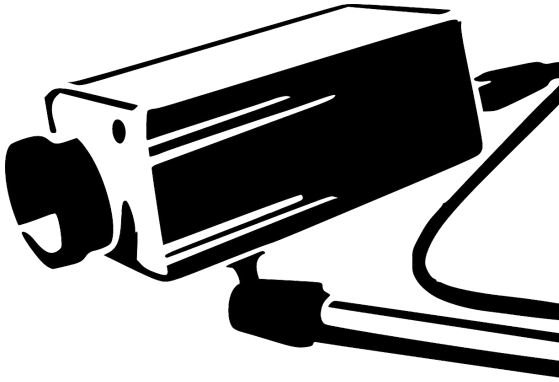
publishers
& Clients

Proposal: Lazy Trust

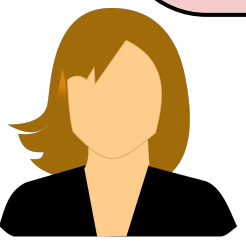
Allow malicious activity but *punish* it eventually

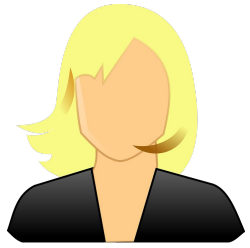
Great! But…

Each data object is in a separate log record. How can we provide a data index for lazy certification?



Trusted Node





read(#5

signed(digest (Log #1 {..x=10..}))

TRU

Log # {..x=10..}

OR

LIE

Log #1 {..x=6..}

Edge Node

Lazy-Certify digest (Log #1 {..x=10..})

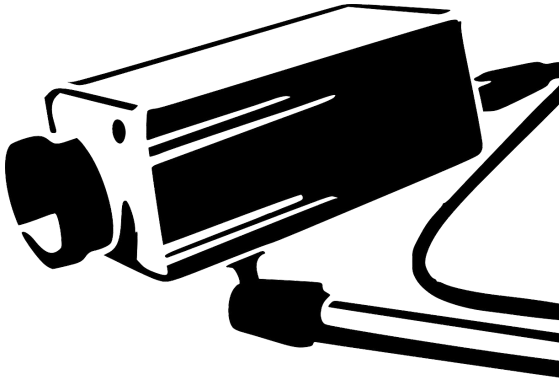
signed (digest (Log #1 {..x=10..}))

Edge Processing (Phase I Commit)

Lazy Certification (Phase II Commit)

Indexing for Lazy Trust

- Possible solution: the trusted node provides a merkle tree
- Merkle tree is ordered and can enable more efficient access
- But, updating the merkle tree for each update is expensive!

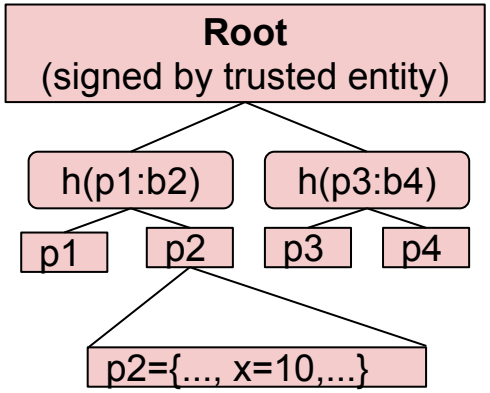


Trusted Node

Lazy-Certify digest ({..x=10..})

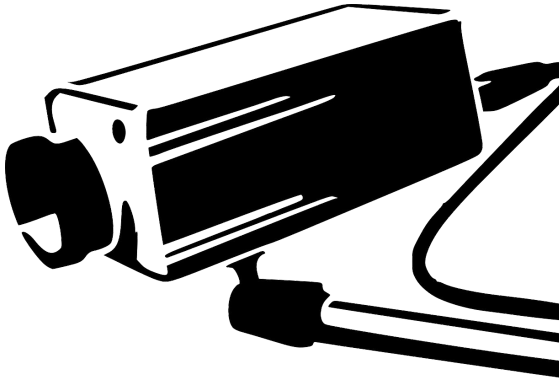
update(x=10)

Edge Node



Proposal: LSMerkle
An Index for Lazy Trust

- **LSMerkle:** a specialized index for lazy trust and designed for efficient data ingestion
- (inspired from LSM Trees)
- Divide the index tree into levels
  - Level 0: most recent data
  - Level 1: compactions of older data
  - Level 2: compactions of oldest data



Trusted Node

update(x=10)

Edge Node

Lazy-Certify digest ({..x=10..} **)**

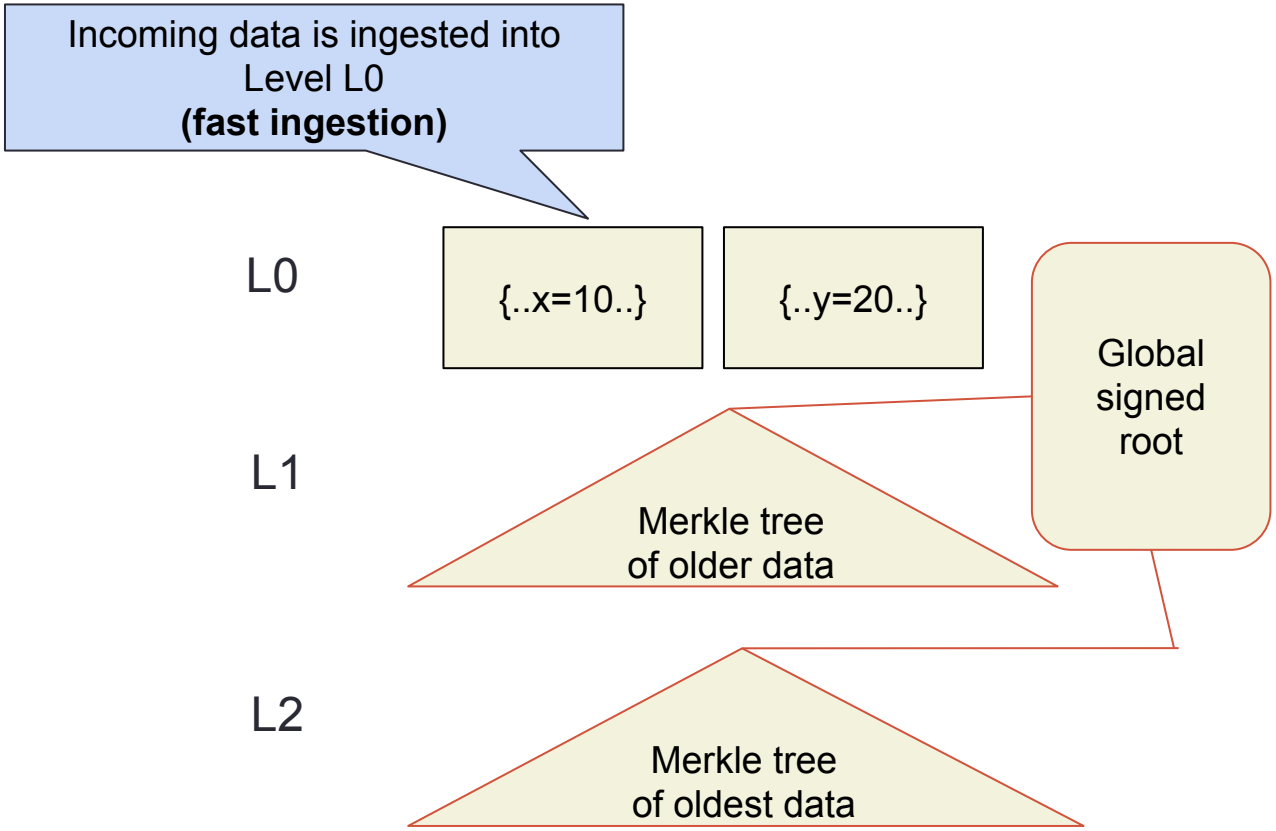
L0 {..x=10..} {..y=20..}

L1 Merkle tree of older data

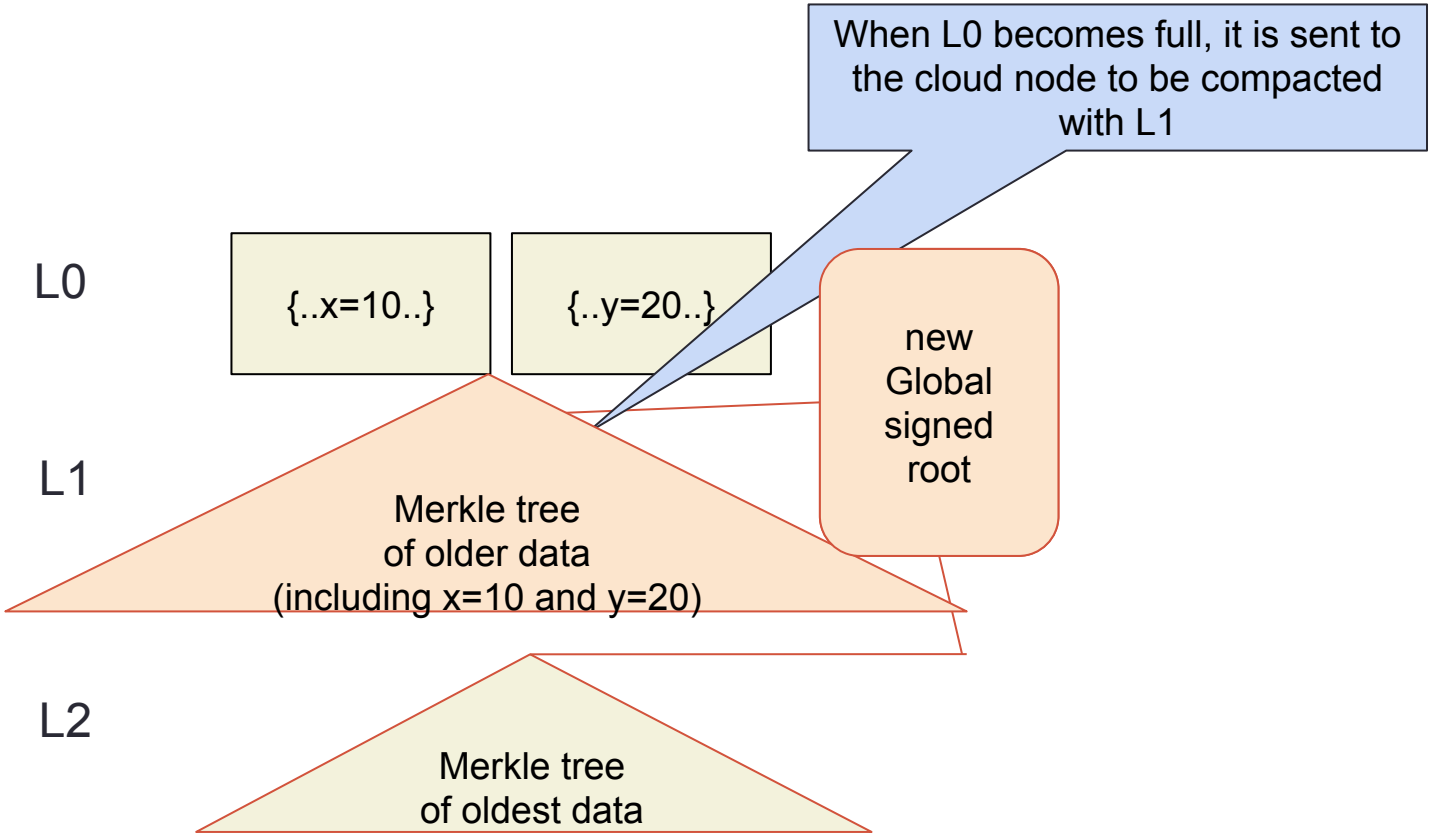
L2 Merkle tree of oldest data

Global signed root

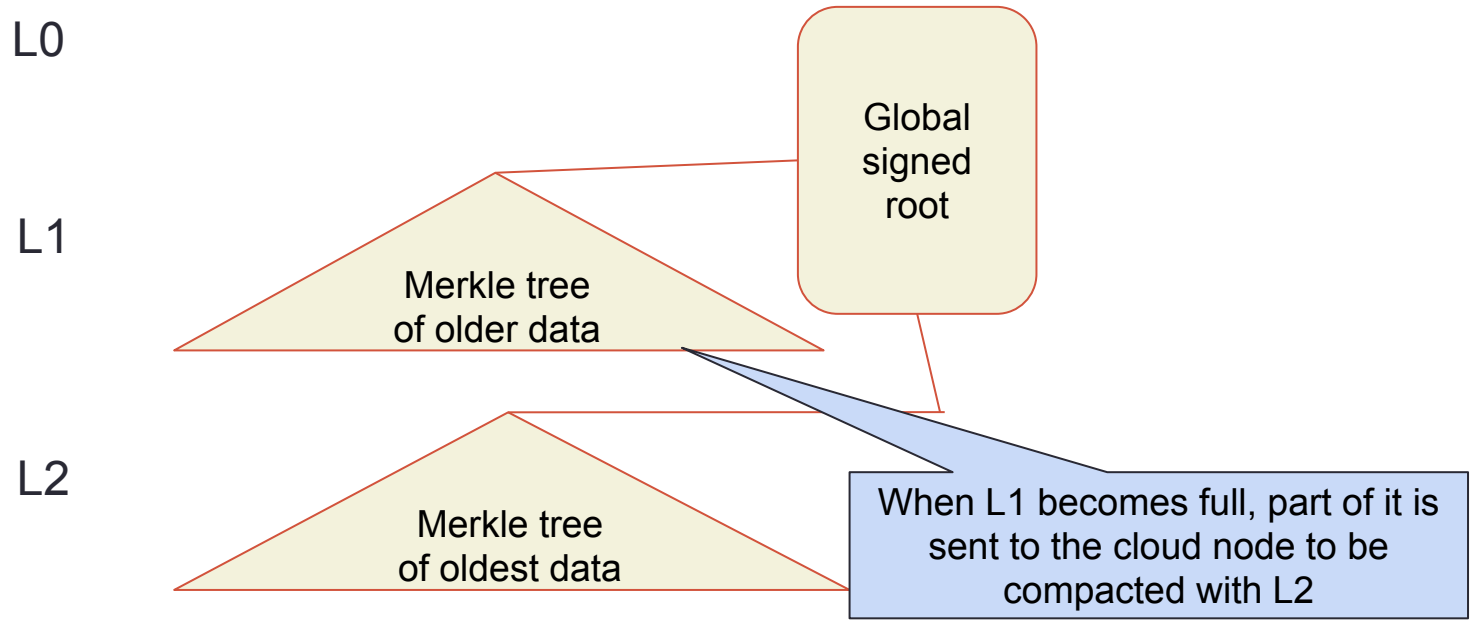
Proposal: LSMerkle
An Index for Lazy Trust



Proposal: LSMerkle
An Index for Lazy Trust



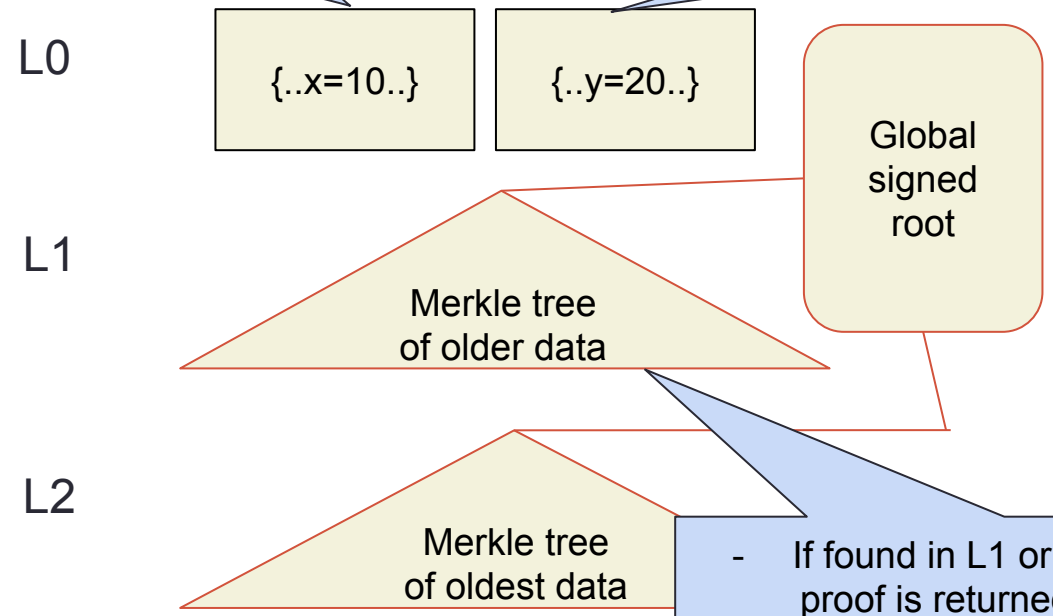
Proposal: LSMerkle
An Index for Lazy Trust



Proposal: LSMerkle reading

- To read a data item x, we start from L0. If found, we return it
- otherwise, we look in L1, and then L2

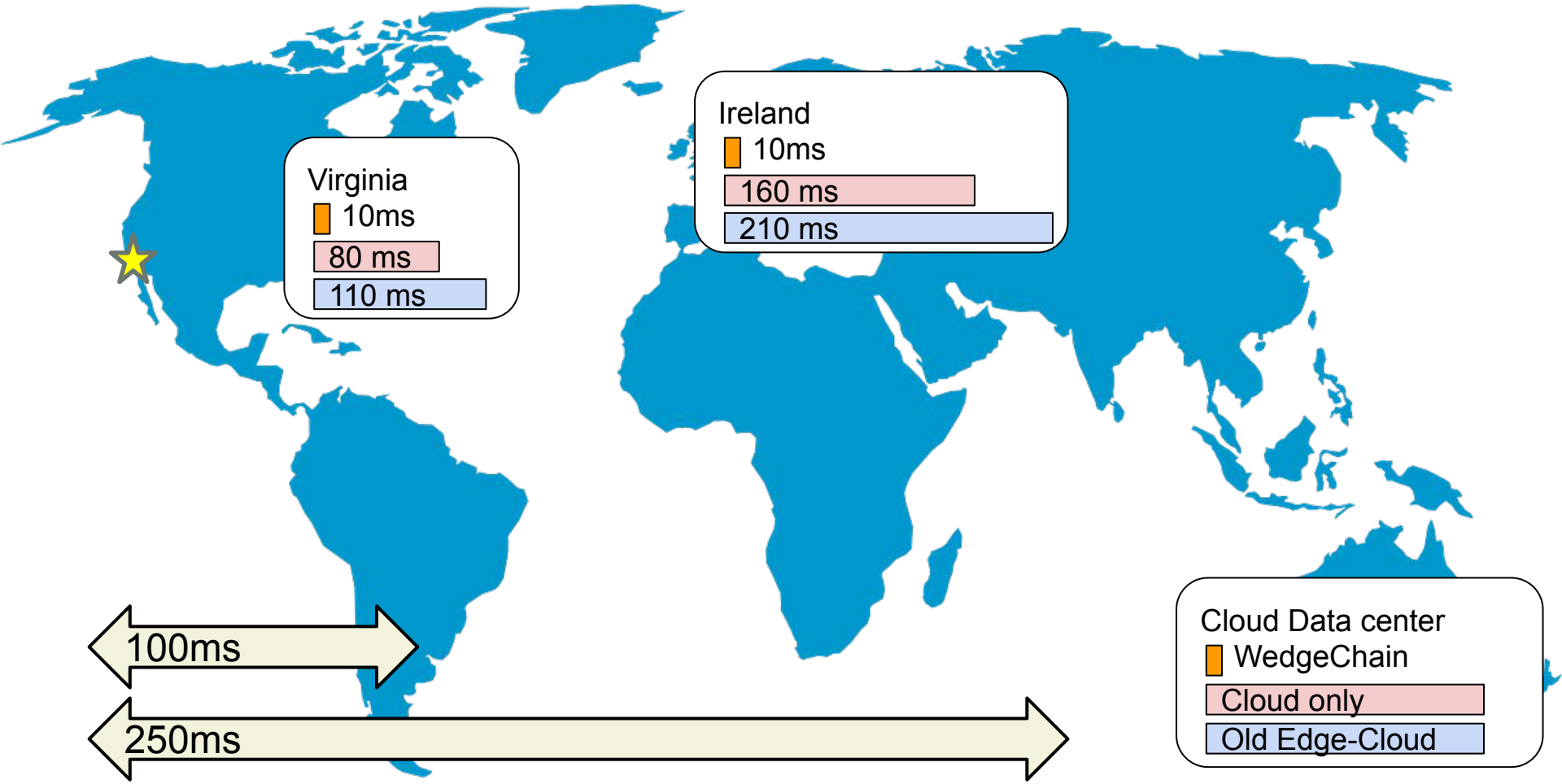
- If found in L0, then the signed record is returned (similar to the original lazy trust example)



- If found in L1 or L2, then the item and a proof is returned to the user. The proof includes:
  - (1) the global signed root
  - (2) the proof of the merkle tree in the corresponding level
  - (3) proof that the key does not exist in higher levels

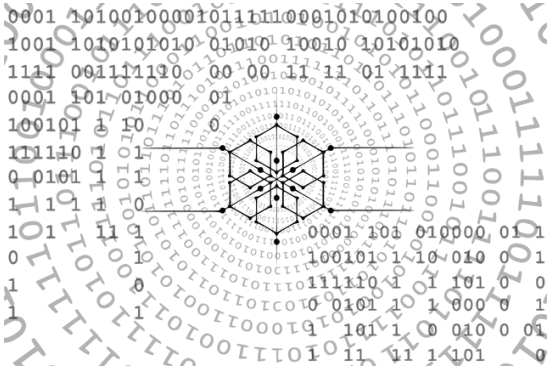
Evaluation (write latency)

- Clients and edge nodes are in California
  - Latency between “edge” nodes in a cluster: emulated 10ms delay



What if the trusted entity is a blockchain? [WedgeBlock EDBT'23]

- Unique opportunities
  - Smart contracts available for anyone as trusted entities
  - Smart contracts to detect and punish malicious nodes
- Unique challenges when using a blockchain smart contract as the trusted entity
  - A smart contract cannot "sign" messages (cannot store private keys)
  - We cannot do a lot of operations on the blockchain (expensive)

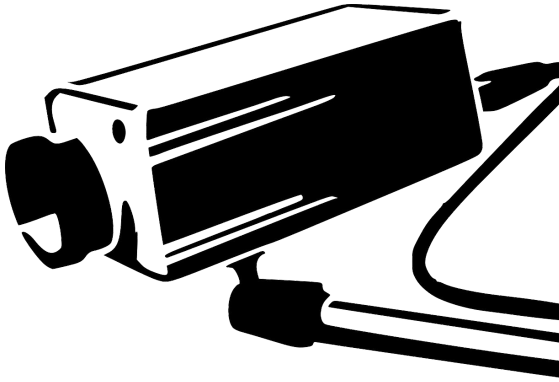


Trusted entity (blockchain smart contract)

update(x=10)

Edge Node

Lazy Trust
WedgeBlock and WedgeChain



1. The trusted node is out of the path of execution.
(Phase I Commit)

2. Malicious activity (lies) are detected, eventually.
(Phase II Commit)

1. Edge Node Efficiency (Energy and Endurance)

2018	2019	2020	2021	2022	2023
			Predict n' Write [ICDE'21] **NVM Lifetime** [VLDB'21Tutorial]		**Hamming Tree** [SIGMOD'23] **E2-NVM** [EDBT'23]

2. Edge-to-Edge Coordination

2018	2019	2020	2021	2022	2023
DPaxos [SIGMOD'18] **Nomadic** [EDBT'18]	**BlockPlane** [ICDE'19] **IoT Blockchain** [IEEE IoT'19]	**AnyLog** [CIDR'20]		**PeloPartition** [Blockchain'22]	**TransEdge** [EDBT'23]

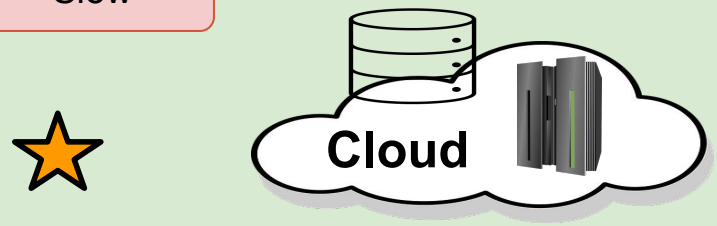
3. Edge-to-Cloud/blockchain Coordination

2018	2019	2020	2021	2022	2023
Edge-Video [VLDB'18Demo]			**CooLSM** [ICDE'21] **WedgeChain** [ICDE'21]	**Croesus** [ICDE'22] **FiME** [Blockchain'22]	**ServerlessBFT** [ICDE'23] **WedgeBlock** [EDBT'23]

CooLSM [ICDE'21]

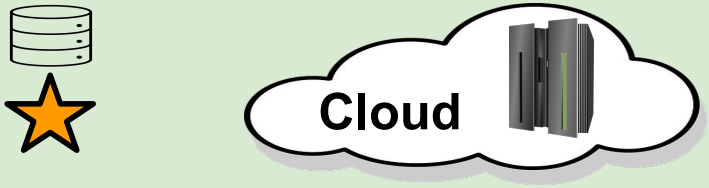
Question 1: how do we build data storage that spans edge and cloud nodes?

Slow



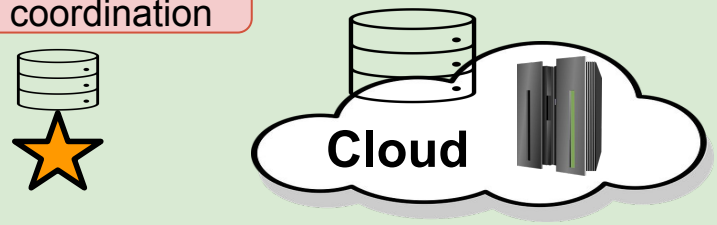
Cloud-only solution

limited capacity



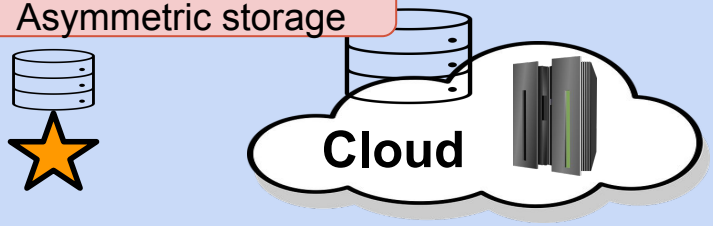
Edge-only solution

expensive coordination



Edge and cloud as part of a distributed/replicated storage system

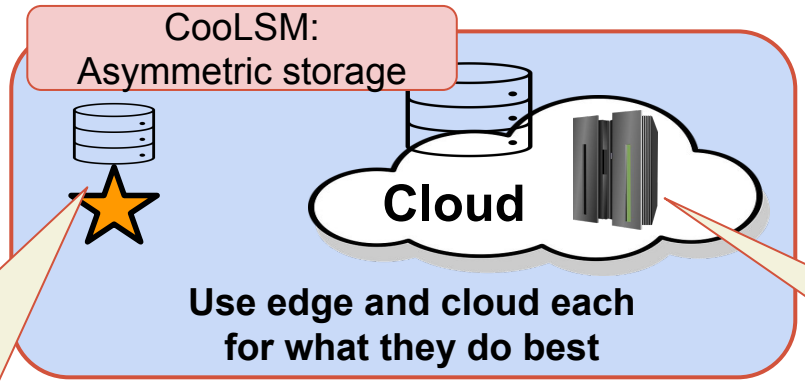
CooLSM: Asymmetric storage



Use edge and cloud each for what they do best

CooLSM [ICDE'21]

Question 1: how do we build data storage that spans edge and cloud nodes?



Edge node

+ close to users
Use for real-time actions

- limited capacity
Do not use for intensive or large jobs

cloud node

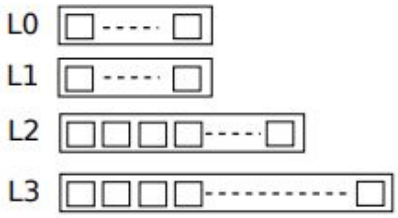
+ large capacity
use for intensive and large tasks

- faraway from users
use only for non-real-time tasks

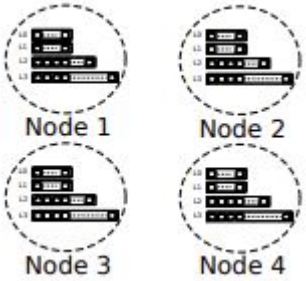
CooLSM [ICDE'21]

Question 2: how do we use these observations for the LSM storage structure?

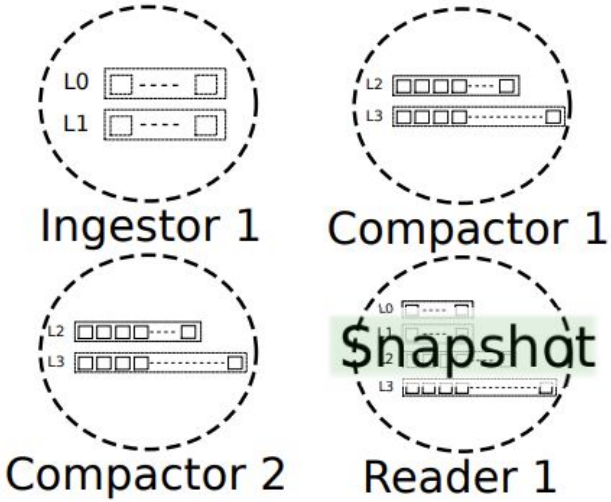
First, let's reflect on the structure of LSM trees



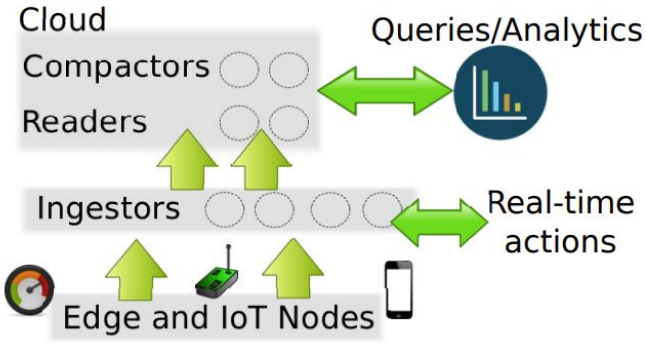
LSM Tree



Distributed LSM Tree



Deconstructed LSM Tree

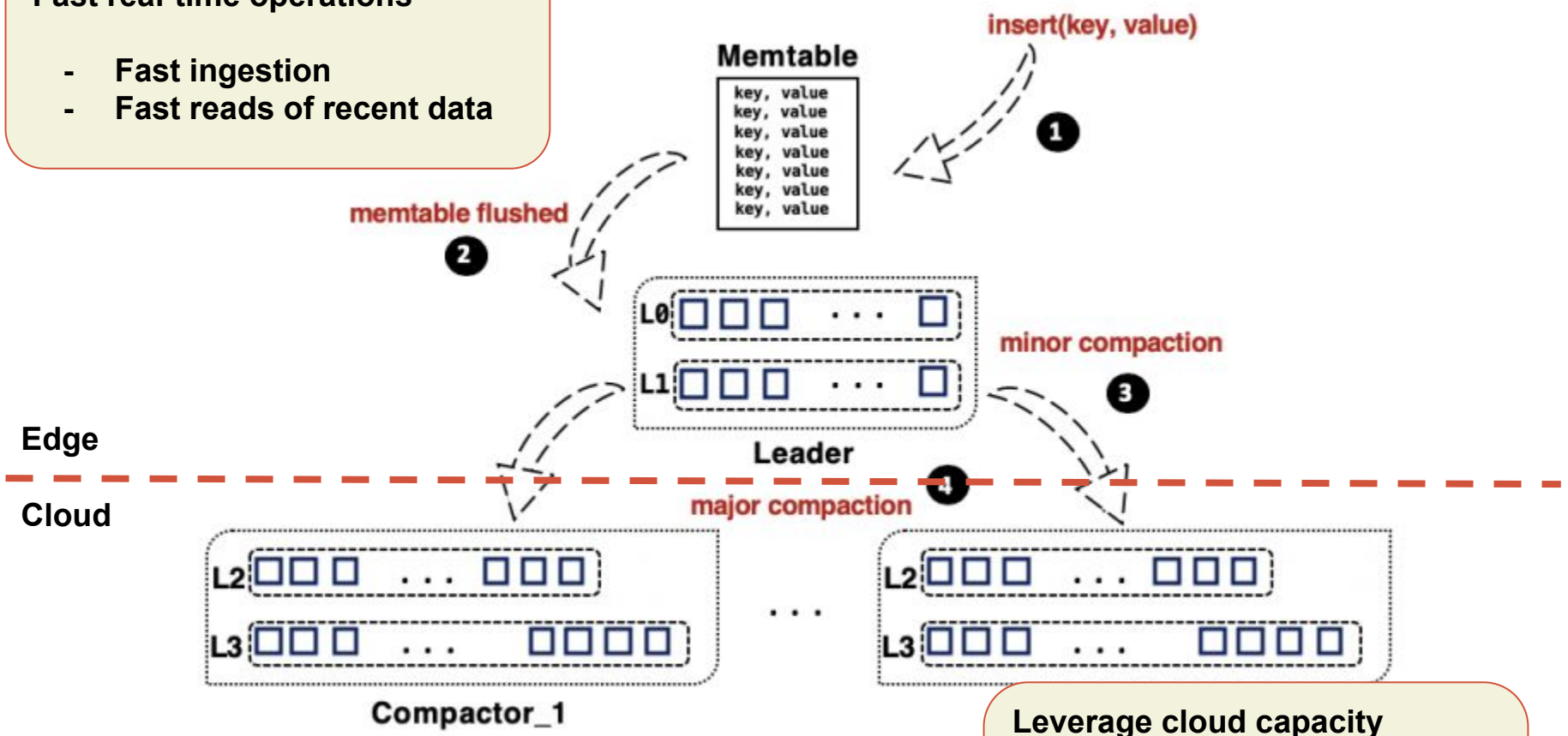


Distribution after deconstruction

CooLSM Data Flow (1 Ingestor/M Compactors)

Fast real-time operations

- Fast ingestion
- Fast reads of recent data



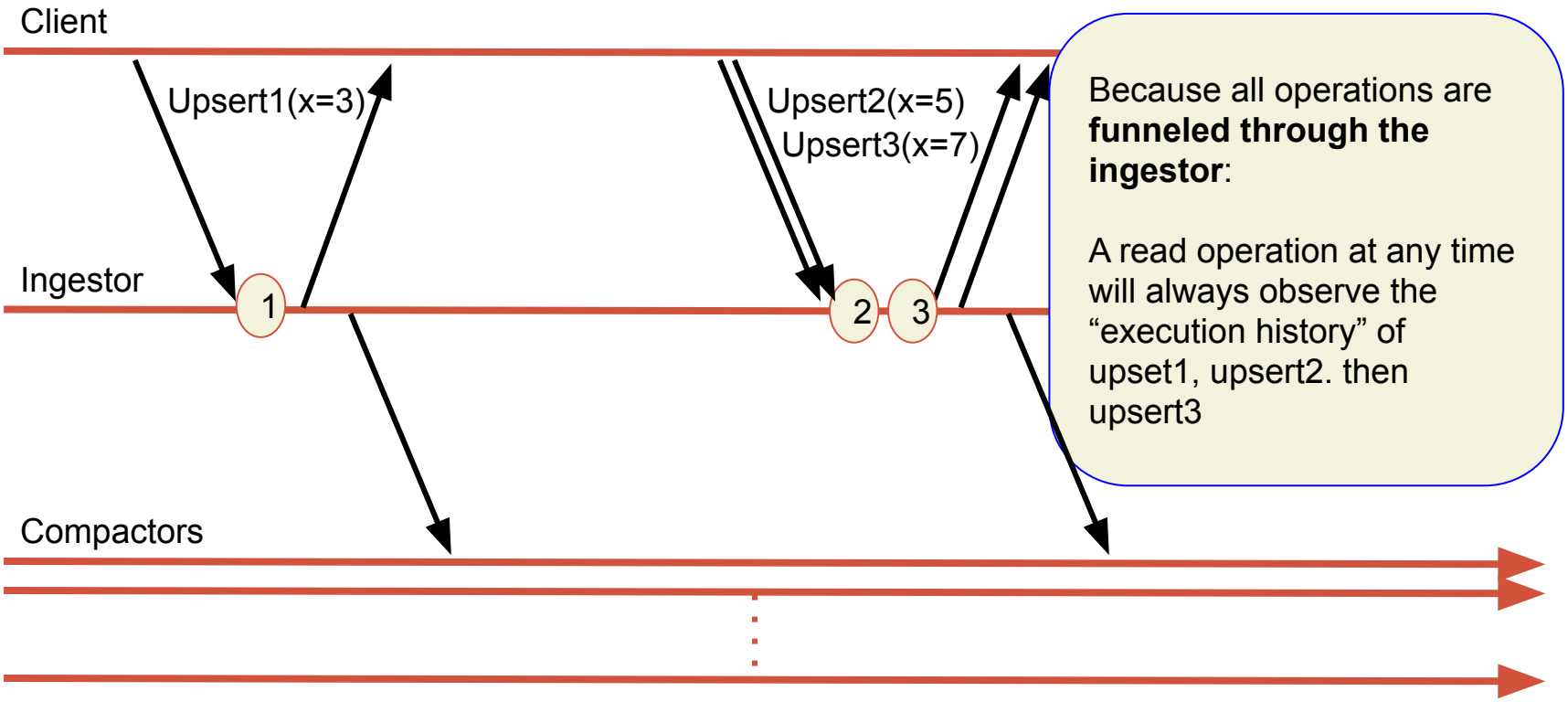
Leverage cloud capacity

- heavy compactions on cloud machines
- Scale compaction with multiple compactors

CooLSM Correctness

CooLSM is linearizable

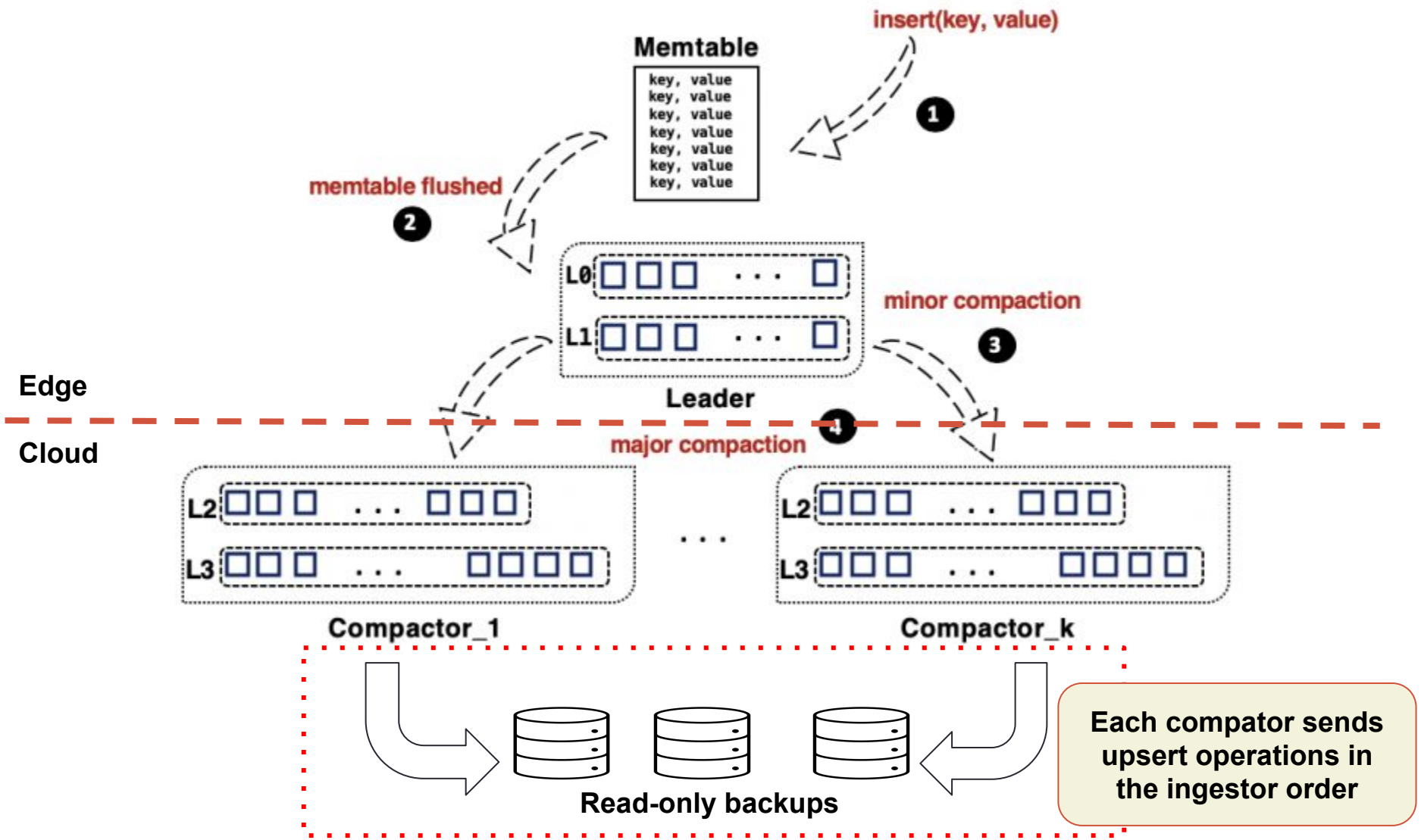
Linearizability is a guarantee that a data operation appears to happen instantaneously at some time between its invocation and return.



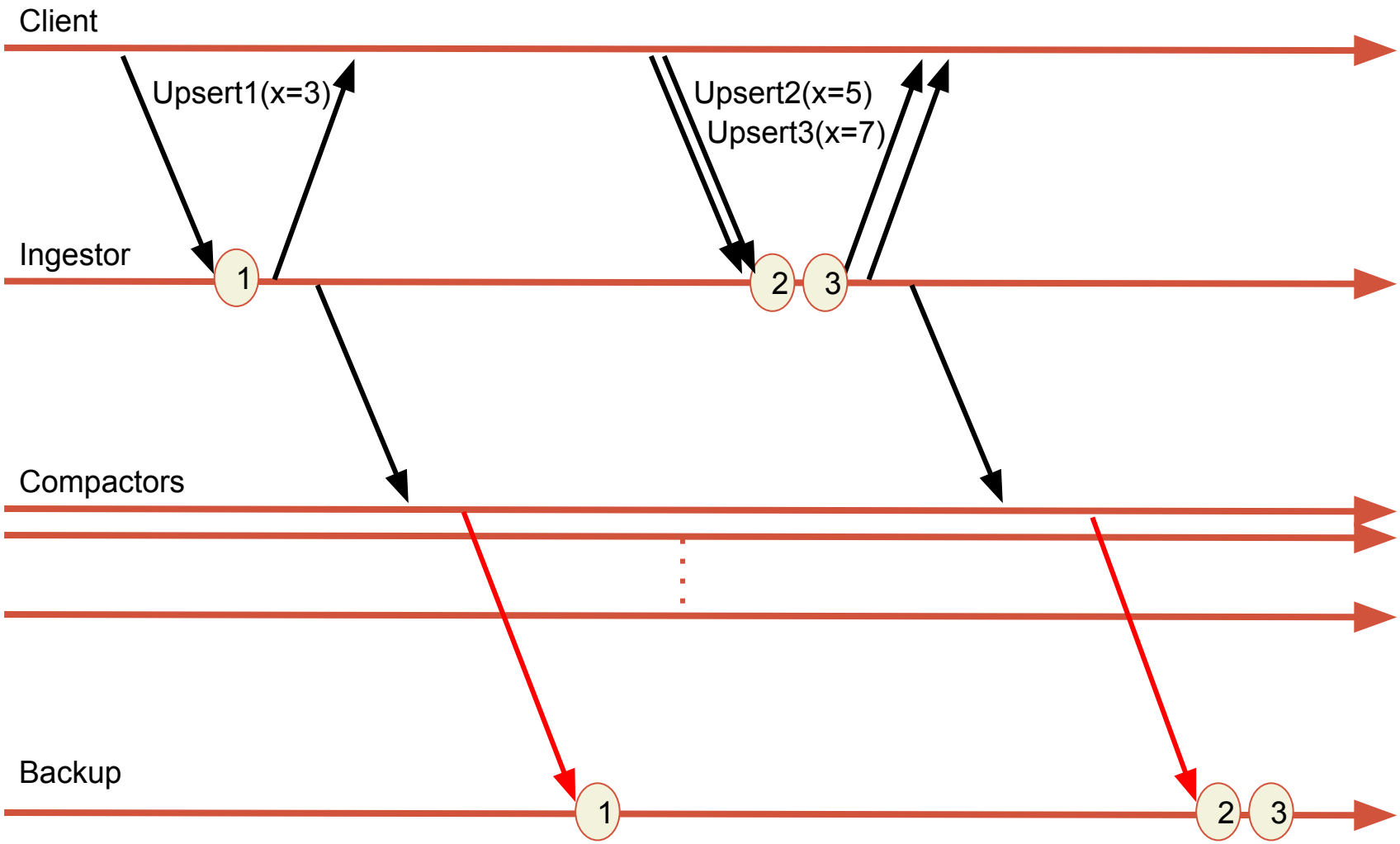
Because all operations are **funneled through the ingestor**:

A read operation at any time will always observe the "execution history" of upset1, upsert2. then upsert3

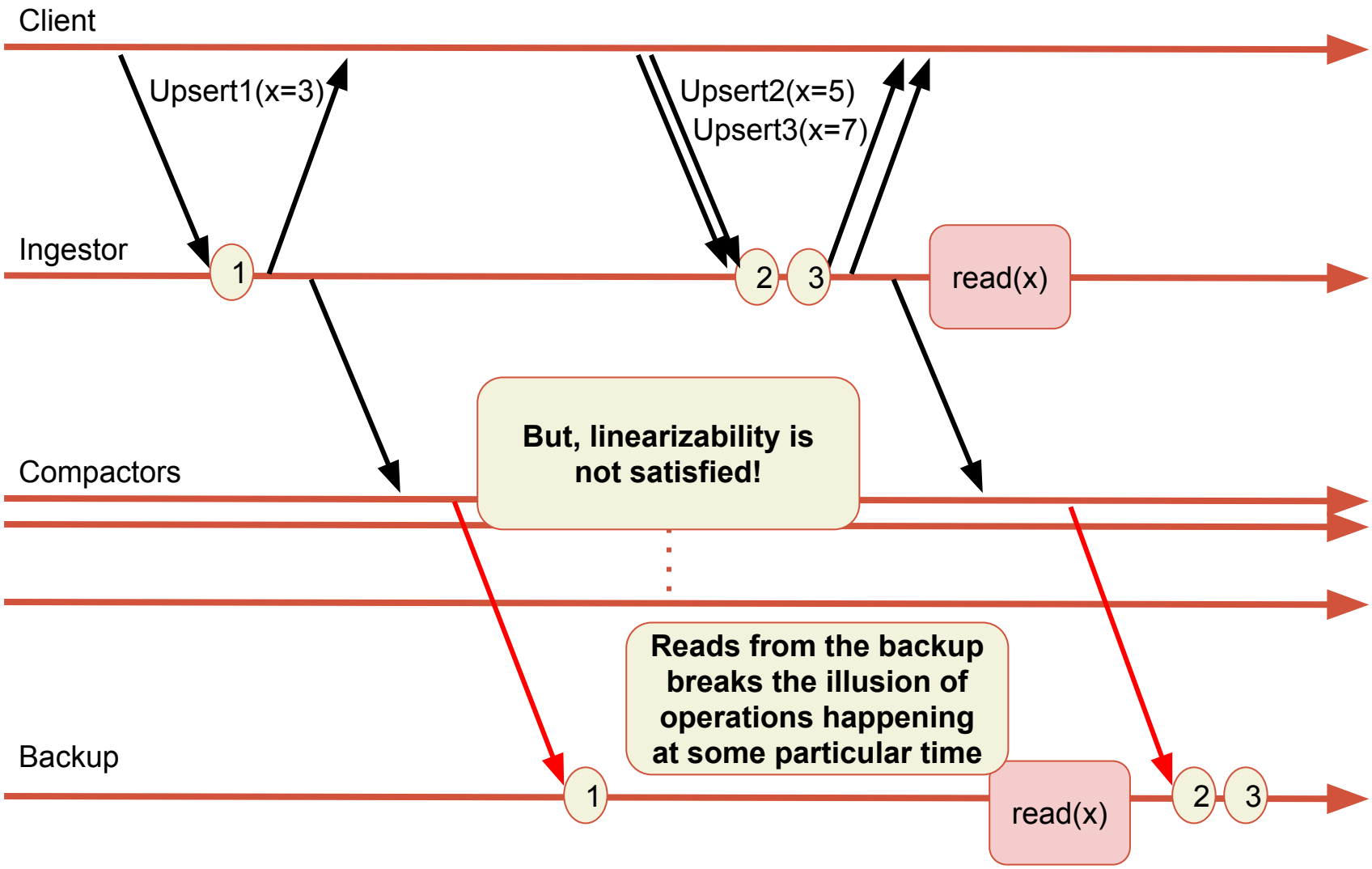
Optimization 1: Add read-only backup nodes



CooLSM Backup data flow



CooLSM Backup correctness



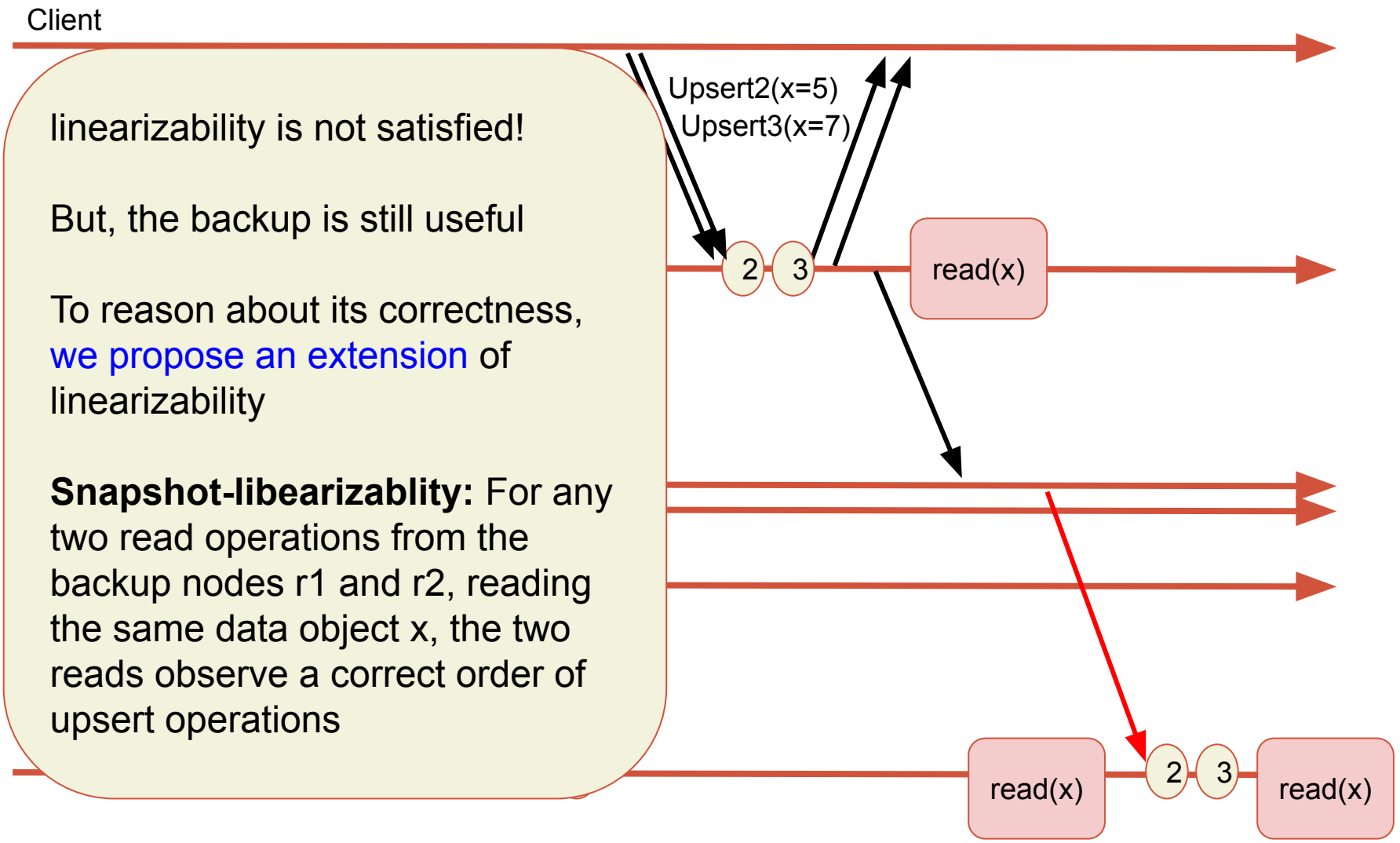
CooLSM Backup correctness

linearizability is not satisfied!

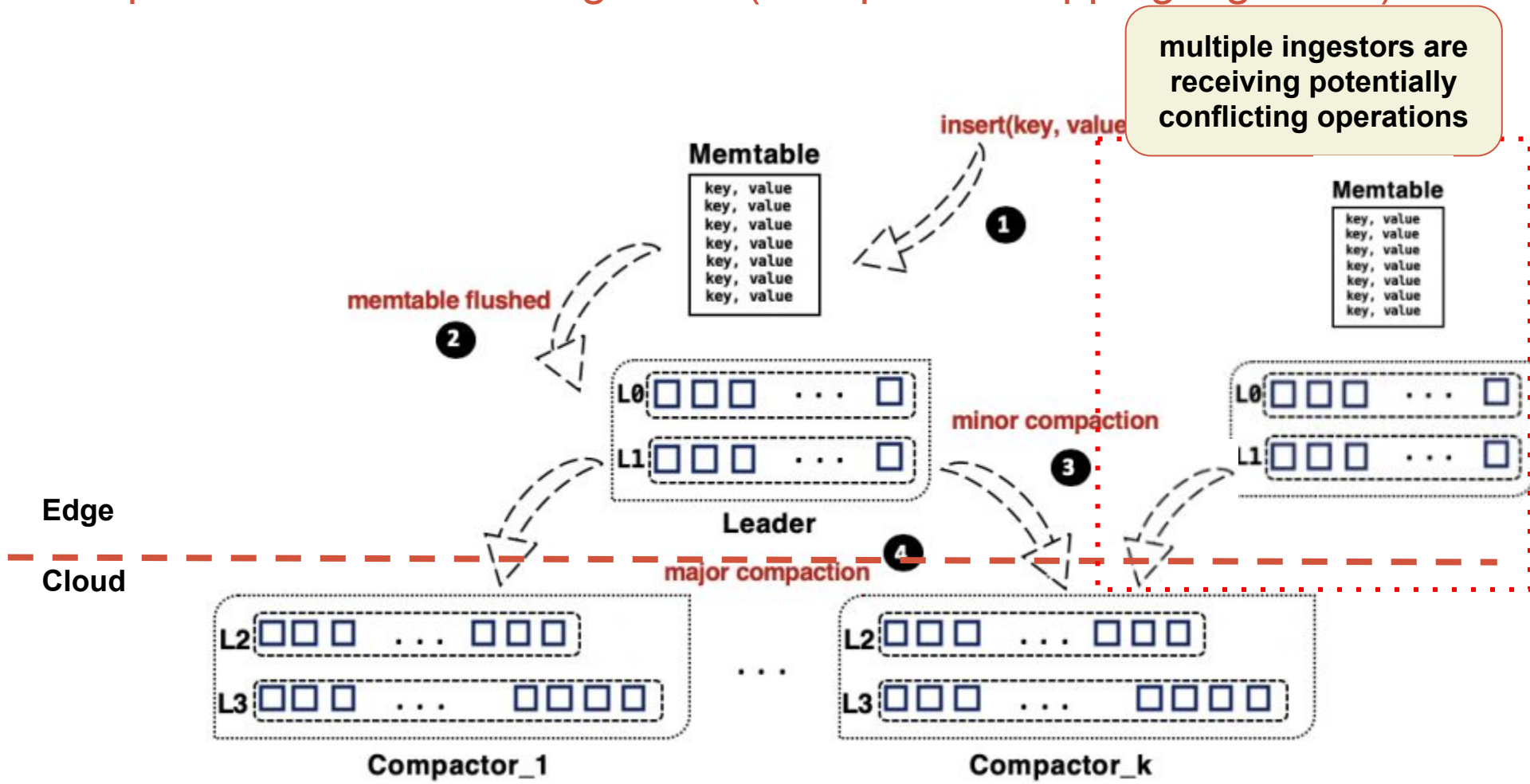
But, the backup is still useful

To reason about its correctness, we propose an extension of linearizability

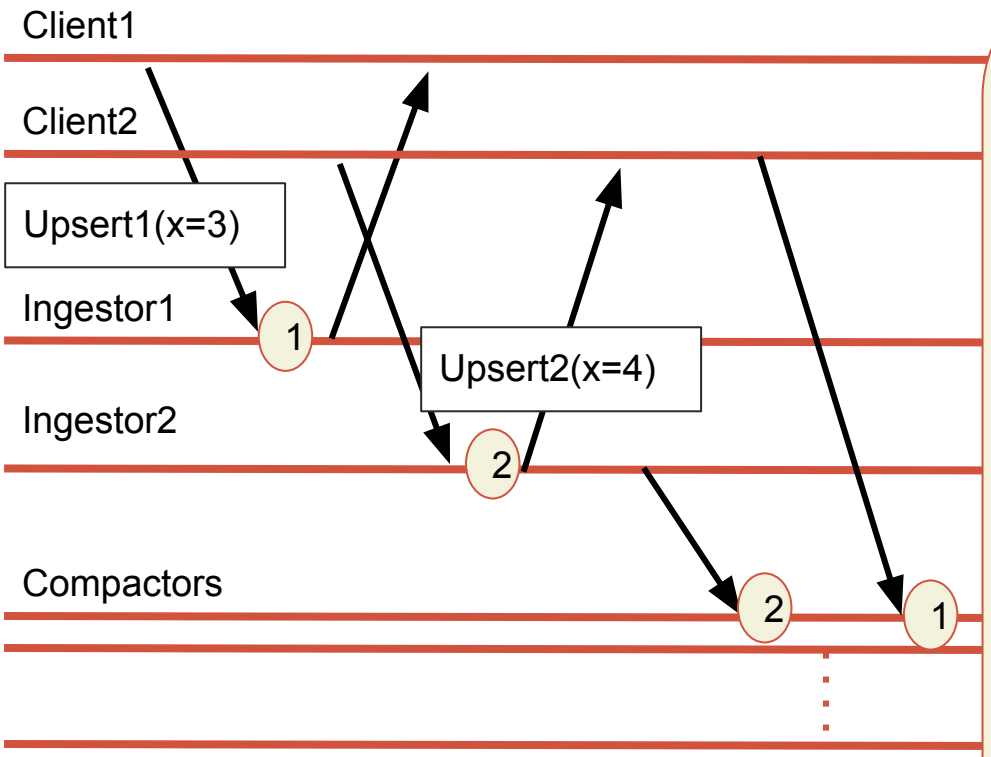
Snapshot-libearizablity: For any two read operations from the backup nodes r1 and r2, reading the same data object x, the two reads observe a correct order of upsert operations



Optimization 2: Scale ingestion (multiple overlapping ingestors)



CooLSM Correctness with multiple ingestors



multiple ingestors lead to anomalies in linearizability!

Either
(1) we make ingestors coordinate – but with high cost, or
(2) we weaken linearizability.

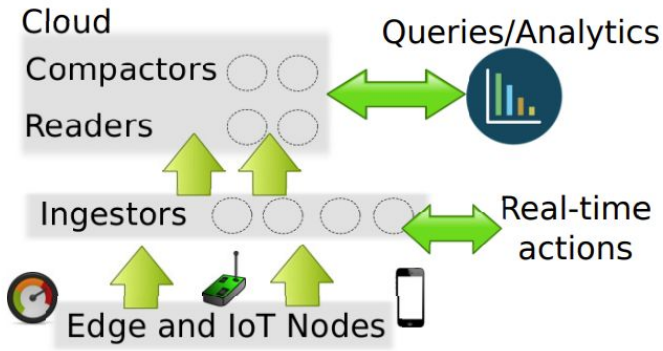
We propose **linearizable+concurrent** isolation

Intuition: make the granularity or ordering based on a window of concurrency.

- If two operations are performed within the same time, then it is fine to be reordered.
- Only ensure the order of operations that do not overlap

CooLSM

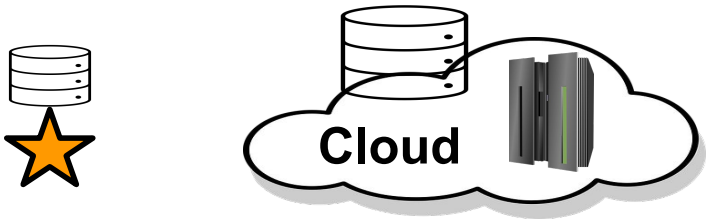
- Deconstructing and distributing LSM storage across edge and cloud nodes
- **Design principle:** consider the asymmetry of resources between edge and cloud nodes
- Useful for **storage disaggregation** in general, e.g., Nova-LSM [SIGMOD'21], dLSM [ICDE'23]
- We apply the **asymmetric edge-cloud principle** to other problems



ICDE 2023

Reliable Transactions in Serverless-Edge Architecture

Suyash Gupta*, Sajjad Rahnama, Erik Linsenmayer, Faisal Nawab†, Mohammad Sadoghi
Exploratory Systems Lab
University of California, Davis
†University of California, Irvine
*University of California, Berkeley

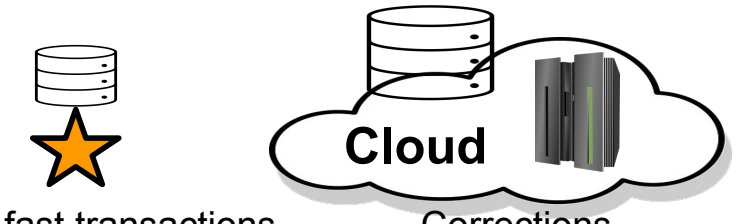


untrusted serverless trusted database

ICDE 2022

Croesus: Multi-Stage Processing and Transactions for Video-Analytics in Edge-Cloud Systems

Samaa Gazzaz, *UC Santa Cruz*, sgazzaz@ucsc.edu
Vishal Chakraborty, *UC Irvine*, vi.c@uci.edu
Faisal Nawab, *UC Irvine*, nawabf@uci.edu



fast transactions Corrections

[SIGMOD'18] [ICDE'19] [CIDR'20] [Blockchain'22] [EDBT'23]

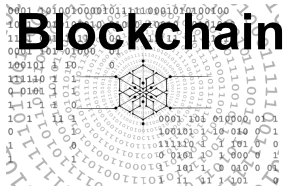
Nomadic **IoT Blockchain**

Blockchain 2022

You've got a Friend in ME (Mobile Edge): Blockchain Pr
with Cloud Node Backup

Zane Karl[1]*, Hayden Freedman[1]*, Ahmad Showail[13], Abhishek Singh[1], Samaa Gazzaz[2], Faisal Nawab[1]
zkarl@uci.edu, hfreedma@uci.edu, ahmad.showail@kaust.edu.sa, abhishas@uci.edu, sgazzaz@ucsc.edu, nawabf@uci.edu
[1]Donald Bren School of Information and Computer Science, University of California, Irvine, Irvine, CA, 92697, USA
[2]Baskin School of Engineering, University of California, Santa Cruz, Santa Cruz, CA, 95064, USA
[3]Department of Computer Engineering, Taibah University, Madinah, Saudi Arabia

Stale knowledge



Ground Truth

In edge-to-cloud and edge-to-blockchain, design for the asymmetry between resources

2021	2022	2023
CooLSM [ICDE'21]	**Croesus** [ICDE'22]	**ServerlessBFT** [ICDE'23]
WedgeChain [ICDE'21]	**FiME** [Blockchain'22]	**WedgeBlock** [EDBT'23]

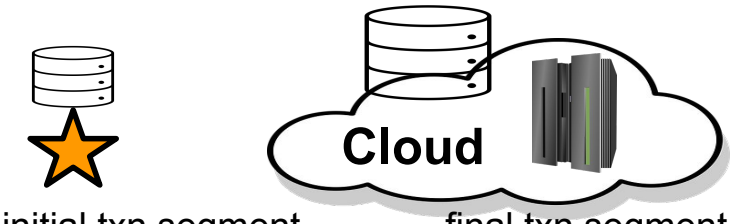
Ongoing/future work

- **Edge-to-cloud Transaction Processing**
- **Croesus [ICDE'22]:** Fast transactions on edge and corrections in the cloud
  - Inspired by Invariant Confluence [P. Bailis, et. al. VLDB'14] and Guesses and Apologies [P. Helland & Campbell CIDR'09]

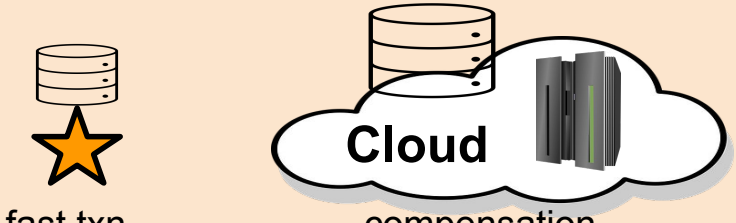
ICDE 2022

Croesus: Multi-Stage Processing and Transactions for Video-Analytics in Edge-Cloud Systems

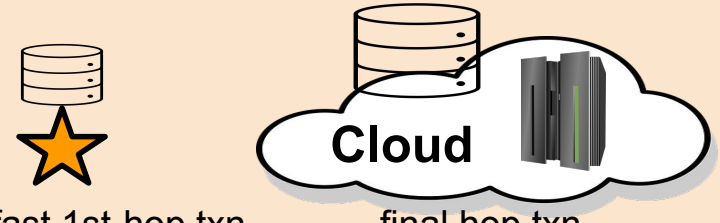
Samaa Gazzaz, *UC Santa Cruz*, sgazzaz@ucsc.edu
Vishal Chakraborty, *UC Irvine*, vi.c@uci.edu
Faisal Nawab, *UC Irvine*, nawabf@uci.edu



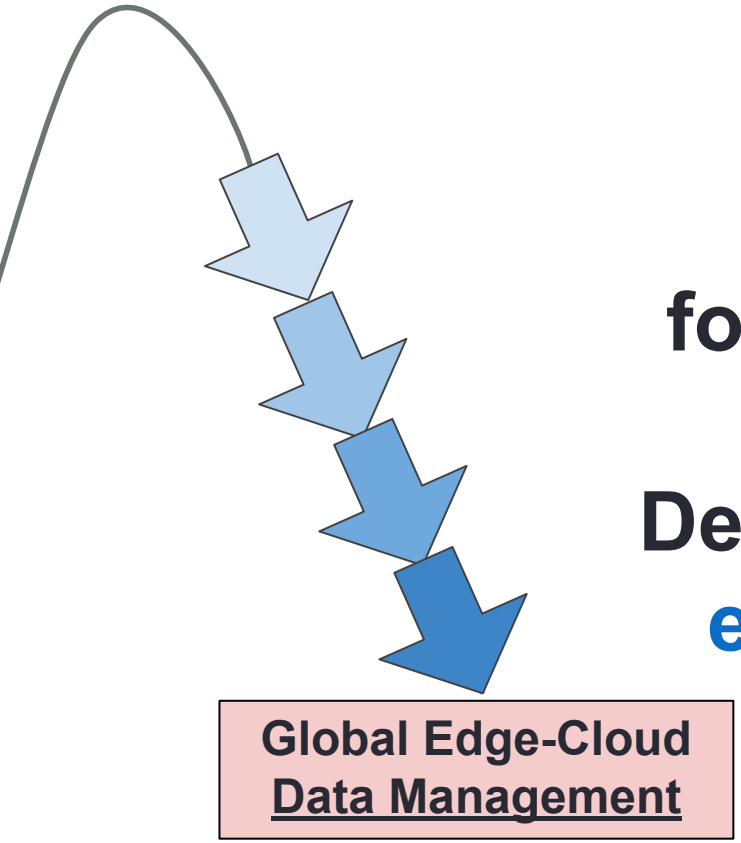
- **Fast transactions in edge and compensations in the cloud**
  - Inspired by Sagas [H. Garcia-Molina and K. Salem SIGMOD'87]



- **Chopping txns to edge hop and cloud hop**
  - Inspired by Transaction Chopping [D. Shasha et. al TODS'95] and Transaction Chains [Y. Zhang et. al SOSP'13]



Edge-Cloud Data Management



Managing and unifying data management for the future of computing

Design for the asymmetry of edge and cloud resource

UCI Edge Lab

https://nawab.me/
nawabf@uci.edu

EdgeLab students

- **Research funding**

  Data management on the **Edge**

  Compliance to **data protection regulations** in smart spaces

  CyberTraining: **Data Science for Engineering**

- **Industry funding**

  Next Generation Data Infrastructure Award

  Resilience of Large-scale Systems

AnyLog