Paper ID: 273

Consistent RDMA-Friendly Hashing on Remote Persistent Memory

Xinxin Liu, Yu Hua, Rong Bai

Huazhong University of Science and Technology

ICCD 2021

Background

- **Persistent Memory (PM)**
  - ✓ Non-volatility
  - ✓ Byte-addressability
  - ✓ Large capacity
  - ✓ DRAM-scale latency
- **Remote Direct Memory Access (RDMA)**
  - ✓ Bypassing kernel
  - ✓ Zero memory copy
  - ✓ High bandwidth/Low latency
  - ✓ Well-known for one-sided RDMA (Do not involve remote CPU)

Background

- **Persistent Memory (PM)**
  - ✓ Non-volatility
  - ✓ Byte-addressability
  - ✓ Large capacity
  - ✓ DRAM-scale latency
- **Remote Direct Memory Access (RDMA)**
  - ✓ Bypassing kernel
  - ✓ Zero memory copy
  - ✓ High bandwidth/Low latency
  - ✓ Well-known for one-sided RDMA (Do not involve remote CPU)

RDMA+PM: Deliver high end-to-end performance for networked storage systems.

- Require rethinking the design of efficient hash structures.

Challenges

Designing hashing indexes for RDMA+PM:

- RDMA Access Amplification:
  - ✓ Accessing non-contiguous remote memory region requires multiple one-sided RDMA round-trips.
- High-Overhead PM Consistency:
  - ✓ Undo/redo logging and copy-on-write require double PM writes, consuming the limited endurance of PM.

Existing Solutions

Existing hashing schemes separately optimize RDMA or PM:

- RDMA-friendly hashing schemes:
  - ✓ Pros: address the problem of RDMA Access Amplification.
  - ✓ Cons: fail to mitigate High-Overhead PM Consistency.

- PM-friendly hashing schemes:
  - ✓ Pros: guarantee crash consistency and optimize PM writes.
  - ✓ Cons: cause RDMA Access Amplification due to indirect layers or non-contiguous standby positions.

Existing Solutions

Existing hashing schemes separately optimize RDMA or PM:

- RDMA-friendly hashing schemes:
  - ✓ Pros: address the problem of RDMA Access Amplification.
  - ✓ Cons: fail to mitigate High-Overhead PM Consistency.

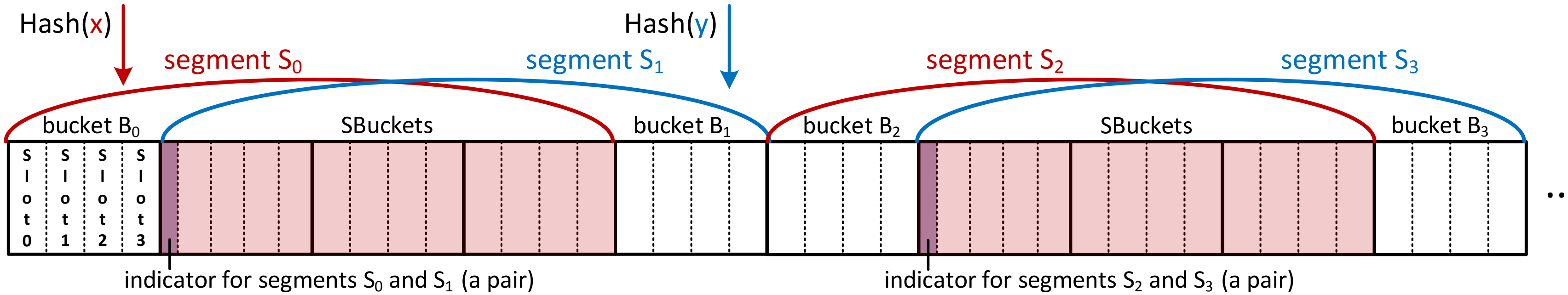
- PM-friendly hashing schemes:
  - ✓ Pros: guarantee crash consistency and optimize PM writes.
  - ✓ Cons: cause RDMA Access Amplification due to indirect layers or non-contiguous standby positions.

Our Continuity Hashing:

- A "one-stone-two-birds" design to optimize both RDMA and PM.

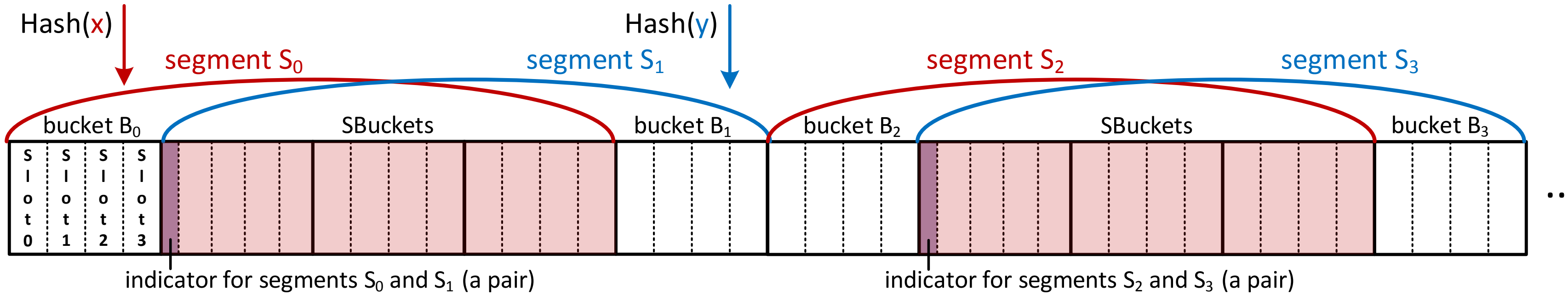
System Design of Continuity Hashing

- Index Structure



System Design of Continuity Hashing

- Index Structure



- Read/Write Operations using RDMA
  - ✓ Reads: use one-sided RDMA read
  - If the bucket number is even, the offset to be read (e.g., S0) is:

$$Ofs = hash(k)\%N/2 * (size_{se} + size_{bu})$$

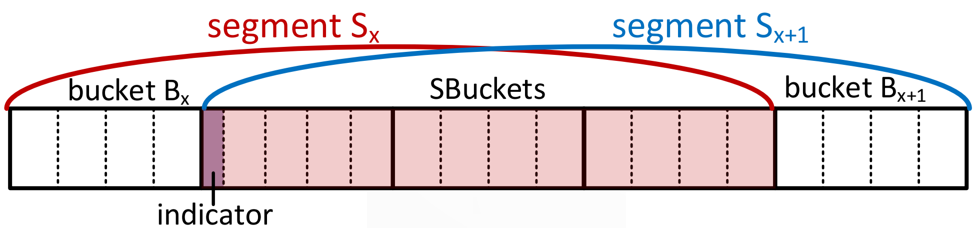
  - If the bucket number is odd, the offset to be read (e.g., S1) is:

$$Ofs = (hash(k)\%N - 1)/2 * (size_{se} + size_{bu}) + size_{bu}$$

  - ✓ Writes:
  - Use *RDMA write_with_imm* operation
  - Servers handle writes

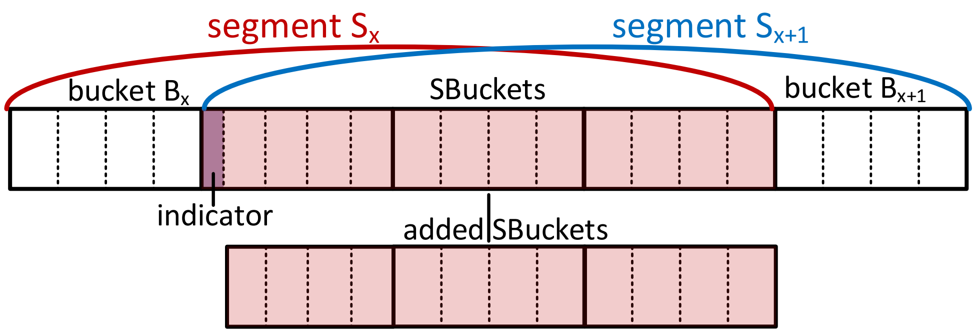
System Design of Continuity Hashing

- Log-Free Failure-Atomicity Guarantee
  - ✓ An indicator:
    - Indicate whether each slot in the segment pair contains valid data.
    - Can be updated in the atomic-write manner.
    - Support atomic insertion/deletion/update & log-free resizing.



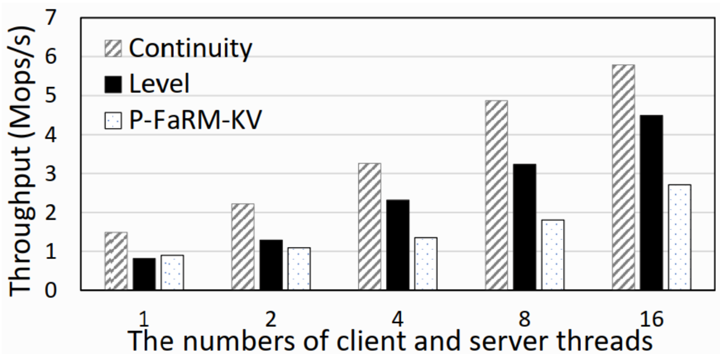
System Design of Continuity Hashing

- Log-Free Failure-Atomicity Guarantee
  - ✓ An indicator:
    - Indicate whether each slot in the segment pair contains valid data.
    - Can be updated in the atomic-write manner.
    - Support atomic insertion/deletion/update & log-free resizing.

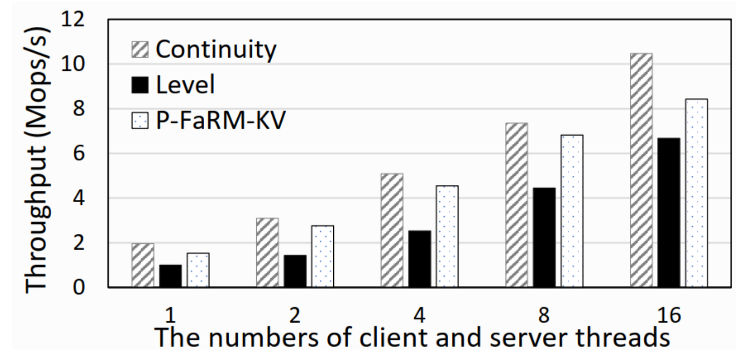


- Optimizing Space Utilization
  - ✓ Dynamically increase the number of SBuckets for 1/10 segment pairs before resizing.
  - ✓ Still support log-free consistency for all the PM writes.
    - The added SBuckets use the same indicator as the original buckets, which can be updated with an atomic write.

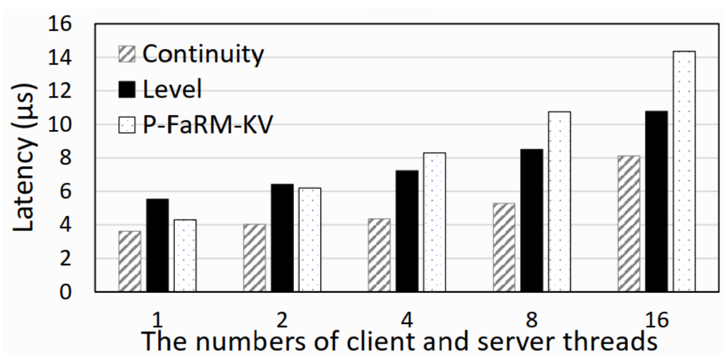
Evaluation

- 

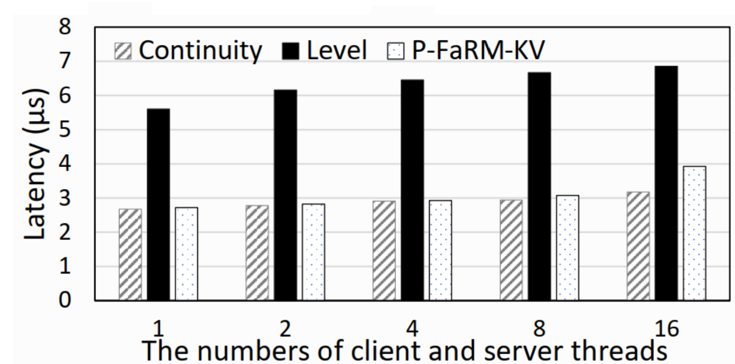
The throughput of the update-heavy workload.



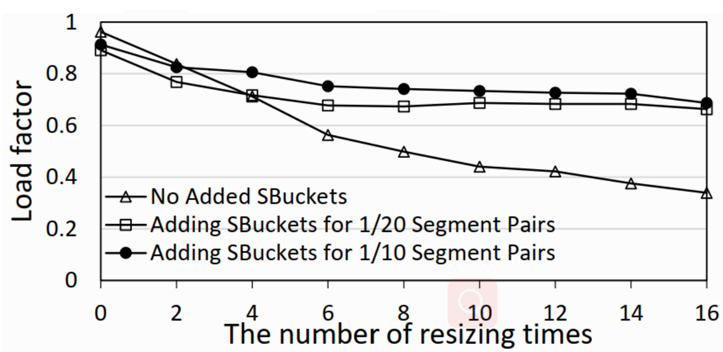
The throughput of the read-only workload.

- 

The latency of the update-heavy workload.



The latency of the read-only workload.

- 

	Insertion	Update	Deletion
Continuity	2	2	1
Level	2 – 2.01	2 – 5	1
P-FaRM-KV	5	5	5

The number of PM writes.

Conclusion

- **Challenges of designing hashing indexes for RDMA+PM:**
  - ✓ RDMA Access Amplification
  - ✓ High-Overhead PM Consistency

- **Our Continuity Hashing:**
  - ✓ Coalescing design for RDMA and PM.
  - ✓ Efficient remote read without access amplification.
  - ✓ Log-free consistency guarantee for all the PM writes.

- Compared with state-of-the-art schemes, **continuity hashing** achieves **high throughput** (1.45X – 2.43X), **low latency** (about 1.7X speedup) and **the smallest number of PM writes**, while obtaining **acceptable load factors**.

Thanks! Q&A