High Performance Packet Forwarding with CuckooSwitch and Integration with Intel DPDK

Dong Zhou, Bin Fan, Hyeontaek Lim, David G. Andersen,
Michael Kaminsky (Intel Labs)

Carnegie Mellon University

Ren Wang, Sameh Gobriel, Christian Maciocco, George Kennedy
Intel Labs

Venky Venkatesan, Bruce Richardson
CSIG

Agenda

- CuckooSwitch: a software switch that
  - uses DPDK as IO engine
  - can handle extremely large forwarding tables while offering line-rate throughput
- Integration with Intel DPDK framework
  - benefit: improve DPDK large table forwarding performance
  - current status

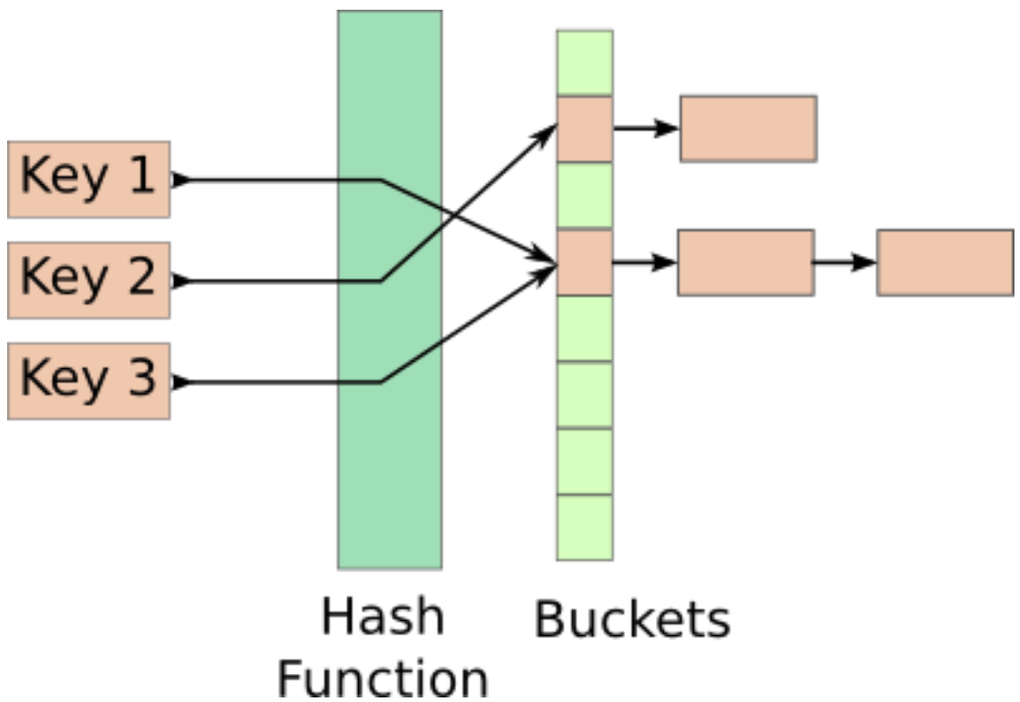
Challenges

- Requirements for network switches

more lookups per second into **larger** tables

Why Not Existing Solutions?

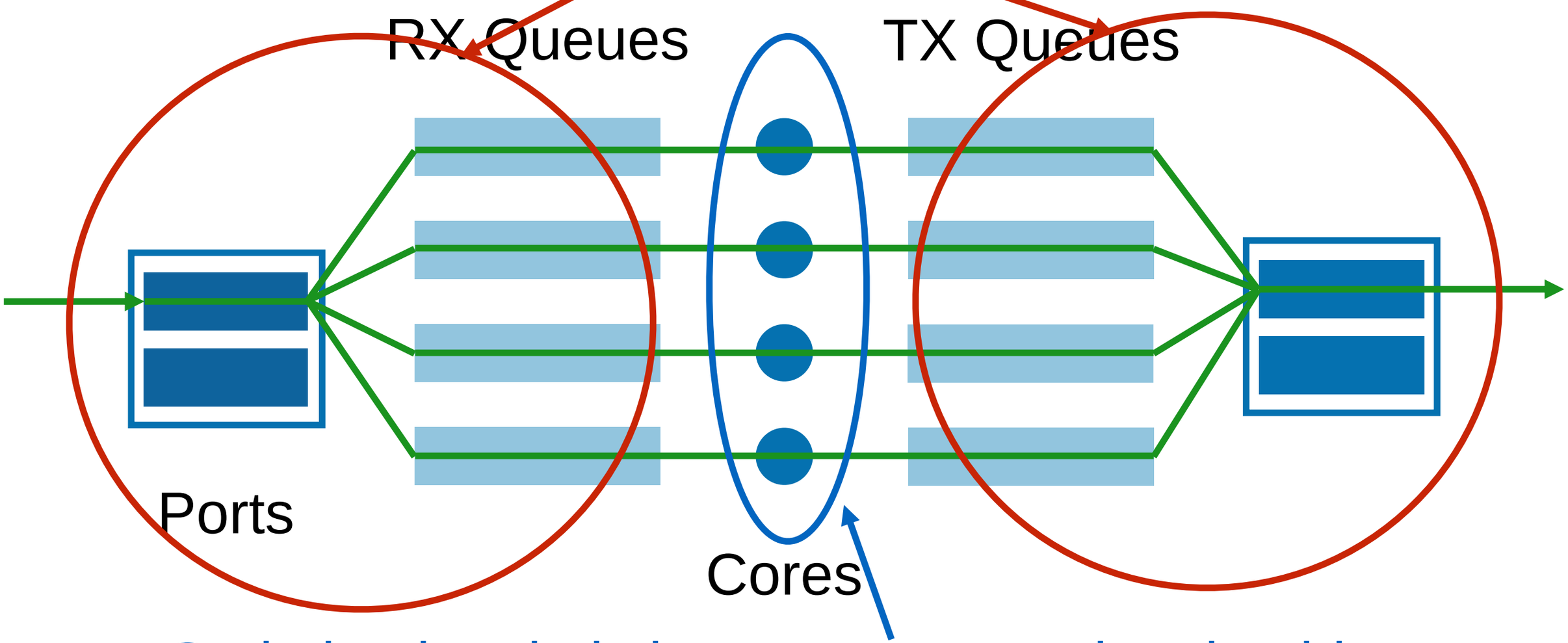
- CAM/TCAM
  - expensive
  - power hungry
  - very limited in size



- Hash tables in DRAM
  - slow
  - memory inefficient
    - hash collisions
    - pointer-swapping

Intel DPDK

high-throughput packet I/O to userspace



Optimized optimistic concurrent cuckoo hashing

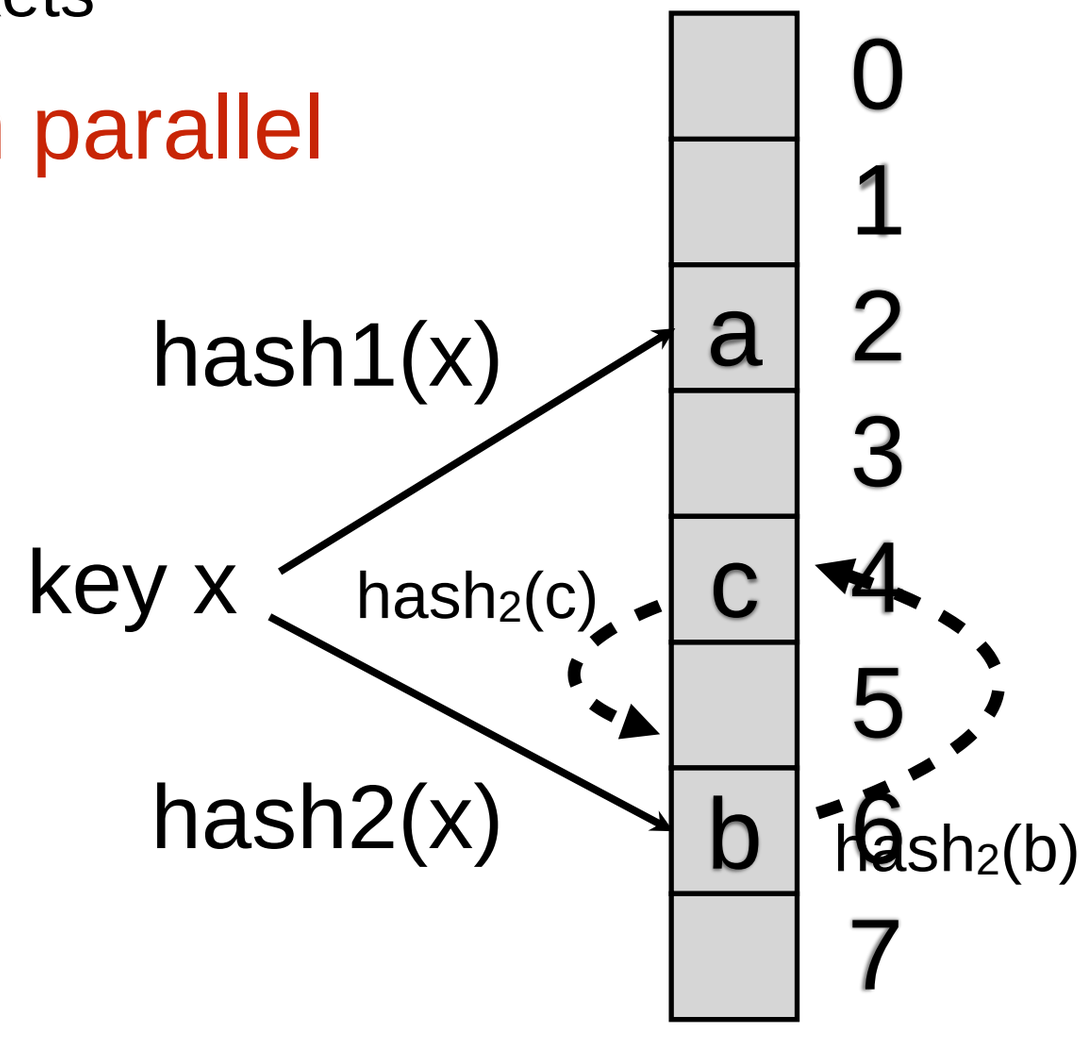
high performance switch FIB

Hash Table

- Desired properties
  - **Fast**
  - **High occupancy**
  - **In-place update**

Cuckoo Hashing[Pagh 01]

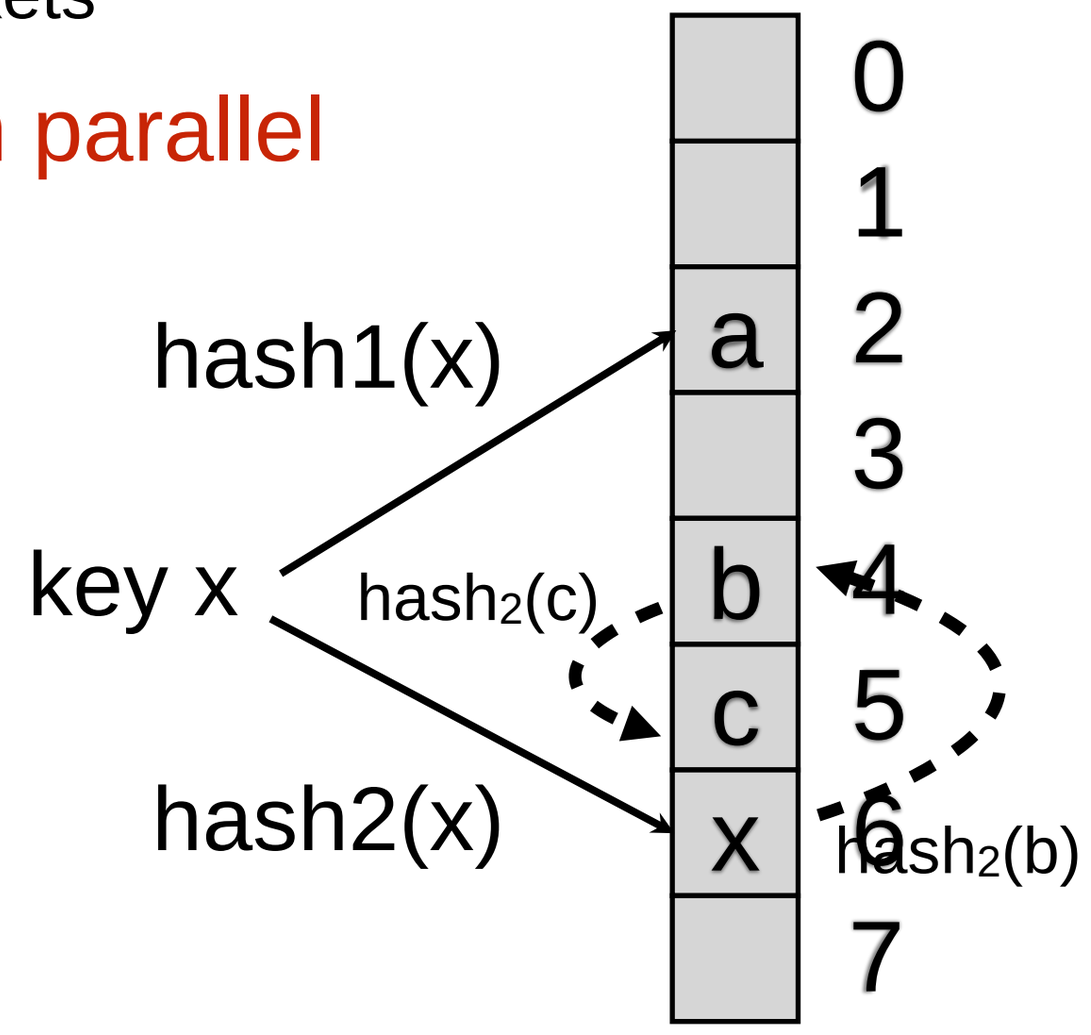
- Each key has two candidate buckets
  - Assigned by hash1(key) and hash2(key)
  - Stored in one of the candidate buckets
  - Lookup: check two buckets in parallel
- Insert: perform key displacement recursively



Cuckoo Hashing[Pagh 01]

- Each key has two candidate buckets
  - Assigned by hash1(key) and hash2(key)
  - Stored in one of the candidate buckets
  - Lookup: check two buckets in parallel

- Insert: perform key displacement recursively
- 95% occupancy when set-associativity is 4



Optimistic Concurrent Cuckoo Hashing[NSDI 13]

- Higher concurrency
  - single-writer/multi-readers by *optimistic concurrency control*

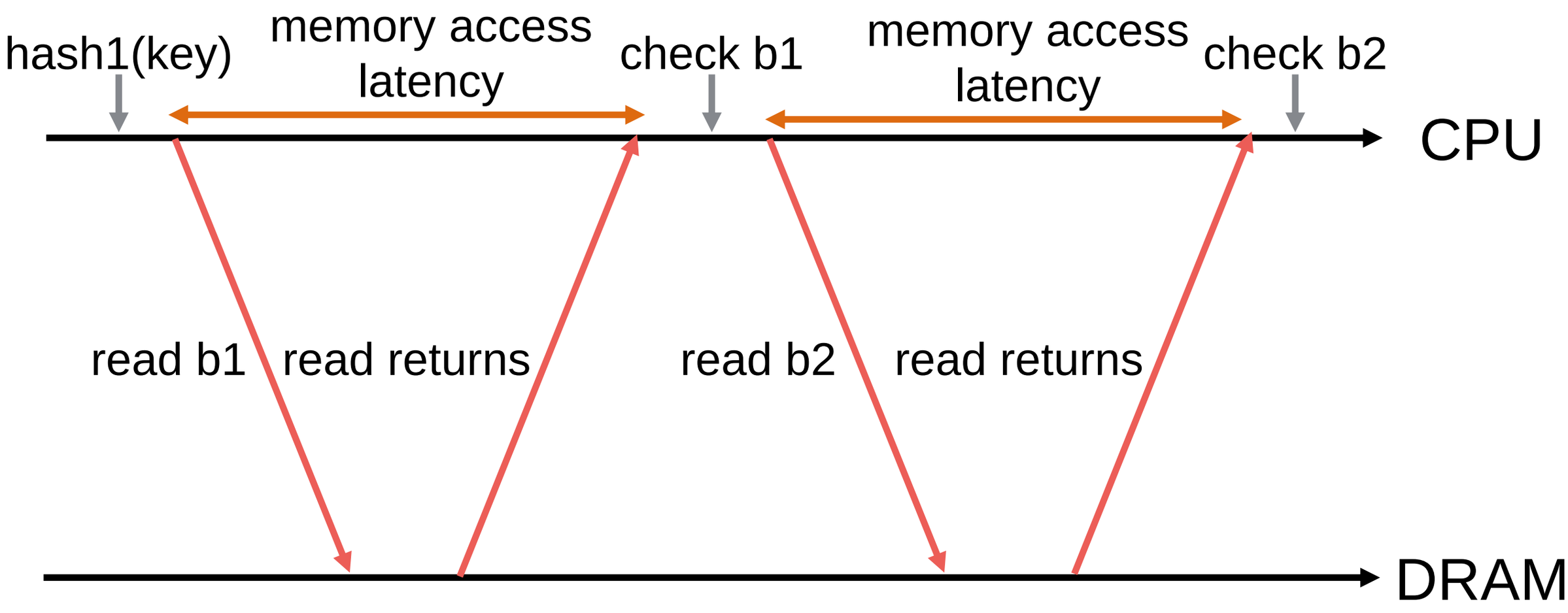
Simplified Multi-Reader/Single-Writer

- Keep a version number for each bucket
- Lookup
  - v = bucket version before lookup
  - v’ = bucket version after lookup
  - Compare v with v’, retry if mismatch or v is odd
- Insert
  - Increase versions of involved buckets for each displacement

System Optimizations

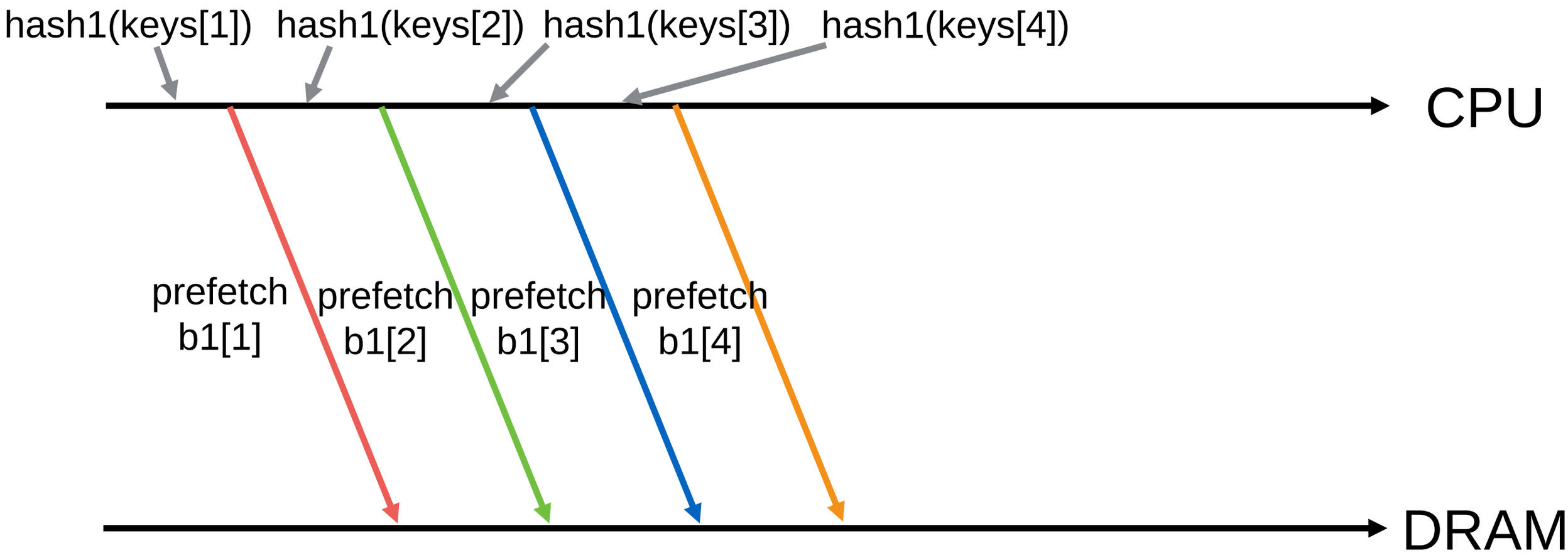
- We share the similar principles with DPDK, especially:
  - Batched hash table lookup with prefetching

Original Lookup

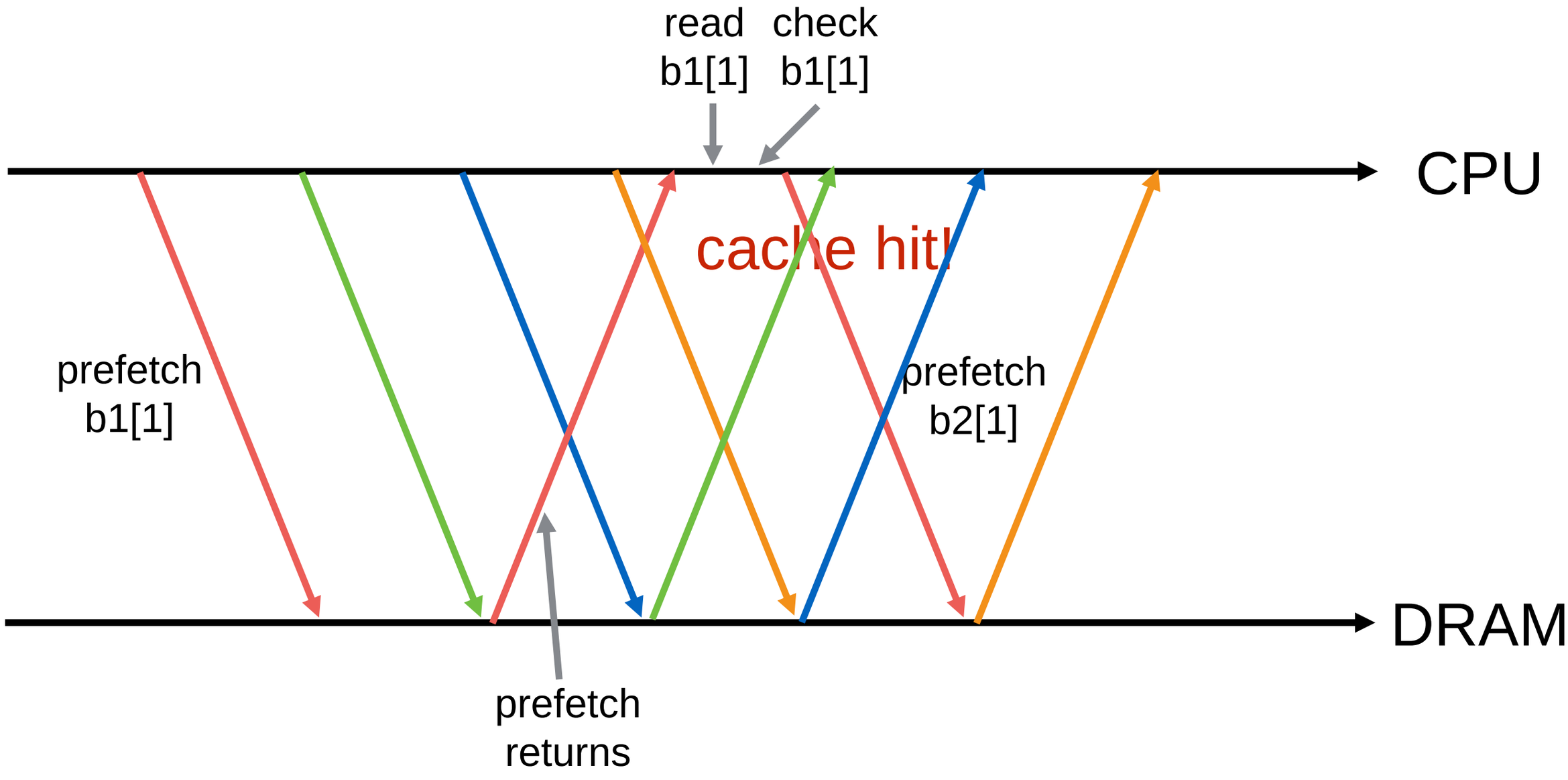


Lookup throughput is very sensitive to memory access latency

Batched Lookup with Prefetching



Batched Lookup with Prefetching

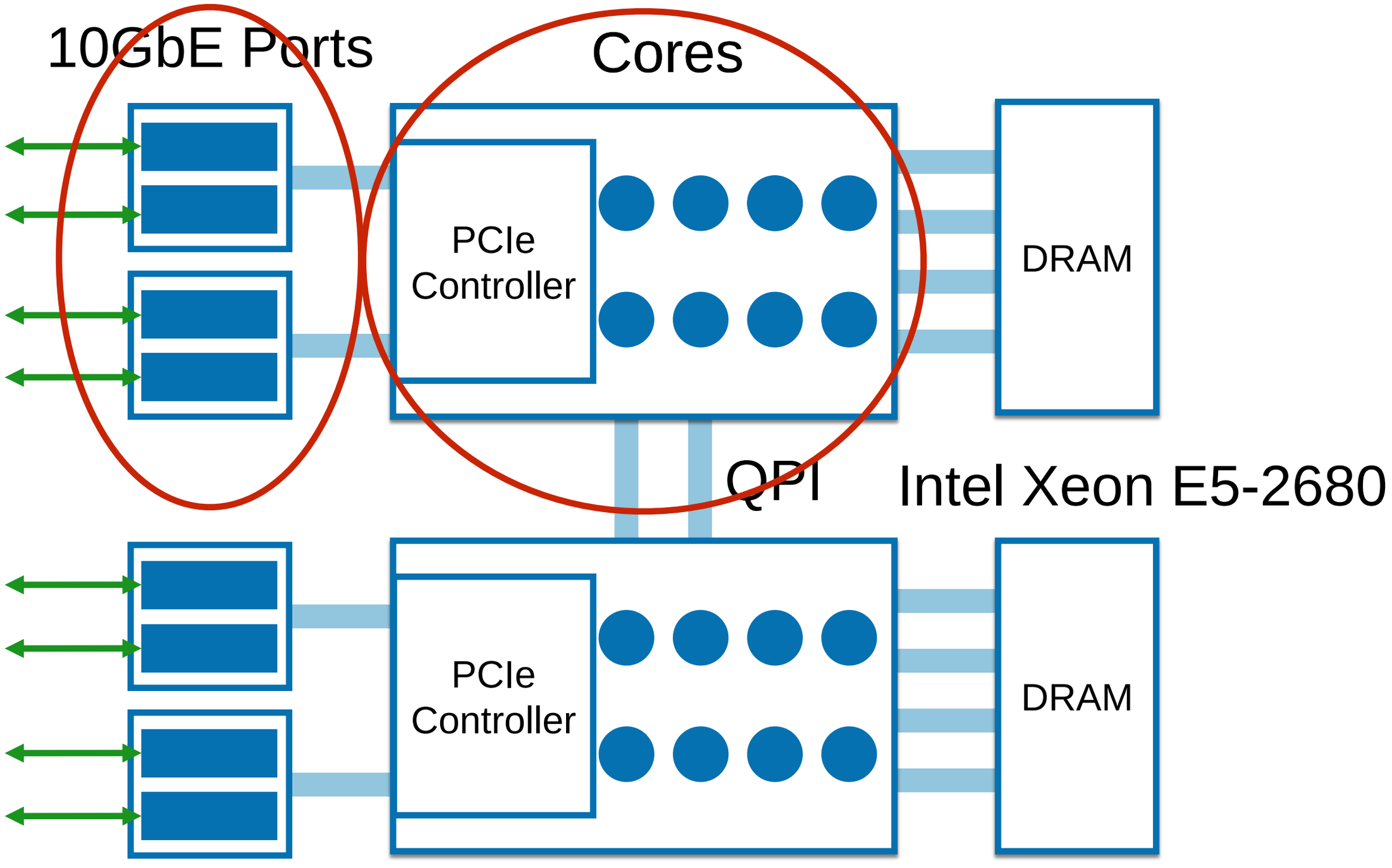


Batched Lookup with Prefetching

- Prefetch one bucket after hash computation
  - Interleave computation w/ memory accesses
  - Better use available execution units and CPU load buffers
- 1.5 cache-line retrievals on average

Performance Evaluation

Experiment Setup



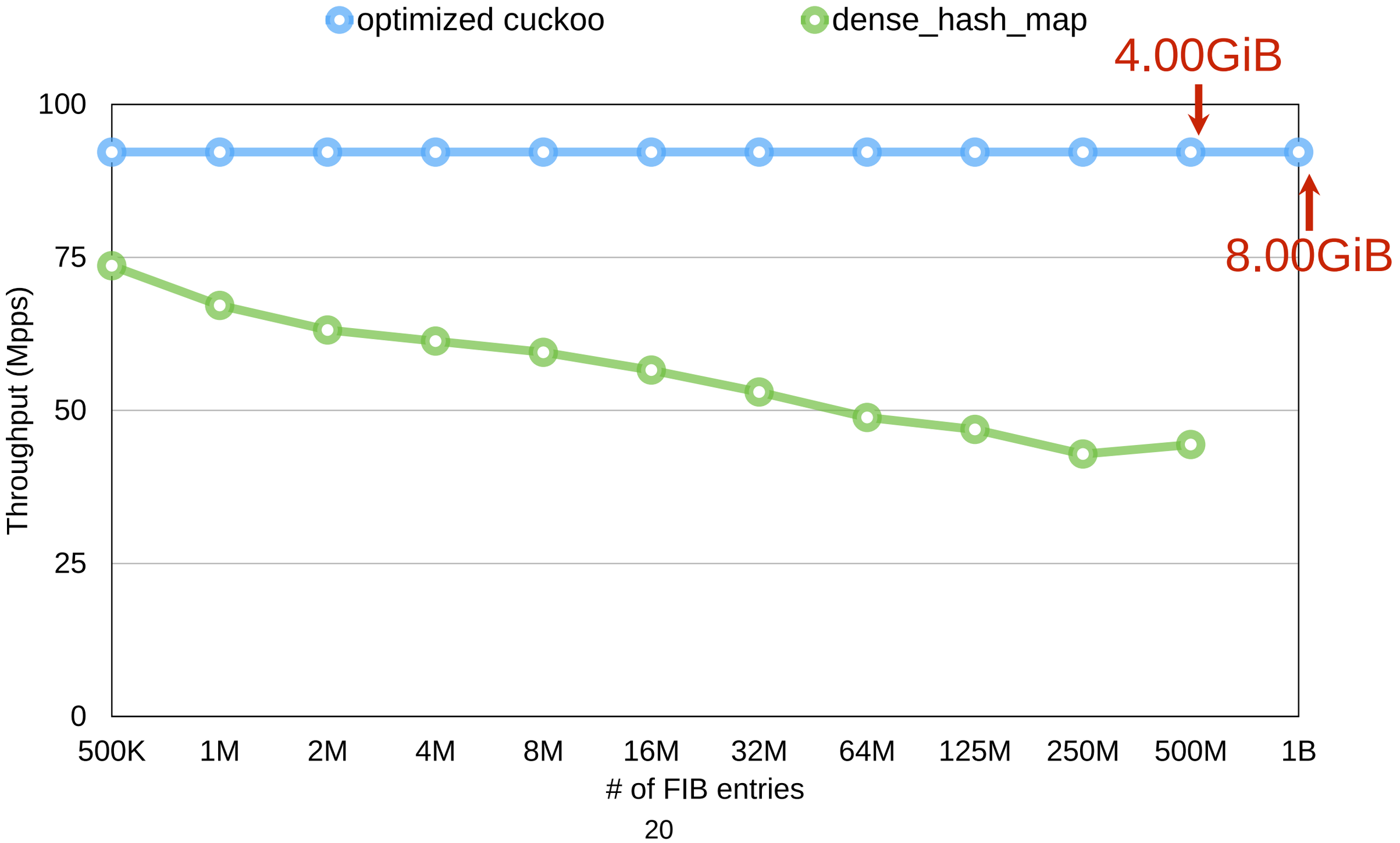
Raw Packet I/O

Packet Size (Bytes)	Throughput (Mpps)	Throughput (Gbps)	Bottleneck
64	92.22	61.97	PCIe B/W

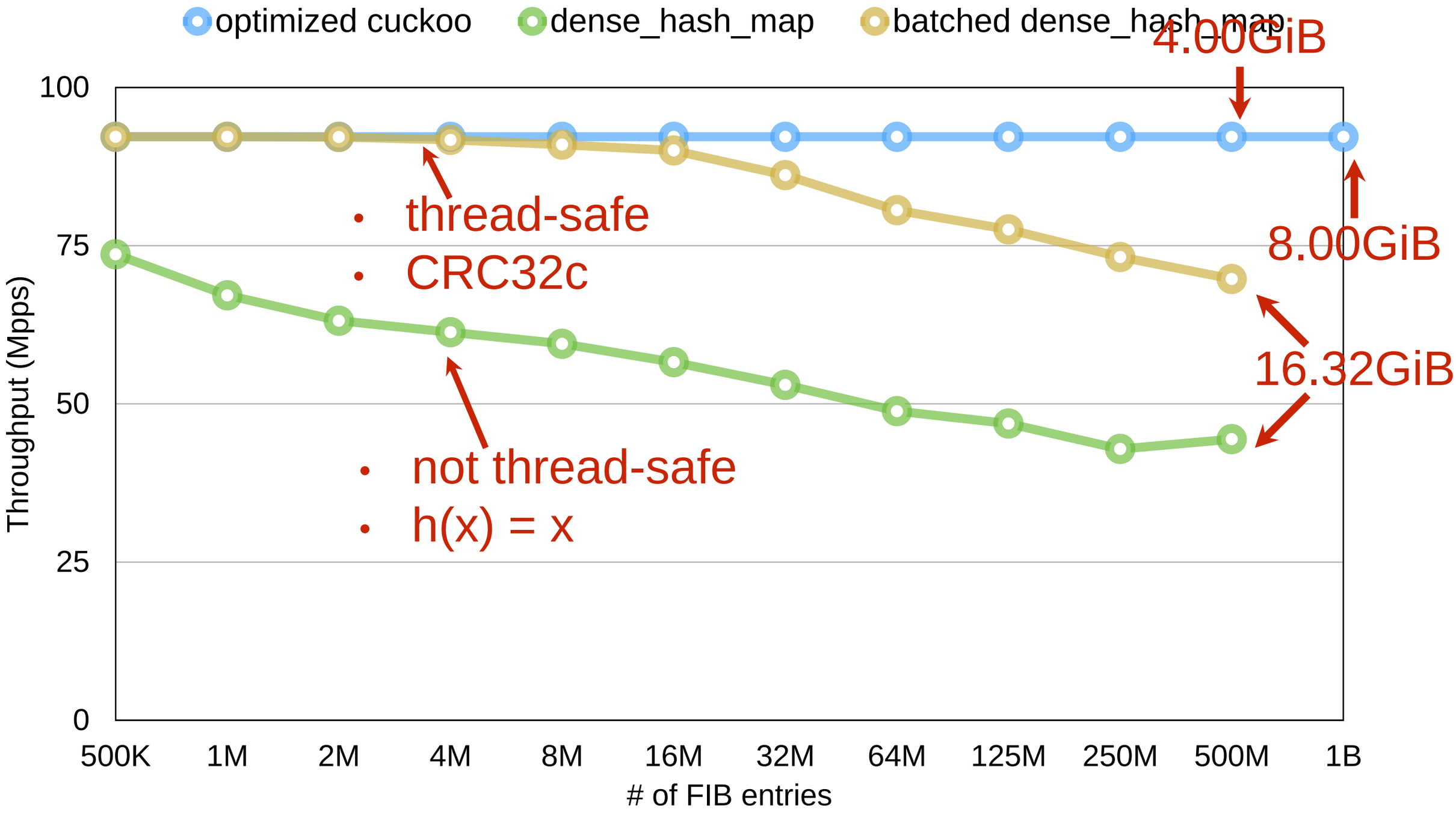
Raw Packet I/O

Packet Size (Bytes)	Throughput (Mpps)	Throughput (Gbps)	Bottleneck
64	92.22	61.97	PCIe B/W
128	66.24	78.43	PCIe B/W
192	47.17	80	Network B/W
256	36.23	80	Network B/W

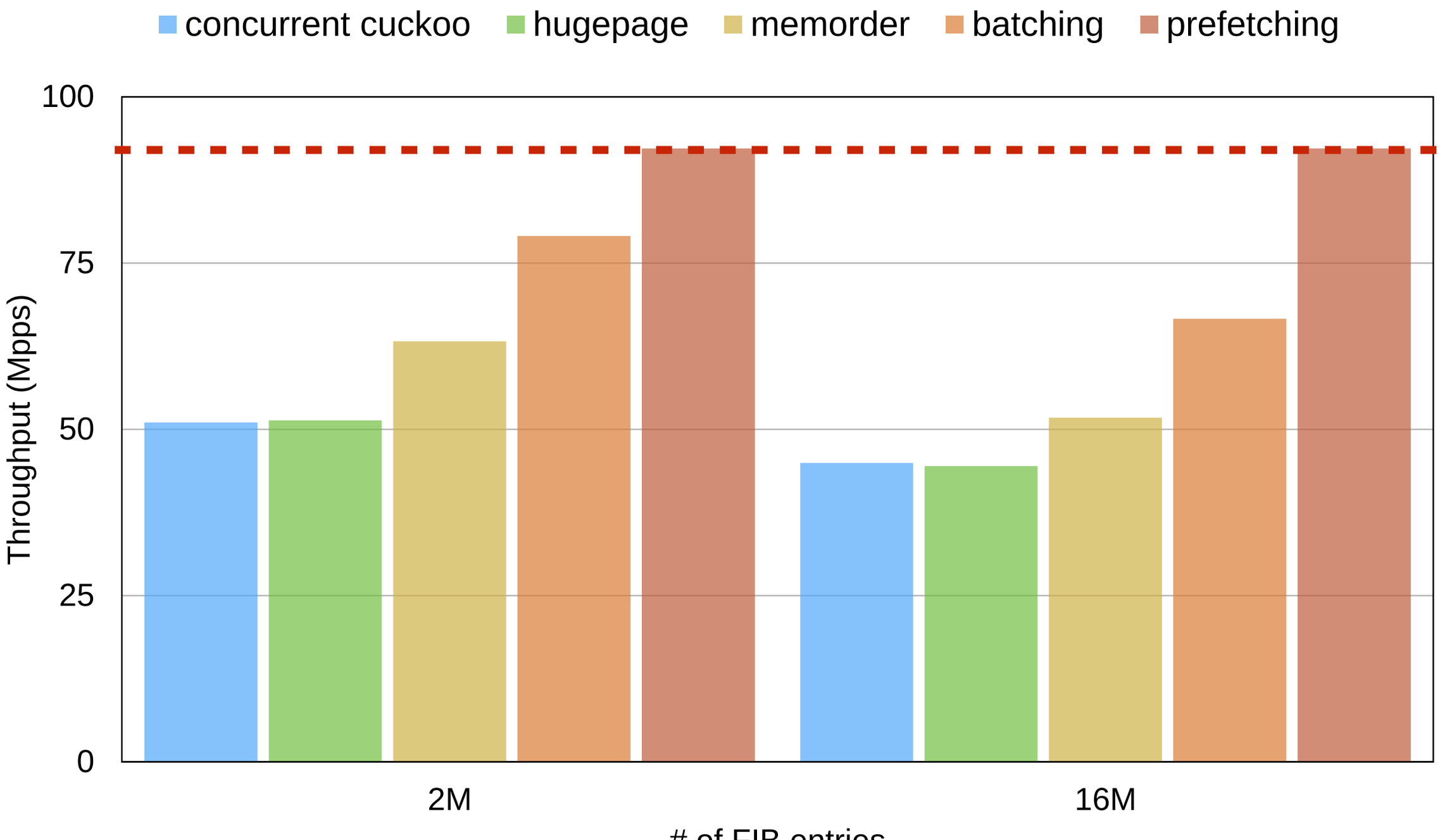
End-to-end Benchmark



End-to-end Benchmark



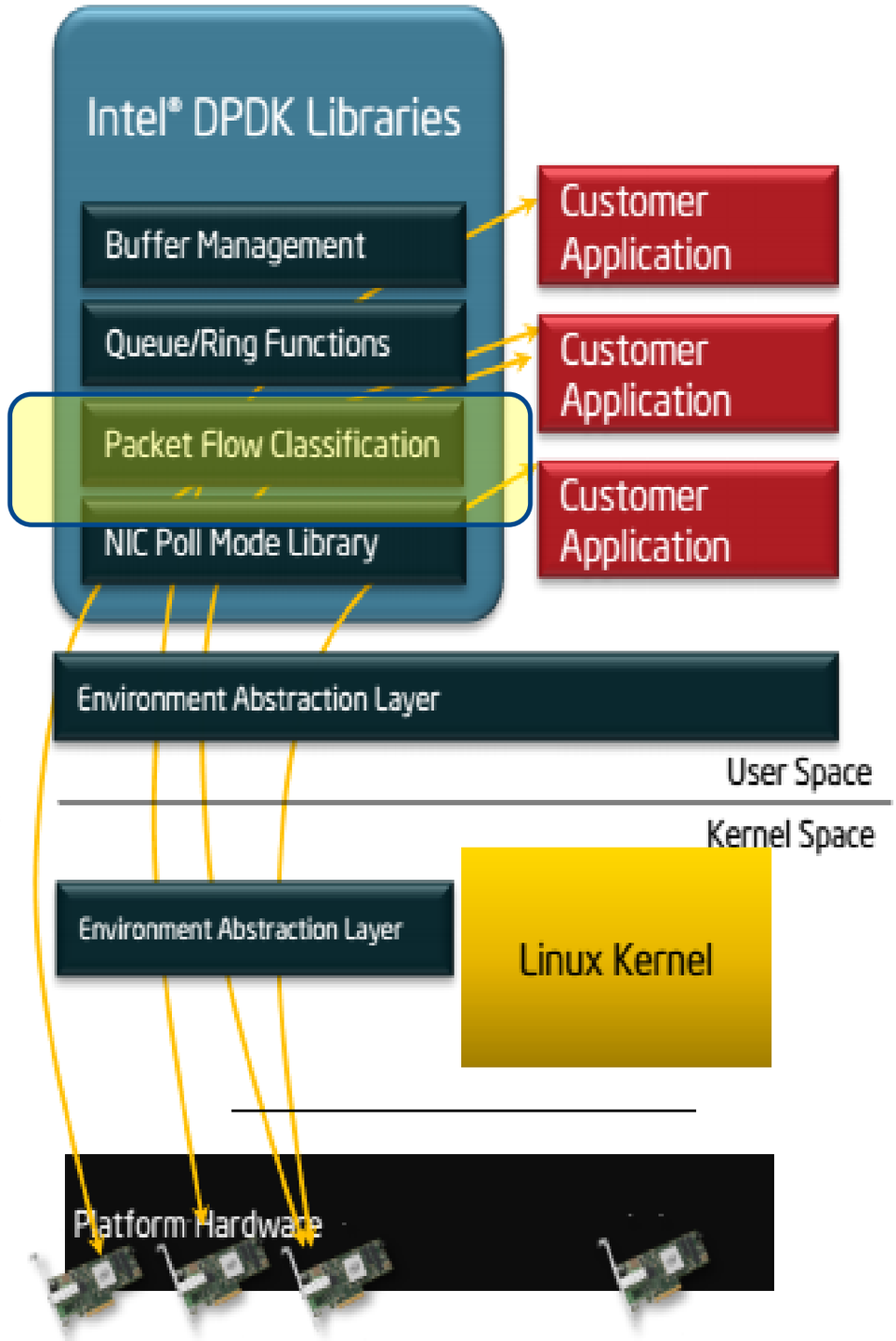
End-to-end Benchmark



Integration with Intel DPDK

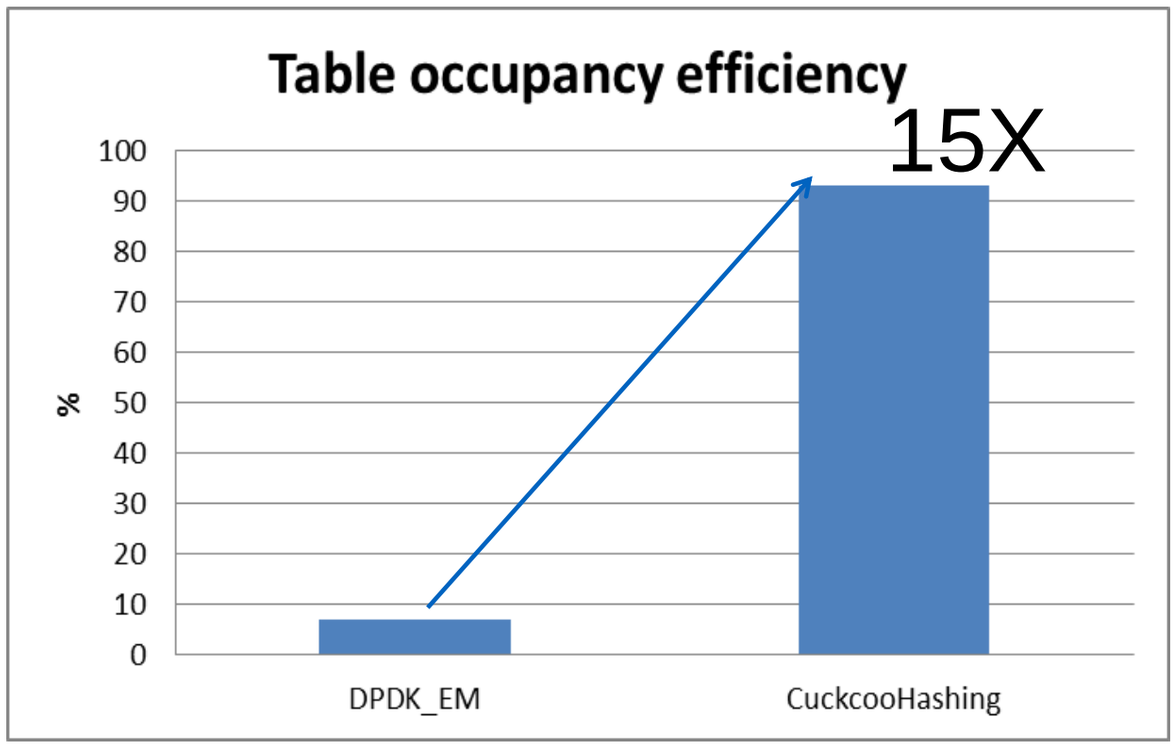
Intel DPDK

- User mode packet processing framework providing great IO performance.
- Packet flow classification library uses hashing to achieve line-rate packet switching
- It is challenging to maintain line rate with current default hashing designs when number of flows grows large.
- CuckooSwitch hash table design naturally aligns with DPDK framework.



The benefit of DPDK with CuckooHashing

- CuckooHashing provides ~15X improvement on table efficiency.



- Reduces memory bandwidth due to higher cache utilization
- Maintains throughput with large table (64M entries) comparing to the current DPDK flow classification hashing.

Integration approach

- Include CuckooHashing as one option in DPDK flow classification library.
  - Working with CSIG closely
- Better address customers' need
  - Especially Telco industry such as AT&T.

Current status

- Licensing
  - BSD licensing (thanks to ISTC ☺)
- Comply with DPDK framework
  - Unified APIs
  - Code optimization and performance evaluation
- Functionality extension
  - E.g., variable key-length hashing
- Architectural and system behavior characterization for understanding and optimization
  - Unit test and optimization for hashing functionalities
  - Architectural characteristics of hashing behavior

Conclusion

DPDK

- High performance IO framework
- Flow classification library for switching

CuckooSwitch

- Build on top of DPDK framework
- Cuckoo hashing handles large number of entries
- System optimizations that aligns with DPDK

- Integrating CuckooHashing into DPDK benefits Intel and industry
- Continue to collaborate towards future communication centric workload optimizations

backup

Intel DPDK framework

Where we can help with Cuckoo Hashing

Latency

The average latency is ~ 35 microsecond under maximum throughput