University of Minnesota

Scaling Up The Performance of Distributed Key-Value Stores Using Emerging Technologies for Big Data Applications

Hebatalla Eldakiky
Advisor: Prof. David H. C. Du
Department of Computer Science and Engineering
University of Minnesota, USA

January 22nd, 2020

Talk Outline

- Introduction
- Background & Motivation
- Completed Work
  - TurboKV: Scaling Up the Performance of Distributed Key-value Stores with In-Switch Coordination
  - Key-value Pairs Allocation Strategy for Kinetic Drives
- Proposed Work
  - TransKV: A Networking Support for Transaction Processing in Distributed Key-value Stores (Proposed Project)
- Conclusion
- Future Plan

The Big Data Era (1/2)

We live in the digital era, where data is generated from everywhere

Bridge Monitoring.

Environment Controls.

Elder Care Monitoring.

Forest Management.

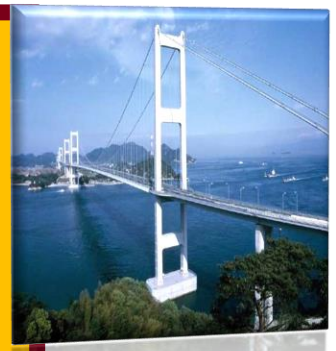
Soil Monitoring.

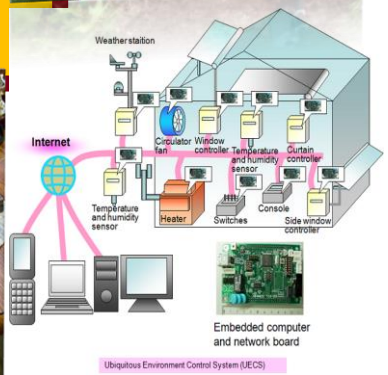
Internet of Things.

Social Media.

Smart Phones.

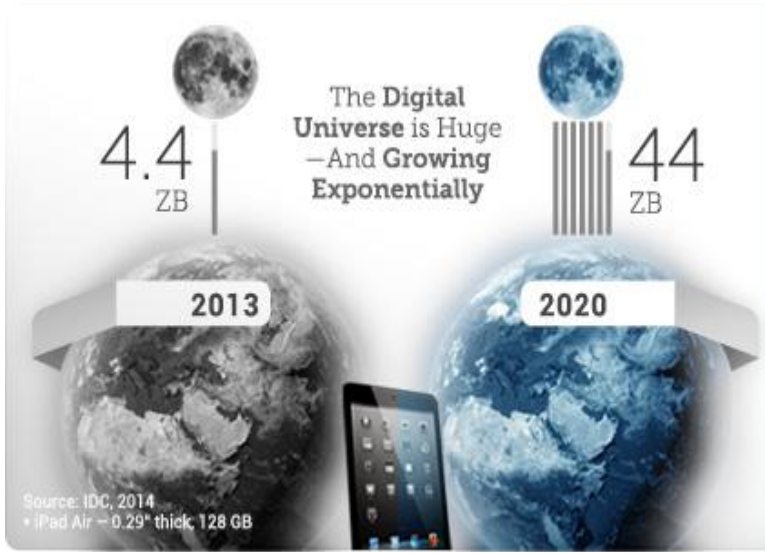
and more…..





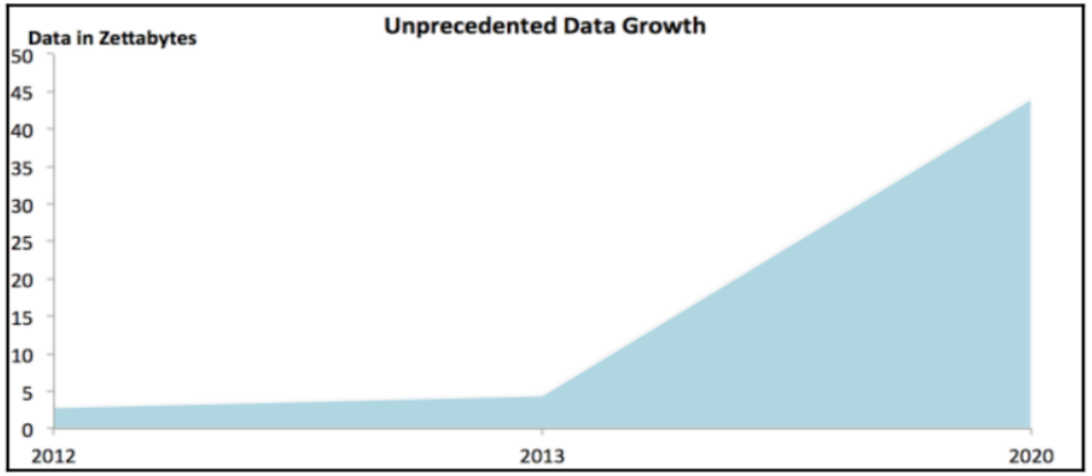
New 4 PB/day

6000 tweets/sec



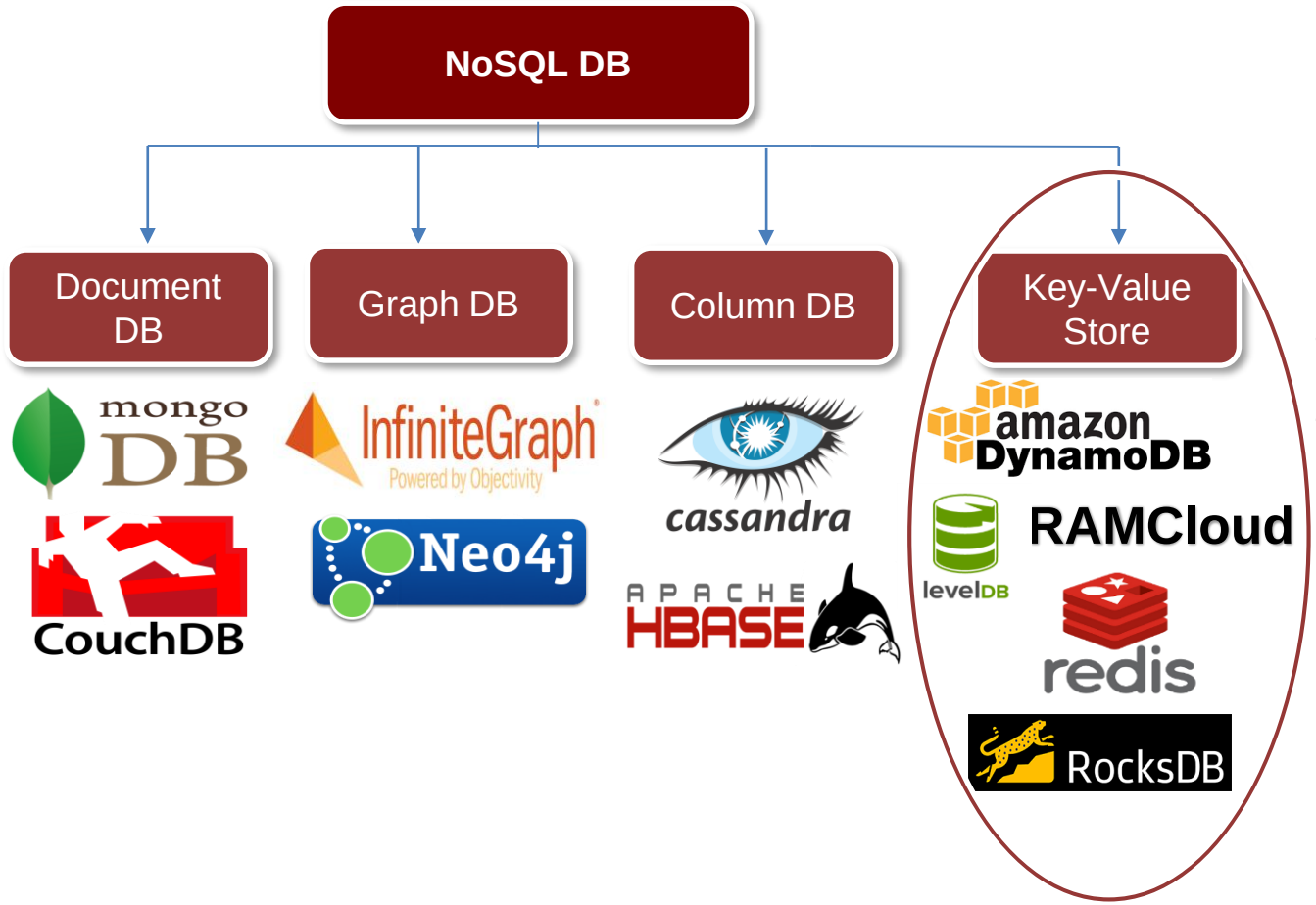
If the Digital Universe were represented by the memory in a stack of tablets, in **2013** it would have stretched two-thirds the way to the Moon*

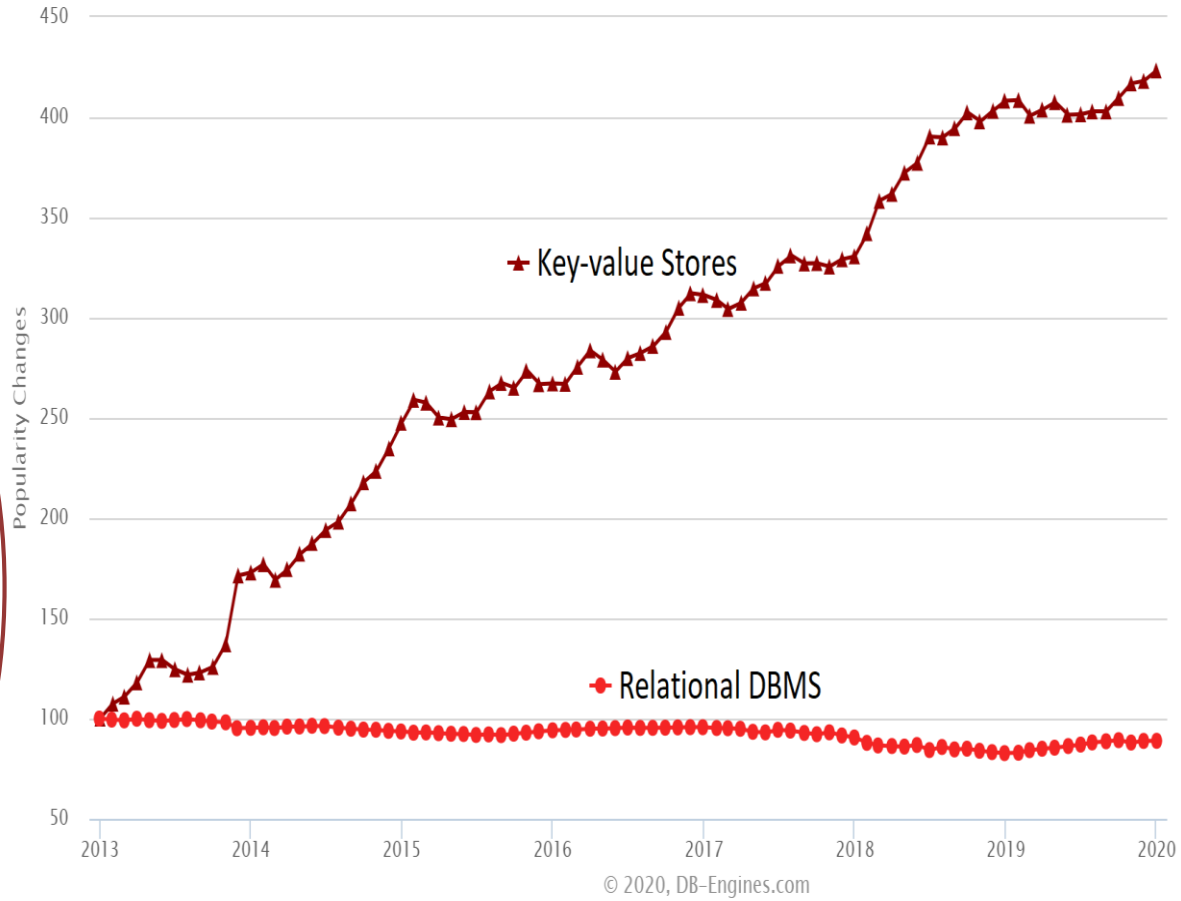
By **2020**, there would be 6.6 stacks from the Earth to the Moon*



The Big Data Era (2/2)

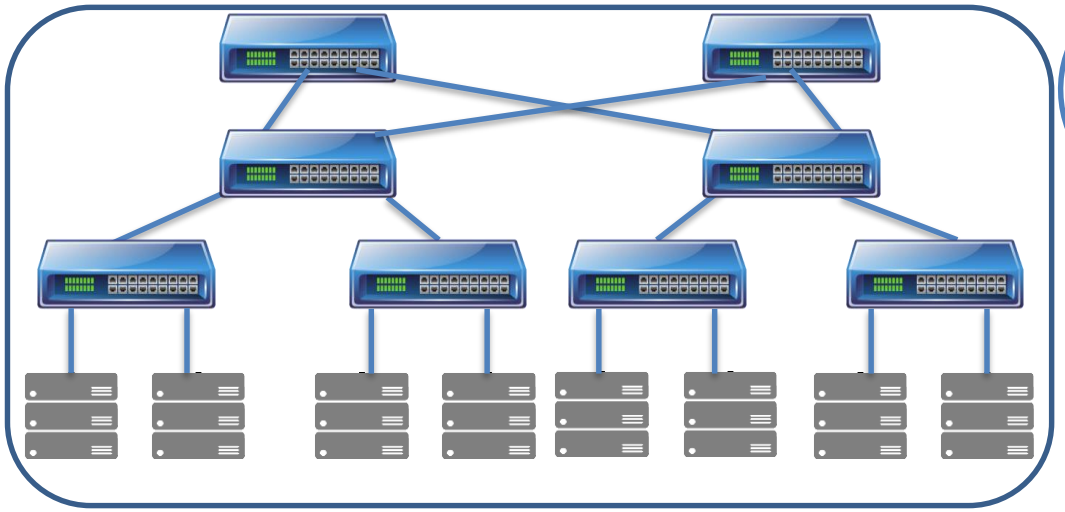
NoSQL Databases become a competitive alternate to the relational DB to store and process the data.





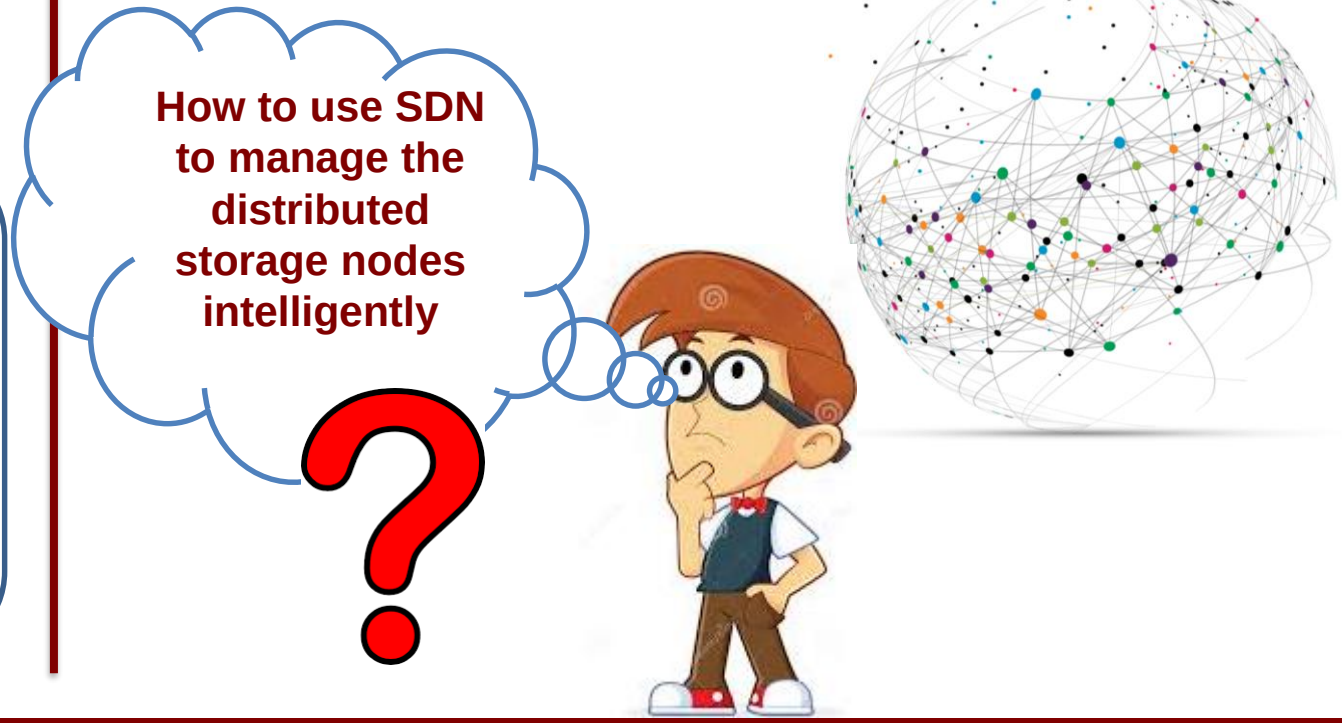
Big Data & Storage Challenges (1/2)

- Storage infrastructure is vital for solving big data problems.
- Enormous amount of data is distributed between several storage nodes which are connected with network switches.
- Network latency plays a critical role in the efficient access of data in this distributed environment.



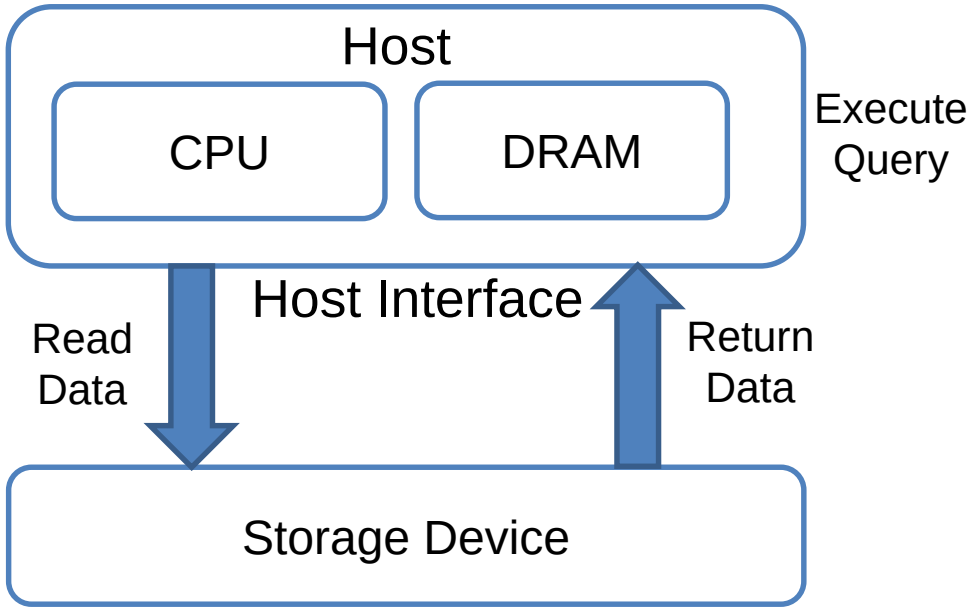
Storage Infrastructure

- Software-defined Networks (SDN) provide efficient resource allocation and flexibility for maximum network performance.
- Network switches also become more intelligent to perform some computational tasks in-network.



Big Data & Storage Challenges (2/2)

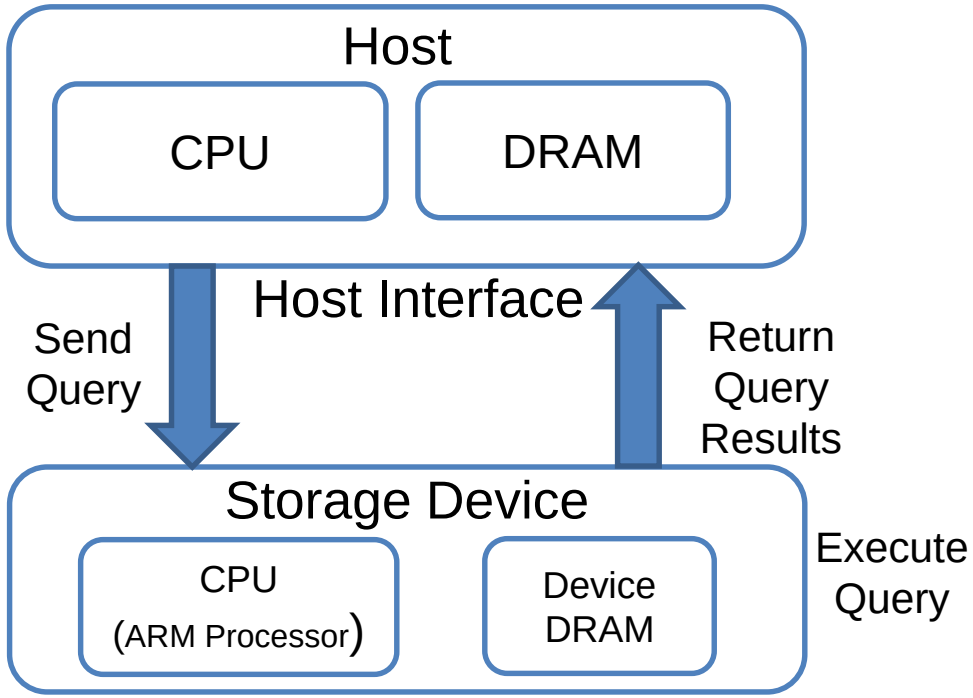
Conventional Architecture



Data movement problem

With data intensive application, amount of data shipped from storage drives to be processed by the host is very large.

In-Storage Computing Architecture

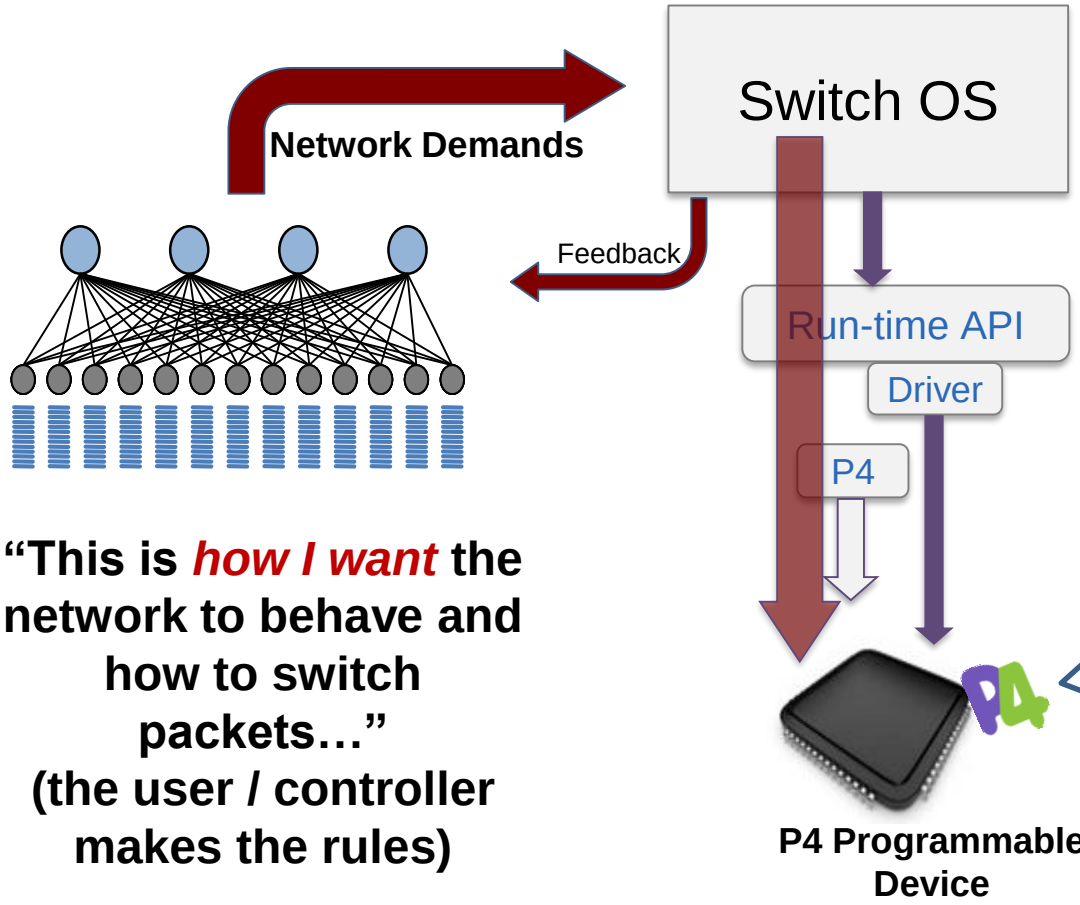


Reduce the amount of data shipped between storage and compute

- ✓ Lower Latency
- ✓ Less energy for data transfer

Programmable Networks → In-Network Computing

Programmable Networks



"This is *how I want* the network to behave and how to switch packets…"
(the user / controller makes the rules)

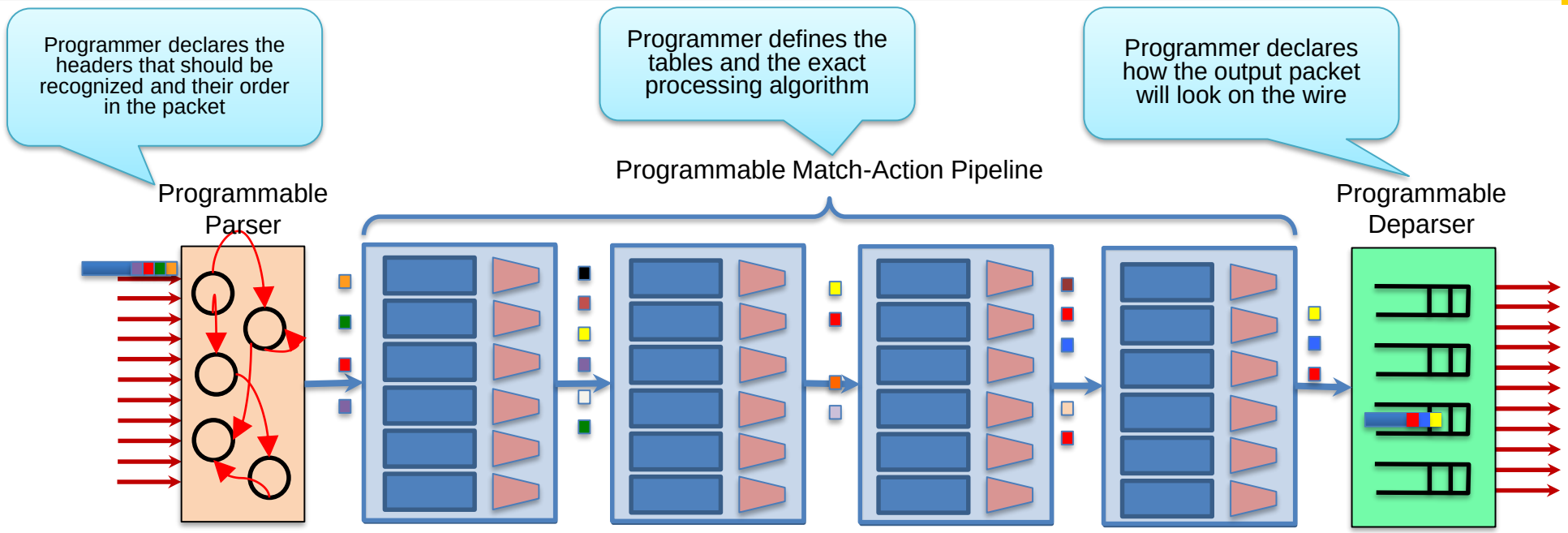
P4 Programmable Device

P4 is a high-level language for programming protocol independent packet processors designed to achieve 3 goals.

- **Protocol independence.**
- **Target independence.**
- **Re-configurability in the field.**

Think programming rather than protocols…

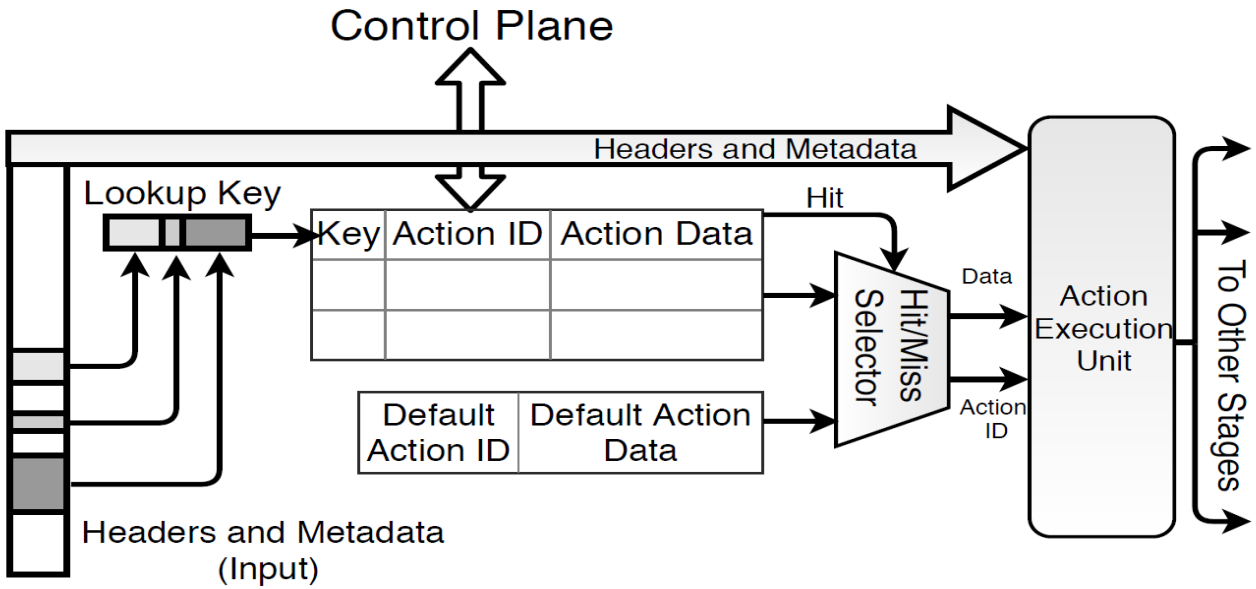
What is PISA ?



- Packet is parsed into individual headers.
- Headers and intermediate results are used for matching and actions.
- Headers can be modified, added or removed in match-action processing.
- Packet is deparsed.

Match-Action Processing

- Tables are the fundamental unit in the match-action pipeline
- Each table contains one or more entries
  - An entry contains: specific key to match on, **single** action, Action data.



BAREFOOT NETWORKS

Tofino

Bandwidth	6.5 Tbps
Processing delay	< 1 µs

NetFPGA SUME

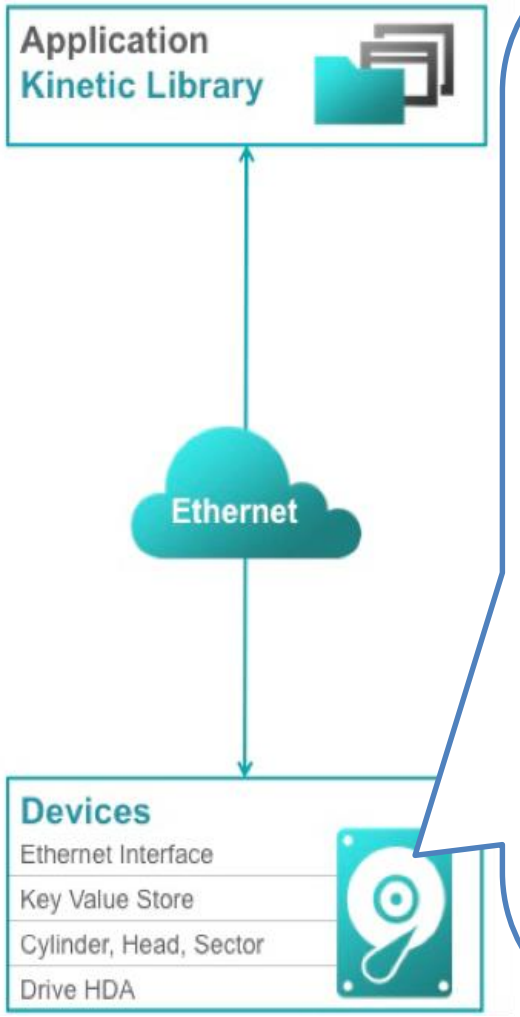
Bandwidth	4x10 Gbps

Systems use programmable switches

- NetCache [SOSP' 17]
  - On-switch cache for Load Balancing (LB).
- NetChain [NSDI' 18]
  - on-switch KV store for small data.
- DistCache [FAST' 19]
  - multiple racks on-switch cache for LB
- iSwitch [ISCA '19]
  - on-switch aggregation for distributed RL

Kinetic Drive → In-Storage Computing

Kinetic Stack



- Active KV storage device developed by Seagate.
- Accessible by an Ethernet connection.
- Has CPU and RAM with built-in LevelDB.
- Handle device to device data migration through P2P copy commands.
- Applications communicate with the drive using the Kinetic Protocol over the TCP network.
- Simple API (get, put, delete).

SEAGATE

Model No.	ST4000NK0001
Transfer rate	60 Mbps
Capacity	4 TB
Key size	Up to 4 KB
Value size	Up to 1 MB

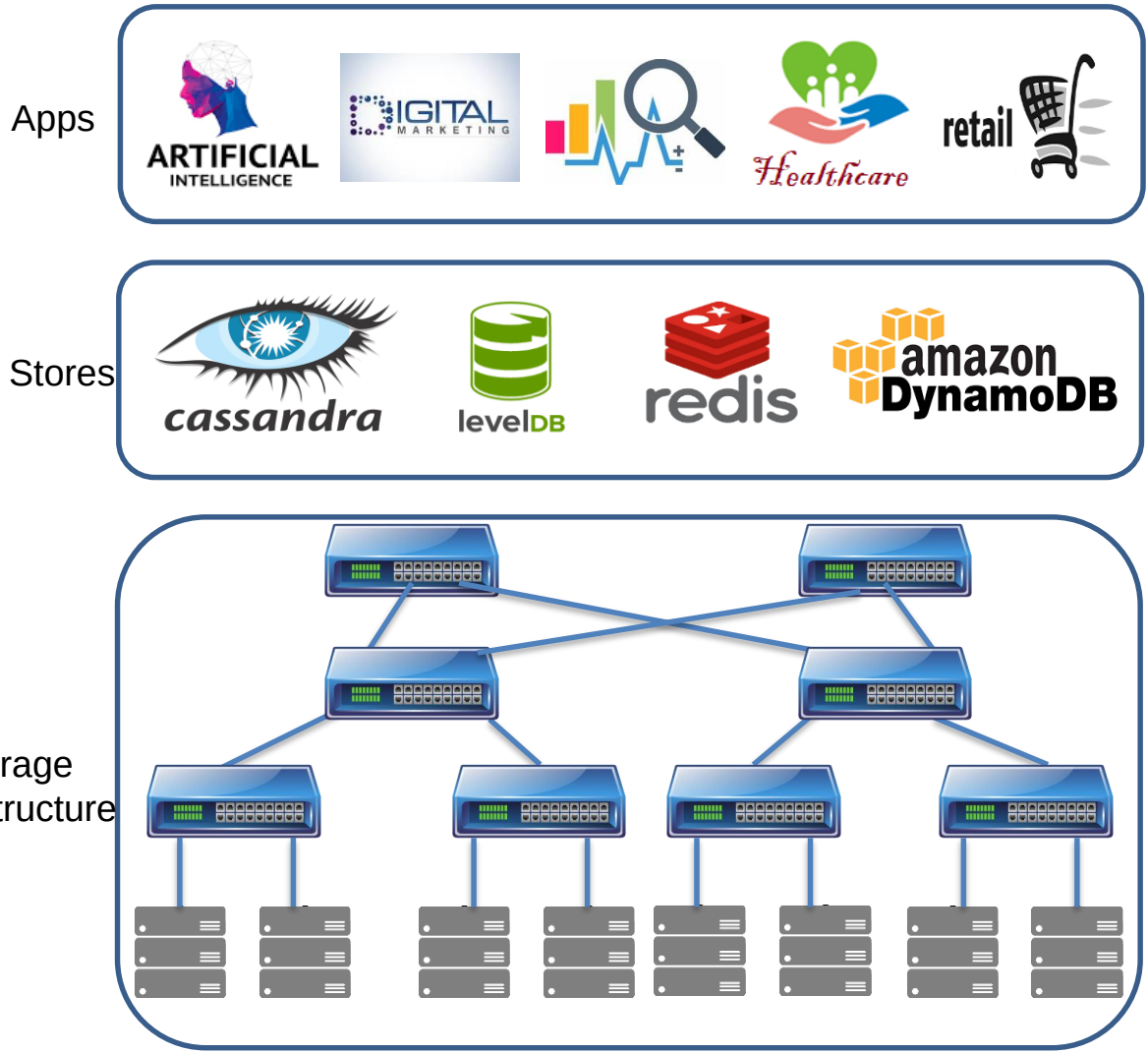
Kinetic Drives Research

- Kinetic Action [ICPADS' 17]
  - Performance evaluation of KD characteristics.
- Data Allocation [BigDataService' 17]
  - 4 data allocation approaches for KD.

Our Mission

- Improve data access performance for distributed KV Stores when applications access storage through network.
- Reduce the amount of data shipped from storage devices to be processed by the host in data intensive applications.

- Completed Work
  - TurboKV: Scaling Up The performance of Distributed Key-value stores with In-Switch Coordination
  - Key-value pair allocation strategy for Kinetic drives.
- Proposed Work
  - TransKV: Networking Support for Transaction Processing in Distributed Key-value Stores.



Completed Work (1/2)

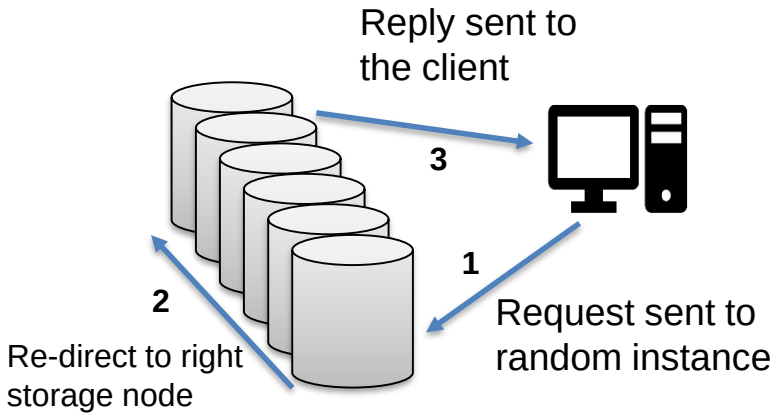
TurboKV: Scaling Up the Performance of Distributed Key-value Stores with In-Switch Coordination[1]

[1] Hebatalla Eldakiky, David H.C. Du, and Eman Ramadan, “TurboKV: Scaling Up the performance of Distributed Key-value Stores with In-Switch Coordination”, under submission to ACM Transaction on Storage (ToS)

Problem Definition

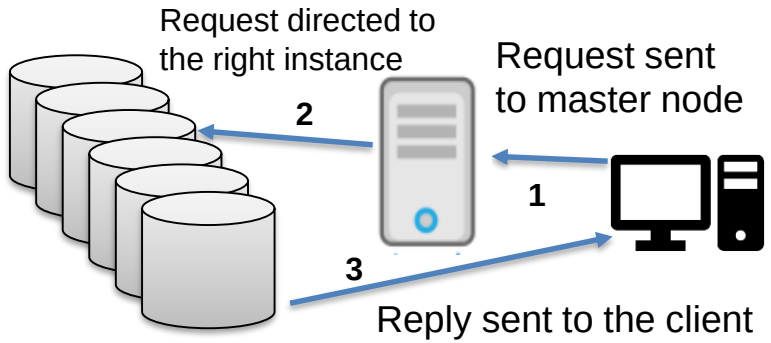
- In distributed Key-value store, data is partitioned between several nodes.
- Partitions management and query routing are managed in three different ways: Server-driven coordination, Client-driven coordination, and Master-node coordination

Server-driven Coordination



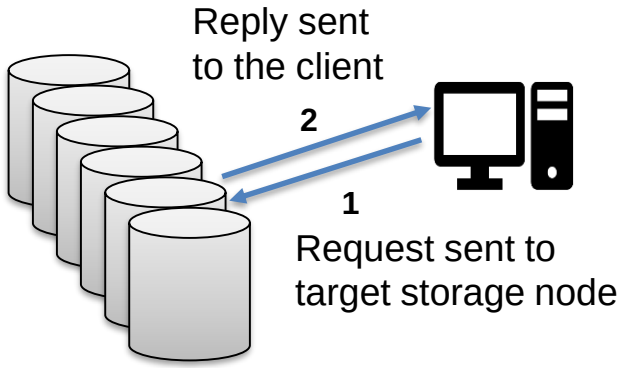
× Increase query response time.
✓ Client doesn't need to link any code to the KV store.

Master-node Coordination



× Increase query response time.
× Single point of failure.
✓ Client doesn't need to link any code to the kV store.

Client-driven Coordination

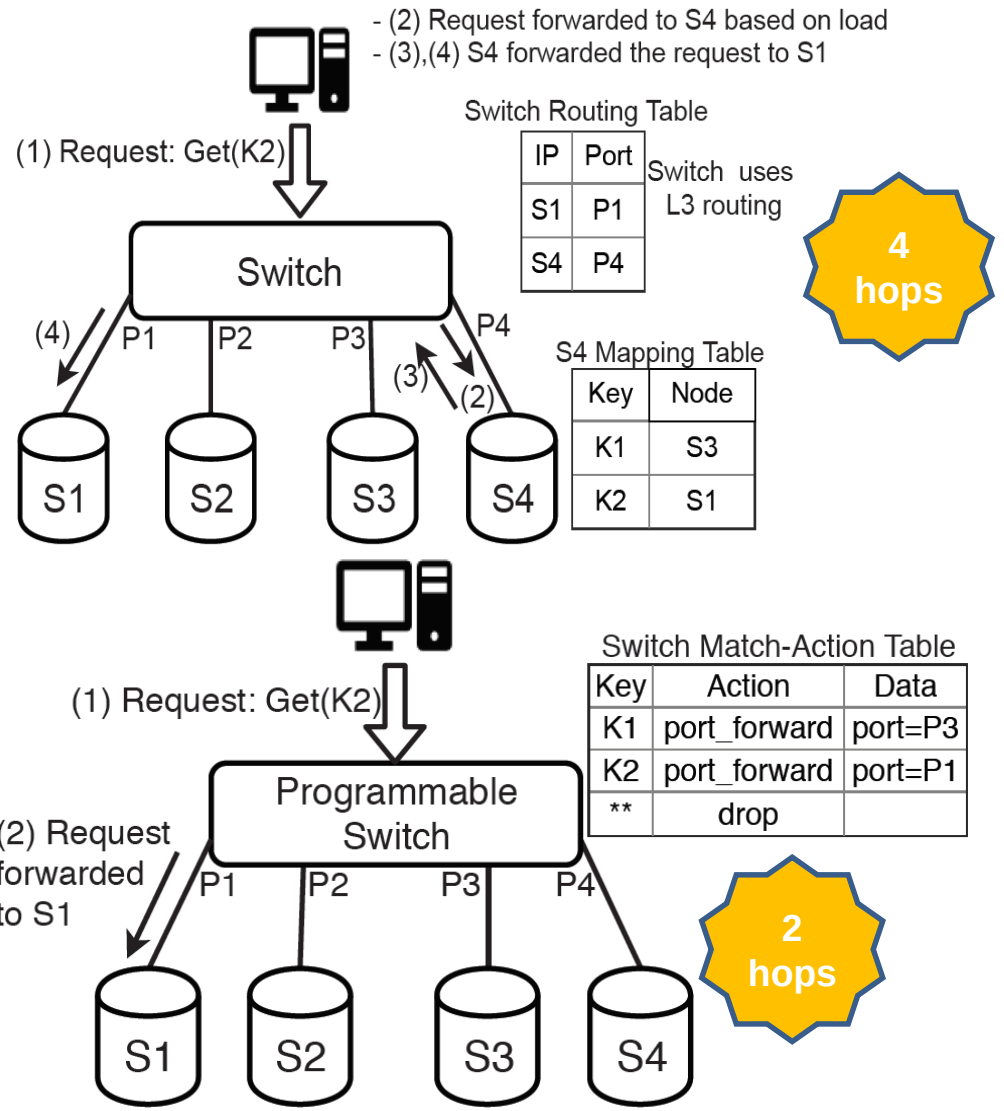


× Periodic pulling of updated directory info.
× client needs to link code related to the used KV store.
✓ Decrease query response time.

Why Switch-driven Coordination?

	99.9th percentile RL	99.9th percentile WL	Average RL	Average WL
Server-driven	68.9	68.5	3.9	4.02
Client-driven	30.4	30.4	1.55	1.9

Performance of client-driven and server-driven coordination approaches (msec) [DeCandia,SOSP'07]



- Requests pass by network switches to arrive at their target.
- Switch-driven Coordination can carry out
  - Partitions management
  - Query routing

  In network switches.

✓ Higher Throughput
✓ Lower R/W Latency

Objectives

- Design in-switch indexing scheme to manage the directory information records.
- Adapt the scheme to the match-action pipeline in the programable switches.
- Utilize switches as a monitoring system for data popularity and storage nodes load.
- Scale up the scheme to multiple racks inside the data center network.

Design Issues

- Data Partitioning
- Data Replication
- Index Table Design
- Network Protocol
- Key-value Operations Processing
- Load Balancing
- Failure Handling
- Scaling up to the data center networks.

TurboKV Overview

Programmable Switches

- Match-action table stores directory information.
- Manages key-based Routing.
- Provide Query statistics reports to controller.

System Controller

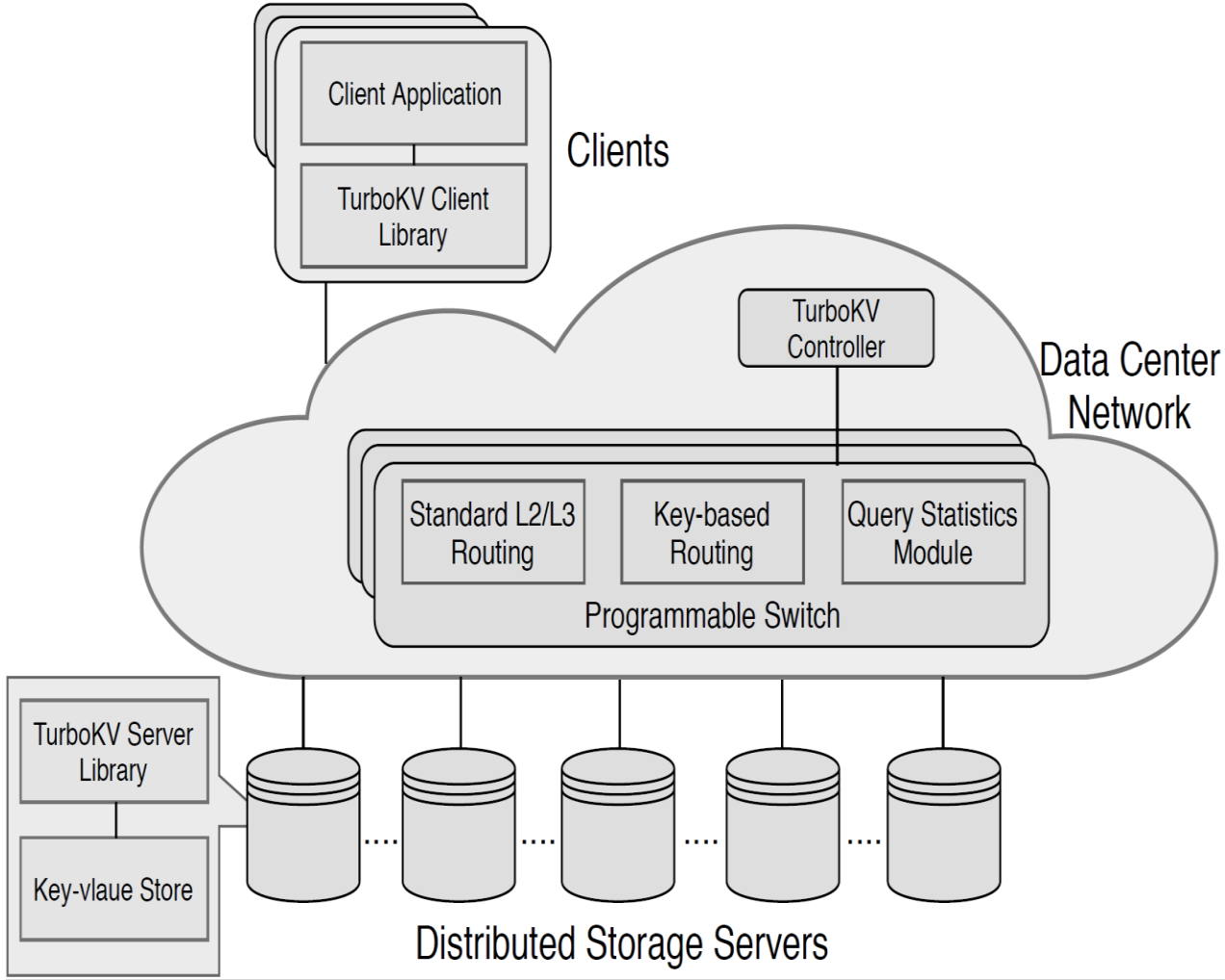
- Load balancing between the storage nodes.
- Updating match-action tables with new location of data.
- Handle failures.

Storage Nodes

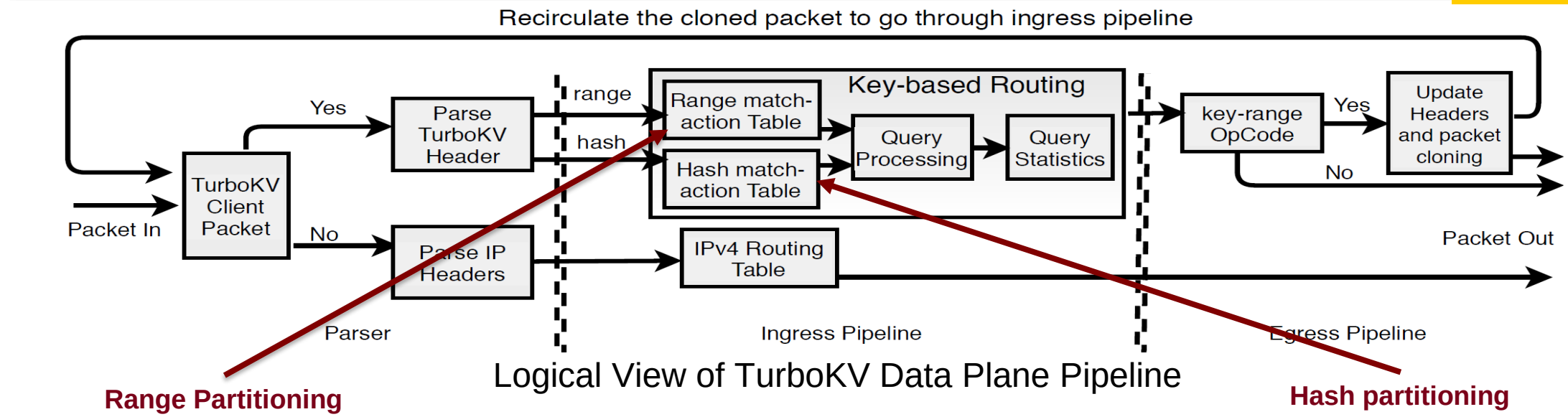
- Server library to translate TurboKV packet to the used key-value store.

System Clients

- Client library to construct TurboKV request packets.



TurboKV Data plane Design (1/3)

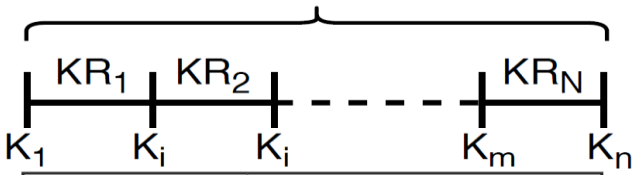


Logical View of TurboKV Data Plane Pipeline

Range Partitioning

Hash partitioning

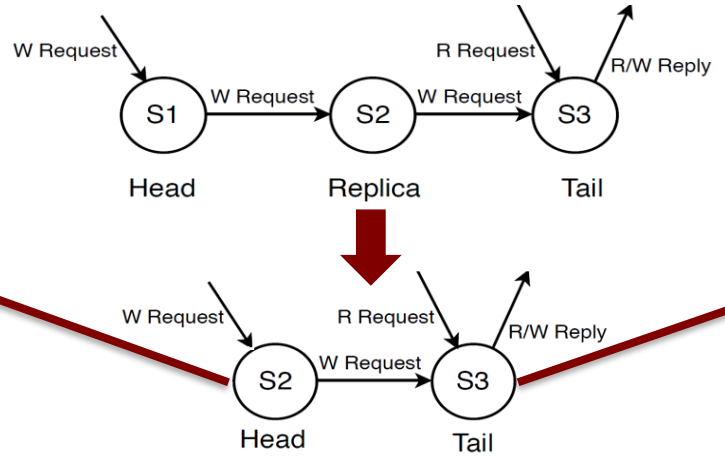
Total Key Span



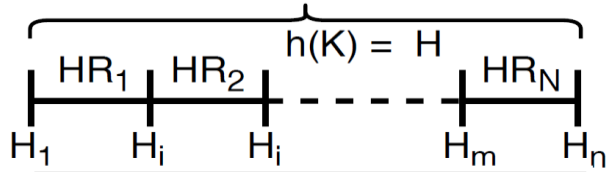
Key Range	Storage Nodes
K_1 -- K_i	R1 = S2, R2 = S3
K_{i+1} -- Kj	R1 = S1, R2 = S4

K_m -- K_n	R1 = S4, R2 = S3

Chain Replication



Hash Function Output Range

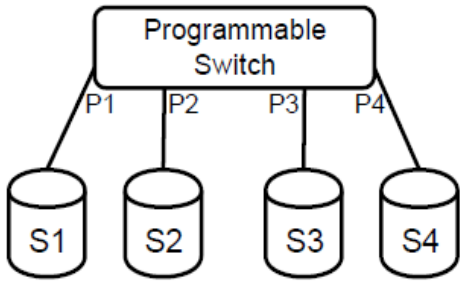


Hash Range	Storage Nodes
H1 -- Hi	R1 = S2, R2 = S3
H_{i+1}-- H_j	R1 = S1, R2 = S4

H_m -- H_n	R1 = S4, R2 = S3

TurboKV Data plane Design (2/3)

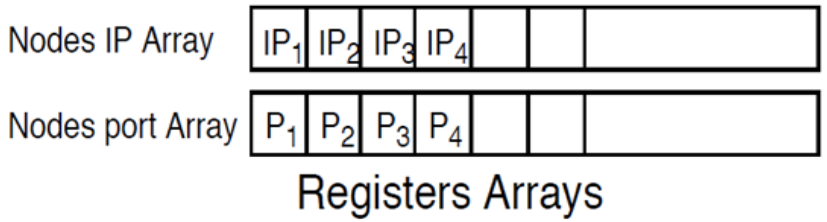
On-Switch Index Table



Sub-range	Storage Nodes
Sub-range1	IP_1, IP_2, IP_3
Sub-range2	IP_2, IP_3, IP_4
Sub-range3	IP_3, IP_4, IP_1
Sub-range4	IP_4, IP_1, IP_2

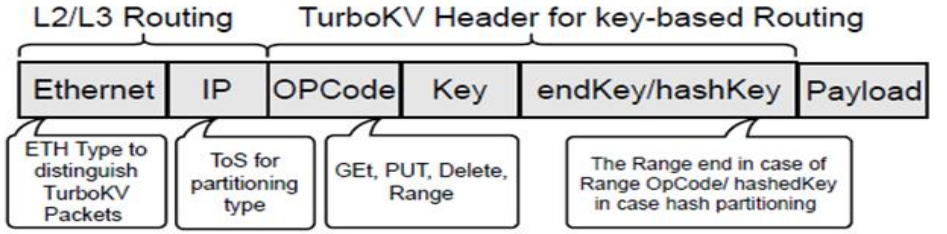
Match	Action	Action data
sub-range1	key_based_routing	chain = 1,2,3 length = 3
sub-range2	key_based_routing	chain = 2,3,4 length = 3
sub-range3	key_based_routing	chain = 3,4,1 length = 3
sub-range4	key_based_routing	chain = 4,1,2 length = 3

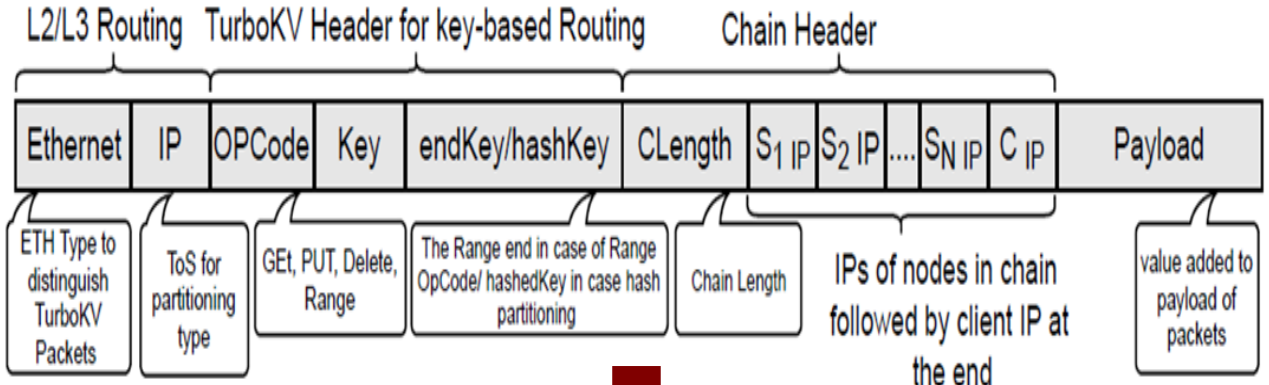
Switch Match-Action Table

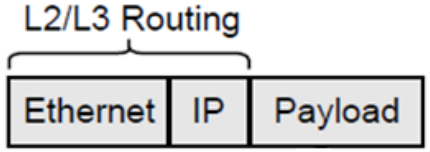


Registers Arrays

Network Protocol



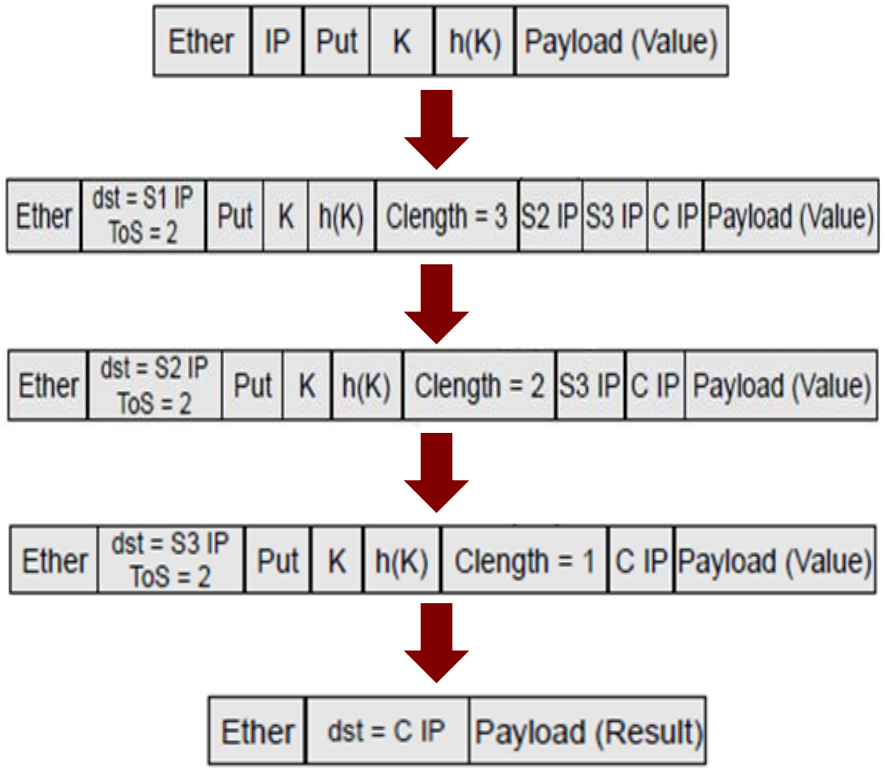


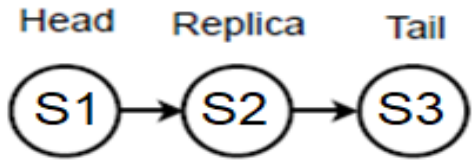


TurboKV Data plane Design (3/3)

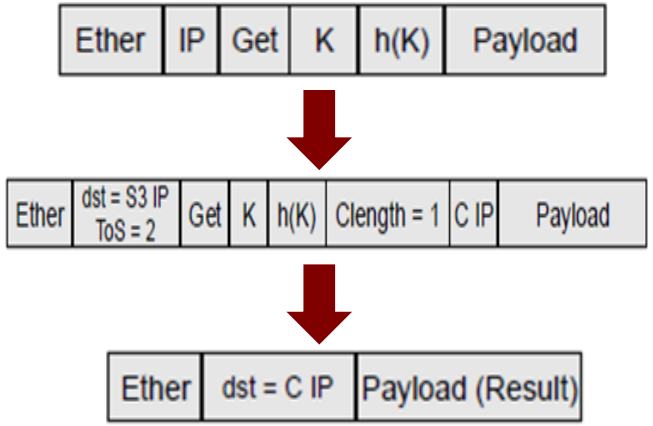
Key-value Operations Processing

PUT (K,value)

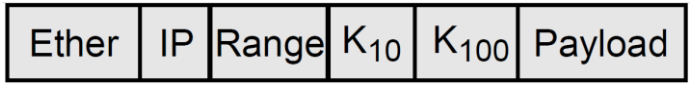




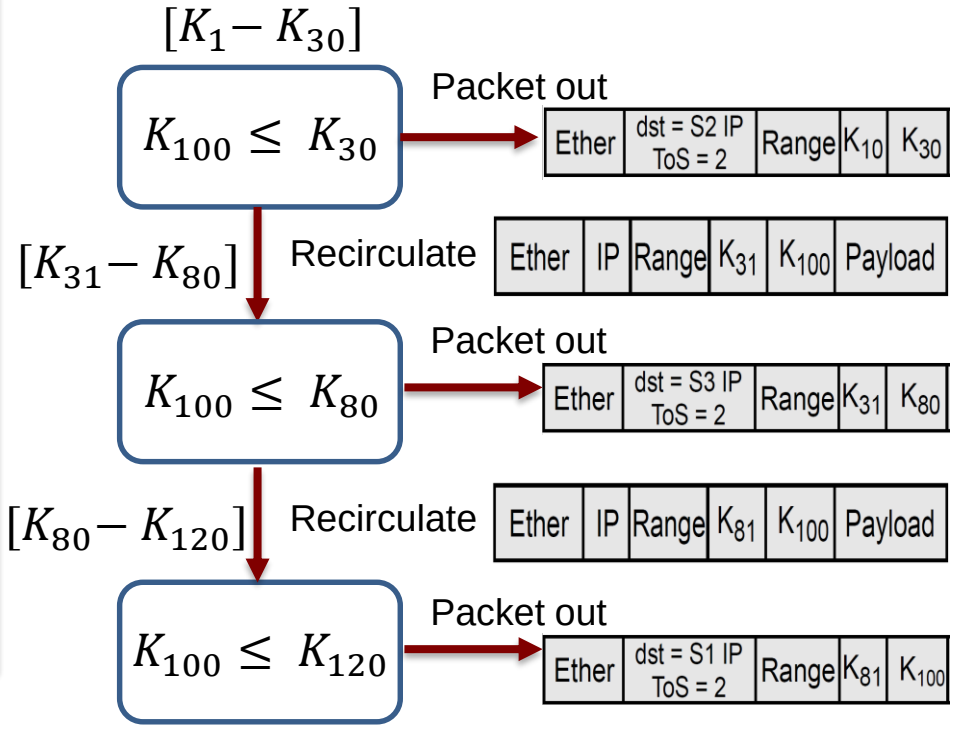
GET (K)



RANGE (K_{10}, K_{100})

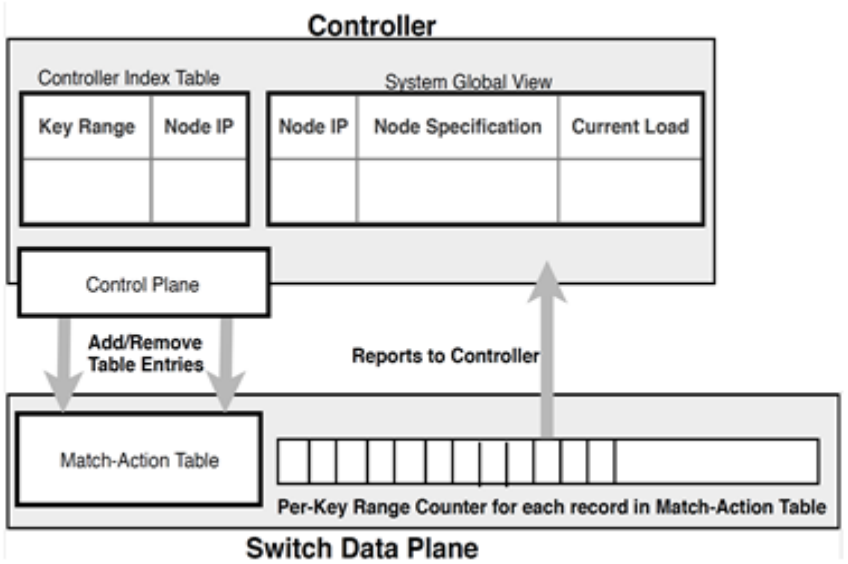


At egress pipeline



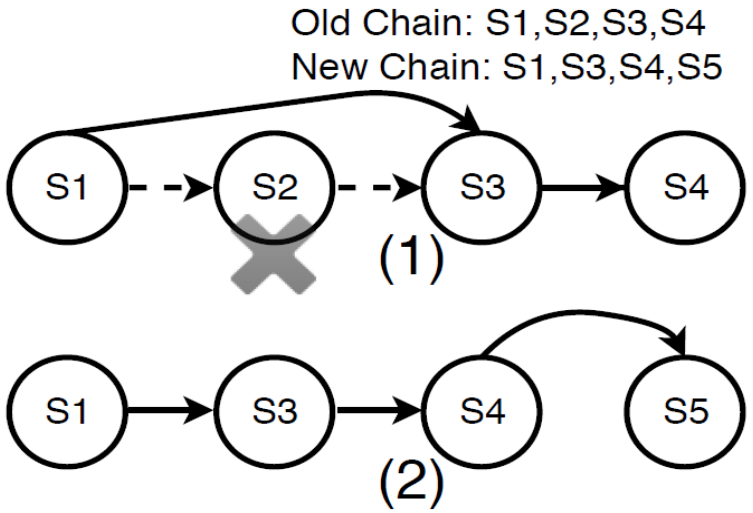
TurboKV Control plane Design

Query Statistics and Load Balancing



- Switches count requests directed to each storage node to estimate the load
- Controller
  - pulls monitoring information from switches.
  - takes migration decisions.
  - updates switches' match-action tables
  - sends data migration commands to storage nodes.

Storage Failure Handling



- Controller reconfigures the chains of all sub-ranges on the failed storage node.
  - removes the failed storage node from all chains.
  - predecessor of failed node will be followed by its successor.
  - distributes the data on the failed node in sub-ranges units among other functional nodes.
  - adds new nodes at the end of sub-ranges' chains.

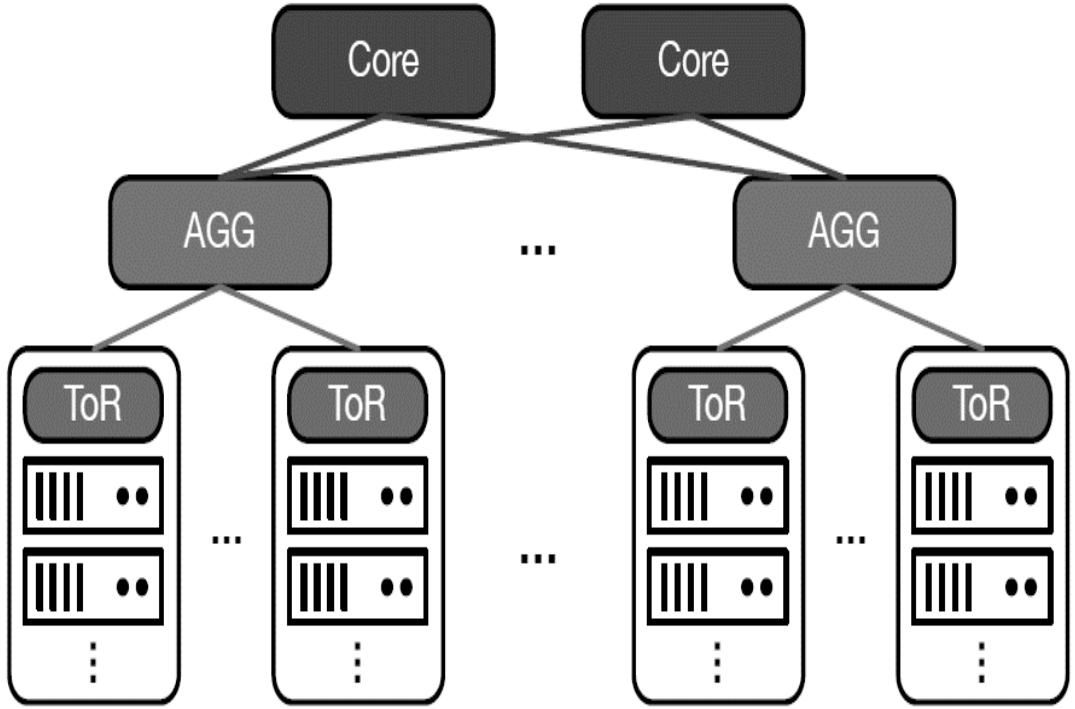
Scaling Up TurboKV to Data Center Network

- Hierarchical indexing directory.
- Top levels switches maintain aggregate information from its connected switches.
- Bottom level switches (ToR) maintain detailed records of their local storage nodes.

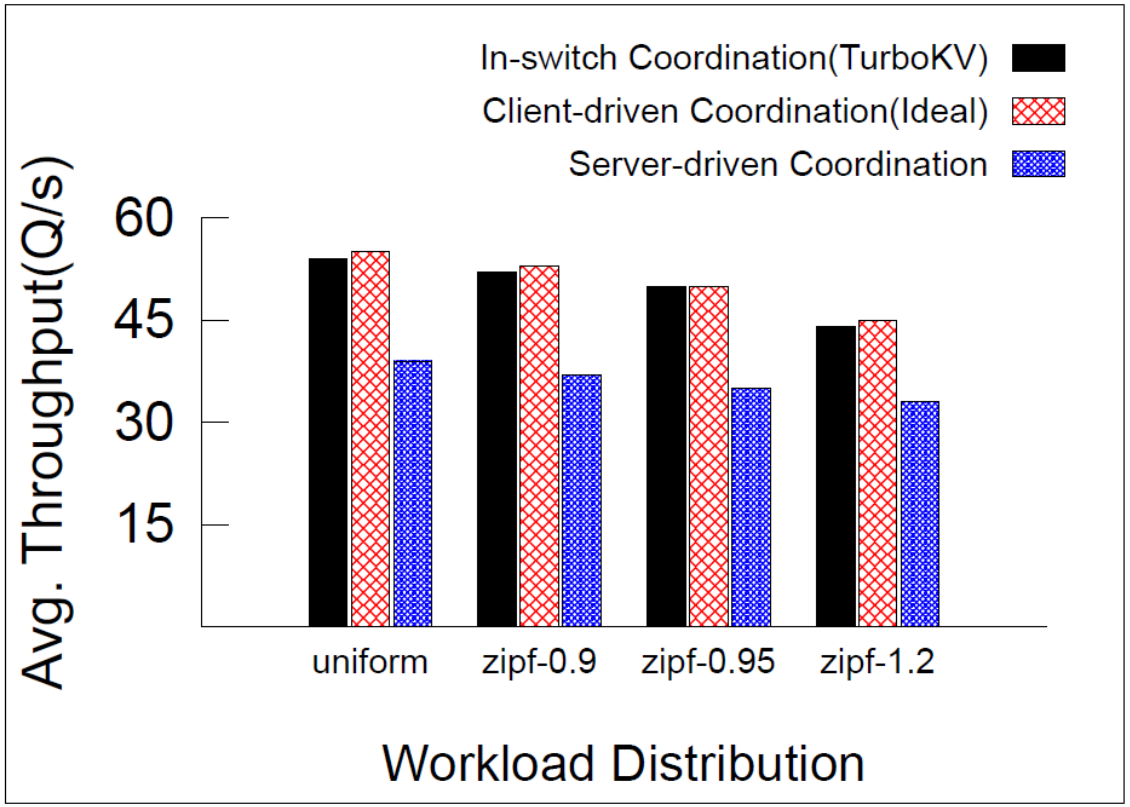
Controller

- keeps track of each index record and its related records on other switches.
- propagates any record's update to all affected switches.

Guarantees consistency between the switches to reflect any data migration or storage node failures

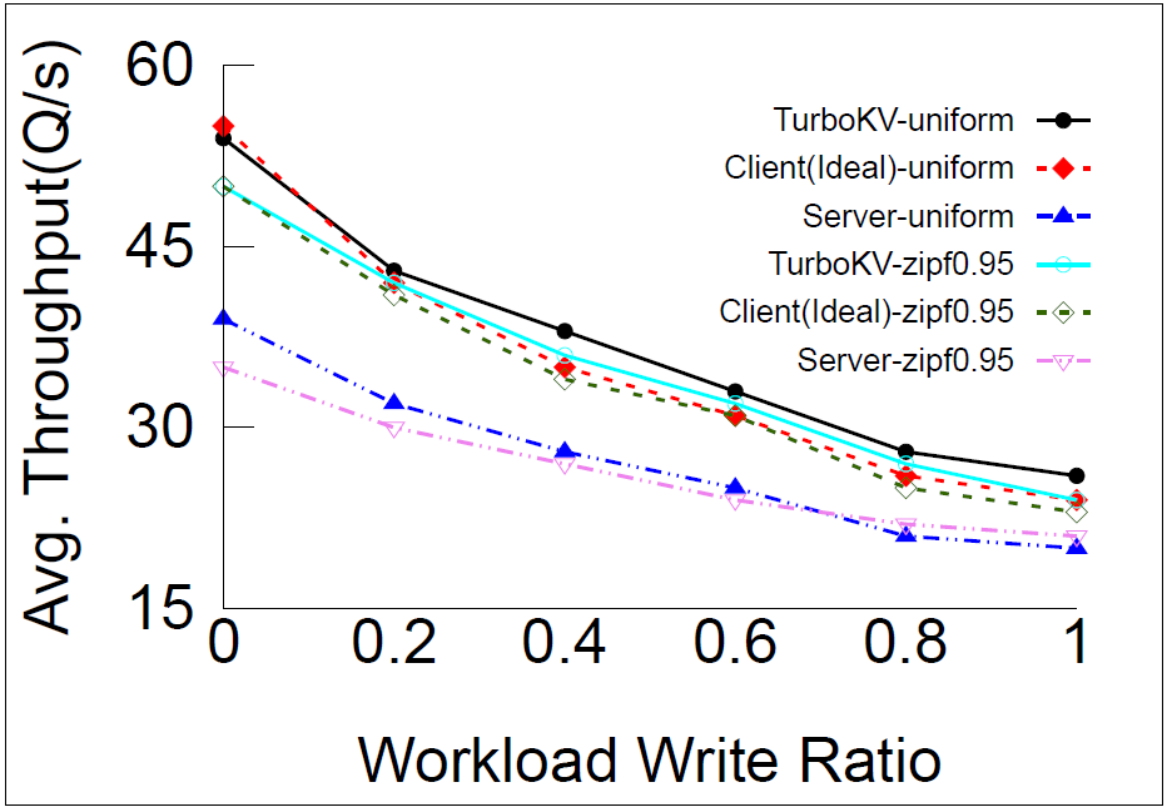


Simulation Results (1/2)



Throughput vs Skewness - Read only

- TurboKV performs as Ideal C. C. while removing the management load from the client side.
- TurboKV outperforms S. C. by **33% -- 42%**.

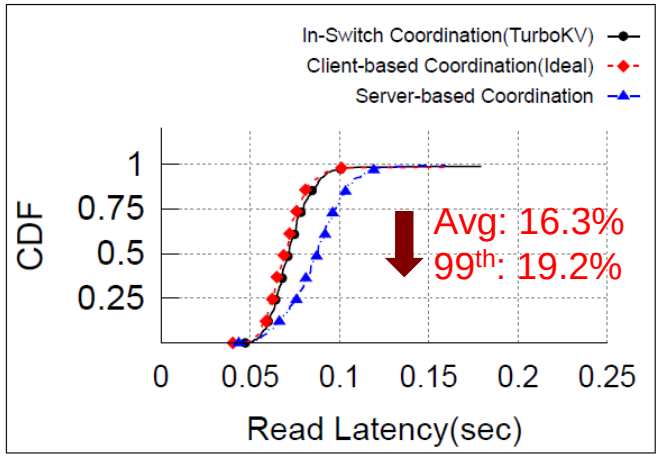


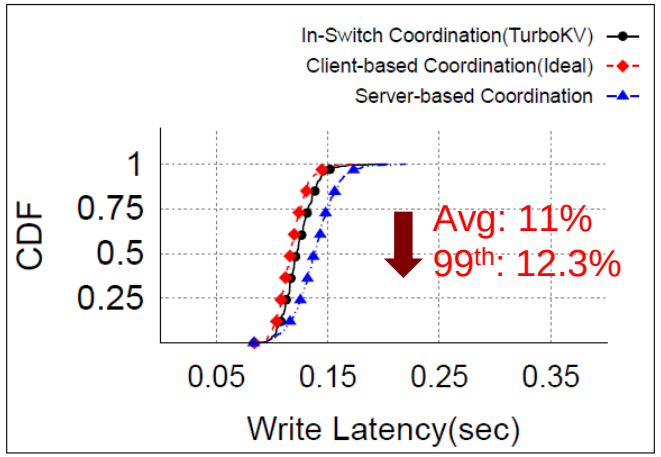
Impact of Write Ratio on System Throughput

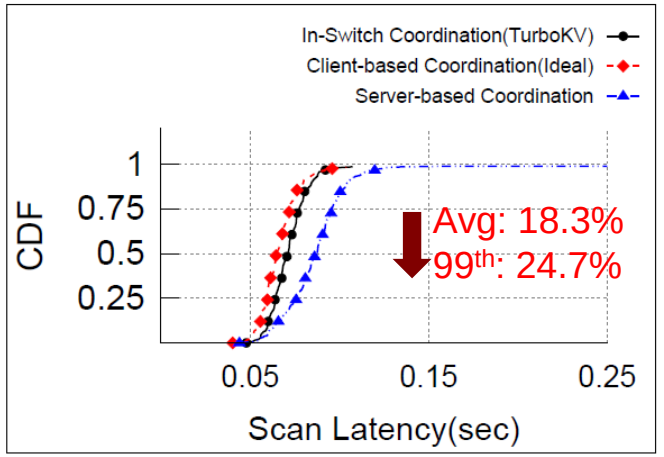
- TurboKV outperforms Ideal C. C. in high write ratio workloads.
- TurboKV outperforms S. C. by **30% -- 38%** in uniform workload, and by **14% -- 42%** in the skewed workload

Simulation Results (2/2)

Key-value operations Latency for uniform Workload

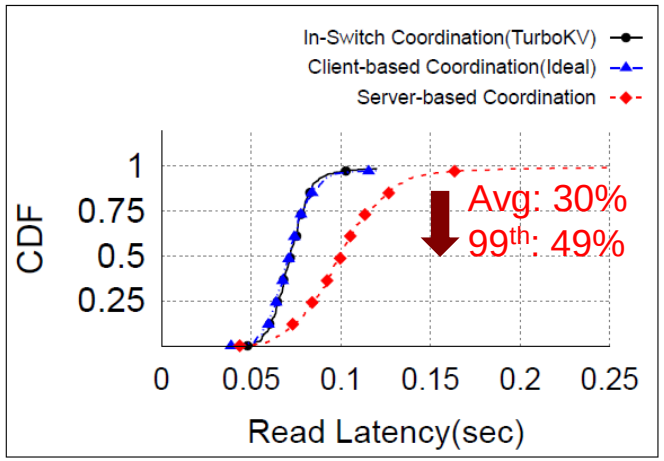


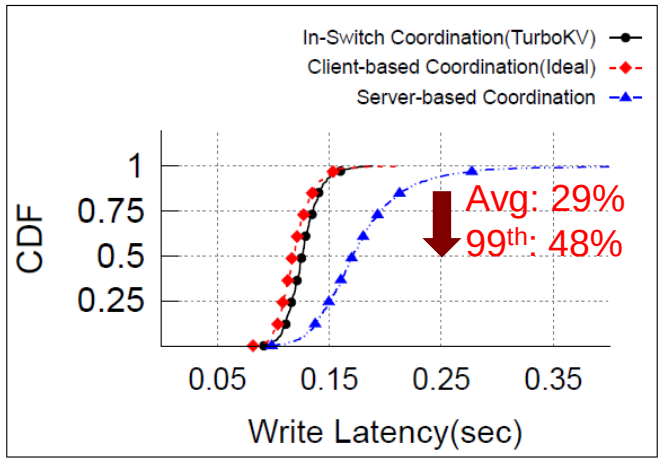


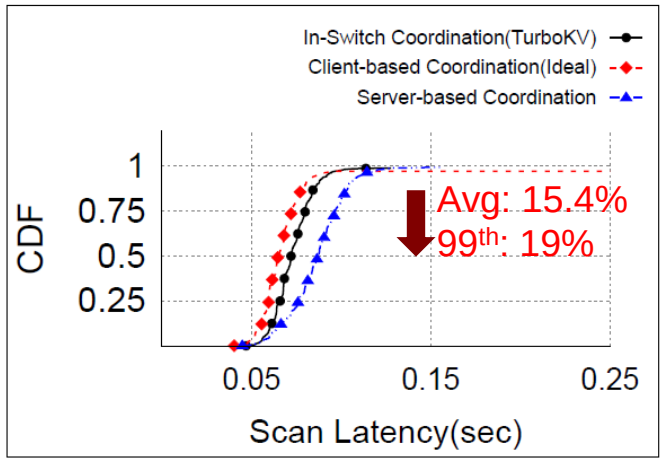


7 -- 10%
With C. C.

Key-value operations Latency for zipf-1.2 Workload





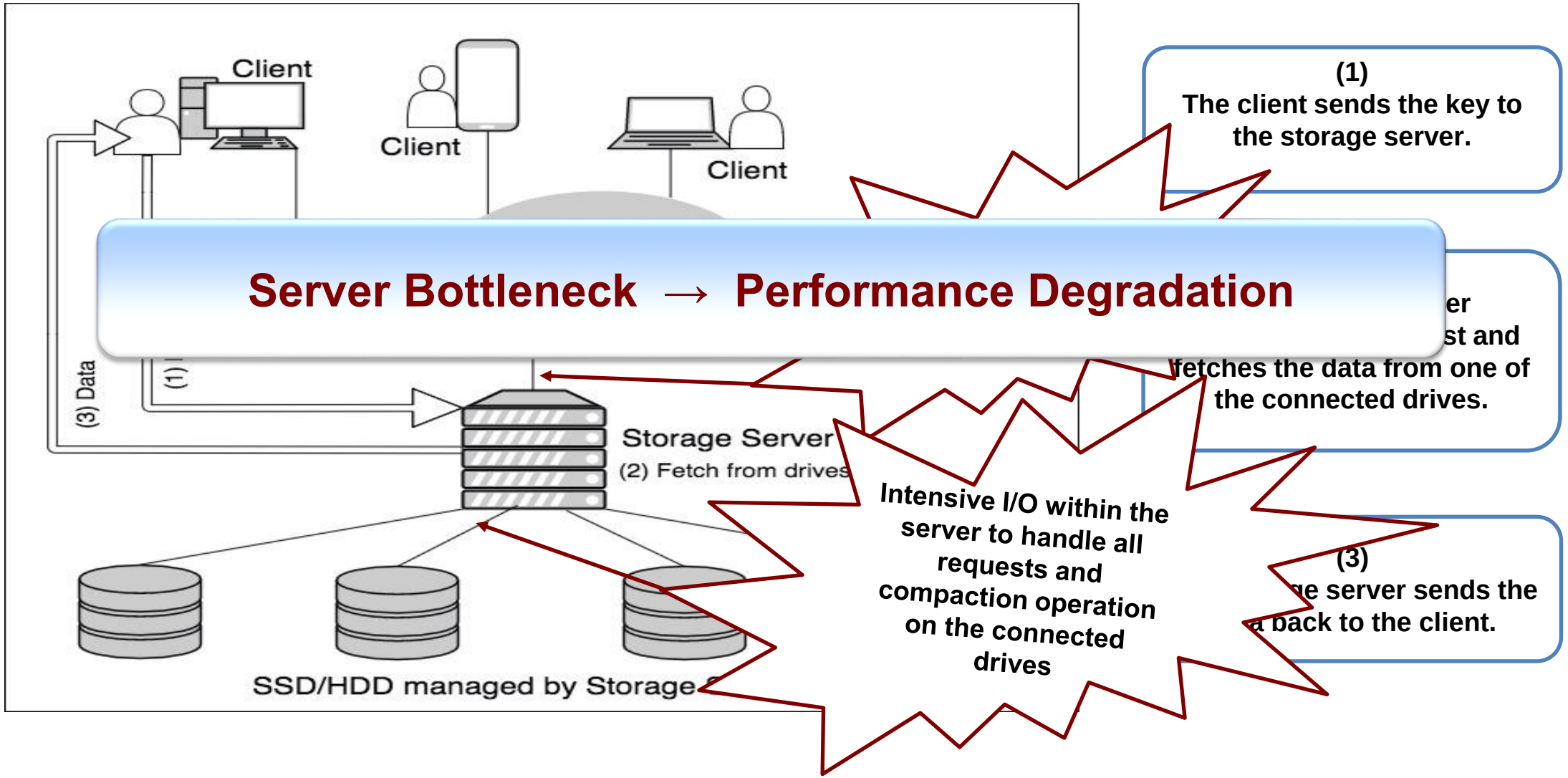


Completed Work (2/2)

Key-value Pairs Allocation Strategy for Kinetic Drives[1]

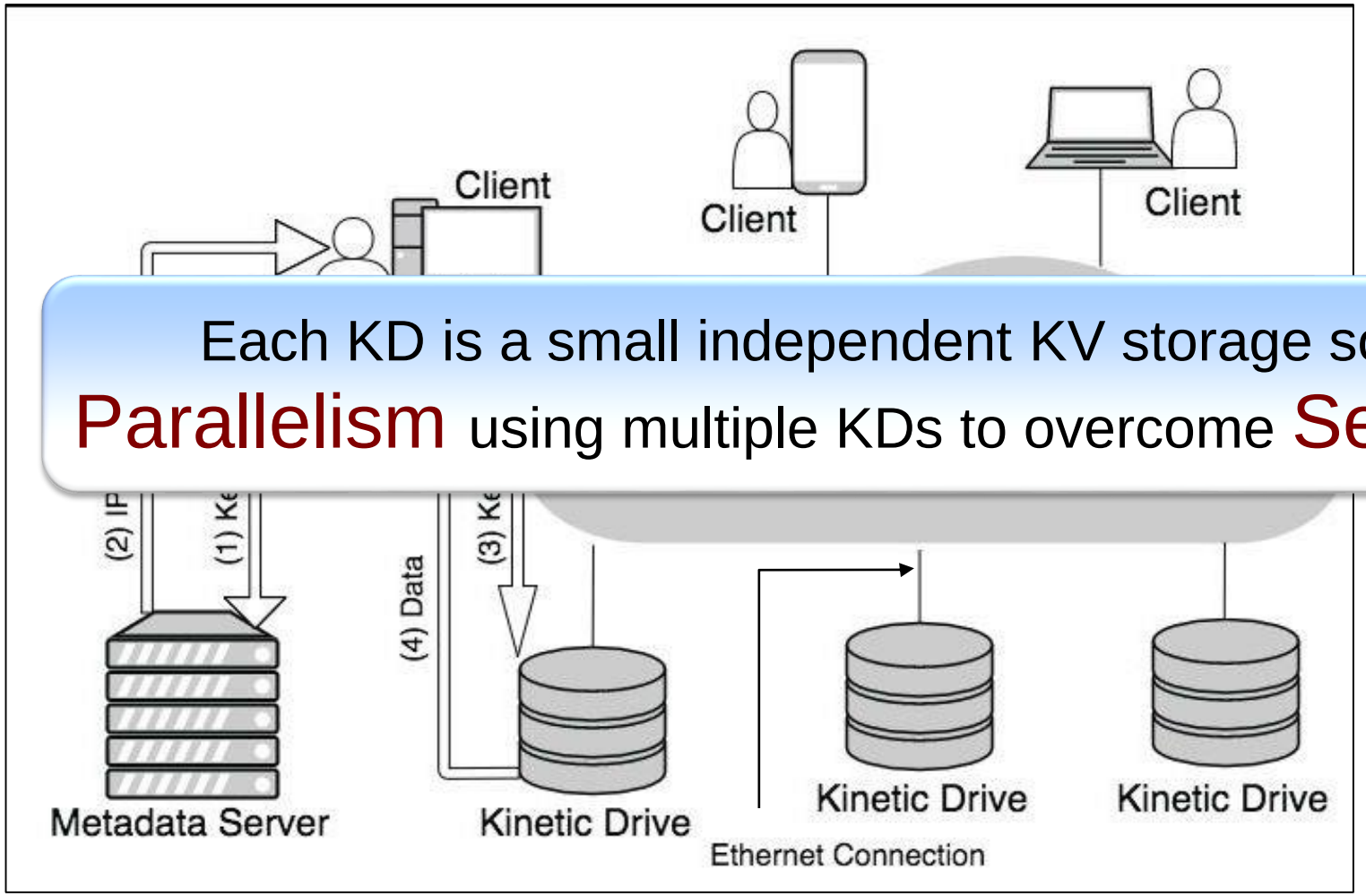
[1] Hebatalla Eldakiky, David H.C. Du, "Key-Value Pairs Allocation Strategy for Kinetic Drives," *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, Bamberg, 2018, pp. 17-24, doi: 10.1109/BigDataService.2018.00012

Traditional KV Store Communication Model



Server Bottleneck → Performance Degradation

Kinetic Drive KV Store Communication Model



Each KD is a small independent KV storage so we can exploit **Parallelism** using multiple KDs to overcome **Server Bottleneck**

(1)
The client sends the key to the Metadata server.

(3)
The client contacts the drive with the IP and sends the key to it.

(4)
The drive processes the request locally and sends the data back to the client.

Motivation

By taking the advantage of Kinetic drive as being an independent active device that can carry out all key-value pairs operations on its own.

Goal

Building a low cost Kinetic based key-value Store with its indexing table to exploit parallelism in satisfying user requests and improve the performance of the storage system

Why we are different from others?

- deal with data popularity and the limited drive bandwidth which may lead to performance bottleneck on the drive.
- minimize the number of drives to reduce the cost of building the distributed kinetic-based Key-value store.

Problem Definition and Challenges (1/2)

Problem Statement

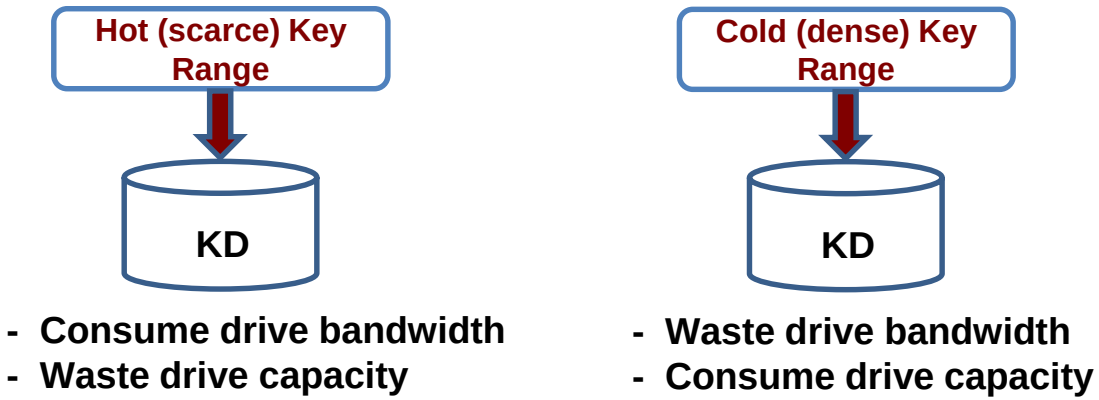
Allocating data into minimum number of kinetic drives to be accessible by applications while satisfying the data size and bandwidth requirements.

Challenges

- Each kinetic drive has **limited size** and **limited bandwidth**.
  - It can only hold certain amount of key-value pairs.
  - It can only serve limited number of requests concurrently.
- User requests are not uniformly distributed across all key ranges (hot key ranges, cold key ranges).
  - Hot key: searched by users frequently (high bandwidth requirement).
  - Cold key: not searched frequently (low bandwidth requirement).

Problem Definition and Challenges (2/2)

- Number of key-value pairs are not uniformly distributed across all key ranges (dense key ranges, scarce key ranges)
  - dense key range: Lots of key-value pairs (high size requirement).
  - scarce key range: few key-value pairs (low size requirement).
- Because of the 80/20 rule in data science, we can see that only 20% of data is accessed 80% of the time and vise versa.

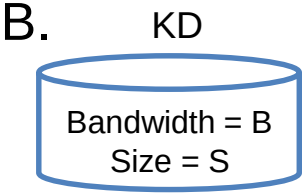


- The metadata server may become a bottleneck point if the searching time for the drive IP takes long time.

Our Approach

Problem Input

- Set of kinetic drives, each of size S and bandwidth B.



- Set of key ranges $KR_1, KR_2, \ldots\ldots, KR_M$ each of them has bandwidth requirement (B_i) and size requirement (S_i).

KR_i

B_i = …..
S_i = …..

- Each of S_i and B_i is a ratio from the drive size and bandwidth.

Min. no. of drives = Max (N_B, N_S)

Theoretical Lower Bounds

$$N_B = \frac{\sum_{i=1}^{M} B_i}{B}$$

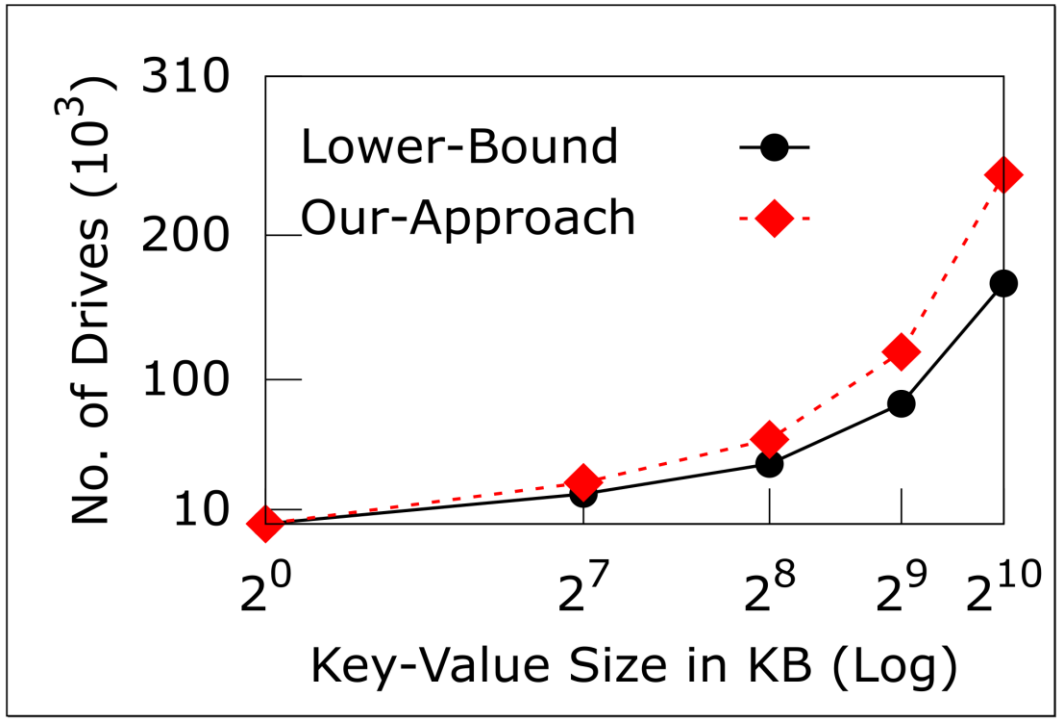
$$N_S = \frac{\sum_{i=1}^{M} S_i}{S}$$

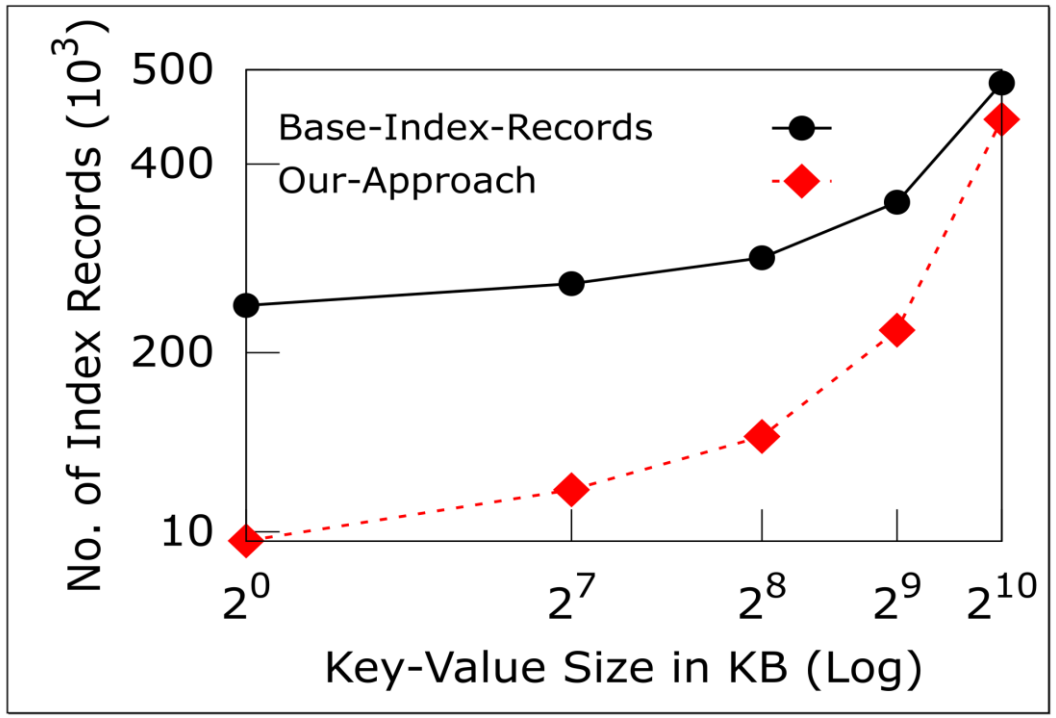
- We modeled the problem as the **multi-capacity bin packing** problem.
  - Each drive represents a bin with multiple capacities (S, B, no. of KR/drive).
  - Each KR represents an item with multiple requirements (size, bandwidth).
- As being a **NP-complete** problem, we develop a heuristic approach to allocate the KR(s) into **near-optimal** no. of drives.
  - key ranges preprocessing to merge some consecutive ranges.
  - Key ranges sorting with weighted sorting function.
  - Key ranges allocation with our proposed best candidate criteria.

Experimental Results (1/2)

- Using the parameters of the current model of Kinetic drive ST4000NK0001 with storage capacity of 4 TB and transfer rate up to 60 MB/s.
- Testing algorithm under different KV pair sizes.
- Performance Metrics
  - the total number of drives used.
  - The size of the index table.
- We compare our approach with the theoretical lower bound on number of drives used and the starting size of index table.

Experimental Results (2/2)





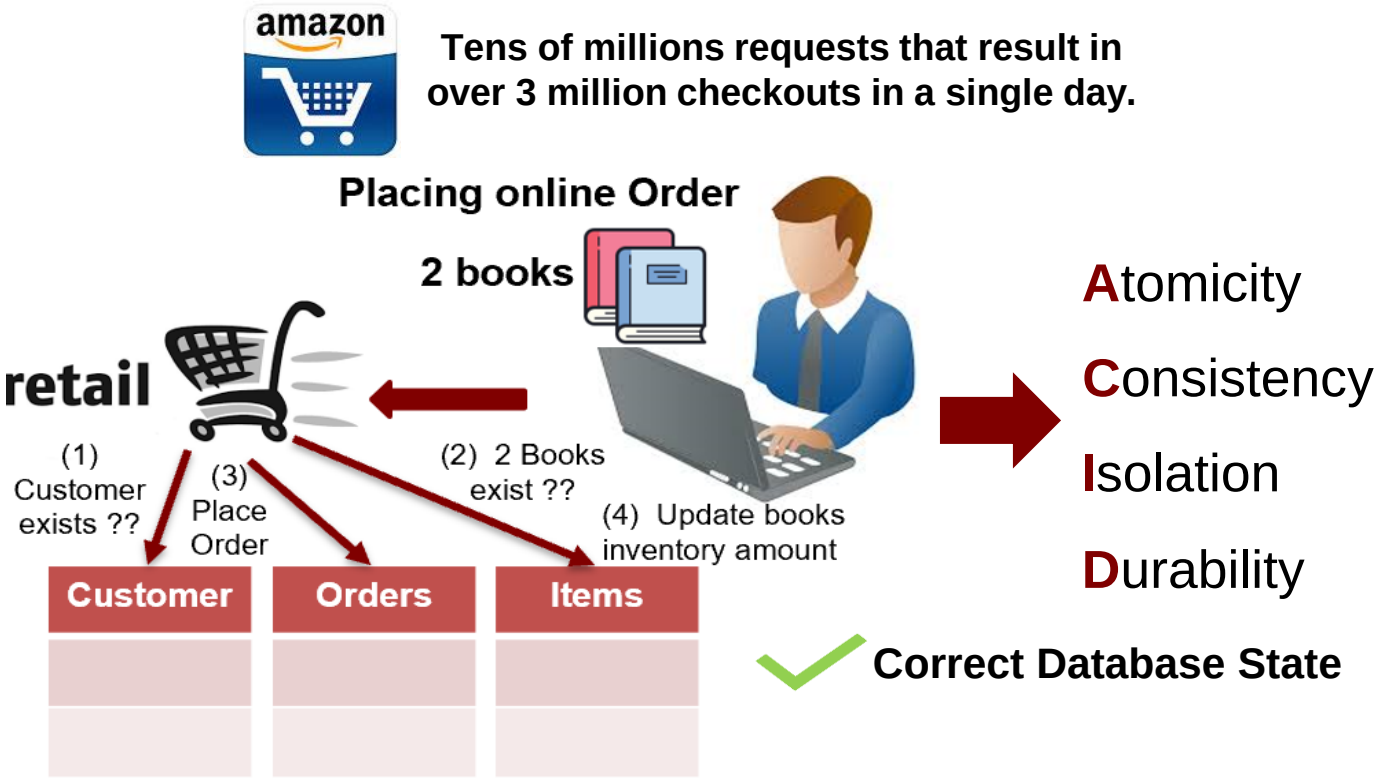
- **No. of drives is closer to the lower bound when KV size is small.**
- **Proposed algorithm results aren't affected by the workload characteristics.**
- **Our approach achieves reduction in the size of the index table up to 57%.**

Proposed Project

TransKV: Networking Support for Transaction Processing in Distributed Key-value Stores

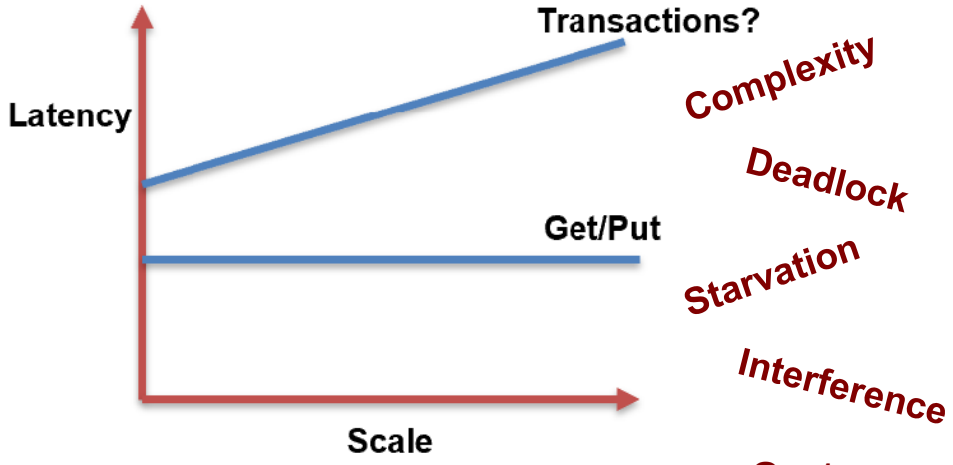
Key-value Stores & Transactions

- Key-value Stores are popular for their simple API, unbounded scalability and predictable low-latency.
- Some applications built on these key-value stores employ non-trivial concurrent transactions from multiple clients.



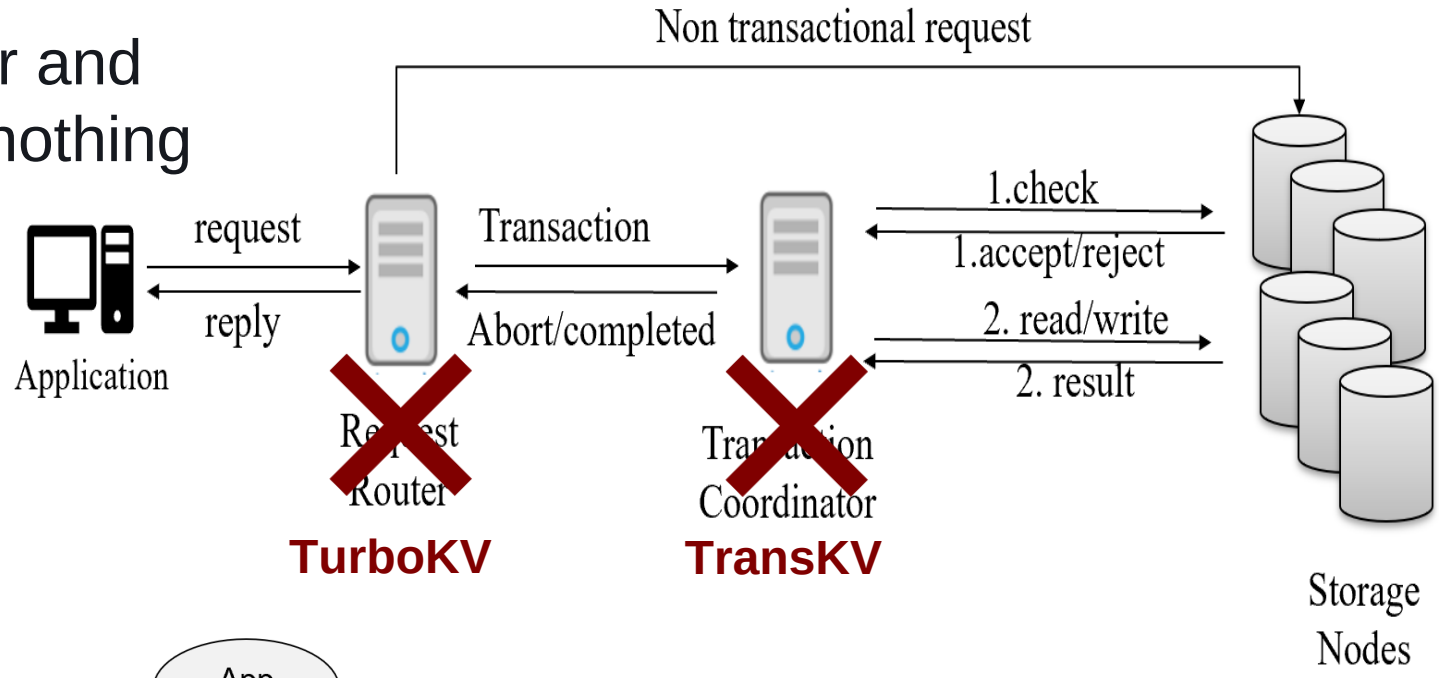
Concern with KV Stores

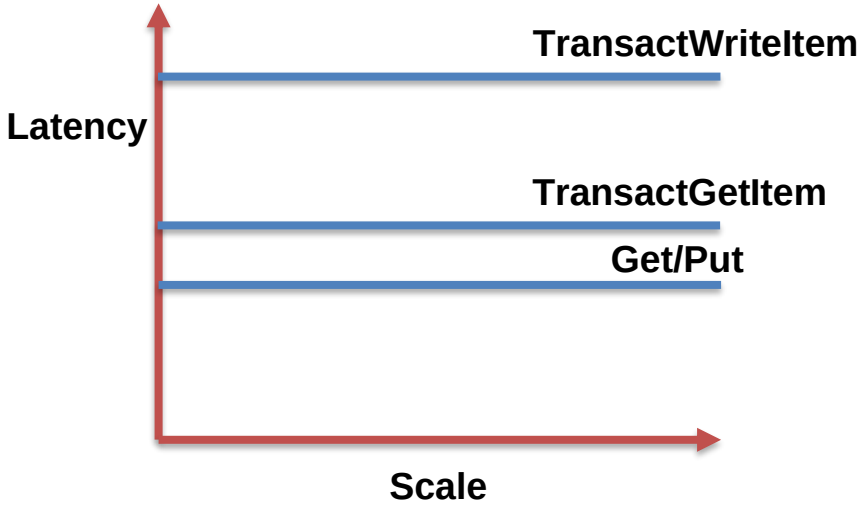
Scalability and Predictable Performance

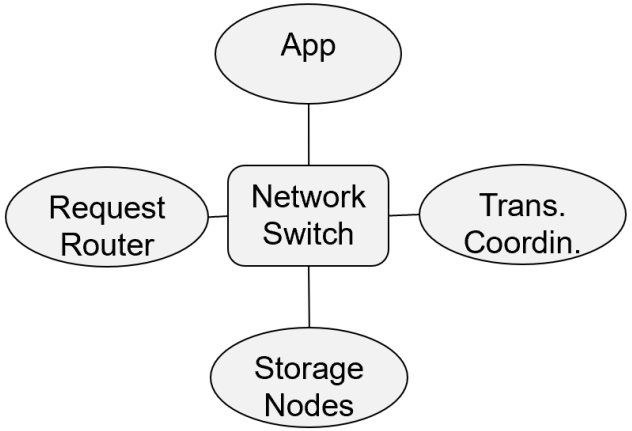


State of art Solution (DynamoDB)

- Group multiple actions together and submit them as a single all-or-nothing operation.
  - **TransactWriteItems.**
  - **TransactGetItems.**





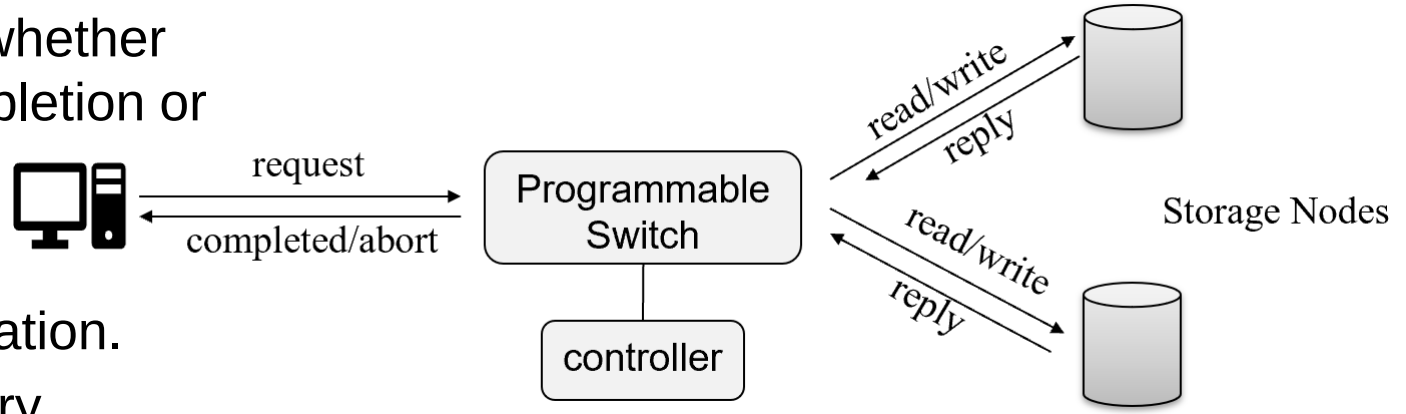


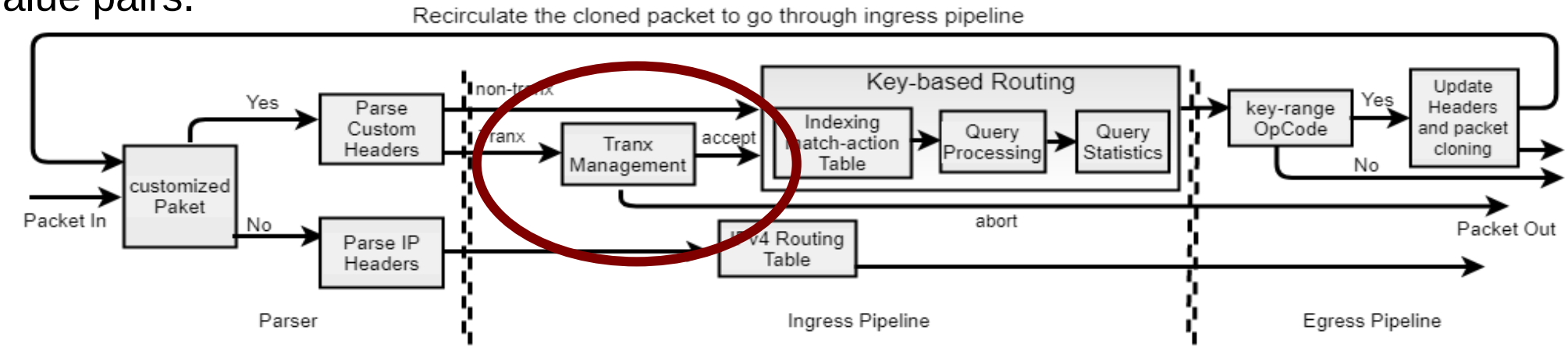
All communications are carried out through network switches→ more forwarding steps

Increase Latency

Proposed Solution (TransKV) (1/2)

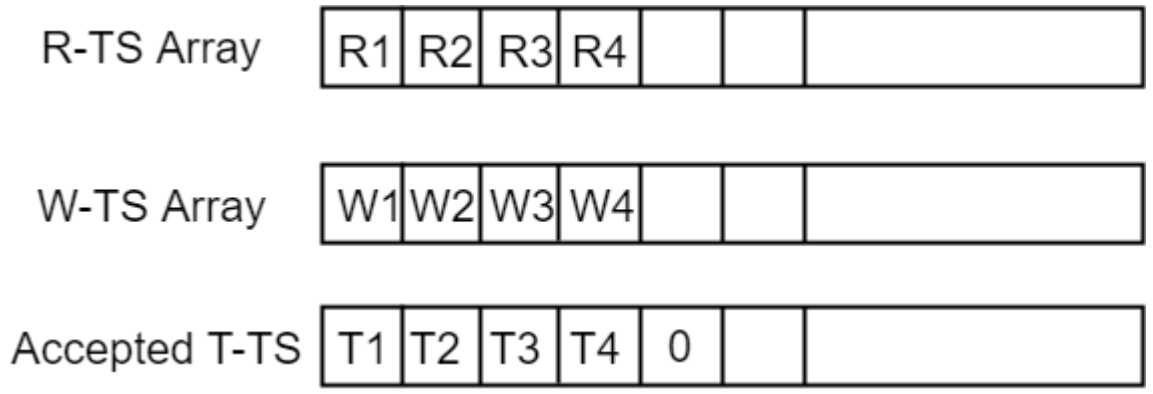
- Programmable Switch
  - Routing requests to target storage nodes.
  - Transaction coordinator to decide whether transaction can be pushed for completion or aborted in the network.
- System Controller
  - Update Cache and indexing information.
  - Log management for failure recovery.
  - Transaction Coordinator for non-cached Key-value pairs.





Proposed Solution (TransKV) (2/2)

- Timestamp Ordering C. C. in the switches and managed by the controller.
- Each transactional operation is cloned and the switch sends a copy to the controller for log management and failure recovery.
- Transaction management is based on the hottest key-value pairs cashed in the switches data plane for space limitation.
- Transactions that span multiple storage nodes with set of operations (read set, write set).
- Hierarchical caching to scale up for data center network.



Match	Action	Action Data
Key1	test-tranx-for-processing	TS-index = 1, val
Key2	test-tranx-for-processing	TS-index = 2, val
Key3	test-tranx-for-processing	TS-index = 3, val
Key4	test-tranx-for-processing	TS-index = 4, val
Key5	test-tranx-for-processing	TS-index = 5, val

Conclusion

- Improve data access performance for distributed key-value stores when applications access storage through network. (In-Network Computing)

- Reduce the amount of data shipped from storage drives to be processed by the host in data intensive applications. (In-Storage Computing)

- Completed Work
  - TurboKV: Scaling Up The performance of Distributed Key-value stores with In- Switch Coordination (In-Network Computing).
  - Key-value pair allocation strategy for Kinetic drives (In-Storage Computing)

- Proposed Work
  - TransKV: Networking Support for Transaction Processing in Distributed Key-value Stores (In-Network Computing)

Future Plan

- Design and Implementation of TransKV.
- December, 2020: Dissertation.
- January, 2021: Defense.

Thank You

Questions