RUTGERS
UNIVERSITY | NEW BRUNSWICK

SAMSUNG

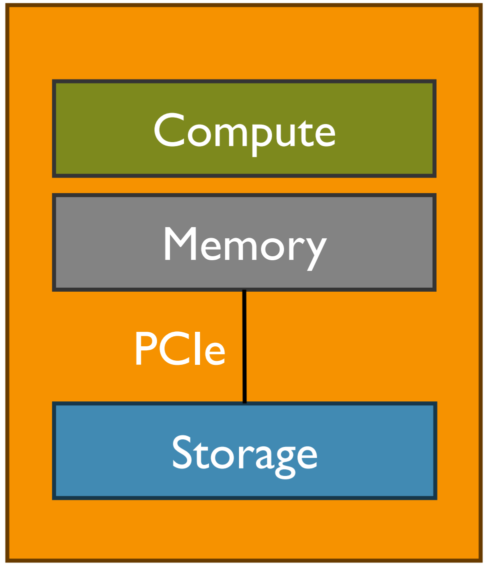
# Can a Client–Server Cache Tango Accelerate Disaggregated Storage?


HotStorage 2025

The 17th ACM Workshop on Hot Topics in Storage and File Systems

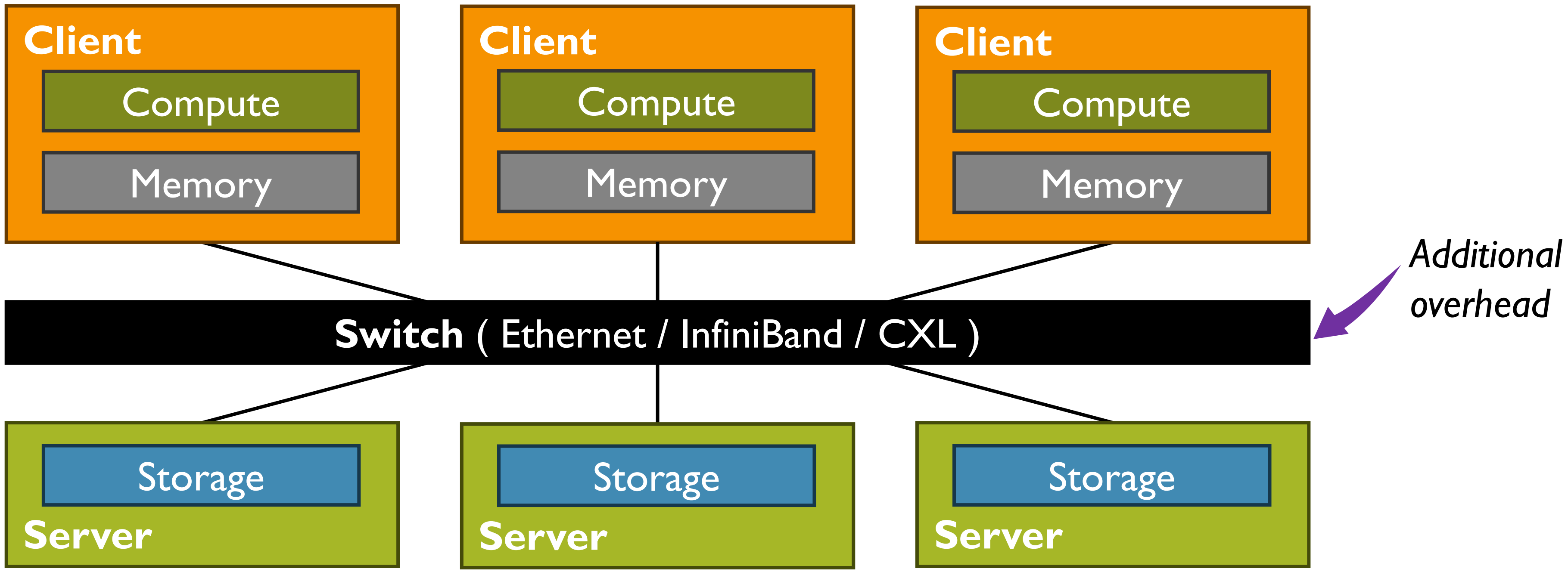
# Direct Attached Flash Storage

- Storage devices directly attached to the host through PCIe
- Compute and memory resources are coupled with storage devices
- Cannot scale to a large capacity (hundreds of TB)



# Disaggregated Flash Storage

- Nodes with compute and memory resources are connected to storage servers
- Benefits of disaggregated storage architecture:
  - Independent scalability
  - Flexible configuration
  - Fault isolation
- However, network brings more communication overhead



# Cache Benefits for Disaggregated Storage

- Absorb requests for frequently accessed data

- Reduce network and storage traffic and contention

- Reduce latency by accessing data in the cache

However, benefits and influences of cache design are under explored/not-public.

# Outline

- **Background**
- Motivation
- Solution
- Future Work and Conclusion

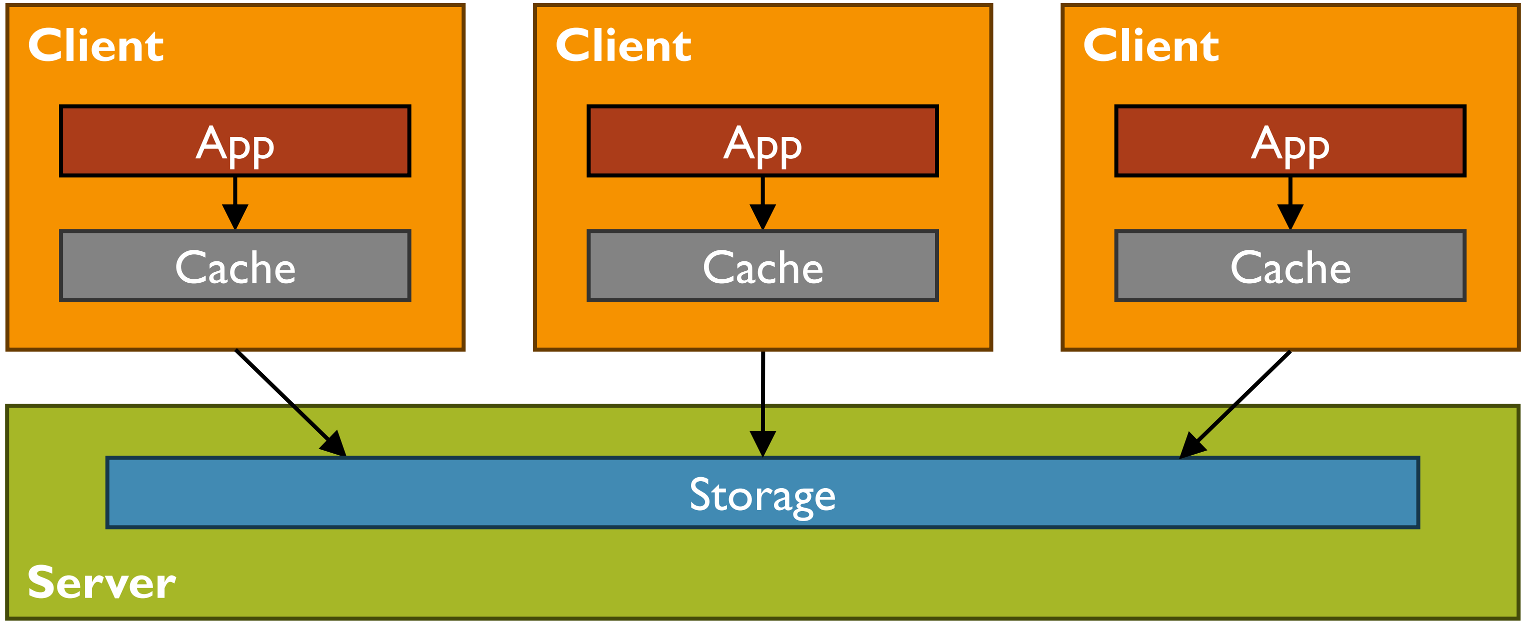
# Cache in Disaggregated Storage Architecture

- **Client Local Caching**
- **Server Only Caching**
- **Client–Server Independent Caching**

# Cache in Disaggregated Storage Architecture

## Client Local Caching

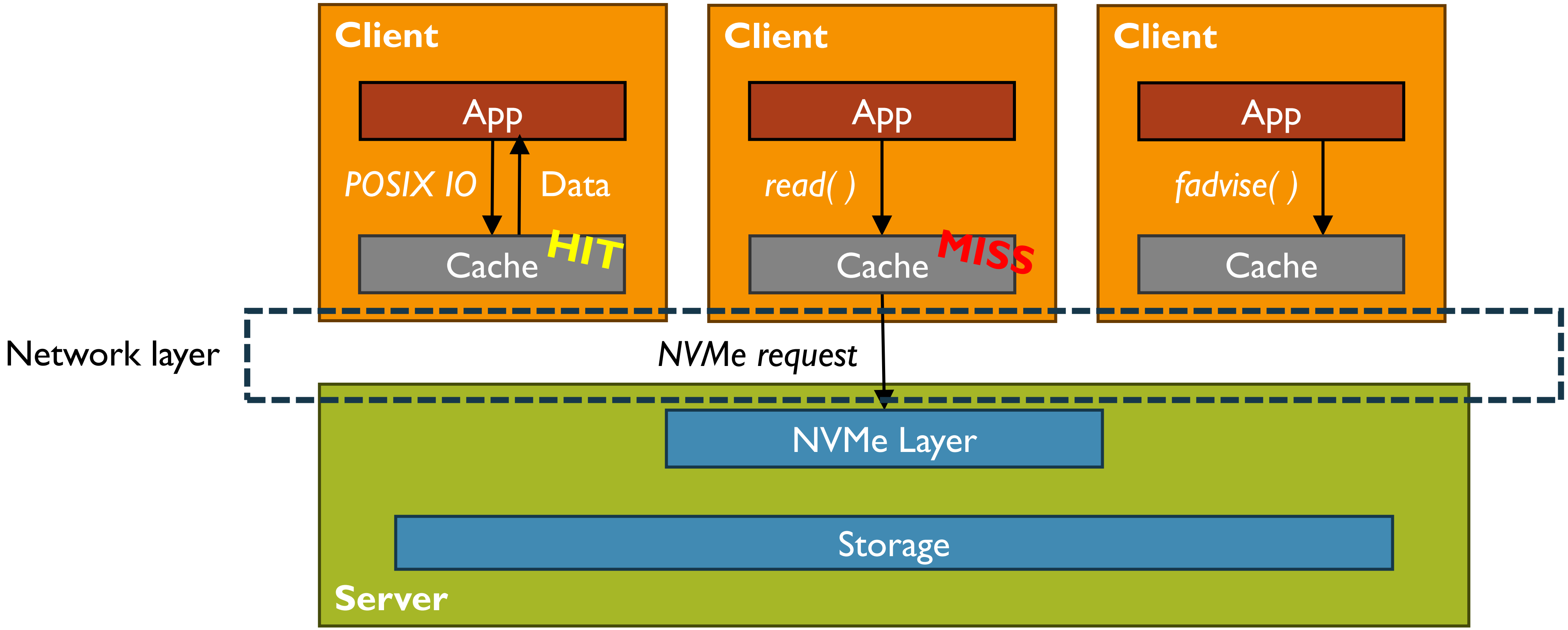
- Utilizing **only the client-side** cache
- **Example:** Linux NVMe over Fabric extension, FS-cache for NFS



# Caching in Disaggregated Storage Architecture

## Client Local Caching

- **Example:** Linux NVMe over Fabric extension



# Cache in Disaggregated Storage Architecture

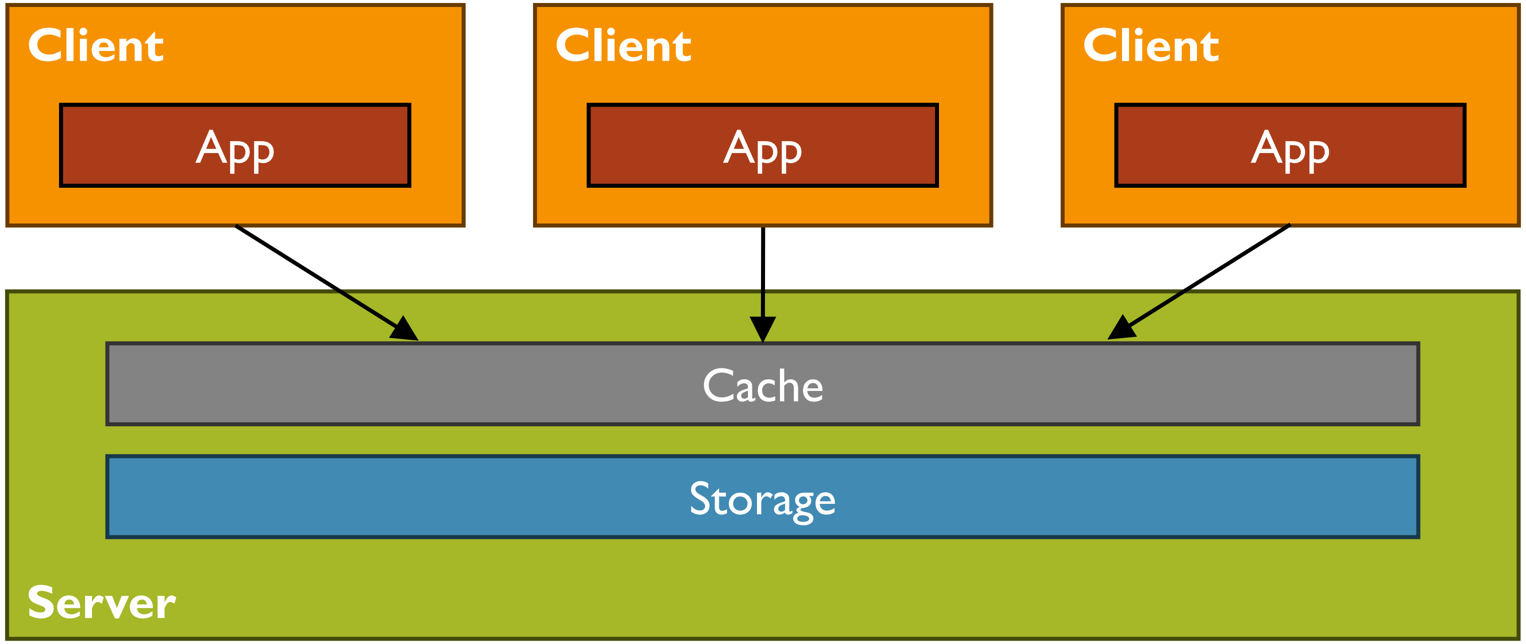
## Client Local Caching

- Benefits:
  - Reduces network communication overhead when locality is good
  - Saves storage server CPU time by letting it only do IO jobs
- Drawbacks:
  - Moves data through network after a cache miss
  - Overlooks memory resource in server-side for caching (up to hundreds of GB)

# Cache in Disaggregated Storage Architecture

## Server Only Caching

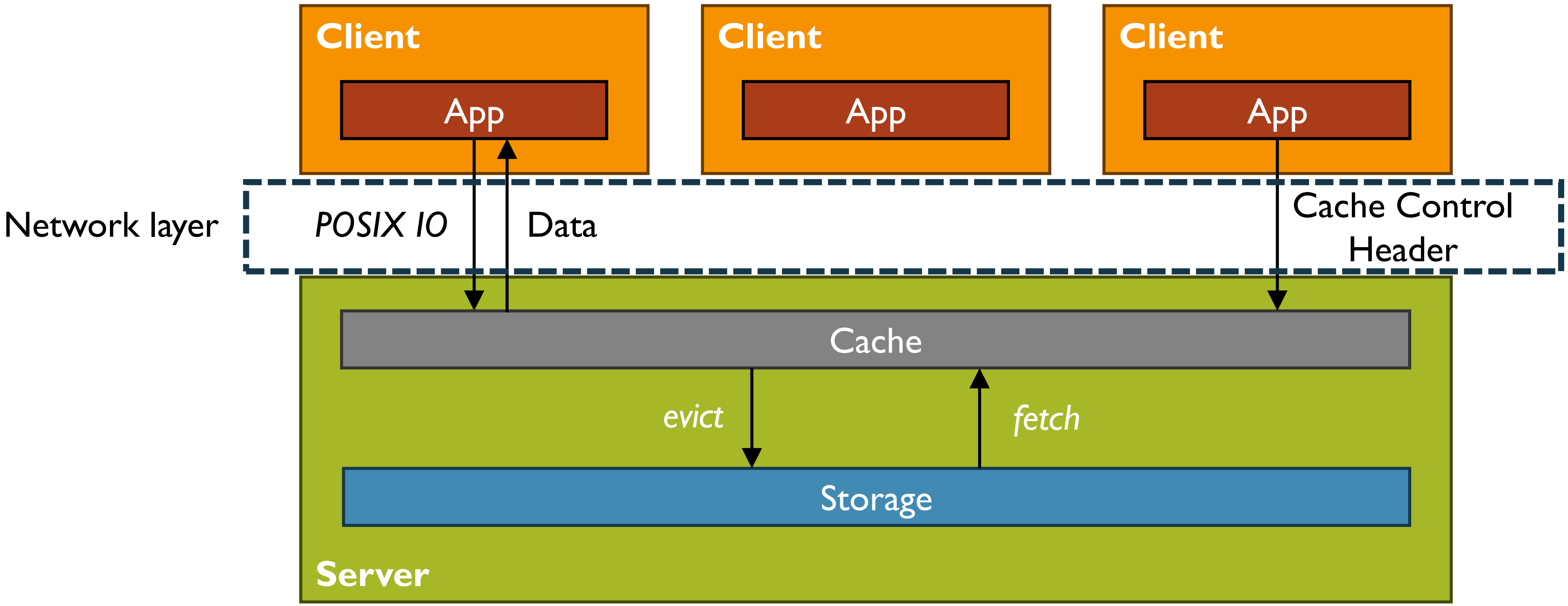
- Deploying cache **only on server side**
- **Example:** Azure Blob Storage, Amazon S3



# Cache in Disaggregated Storage Architecture

## Server Only Caching

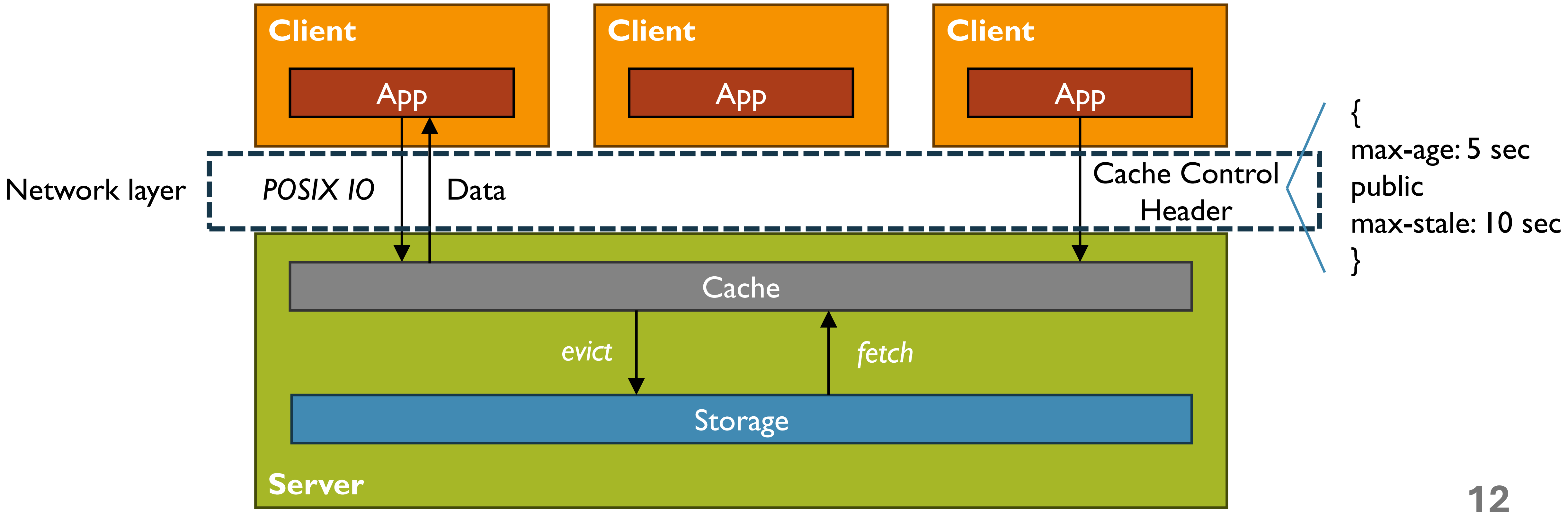
- **Example:** Azure Blob Storage



# Cache in Disaggregated Storage Architecture

## Server Only Caching

- **Example:** Azure Blob Storage



# Cache in Disaggregated Storage Architecture

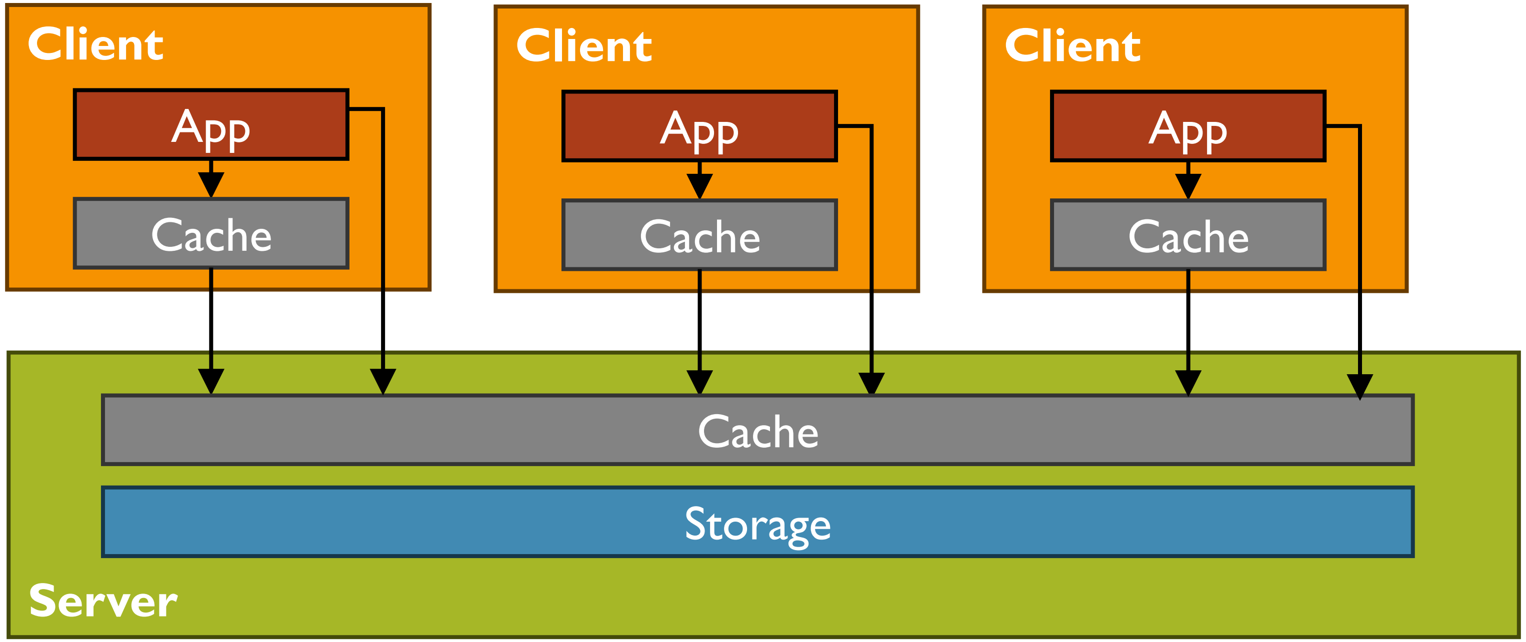
## Server Only Caching

- Benefits:
  - Enables easy client data sharing as all requests go through the server
  - Saves client-side memory for application use
- Drawbacks:
  - Suffers from network latency for every request
  - Causes high server CPU usage due to cache management

# Cache in Disaggregated Storage Architecture

## Client-Server Independent Caching

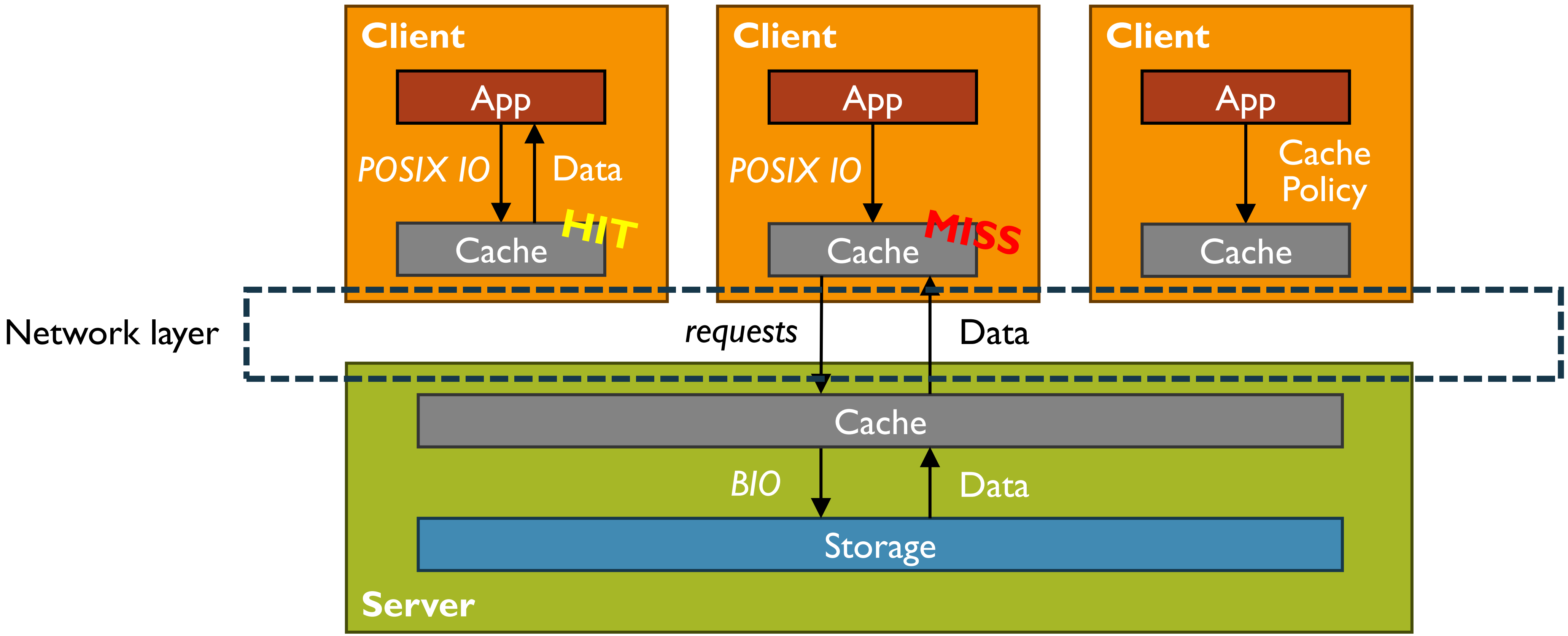
- Leveraging caches on **both sides** that are **managed independently**
- **Example:** Azure Managed Disk, Databricks Query Caching



# Cache in Disaggregated Storage Architecture

## Client-Server Independent Cache

- **Example:** Azure Managed Disk, backed by Azure Blob Storage



# Cache in Disaggregated Storage Architecture

## Client–Server Independent Caching

- Benefit:
  - Utilizes memory resources as caches in both clients and servers
  - Absorbs write-back with a secondary cache layer, thereby reducing stalls
- Drawback:
  - Causes data duplication due to invisibility between clients and servers
  - Leads to suboptimal behavior due to cache management conflict

# Cache in Disaggregated Storage Architecture

**To sum up:**

- Different caching approaches have fundamentally different architectures
- Strengths and weaknesses of each approach vary across dimensions
- No single approach is optimal in all scenarios

Need a **multi-dimensional** evaluation to uncover the characteristics

# Outline

- Background
- **Motivation**
- Solution
- Future Work and Conclusion

# Experiment Setup

- **Hardware Platform**
  - 1 client node and 1 server node
  - Each machine has
    - one 16-core Intel Xeon Silver CPU
    - 960 GB Samsung PCIe4 NVMe SSDs
    - 100 Gbps Mellanox ConnectX-6 NIC, with a 100 Gbps Ethernet switch
- **Workload**
  - Total of 128 GB split across 32 threads, with cache warm-up
  - For all 3 approaches, total cache size is limited to 36 GB

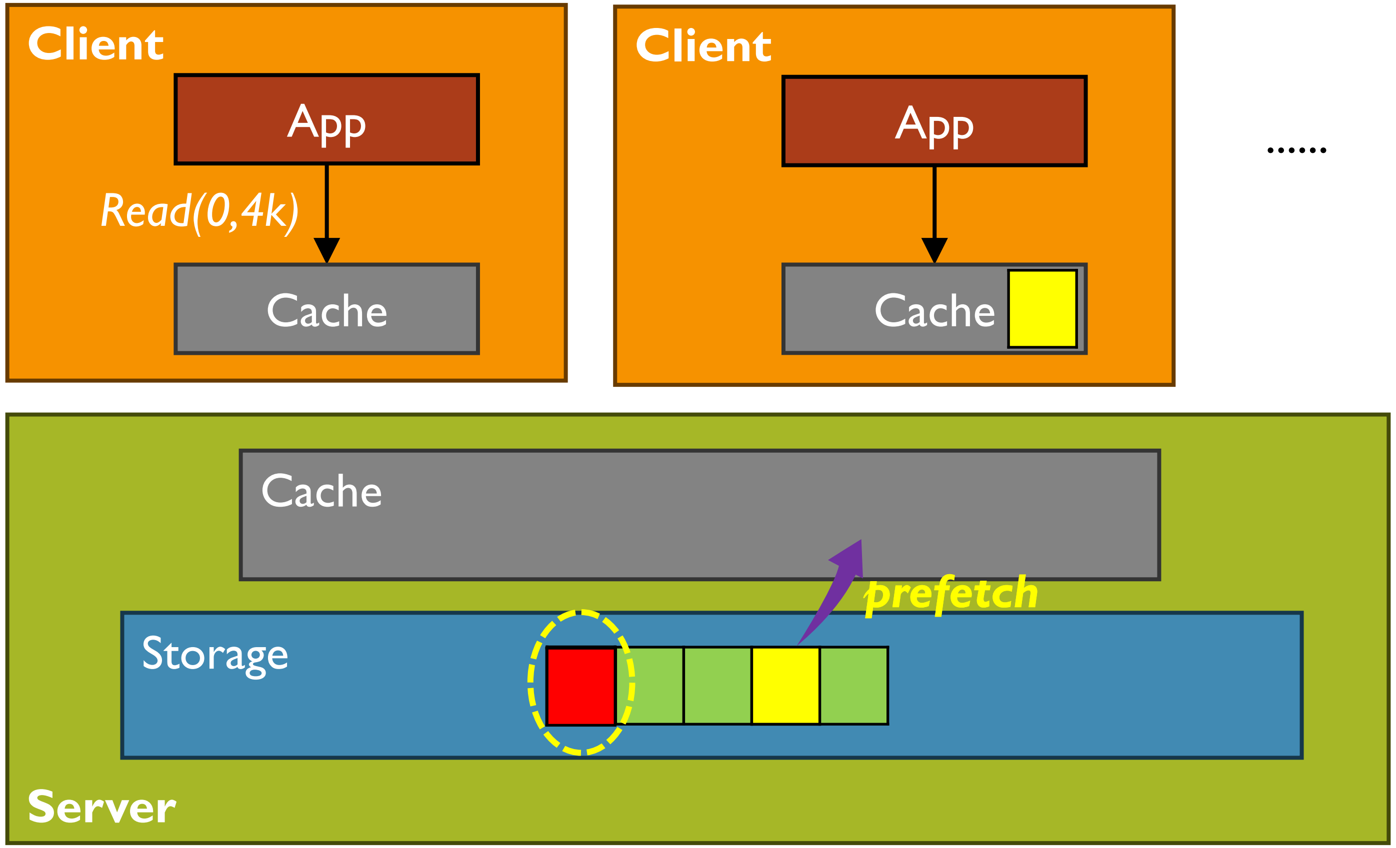
# Cache Duplication

| Workload | Rand read | Rand write | Seq read | Seq write |
|---|---|---|---|---|
| Duplication ratio | 22.73% | 22.56% | 21.45% | 23.73% |

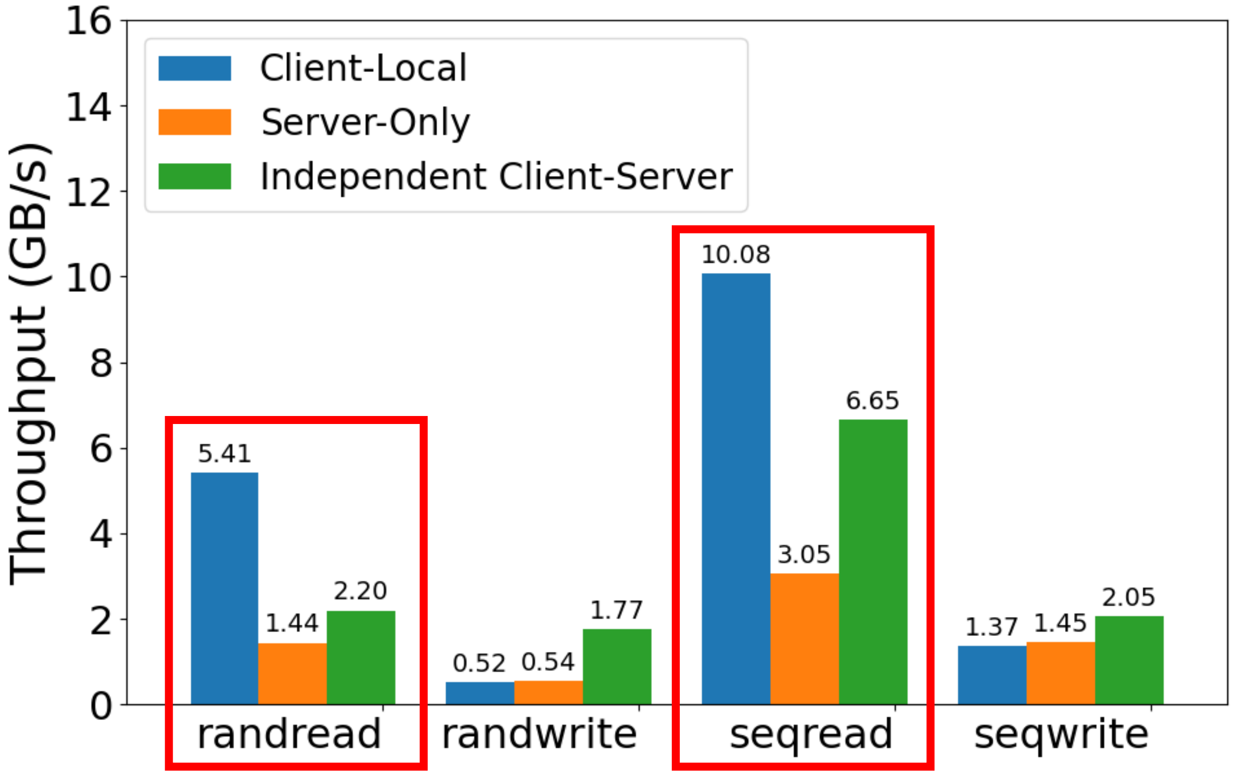
**Client-Server Independent** caching has **high cache duplication**

# Cache Duplication

| Workload | Rand read | Rand write | Seq read | Seq write |
|---|---|---|---|---|
| Duplication ratio | 22.73% | 22.56% | 21.45% | 23.73% |

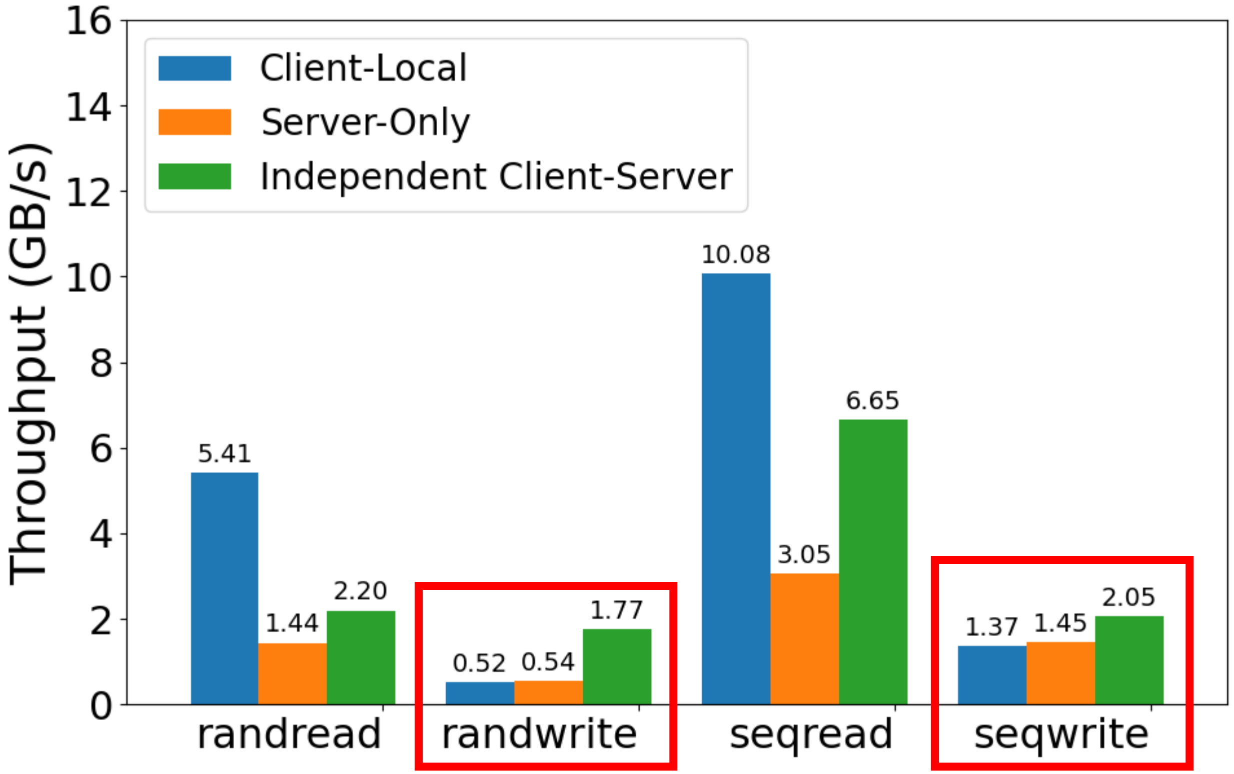


# Throughput Comparison




- **Independent client-server caching** performs worse under **read workload**
  - Cache duplication reduces effective cache space and degrades read performance

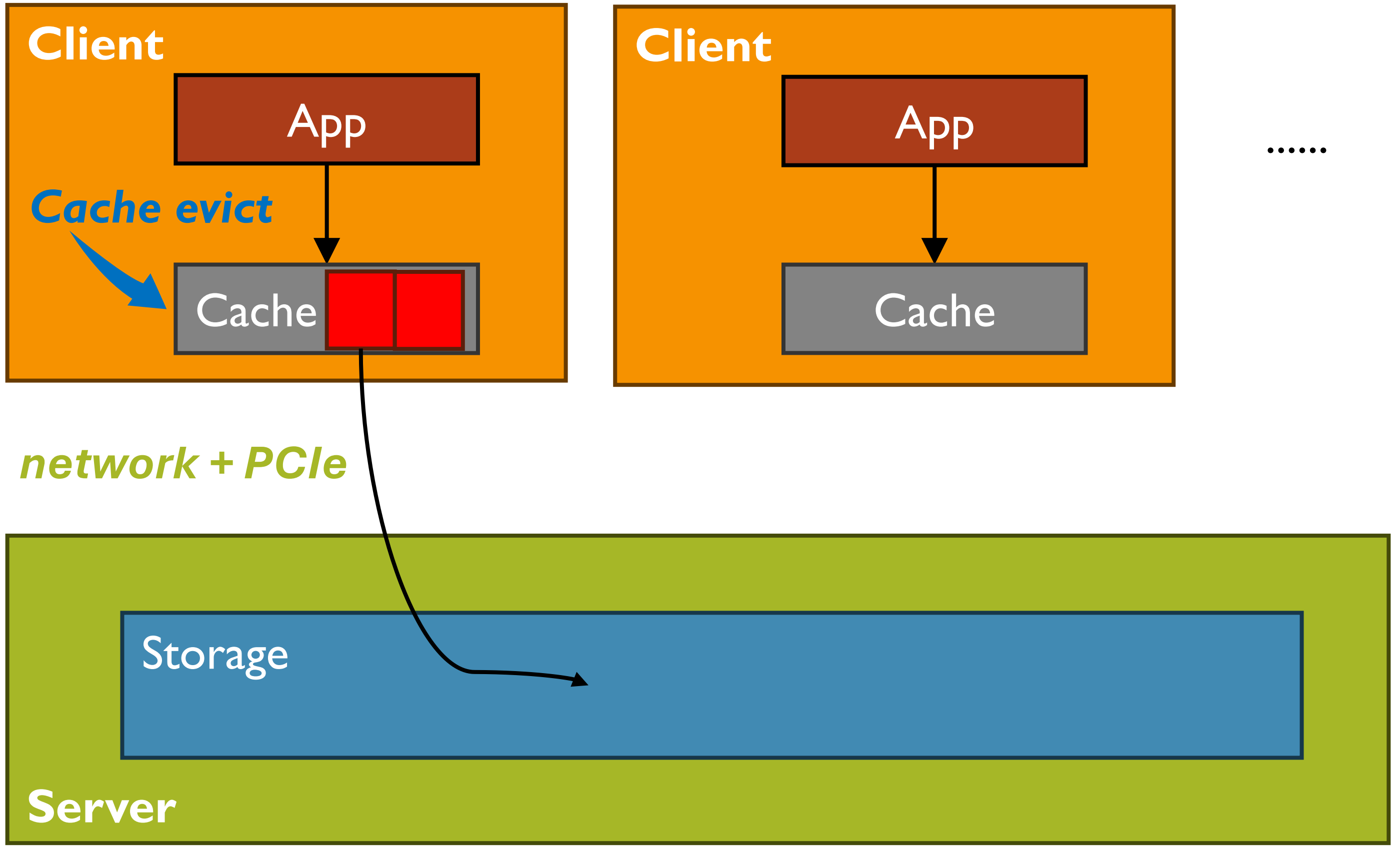
# Throughput Comparison




- **Client-local** and **Server-only** caching performs worse under **write workload**
  - Absence of a secondary cache layer worsens eviction-induced stalls

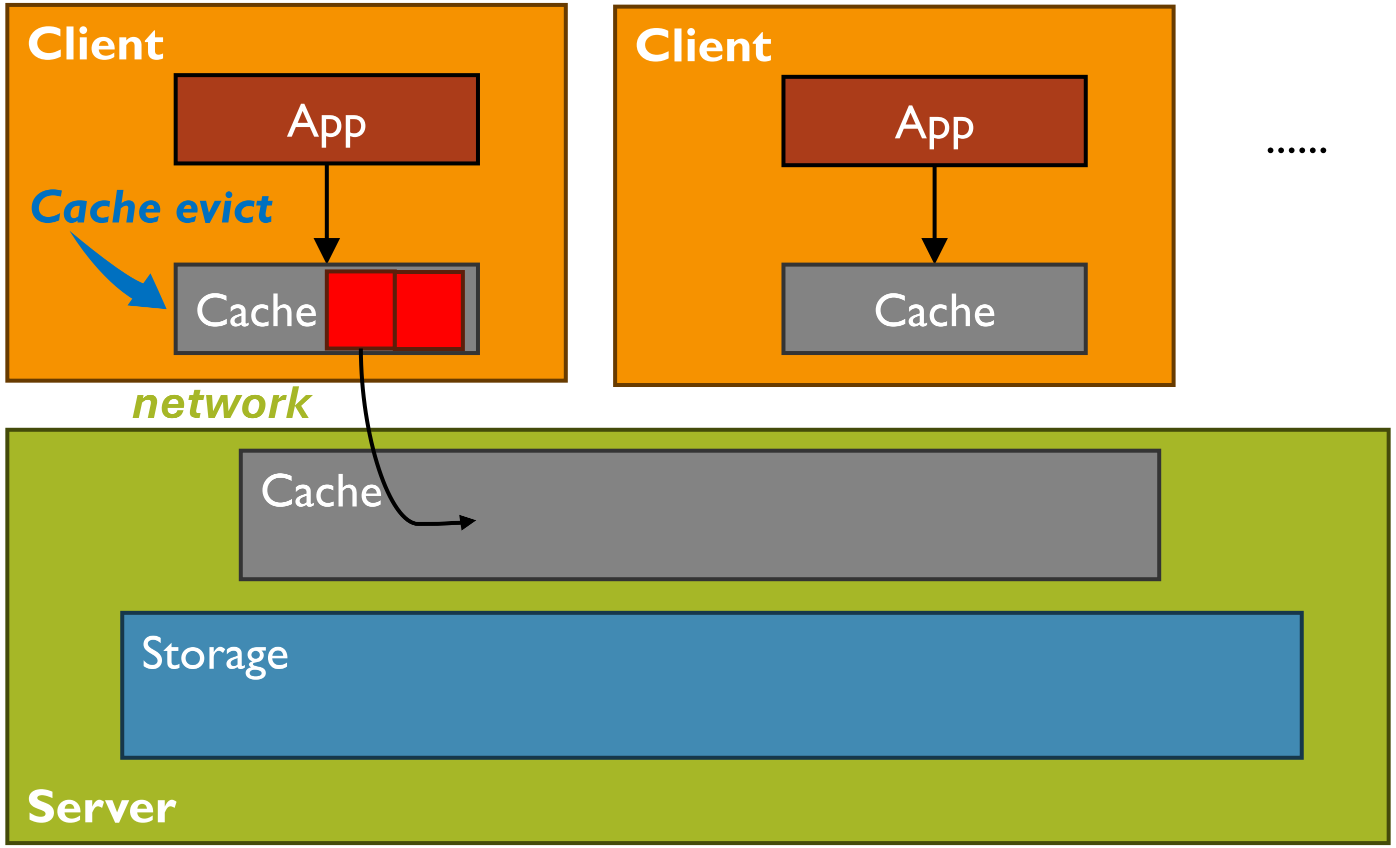
# Throughput Comparison

- **Client-local** and **Server-only** caching performs **worse under write workload**
  - Absence of a secondary cache layer worsens eviction-induced stalls



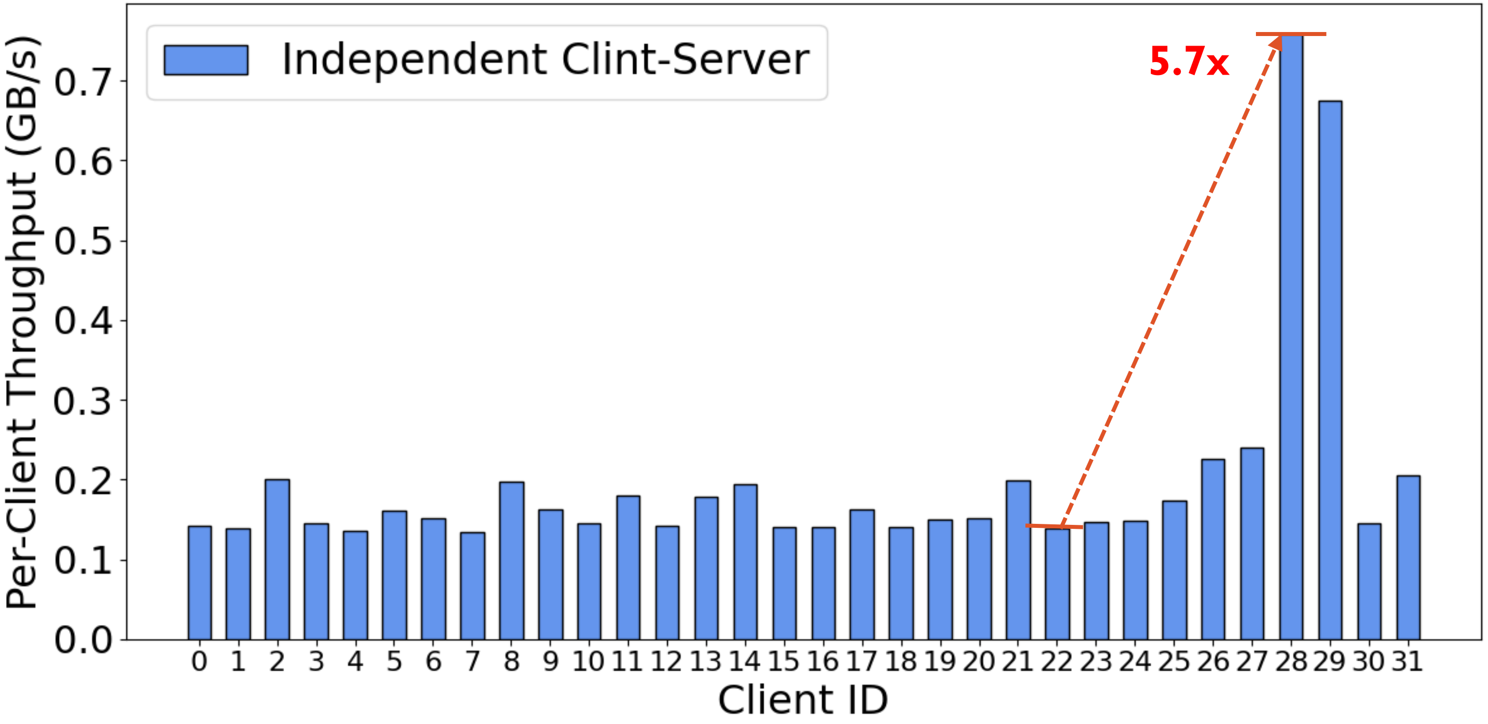
# Throughput Comparison

- **Client-local** and **Server-only** caching performs **worse under write workload**
  - Absence of a secondary cache layer worsens eviction-induced stalls



# Unfairness between Clients

- Performance of different threads **vary dramatically** using **Independent** caching
  - Skewed cache occupancy across different caches leads to unfairness



# Summary of Challenges

- Independent caching loses effective cache capacity due to invisibility
- Client local and Server only caching suffer from lack of a secondary cache layer
- All caching approaches lack cache fairness guarantee

# Outline

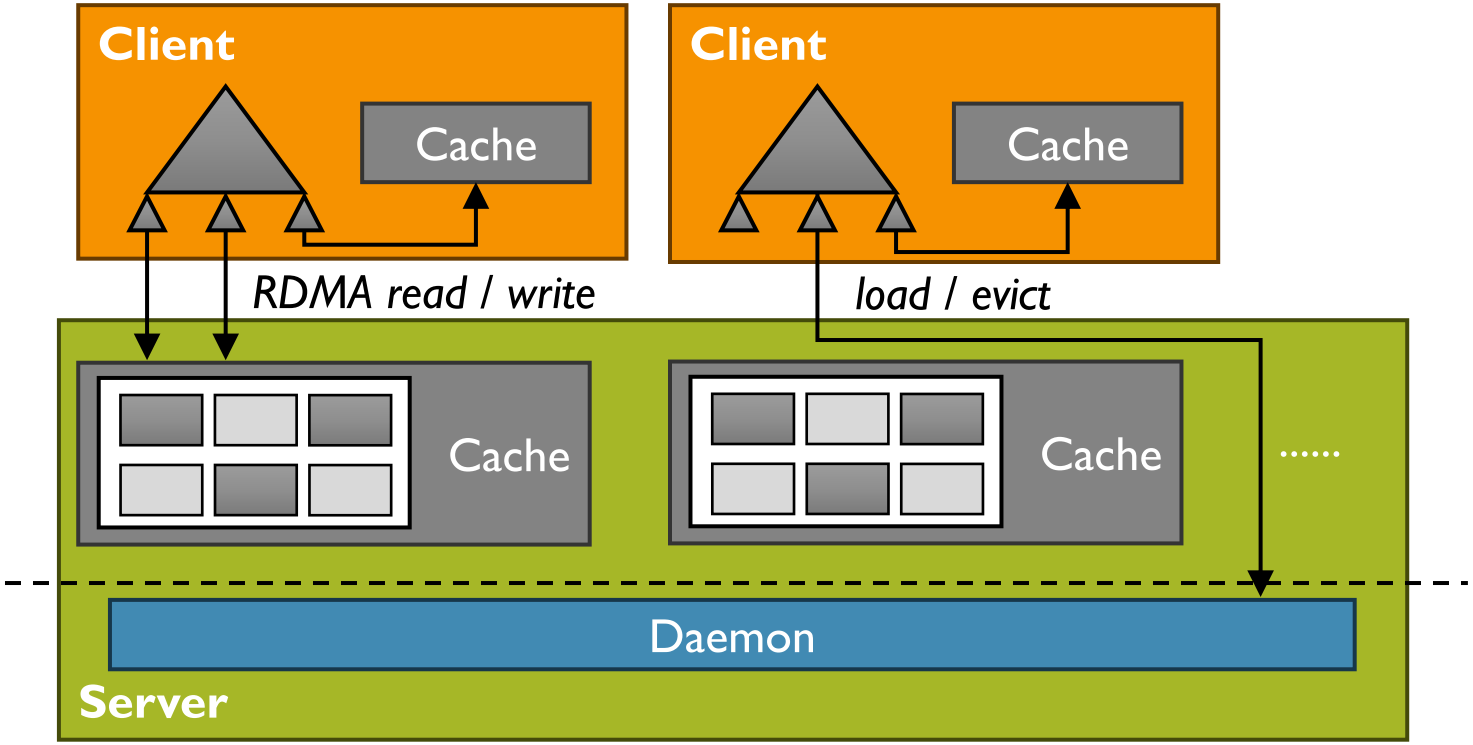
- Background
- Motivation
- **Solution**
- Future Work and Conclusion

# Proposed – Coordination Cache across Clients and Servers

- Expose visibility and control of server cache to clients through richer abstractions
- Split cache management workload across clients and servers
- Deploy different forms of cache indexes on the client and server side
- Manage server-side resource for cache fairness

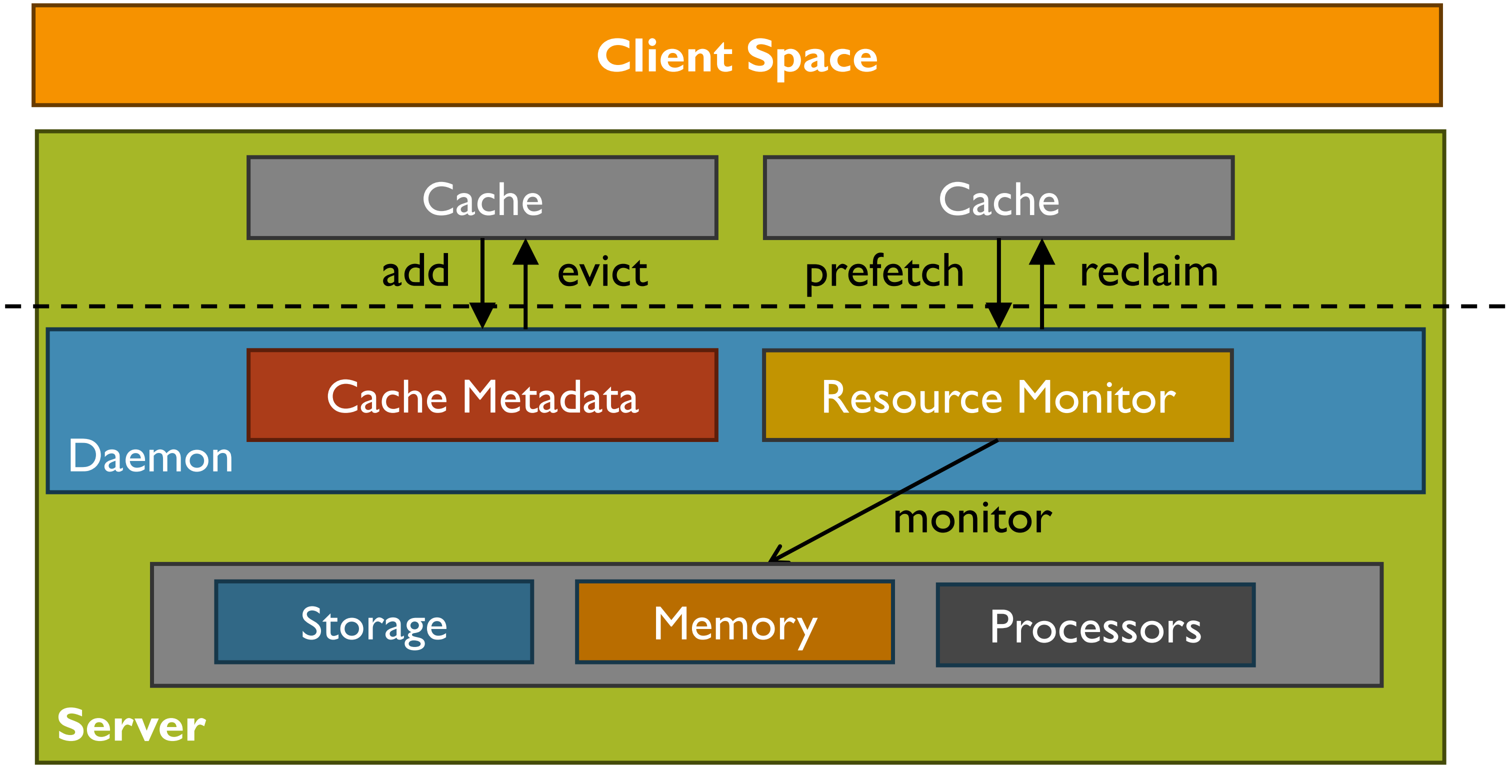
# Client-Side Cache Management

- Unified client-side index for client and server caches
- One-side RDMA operations for accessing server cache entities
- Message passing interfaces for cache control and management



# Server-Side Cache Management

- Lightweight cache metadata for tracking server-side cache
- Resource monitoring for cache fairness
- Utilizing clients' hints for adaptive cache policy



# Outline

- Background
- Motivation
- Solution
- **Future Work and Conclusion**

# Challenges for Cache Coordination

- Efficient abstractions for clients to provide hints and control server-side cache
- Coordinated client-server cache index for minimizing synchronization overhead
- Isolation and security in multi-tenant environments
- Consistency control across multiple clients' local caches

# Conclusion

- Analyze different caching approaches in disaggregated storage architecture
- Conduct a multi-dimensional evaluation
- Propose a preliminary solution for client-server cache coordination

**Linjie Ma (linjie.ma@rutgers.edu)**
**Rutgers University**