A Remote Dynamic Memory Cache

Qizhen Zhang, Phil Bernstein, Daniel Berger,
Badrish Chandramouli, Vincent Liu, Boon Thau Loo

March 28, 2024

Goal

Determine how best to use RDMA-accessible memory for database services

Hardware latency & bandwidth (your mileage may vary)

- Level 0 – DRAM 0.1 μs, 500 Gbps
- Level 1 – RDMA 1.0 μs, 100 Gbps
- Level 2 – SSD 100 μs, 48 Gbps

Why Use Remote Memory as a Cache?

- Database workload benefits from a larger cache
  - But VM’s physical server has no available memory
  - Workload changes permanently or periodically
- There is lots of unused memory in data centers
  - External fragmentation, due to bin packing of VMs on a server
  - Internal fragmentation, because VMs overprovision memory
- Faster datacenter networks → disaggregated memory is coming

Unallocated Memory in Azure

- Unallocated memory across clusters and time
  - Median – 46%
  - 10^{th} percentile – 37%
  - 1^{st} percentile – 28%

- Daily peak-to-trough is 2x

- Extreme case is *stranded memory*, on servers with no available cores

Stranded Memory

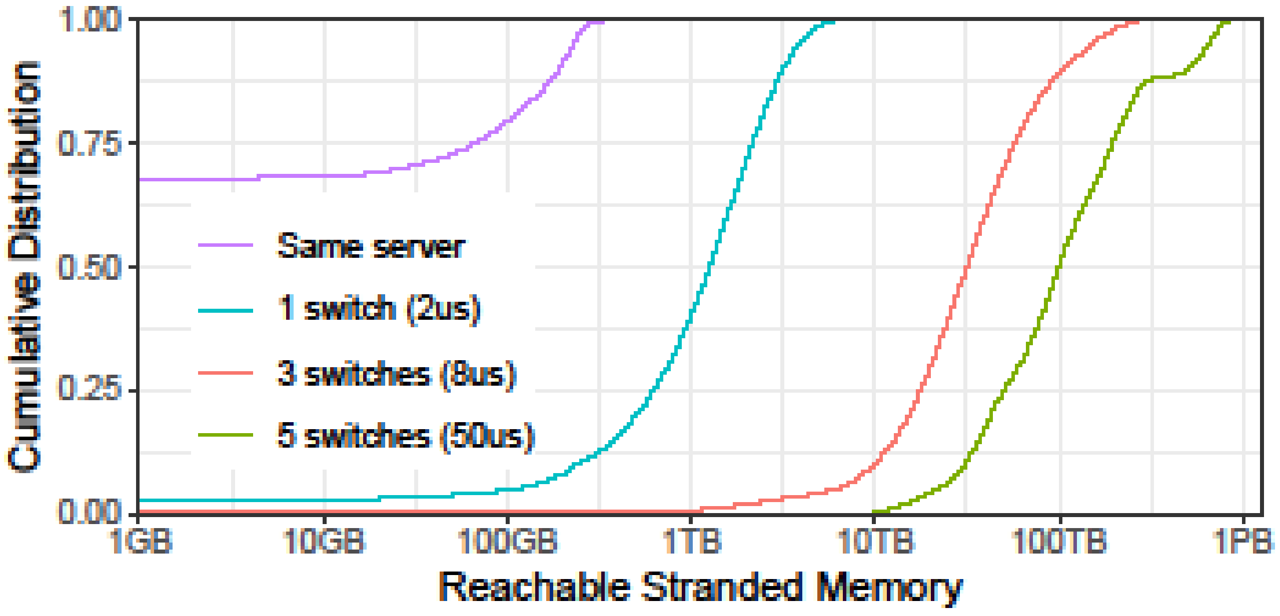
Fraction of memory stranded

- Median cluster, 8% stranded
- 10th percentile, ≥ 16% stranded
- 1st percentile, ≥ 23% stranded

Stranding duration

- 75th percentile – 22 minutes
- Median – 13 minutes
- 25th percentile – 6 minutes

Location



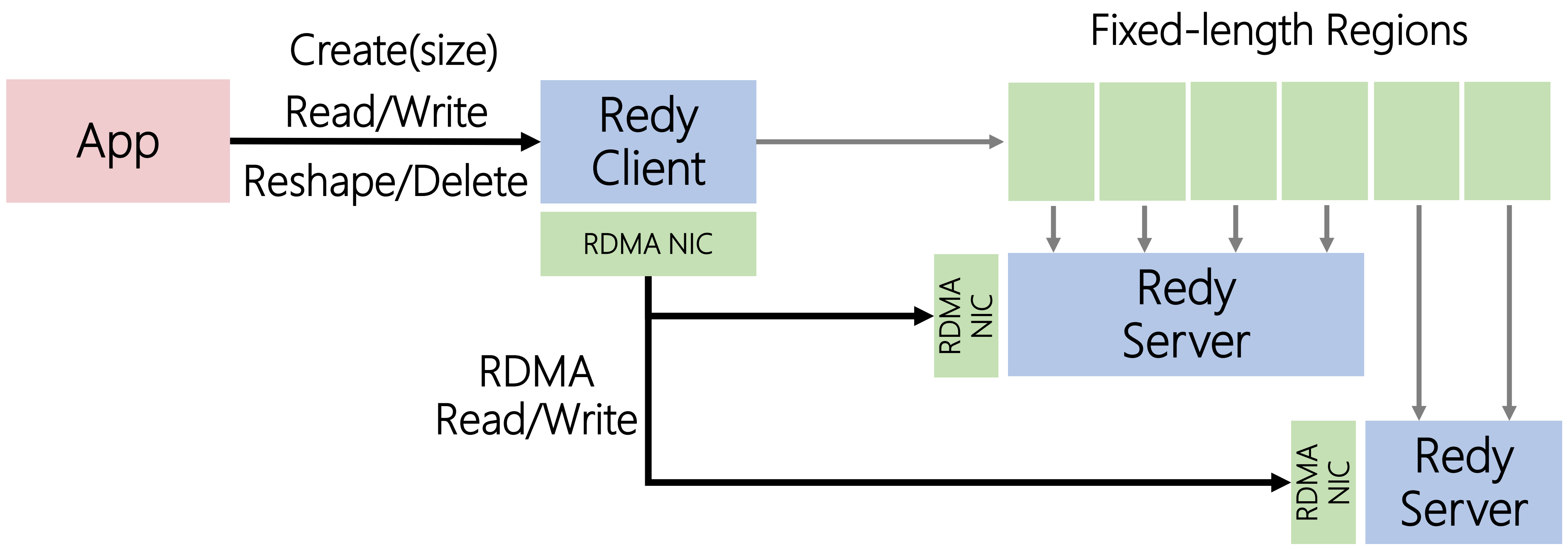
Outline

✓Motivation

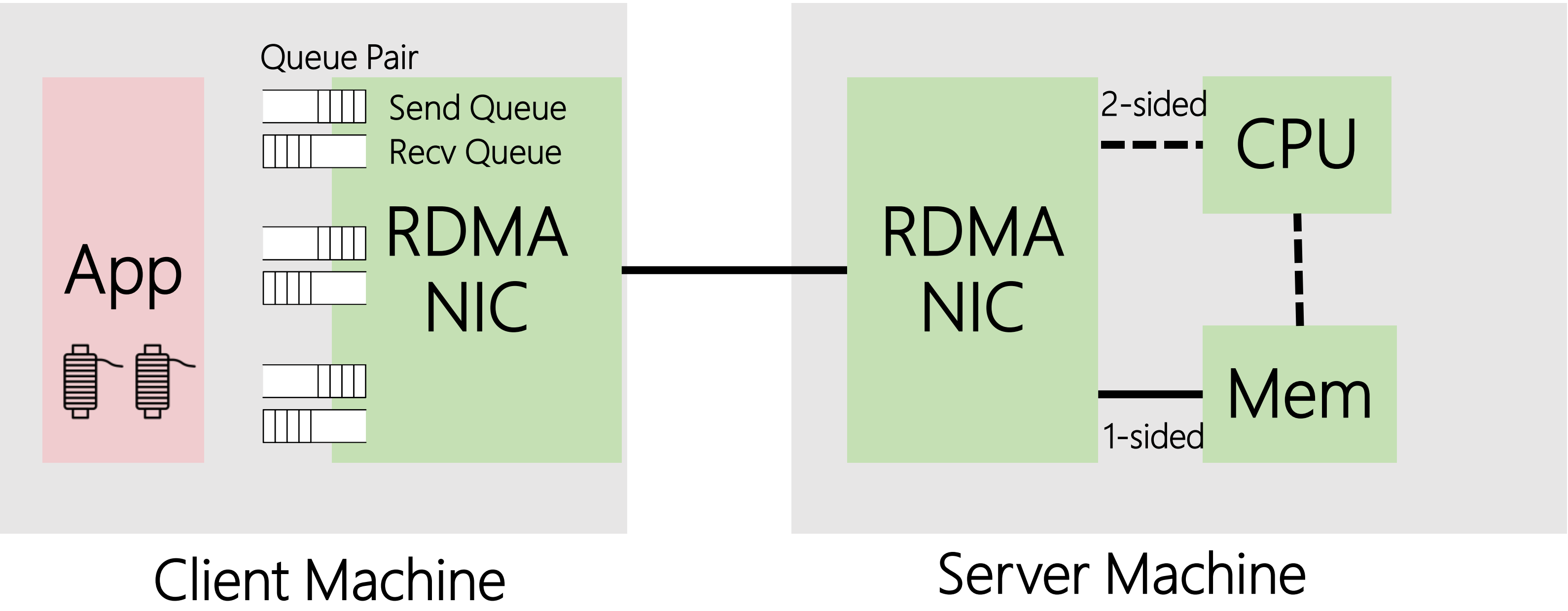
- Redy, a remote-cache manager
  - Optimizing for throughput vs. latency
  - Implementation
- Experiments
- Case study – FASTER key-value store
- CompuCache - Adding stored procedures

RDMA-accessible Cache

- Front End: an easy-to-use and general device abstraction
- Back End: efficient remote memory accesses via RDMA



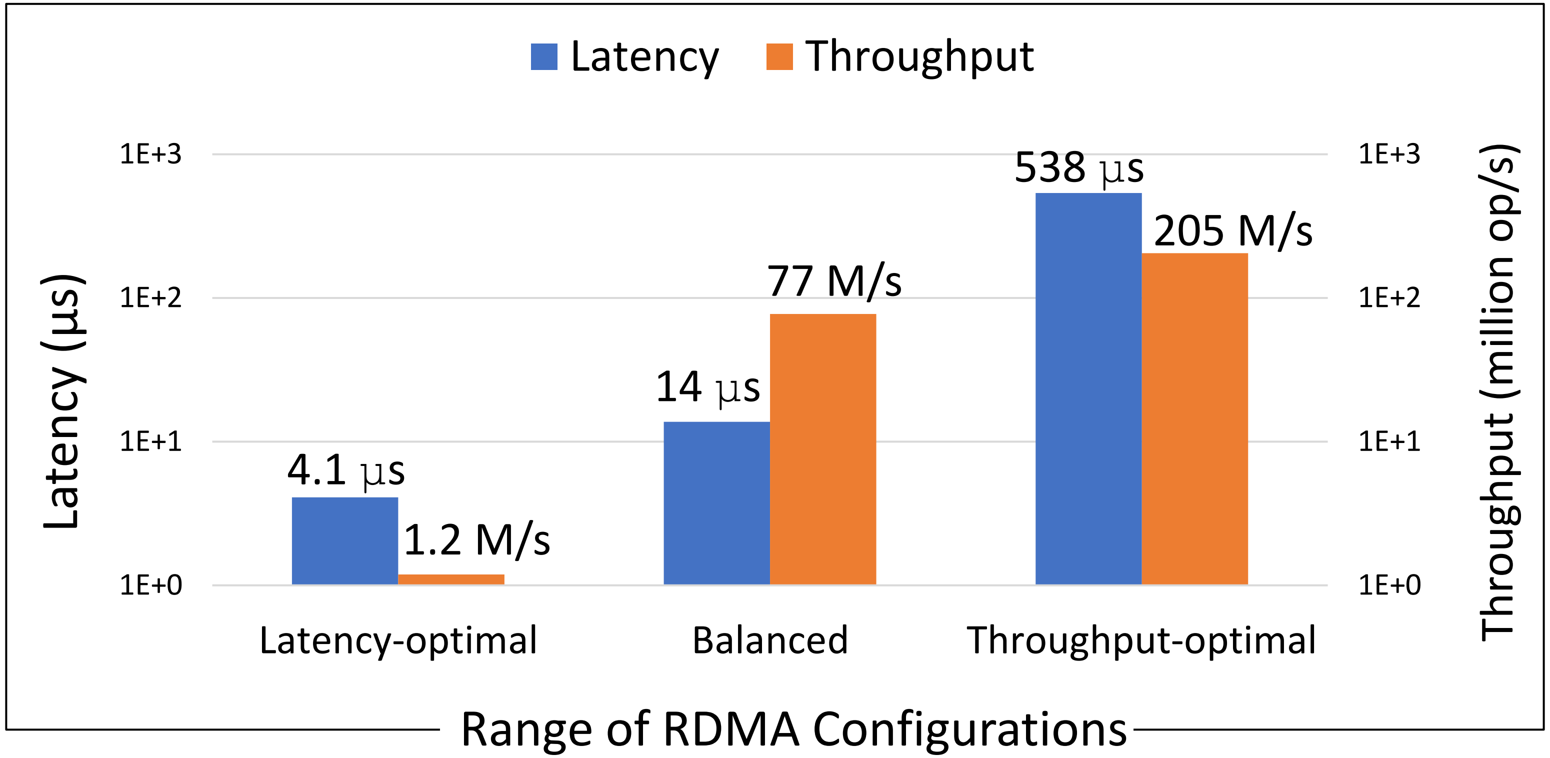
RDMA



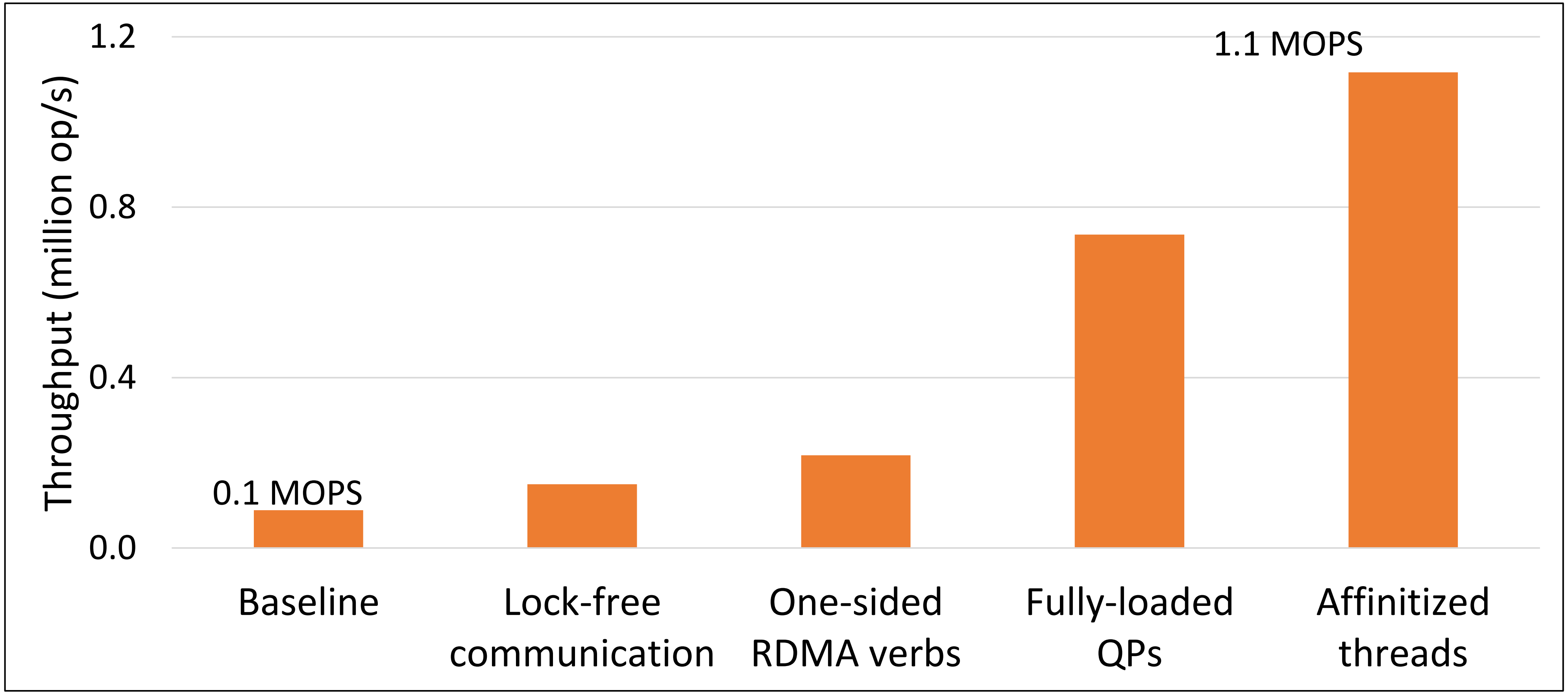
Queue Pair
Send Queue
Recv Queue
App
RDMA NIC
RDMA NIC
2-sided
CPU
Mem
1-sided
Client Machine
Server Machine

Research Challenge
Tuning RDMA performance

- Many RDMA tuning knobs for best latency or throughput
- Parallelization, asynchrony, batching, 1-sided/2-sided, …
- Optimal choice depends on record size, VM size, SLO, network configuration, …
- Experiment shows YCSB-like workload for 8-byte payloads

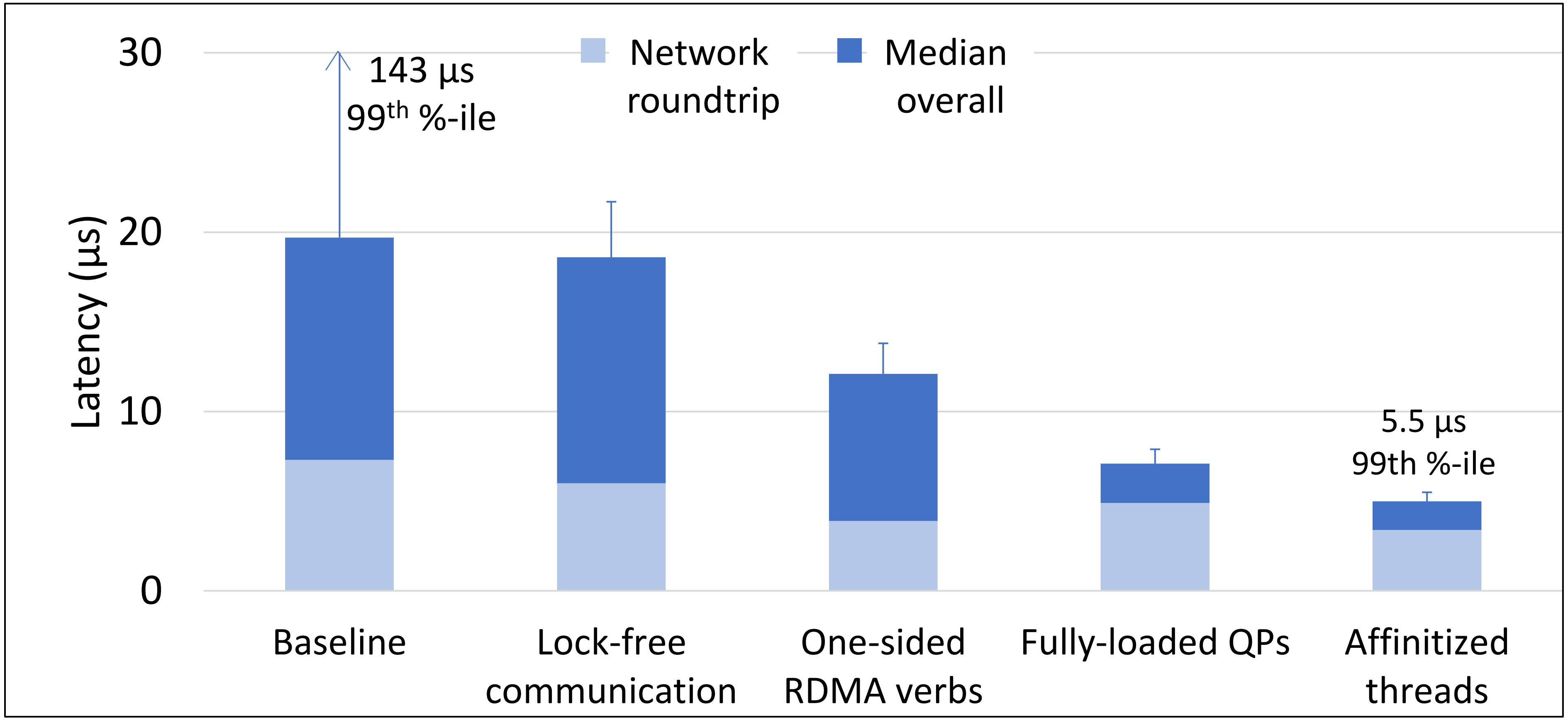


Throughput Benefit of Static Optimizations



- One client thread, one server thread, 8-byte read/write, no batching

Latency Benefit of Static Optimizations



- One client thread, one server thread, 8-byte read/write, no batching

Workload-dependent Optimization

- Primary determinants of Redy performance

Variable	Description	Lower Bound	Upper Bound
c	# client threads	1	Client cores
s	# server threads	0	c
b	# requests per batch	1	$\lceil$4KB / record-size $\rceil$
q	# in-flight operations	Static opt	NIC spec

- (Offline) For each network distance, message size, and [c, s, b, q] measure the latency and throughput.
- (At runtime) Find the cheapest [s, c, b, q] that satisfies the SLO.

Offline - Too Many Configurations to Measure

- Test parameter values that are powers of 2
  - [1,1,1,1], [1,2,1,1], [1,4,1,1], [1,8,1,1], [1,16,1,1], etc.
- *Throughput*(s,c,b,q) increases w.r.t. all parameters …
  - Until thread and connection contention cause it to drop
  - At that point, stop increasing that parameter
- Offline: 15 hours for [0:30, 1:30, 1:500, 1:16]
- Online: scan parameters until throughput is achieved
  - Then check latency.
  - Average search time is 27 ms.

VM Allocation

- Cache manager chooses VM with enough memory and cores
- Sometimes it’s better to allocate multiple VMs that together have enough memory, each with enough cores/memory.
- Can use spot instances
  - Requires cache migration on short notice
- Periodically check for cheaper VMs
  - If successful, allocate and migrate

Dynamic Memory Management

- Cache failure – allocate a new cache [and populate it from a checkpoint]
- Spot instance reclamation – migrate cache to a newly allocated cache
  - Use bandwidth-optimized connection from new to old
  - Use one-sided reads to migrate the content
- Optimizations
  - Allow application reads of old cache during migration
  - Migrate region-by-region and stop writes only to the region being migrated

Implementation

- 13,700 lines of C++
- CLR wrapper for access by .NET applications
- Uses NDSPI to access RDMA on Windows.

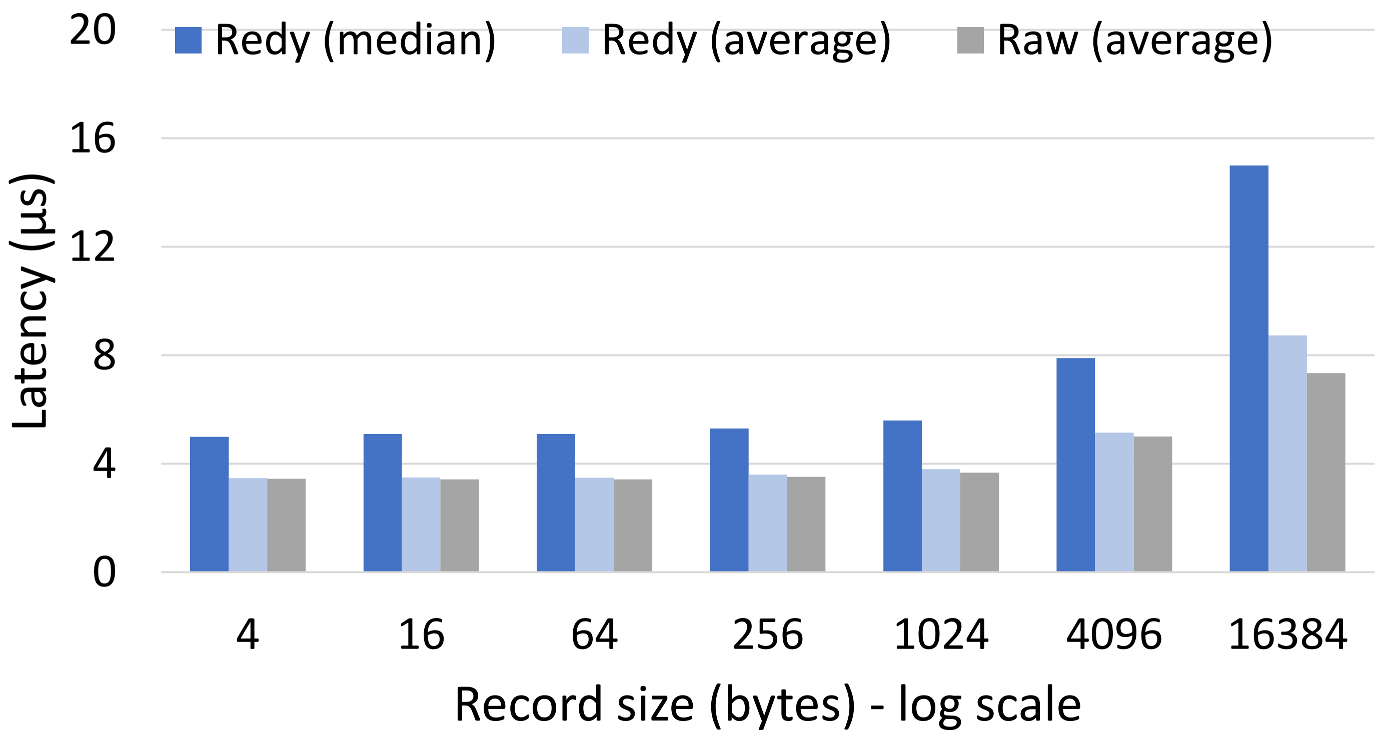
Outline

- ✓ Motivation
- ✓ Redy, a remote-cache manager
  - Optimizing for throughput vs. latency
  - Implementation
- Experiments
- Case study – FASTER key-value store
- CompuCache - Adding stored procedures

Experiments

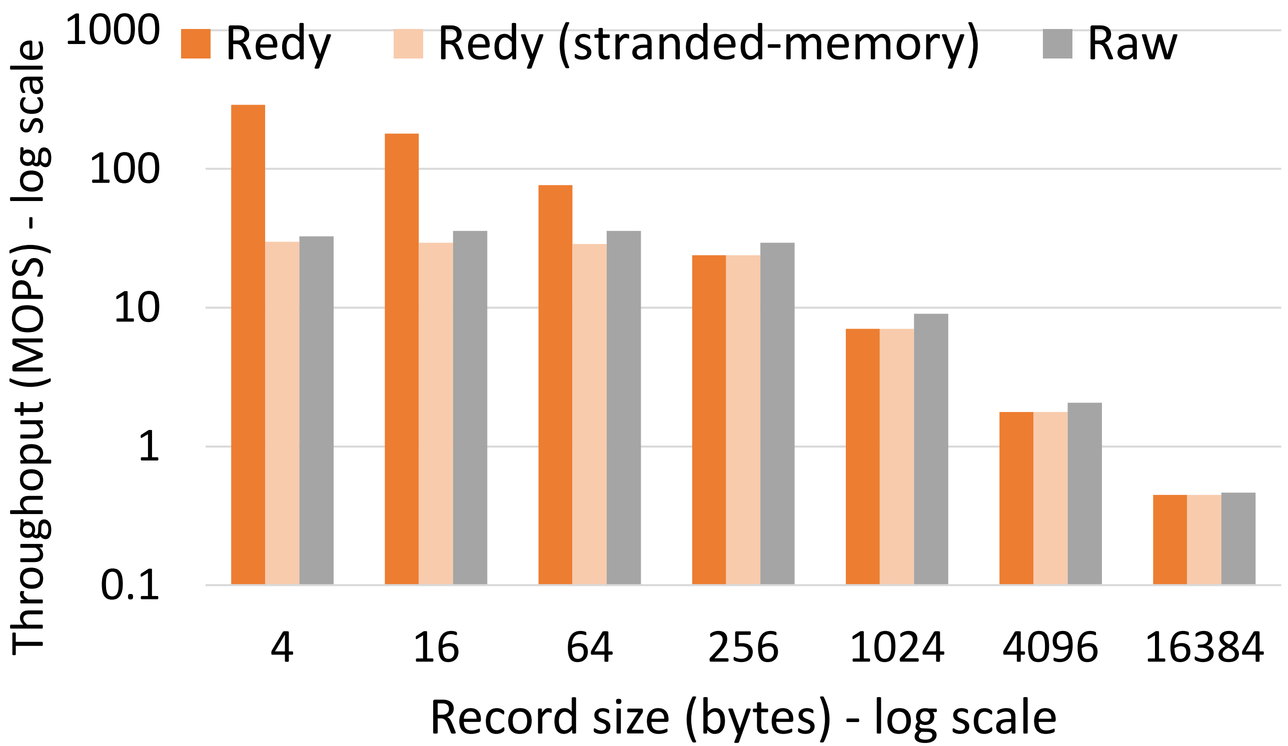
- Azure HPC Standard_HB60rs VMs.
- Each VM has 60 vCPUs, with 2.0 GHz AMD EPYC 7551
  - 228 GB of memory
  - 700 GB Azure premium SSD
- Mellanox ConnectX-5 NIC, supporting 100 Gb/s EDR Infiniband
- Windows 2019 Datacenter

Latency-Optimized Configuration (Reads)



- Reads are close to raw network speed of 3-4μs
- Small writes are a bit faster by in-lining them in the request (not shown)

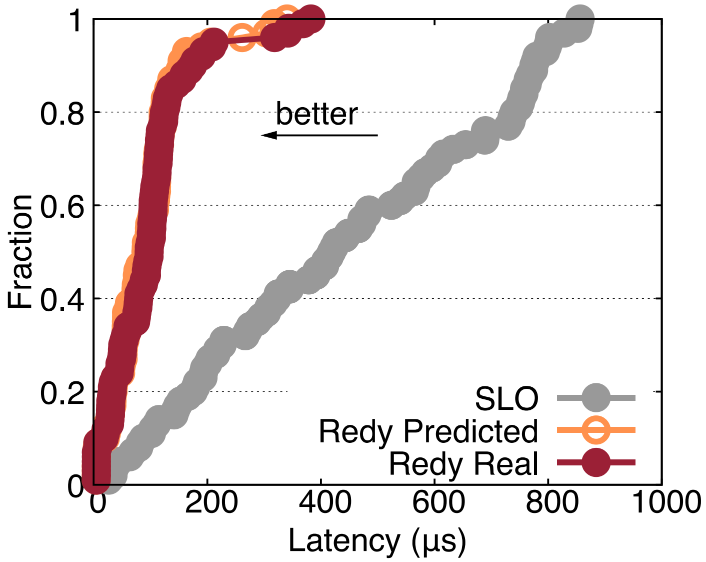
Throughput-Optimized Configuration (Reads)



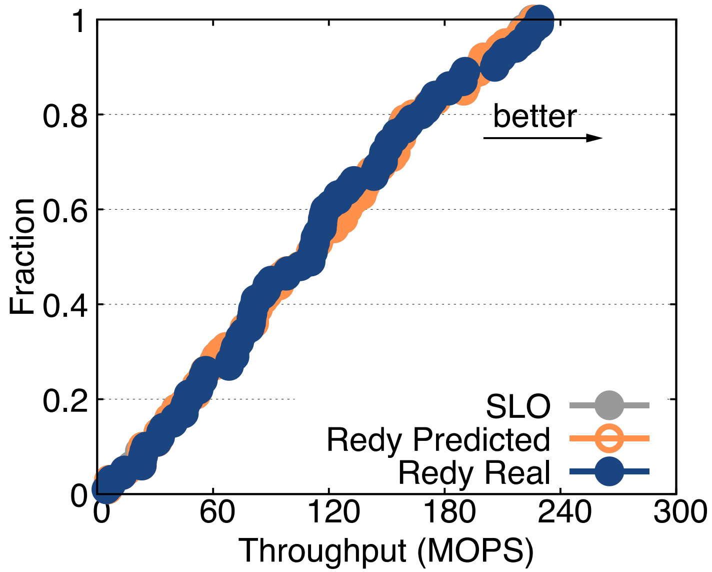
- For 16-byte records, throughput is 10x raw.
- Batching benefit stops at 256-bytes, the RDMA minimum packet size.
- Throughput of writes is similar.

Accuracy of Throughput SLOs

- Use interpolation to map SLO to a configuration
- We randomly choose 100 SLO’s, allocate a cache with the recommended configuration, and measure its performance

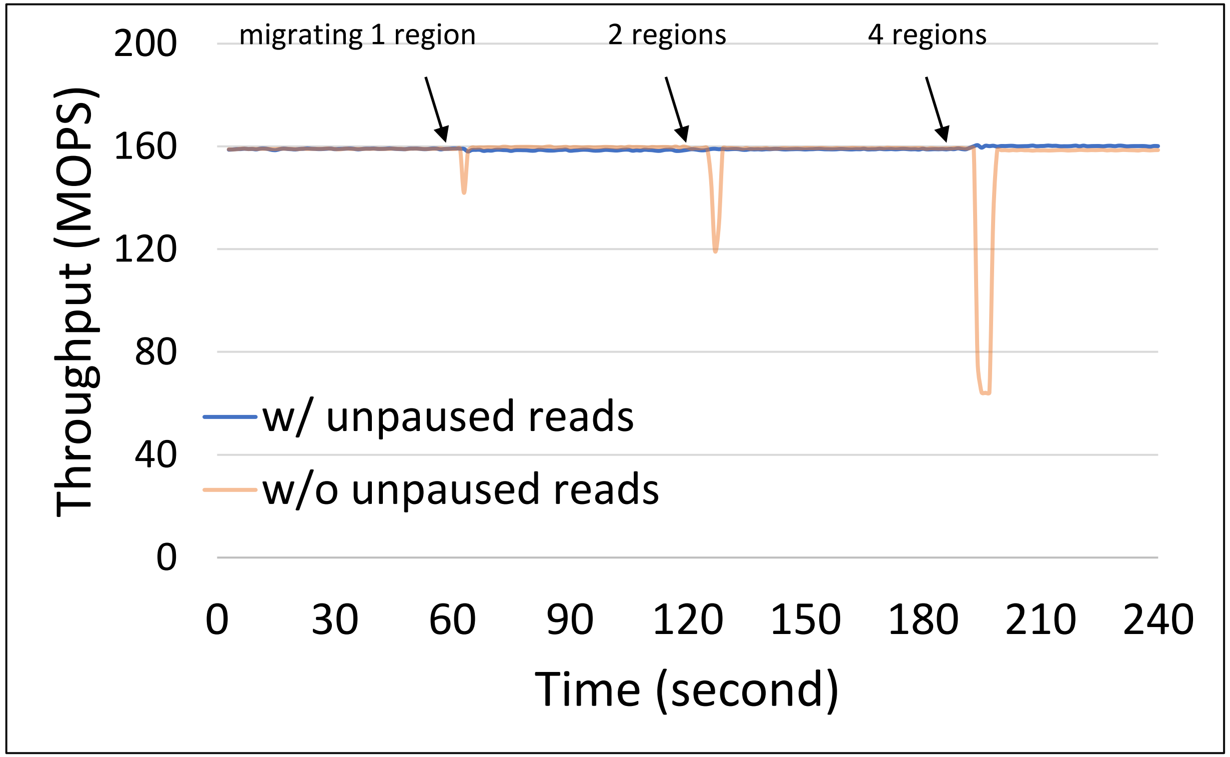


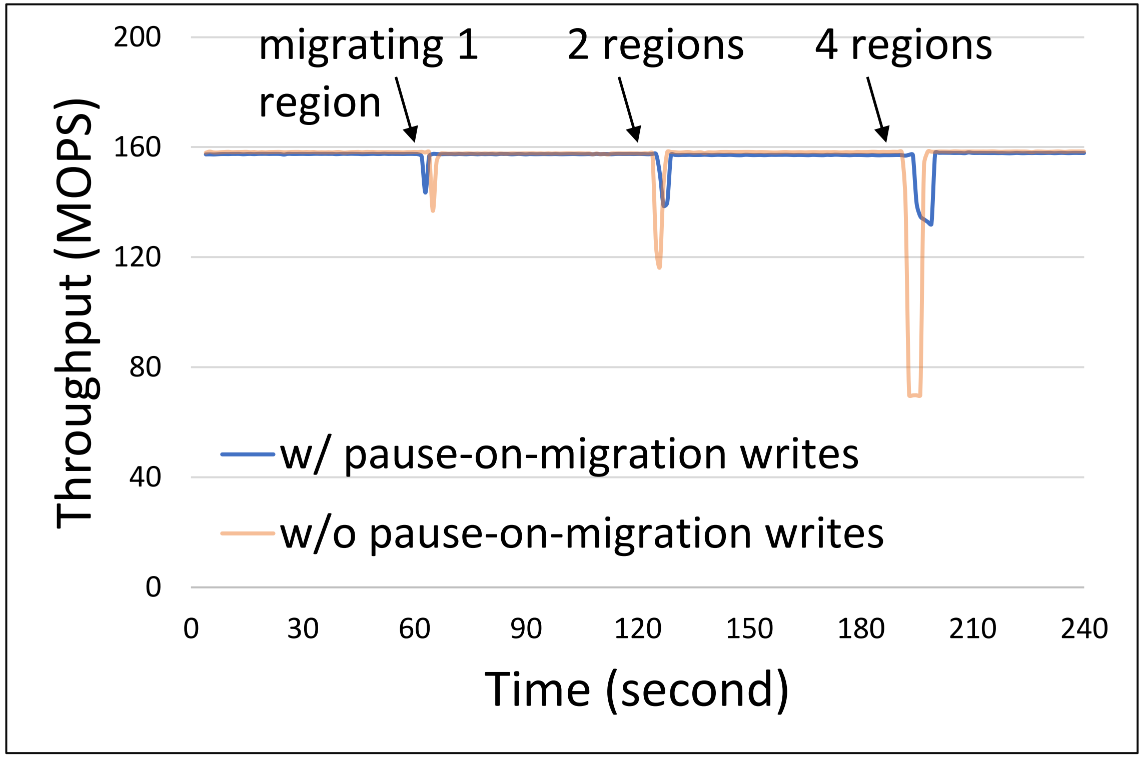
- Measured latency is always much better than the SLO and close to predicted (not shown)



- Real throughput is always better than SLO

Optimizing Region Migration





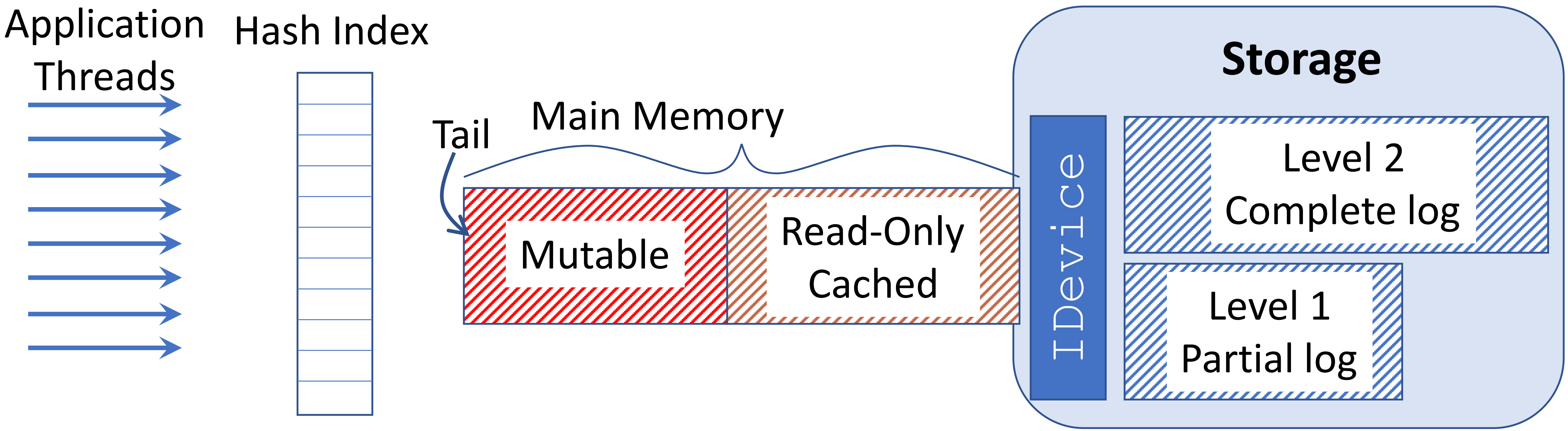
- Successively migrate 1GB regions

Outline

- ✓ Motivation
- ✓ Redy, a remote-cache manager
  - Optimizing for throughput vs. latency
  - Implementation
- ✓ Experiments
- Case study – FASTER key-value store
- CompuCache - Adding stored procedures

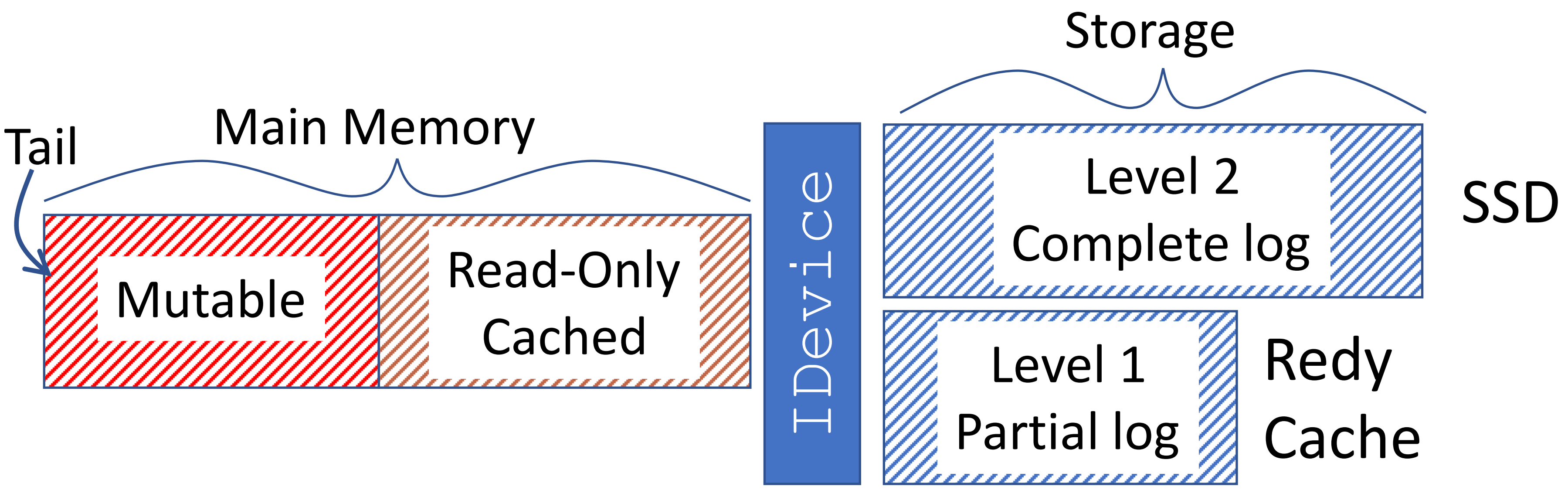
FASTER's Multi-tier Storage

- The top tier stores the entire database and is the slowest
- Lower tier replicates a tail of next higher tier and is faster
- FASTER services cache misses from the lowest tier that has the data
  - Uses address calculation to pick the correct tier

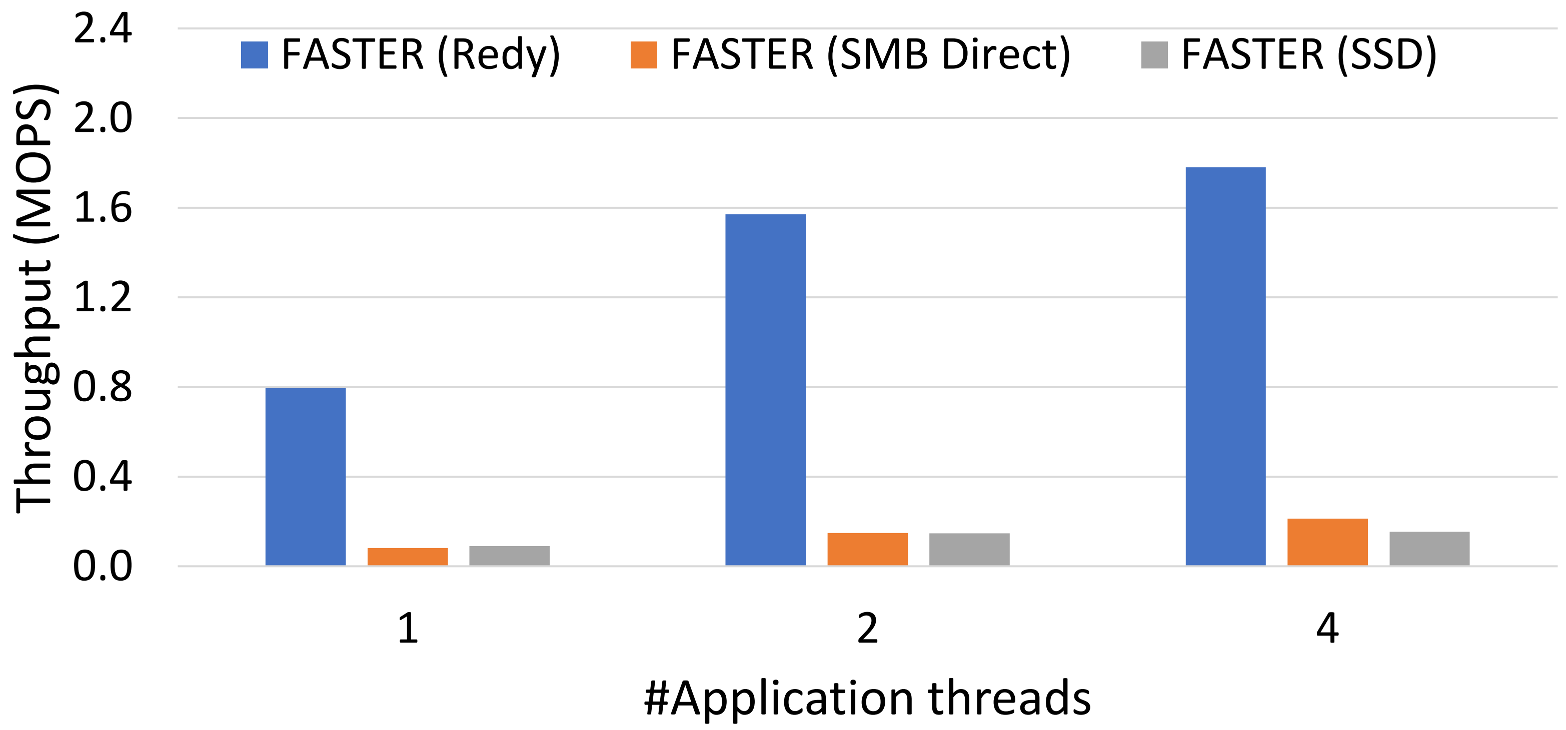


FASTER's Multi-tier Storage (cont'd)

- Our target
  - Level 0 is client cache (where FASTER executes)
  - Level 1 is a Redy cache
  - Level 2 is client-attached SSD



FASTER vs. SSD and SMB Direct



- FASTER local memory is 1GB
- 24-byte records
- Random reads from ~6GB database
- All devices can hold the complete log
- Redy uses batching

Outline

- ✓ Motivation
- ✓ Redy, a remote-cache manager
  - Optimizing for throughput vs. latency
  - Implementation
- ✓ Experiments
- ✓ Case study – FASTER key-value store
- CompuCache - Adding stored procedures

CompuCache

- Extend Redy with stored procedures and server-side pointer chasing

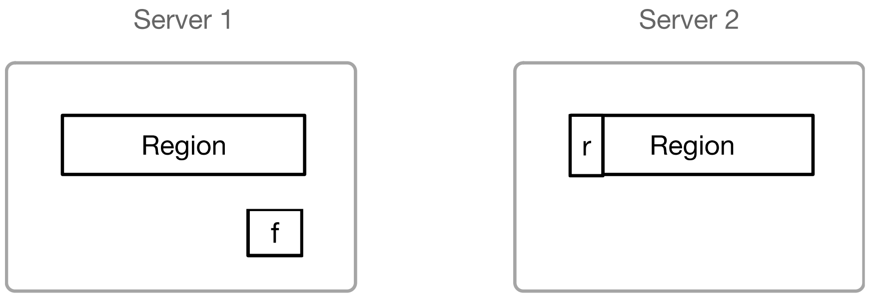
- Server-side pointer chasing
  - Apps know cache addresses, but chasing pointers requires physical addresses
  - Solution: LocalTranslator

`l_addr, l_size ← Translate(c_addr, c_size)`

physical address   cache address

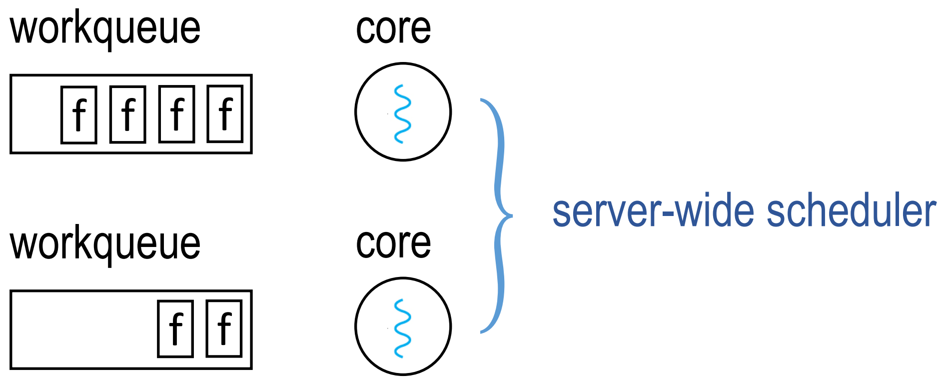
CompuCache Out-of-Bounds Exceptions

- A sproc may span VM boundaries
  - Data Shipping - Flow the input data with Dflow: `l_addr ← DFlow(c_addr, c_size)`
  - Function Shipping - Flow the sproc function with Fflow: `FFlow(c_addr, c_size)`
  - Stop and return



CompuCache Execution

- Transport is eRPC over DPDK or RDMA
  - Batches all operation requests
  - Dynamically chooses batch size of responses
- Server uses one thread per core
  - Polls for requests and executes them on the same thread
  - Server-wide scheduler load-balances by moving requests between threads



CompuCache Execution (cont'd)

- On DFlow, the request becomes inactive and resumes after data transfer
- FFlow dispatches request to another thread

Server Migration

- Client maintains mapping for LocalTranslator
- When a server VM is reclaimed, client allocates new VM(s)
- For each region
  - Client pauses reads and writes for the region
  - Migrates the region
  - Updates LocalTranslator mapping
  - Broadcasts mapping to servers
  - Resumes reads and writes for the region
  - Server forwards future stale DFlow and FFlow requests to the new region
- After server is migrated, it sends remaining request to new VMs

Conclusion

- A remote cache is a must for data management
  - It uses memory that currently goes to waste
  - With RDMA, it offers big performance gains
  - Use stored procedures for pointer-chasing
- Remaining work
  - Integrate with Azure's VM allocator

Redy https://arxiv.org/ftp/arxiv/papers/2112/2112.12946.pdf

CompuCache https://www.cidrdb.org/cidr2022/papers/p31-zhang.pdf