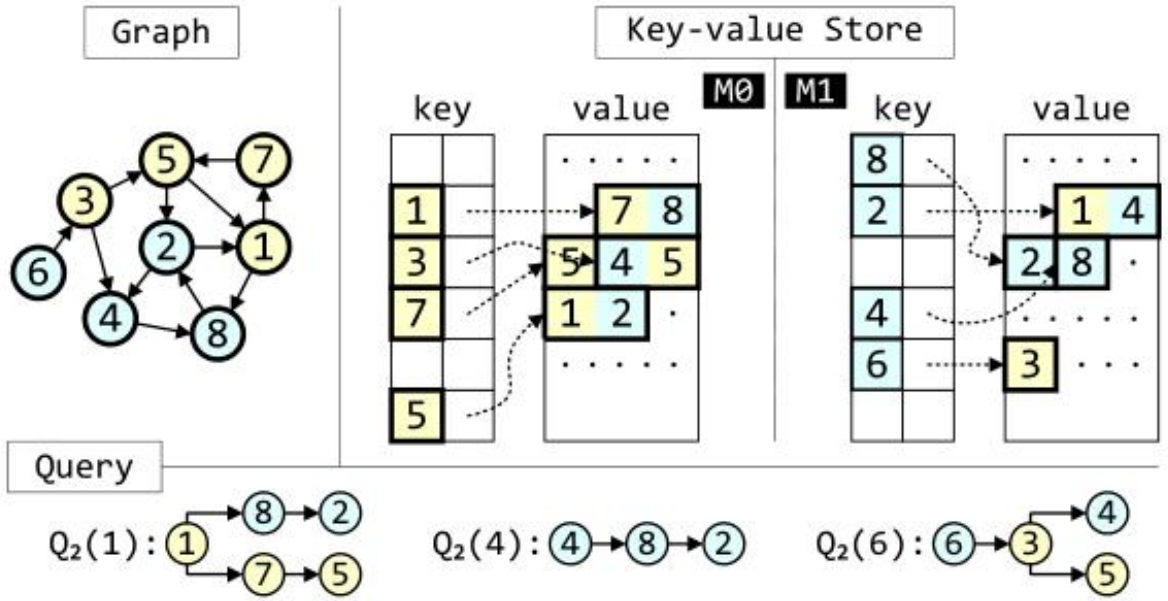
Pragh: Locality-preserving Graph Traversal with Split Live Migration

Xiating Xie, Xingda Wei, Rong Chen, Haibo Chen
Shanghai Key Laboratory of Scalable Computing and Systems
Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University
Contacts: rongchen@sjtu.edu.cn

slide made by wgl

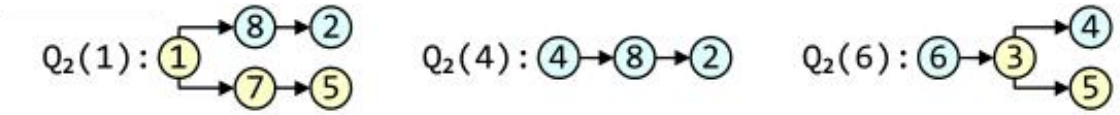
Background

➢**Graph Store and Traversal Workload**



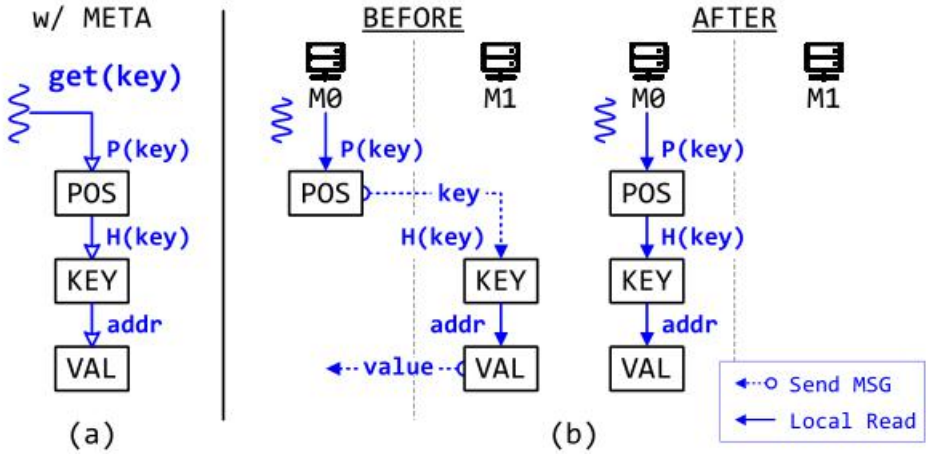
Motivation

- **Poor Locality and Partitioning**
  - For distributed in-memory stores



  - Locality-aware graph partitioning algorithms

- **Live migration**
  - Meta-data (POS)
  - Shard-based: Group the data into shards

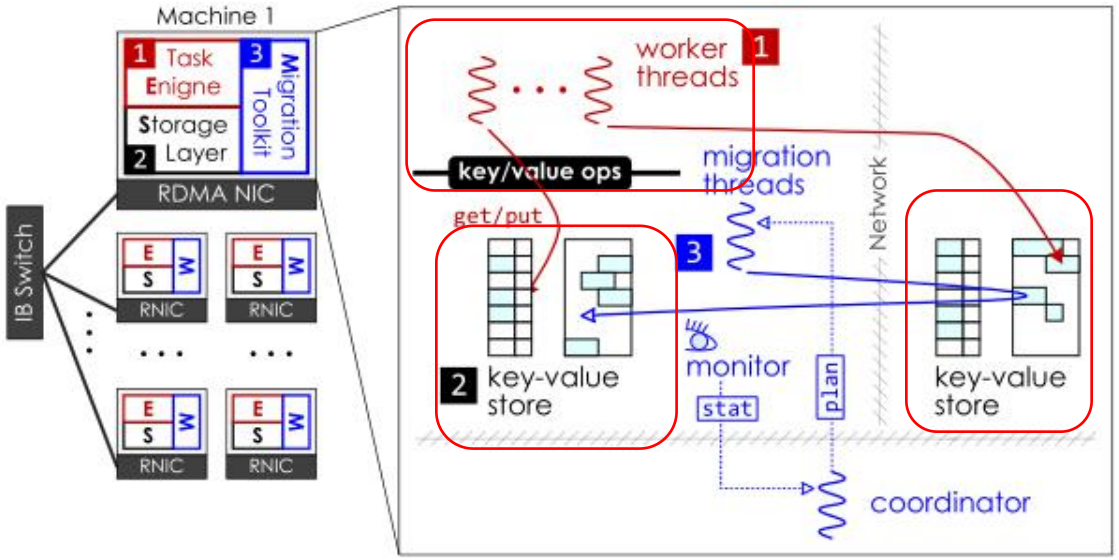


Overview

- **Design**
  - Pragh
  - Split live migration
    - Basic Split Migration *
    - Location cache
    - Lightweight Monitoring
    - Check-and-forward
- **Evaluation**
  - Migration Benefits & Speed & Migration Plan
- **Conclusion**

Architecture

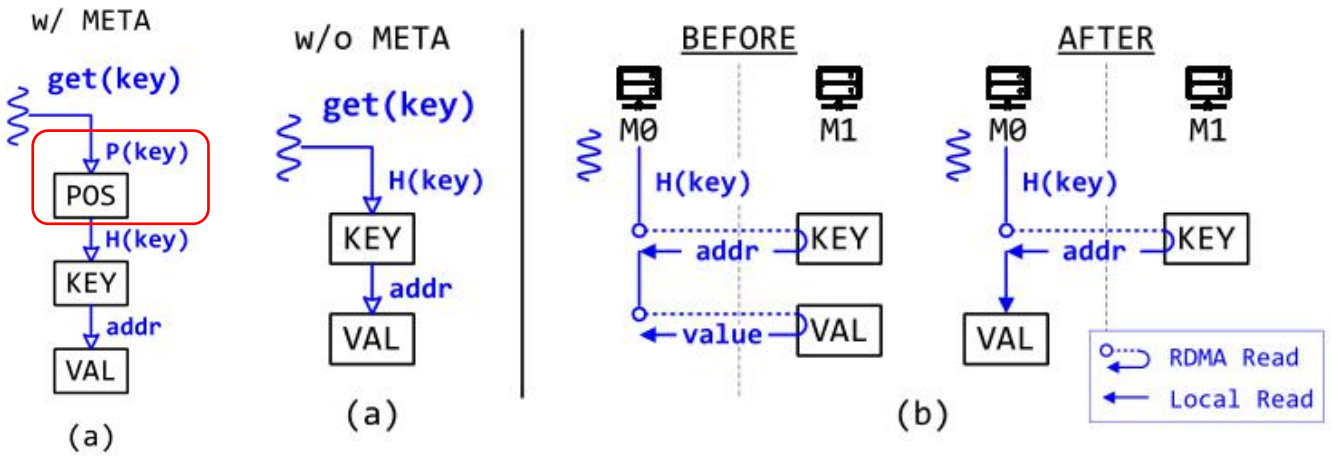
- **Pragh**



- Task engine : Worker thread(GET & PUT)
- Storage layer : RDMA-enabled key-value store
- Migration toolkit : Monitor & Coordinator & Migration threads

Design

➢ Basic Split Migration *



No need for metadata

➢Key is stationary (50% remote)

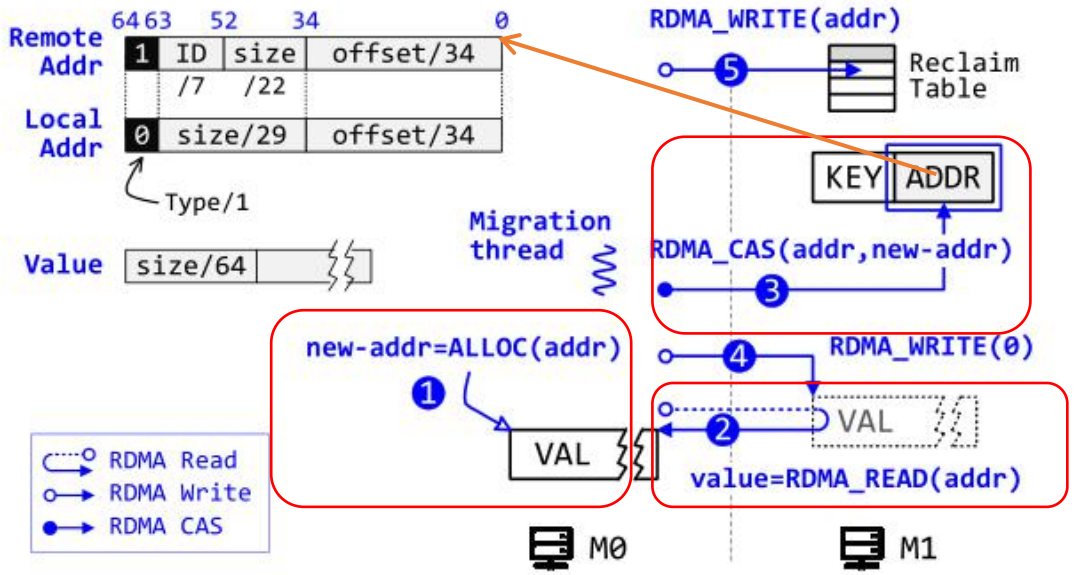
➢Only move the value to target machine

➢RDMA(Remote Direct Memory Access)

- READ & WRITE & CAS
- READ / WRITE to retrieve/update
- Compare-and-swap

Design

➢ Basic Split Migration



➢Address layout

➢Unilateral migration protocol

❶ Allocate memory space in local

❷ RDMA READ: retrieve the value

❸ RDMA CAS: original (local)→new (remote)

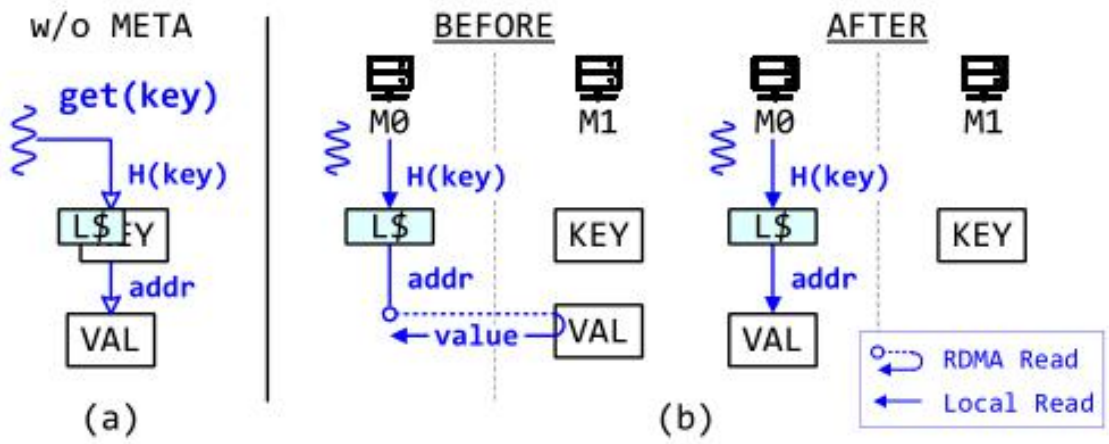
➢Invalidation and reclaim

❹ Invalidate the original memory→0

❺ Reclaim the memory of migrated

Design

- **Location cache**

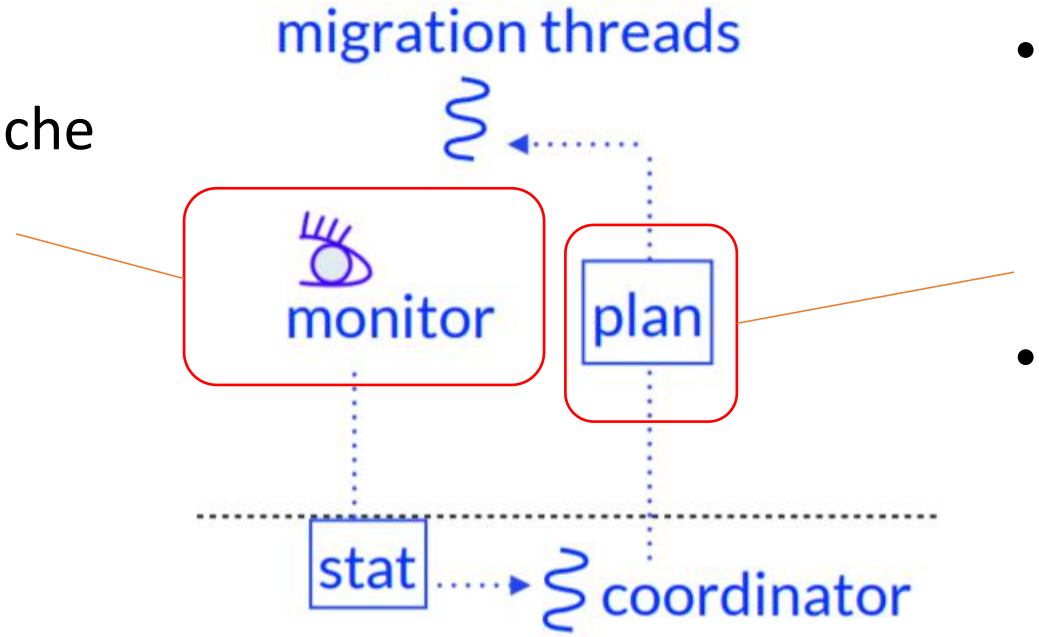


- (L$) stores the address of key-value pair
- Fully-localized

Design

➢Lightweight Monitoring

- Remote access
  Reuse location cache
- Local access
  Separate table

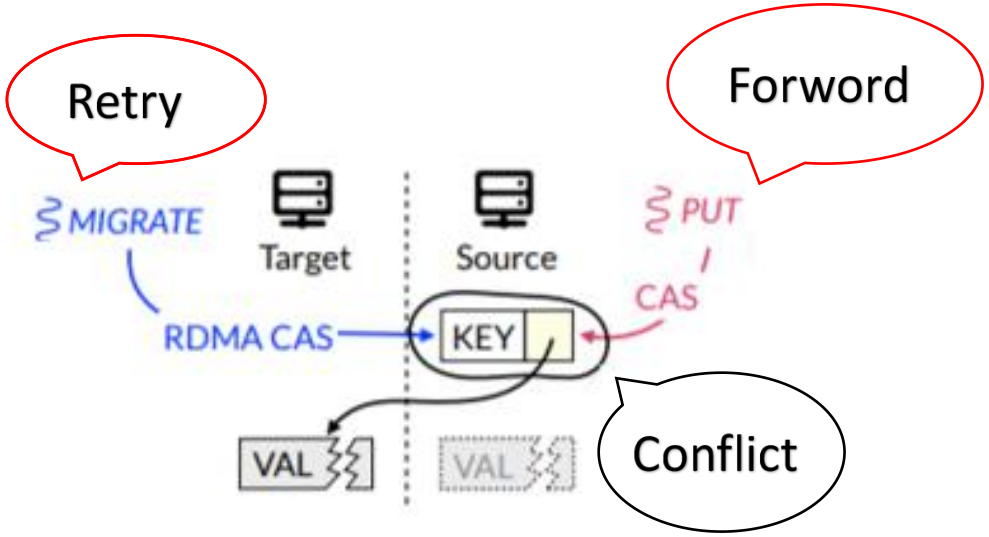


- Eager migration
  Eagerly approving &
  Second migration
- Deferred migration
  Track the local accesses
  using a separate table

Design

➢ **Full-fledged Split**

- Check-and-forward
- Detect conflicts in keys

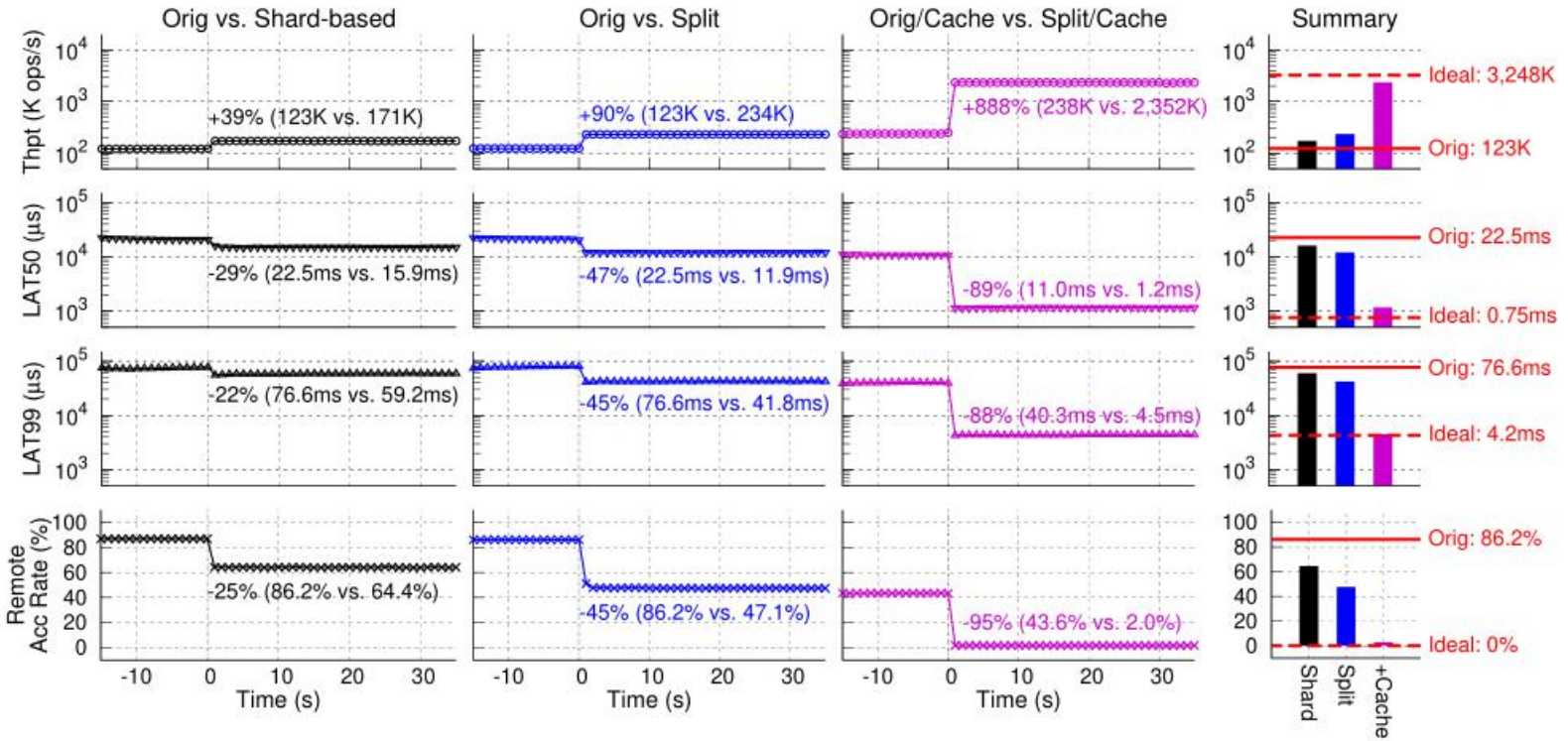


Evaluation

- **Traversal benchmark.**
  - Default dataset: graph with 2^26 vertices and 2^30 edges & rack-scale cluster with 8 nodes
  - 95% two-hop queries (Get) & 5% edge updates/inserts (Put)

Evaluation

➢Migration Benefits

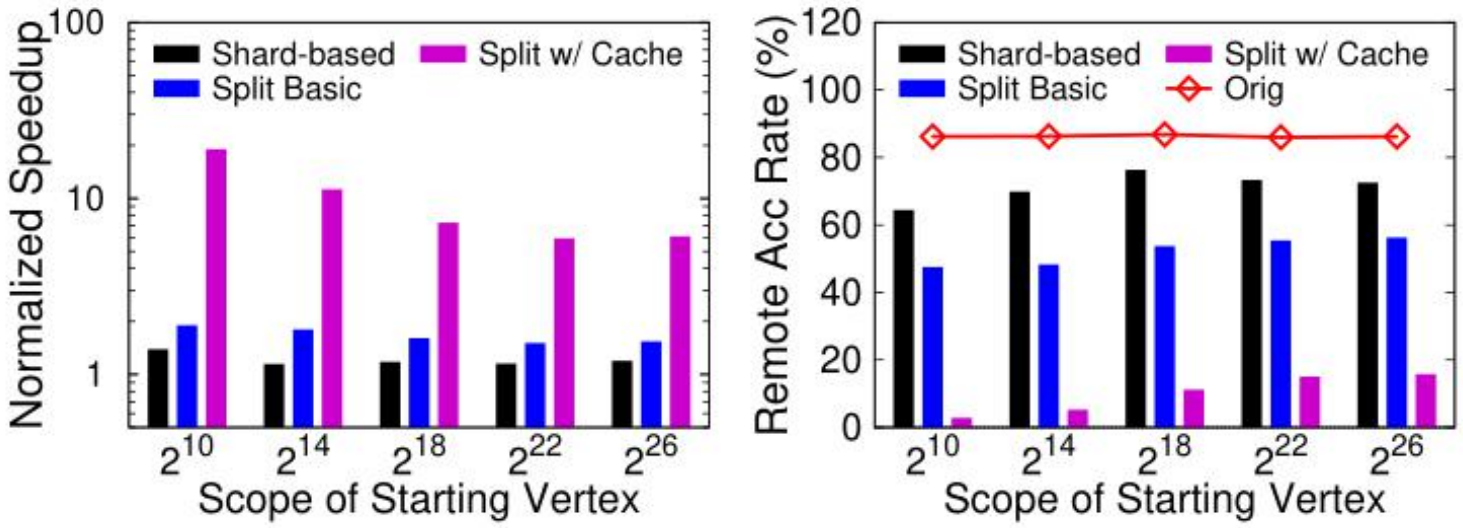


- ➢ Orig
- ➢ Ideal
- ➢ Shred-Based
- ➢ Split and Split/Cache

Pragh can almost double the throughput and reduce the latency by half

Evaluation

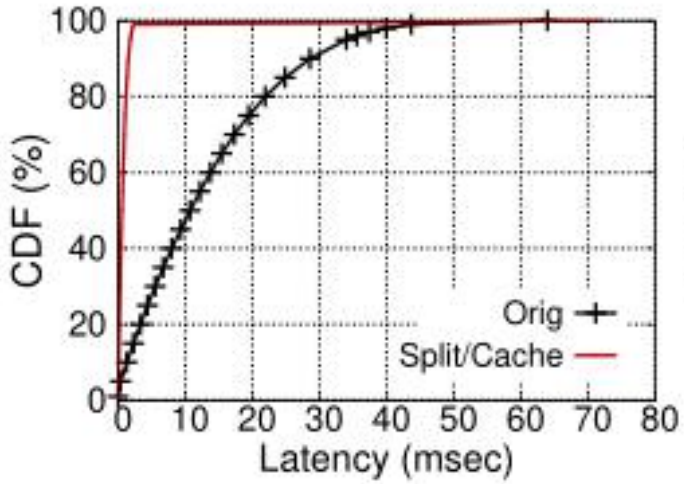
➢Migration Benefits

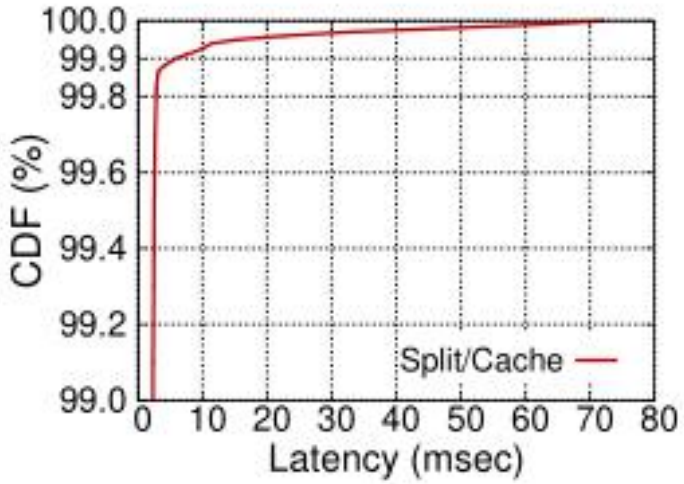


- The speedup decreases & the remote accesses increase with the increase of scope

Evaluation

➢Migration Benefits



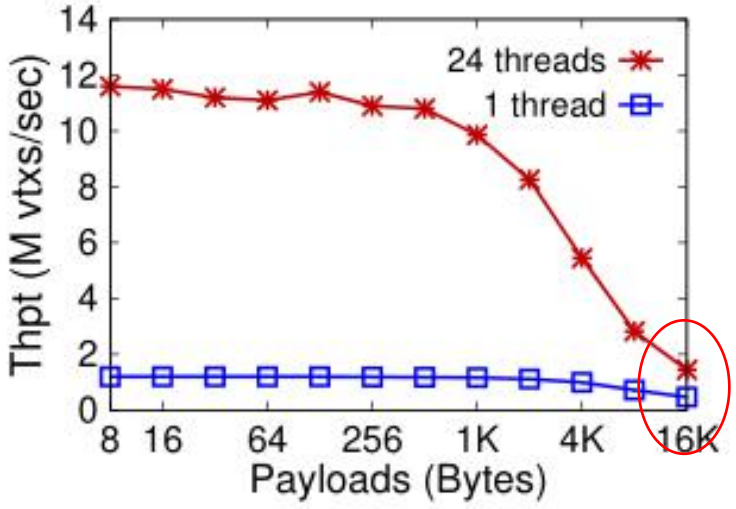


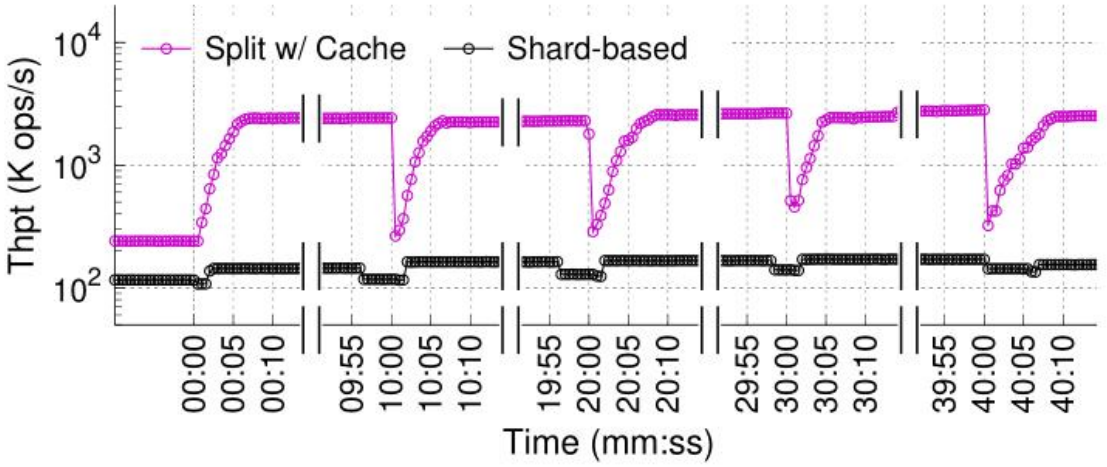
- 50% two-hop queries (Get) & 50% edge updates/inserts (Put)

- After migration, the latency of 99.9% Put operations decreases significantly, 0.11% Put operations→forwarded

Evaluation

➢Migration Speed

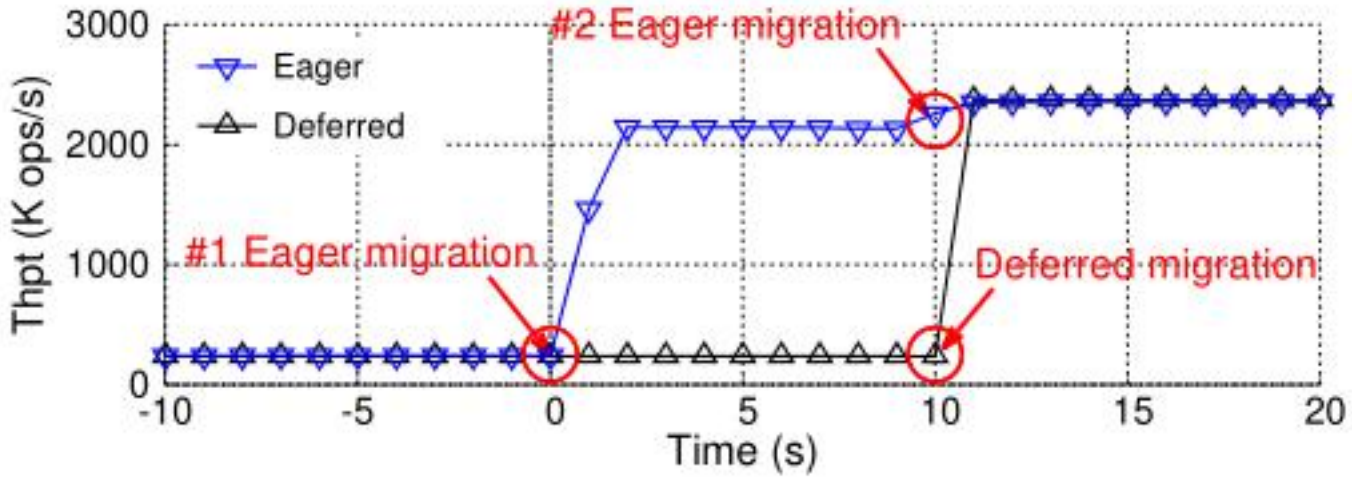




- With the increasing size of payloads, Thpt decreases ,a single thread is enough for 4KB
- The performance notably drops every time the workloads change

Evaluation

➢ **Eager Migration vs Deferred Migration**



- Eager : 0-starts directly, 10-second migration happens
- Deferred : 10-do the migration with an optimal plan

Conclusion

- **Pragh**
  - An efficient locality-preserving live migration
  - Split live migration:
    - Fine-grained migration & no nedd for metadata
    - Location cache
    - Lightweight Monitoring
- **Evaluation**
  - Migration Benefits & Speed & Migration Plan

Append

➢Hardware configuration

- Each node has two 12core Intel Xeon E5-2650 v4 processors and 128GB DRAM.Each node is equipped with two ConnectX-4 MCX455A, 100Gbps InfiniBand NIC via PCIe 3.0 x16 connected to a Mellanox SB7890 100Gbps IB Switch, and an Intel X540 10GbE NIC connected to a Force10 S4810P 10GbE Switch.

➢Wukong