华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

武汉光电国家研究中心
WUHAN NATIONAL LABORATORY FOR OPTOELECTRONICS

SEPH: Scalable, Efficient, and Predictable Hashing on Persistent Memory [OSDI’23]

报告人：邹亦舸

日期：2023.12.14

Background

- **Persistent Memory (PM)**

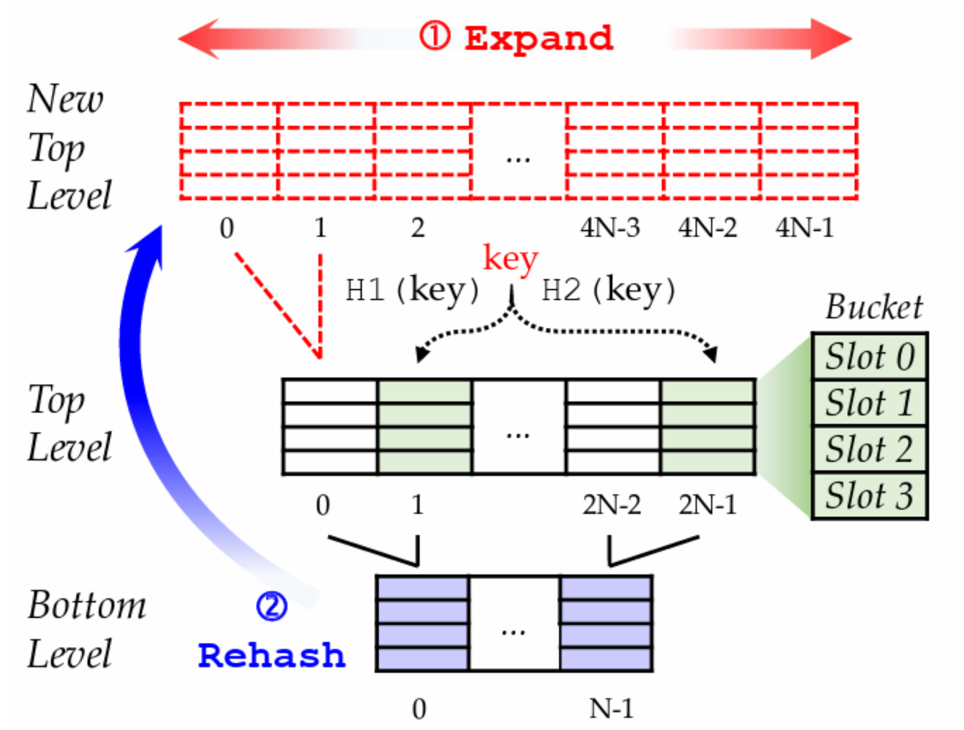
第一款商业PM产品

PM与DRAM的性能比较

	DRAM	PM	PM/DRAM
Latency of Seq. Read (ns)	81	169	208.64%
Latency of Ran. Read (ns)	101	305	301.98%
Latency of Write (ns)	57	62	108.77%
Bandwidth of Read (GB/s)	105.6	37.6	35.61%
Bandwidth of Write (GB/s)	76.8	12.5	16.28%

与DRAM相比，PM的随机读延迟和写带宽问题最为突出

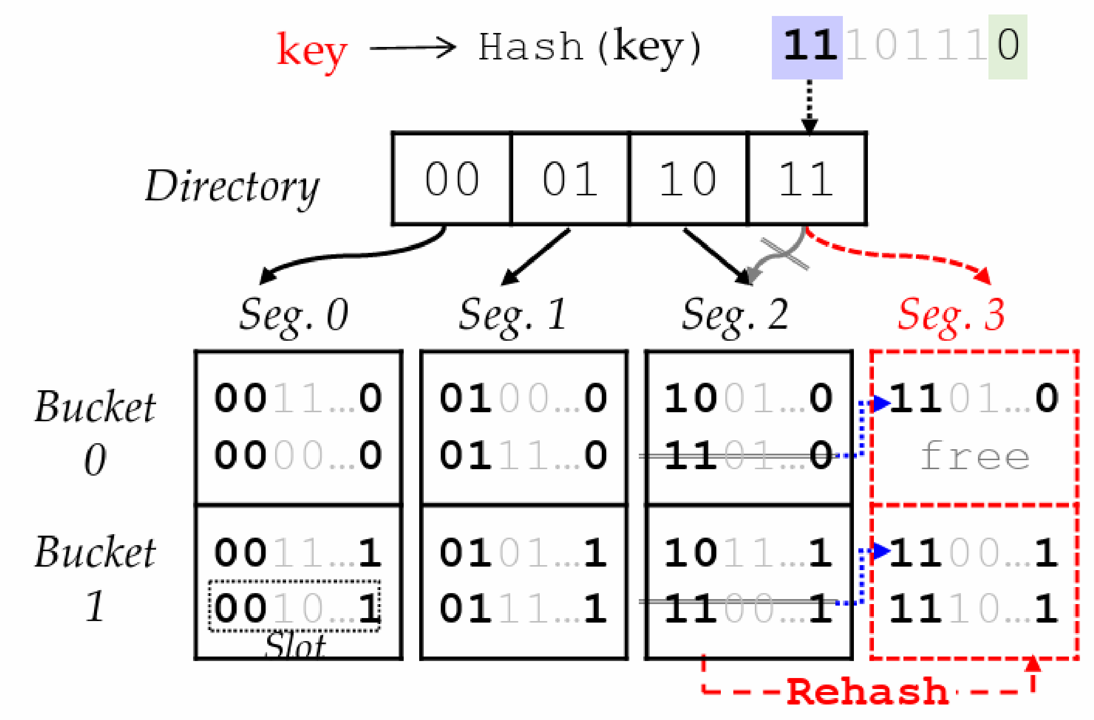
Existing Hashing Schemes in PM



Level-based Hashing

(Level Hashing[OSDI'18], Clevel Hashing[ATC'20])

双层结构，高效扩容



EH-base Hashing

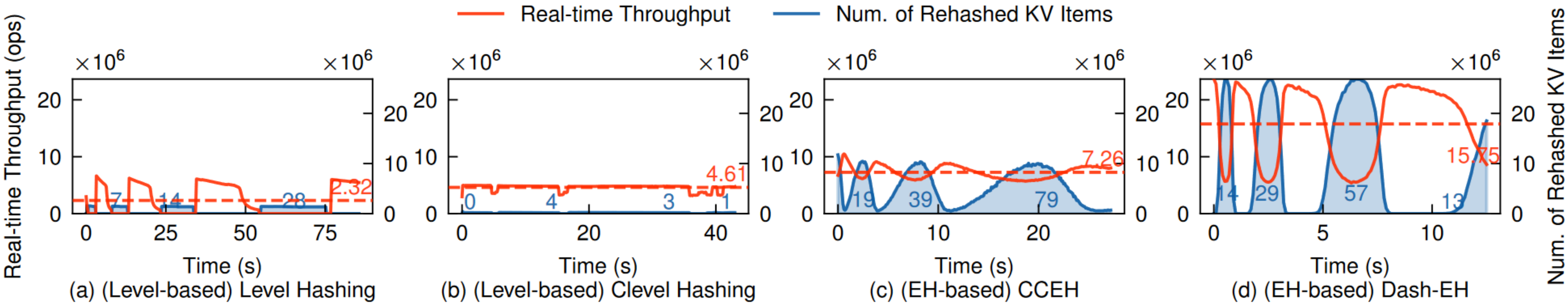
(CCEH [FAST'19], Dash [VLDB'20])

以桶为粒度扩容

Motivation 1: Dilemma between Efficiency and Predictability

Performance Efficiency：平均性能　　**Performance Predictability：高百分位性能**

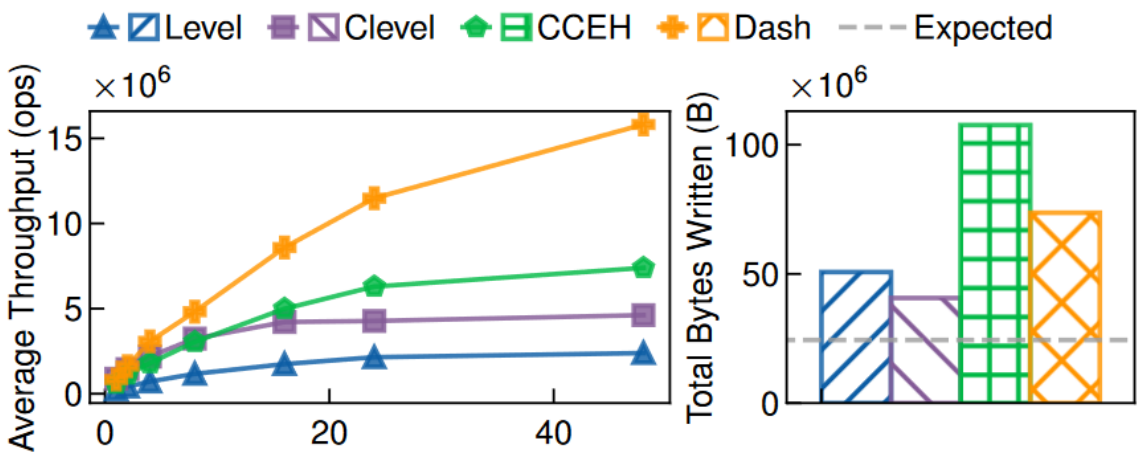
在混合负载（50%查找，50%插入）下，测试实时吞吐量和扩容开支：



- EH-based连续读，Level-based随机读，连续读更快，因此平均性能更高
- Dash平均性能高于CCEH，但由于高扩容开销，高百分位性能较差
- Level-based扩容开销低，但由于性能总体较差，高百分位性能更差

Motivation 2: Limited Scalability

Performance Scalability：处理高并发请求时的性能



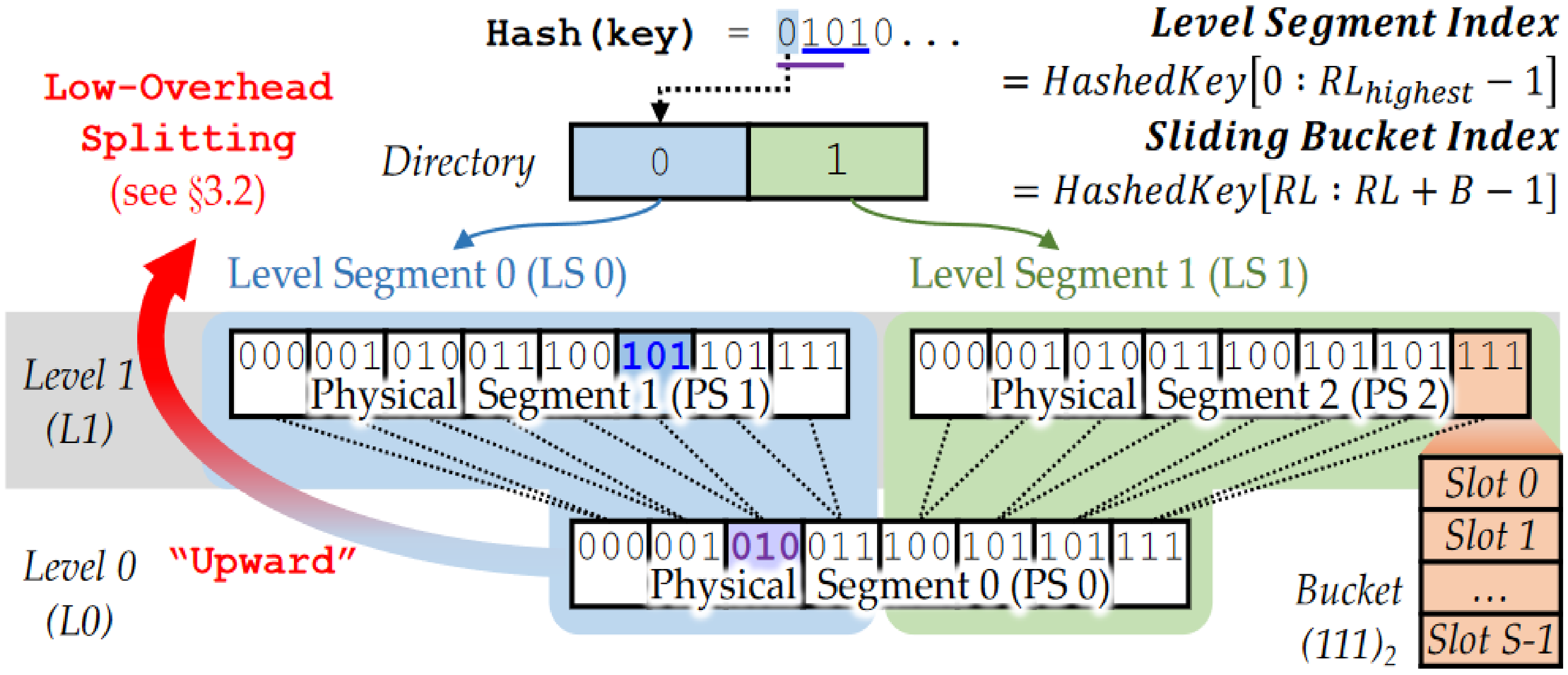
(a) Average Throughput S50% I50%
(b) Total Bytes Written to PM

混合负载（50%查找，50%插入）
不同数量的并发线程（1~48）
测试平均吞吐量和实际数据写入量

- 并发线程量超过24以后，平均吞吐量提升较少
- 实际数据写入量超过预期值，额外的PM写成为性能瓶颈
  - Level, CCEH, Dash为基于锁的哈希，需要一个PM写来锁定和插入KV项，另一个PM写来解锁，因此写数量超过预期值的两倍
  - Clevel需要额外的PM写操作来保存元数据以保证崩溃一致性

Level Segment based Hash Table

- **Structure and Indexing**



Physical Segment (PS)

- PM中的固定大小空间
- 包含固定数量2^B个桶
- 每个桶包含固定数量的槽

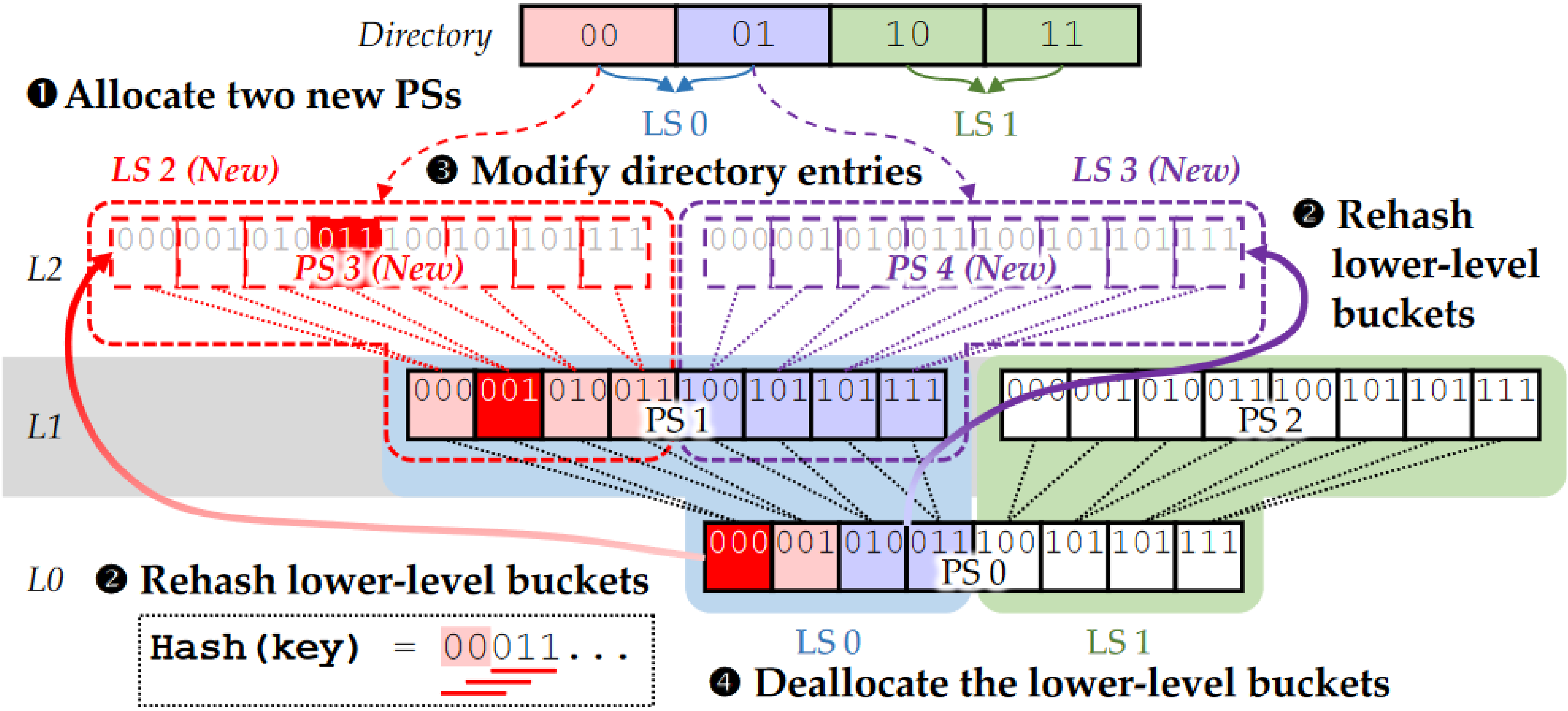
Level Segment (LS)

- 半个低层PS和一个高层PS组成一个LS
- 高层PS中的两个相邻桶共享一个低层桶
- KV项优先存放在低层桶中

Residing Level (RL): PS在表中所处的层级，最低层为0

Low-Overhead Split

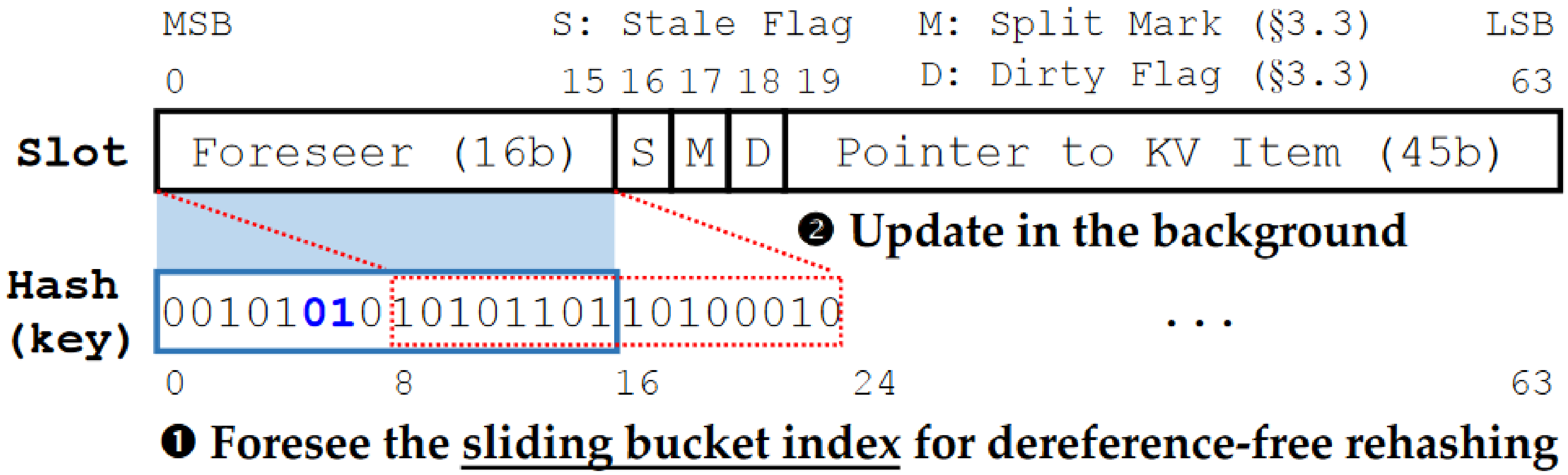
- **One-Third Splitting**



一个LS分裂成两个LS，但是只有1/3的KV项需要重哈希

Low-Overhead Split

- **Dereference-Free Rehashing**
  - 大多数哈希方案都在槽中储存指向KV项的指针
  - 在重哈希时需要Key，要将指针解引，以得到Key
  - 指针解引需要随机读，因此增加了重哈希的开销



Bucket Index Foreseer

- 重哈希时桶索引向后滑动两位（即使用2个未用作索引的bit)
- 预先保存16个bit
- 当unused bit少于一半时用一个后台线程来更新预测器

Semi Lock-Free Concurrency Control

- Scalability ☹ ⇐ excessive PM writes for concurrency control

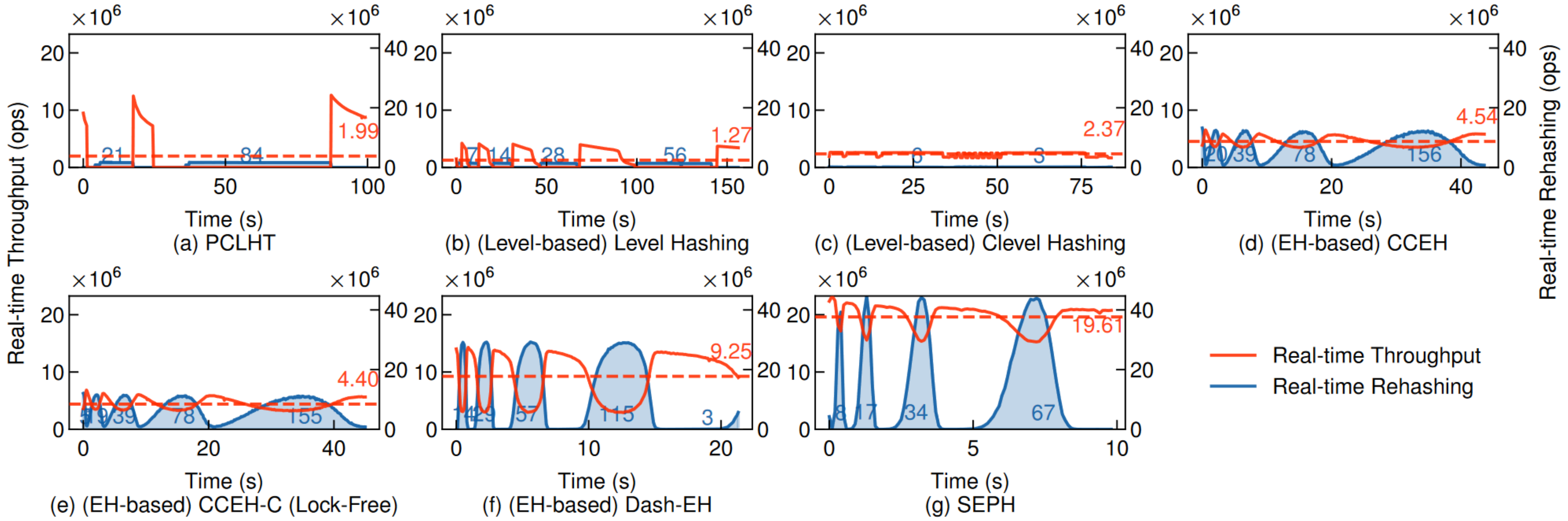
	Lock-based Designs (e.g. Level Hashing, CCEH, Dash)	Lock-free Designs (e.g. Clevel Hashing)
PM writes are to	Manage Locks	Guarantee Correctness

- SEPH solves it by

Frequent Operations (e.g. insert, search, update, delete)	Infrequent operatoins (e.g. split)
Be Lock-free (to save PM writes)	Be Lock-based (to ease correctness guarantee)

Evaluation: Efficiency & Predictability

在混合负载（50%查找，50%插入）下，测试实时吞吐量和扩容开支：

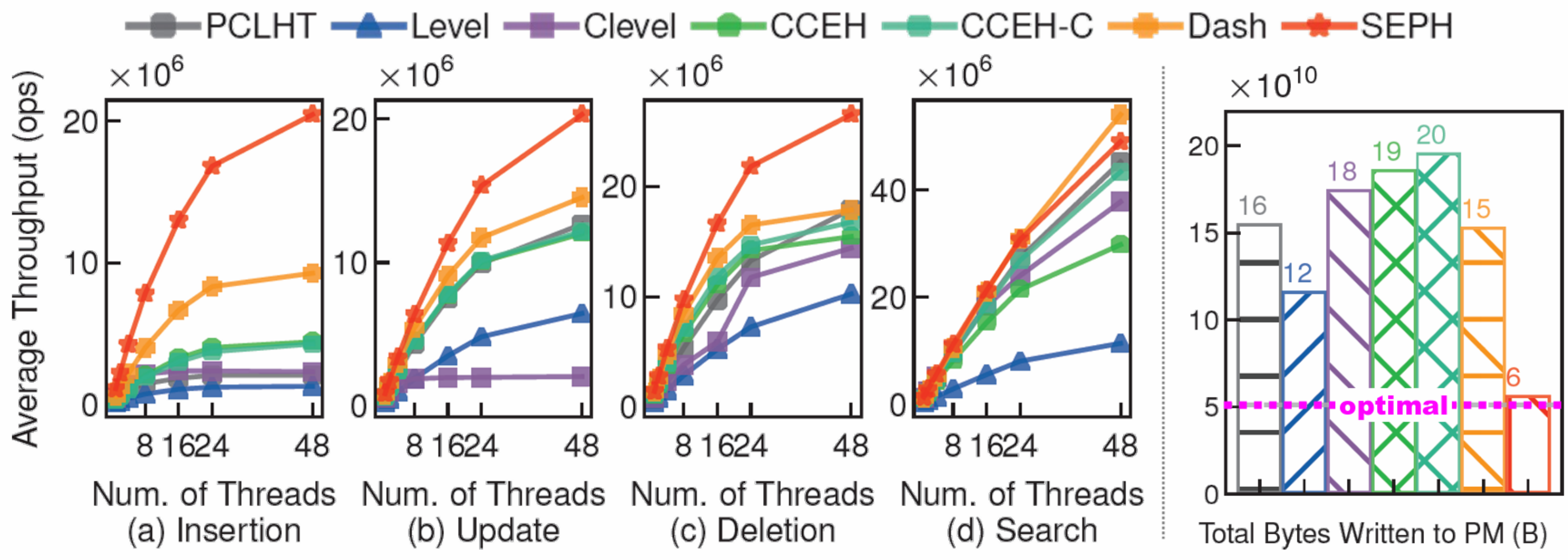


Efficiency：平均性能是其他哈希方案的平均的2.12倍

Predictability：最差性能比其他方案的最好性能还好

Evaluation: Scalability

在混合负载、不同并发线程数量下，测试实时吞吐量和实际数据写入量：

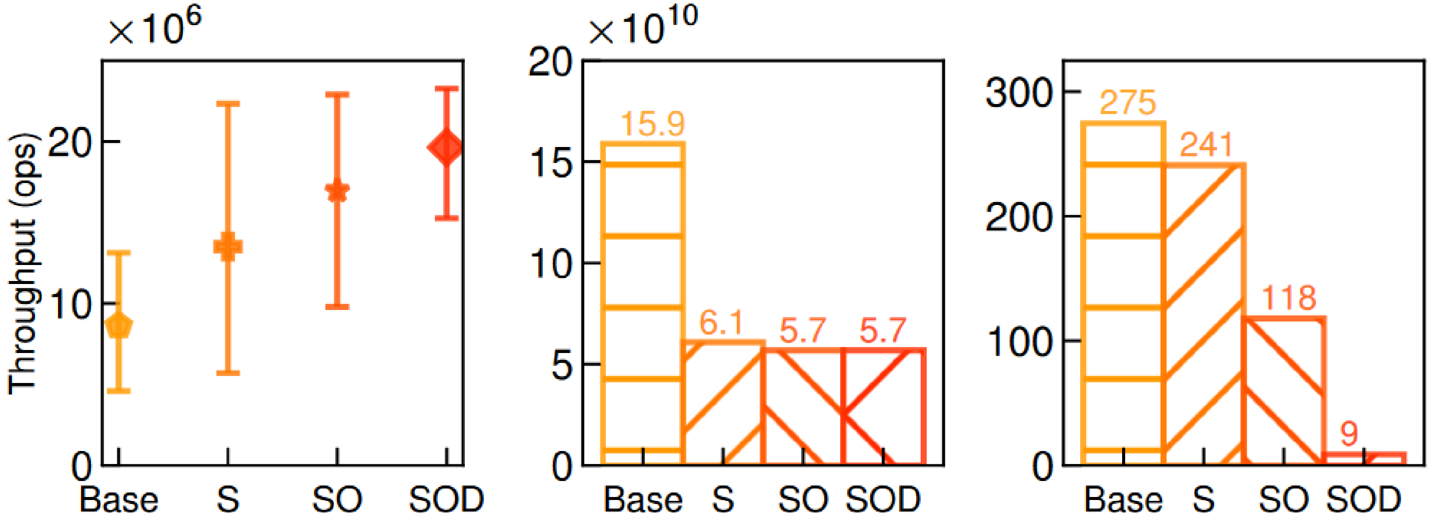


由于半无锁机制，实际数据写入量降低，SEPH的插入、更新、删除操作都具有很好的高并发性能；而无需写操作的查找也只是略逊于Dash

Evaluation: Performance Breakdown

SEPH Variants	Semi Lock-Free	One-Third Splitting	Dereference-Free Rehashing
SEPH-Base	×	×	×
SEPH-S	✓	×	×
SEPH-SO	✓	✓	×
SEPH-SOD	✓	✓	✓

选取带有特定功能的SEPH变种，测试SEPH三种功能分别为性能带来的影响



(a) Min/Avg/Max Throughput (b) Total Bytes Written to PM (c) Total Resizing Time (s)

- 半无锁机制减少了写入量，提高了整体性能，因此使Efficiency和Scalability提升
- 1/3分裂减少了扩容总时间，提升了Efficiency和Predictability
- 无解引的重哈希极大地减少扩容开支，进一步提高Predictability

Summary

SEPH: Scalable, Efficient, and Predictable Hashing for PM

-Efficiency vs. Predictability

- Level segment structure & low-overhead split algorithm
- 结合了两种PM哈希的优点

-Scalability

- Semi lock-free concurrency control
- 减少了用于并发控制的PM写

谢谢！