Fine-Grained Checkpointing with In-Cache-Line Logging

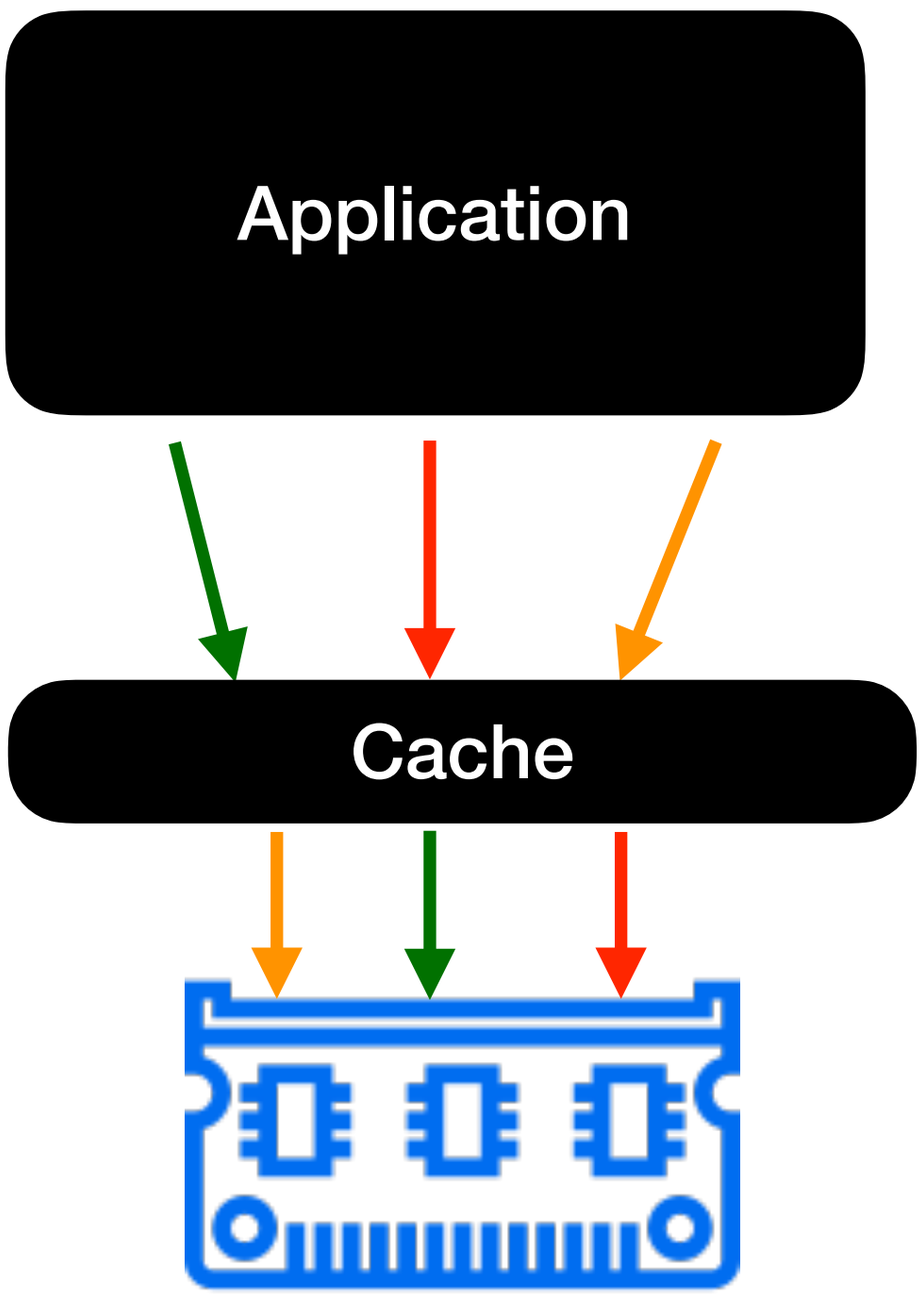
Nachshon Cohen
Hillel Avni

David T. Aksun
James R. Larus

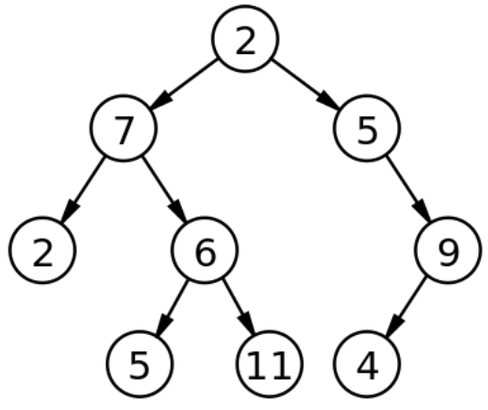
Background: Non-Volatile Memory

- DRAM-like performance, disk-like durability
  - Data is **retained** after shutting down the machine
    - Planned or **unexpected**

Challenge: Cache Reorder Writes



Data structures in NVM



Durable Data Structures

- **Challenge**: design a **durable** data structure for NVM
- **Subject to**: cache can **reorder** writes
- **And**: without **reducing performance** a lot

Existing Approaches

- Log modifications (undo log: old value, redo log: new value)
- **Explicitly force a write back (flush)** modified cache lines
  - Both log and data

Access to memory is expensive

Can we do better?

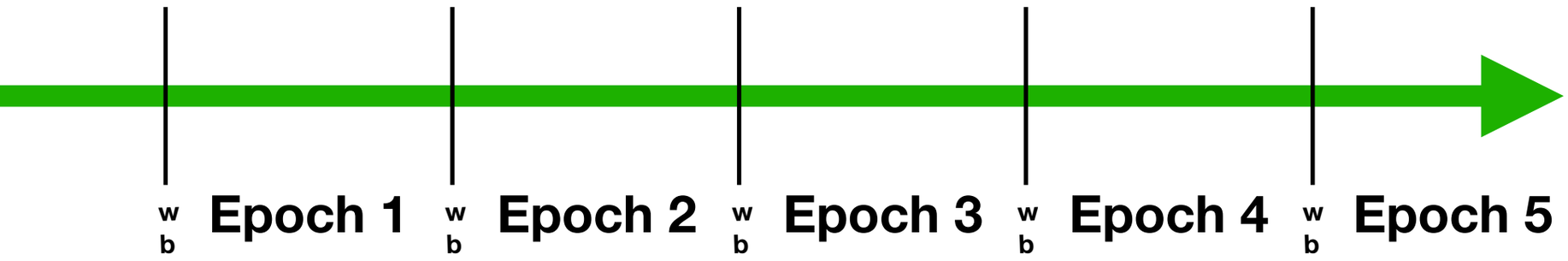
Our Approach

- Algorithm
  - Periodic persistency
  - In Cache Line Log (InCLL): our novel contribution

- **Zero** explicit writes back on the fast path of the data structure

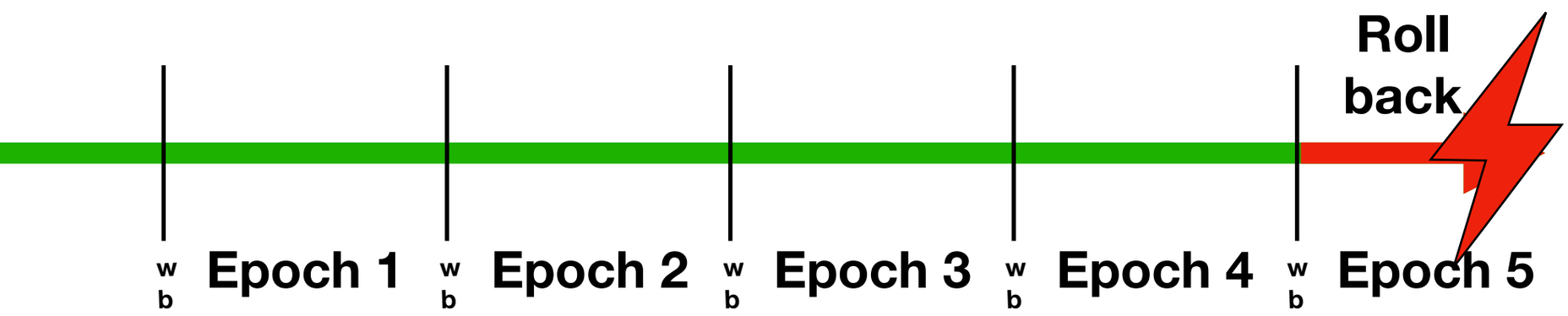
Periodic Persistency

- Flush entire cache infrequently (e.g., every 64ms)
  - E.g., x86's ***`wbinvd`*** instruction



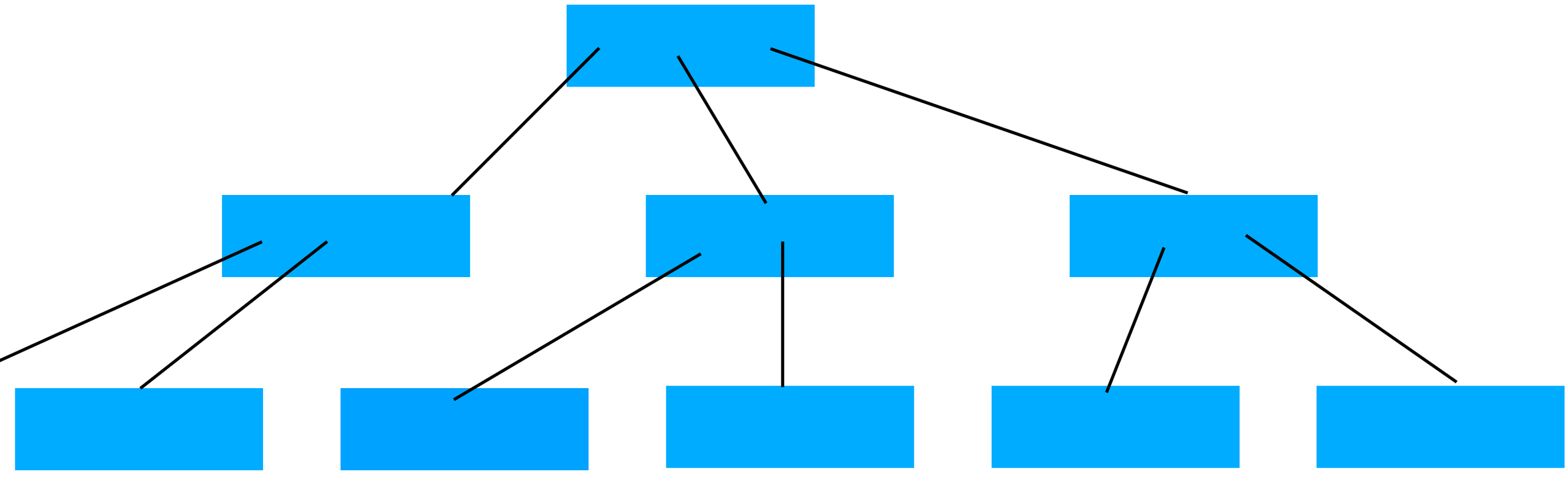
Periodic Persistency

- Flush entire cache infrequently (e.g., every 64ms)
  - E.g., x86's ***`wbinvd`*** instruction
- Return to a consistent state at the end of an epoch
  - Using **undo log**



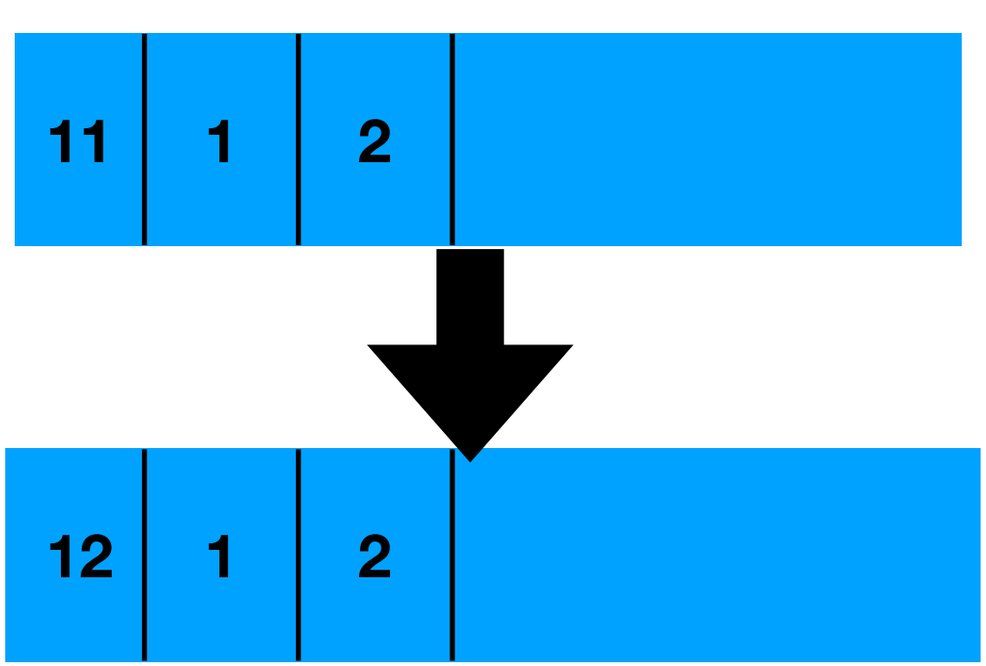
Ensuring Consistent State: B+ Tree

- put(key: 10, value: 12)



Ensuring Consistent State: B+ Tree

- put(key: 10, value: 12)
- node.value[0] = 12



Can we avoid write backs?

```
ModifyData tructureNode
1.Log <- O dValue
2.WriteBack(Log)
3.Node <- NewValue
```

Concurrency

- Modify multiple variables is hard
  - Requires a lock or TM
- Modify a single variable is easy
  - Fetch and Add
  - Compare and Swap

In Cache Line Log

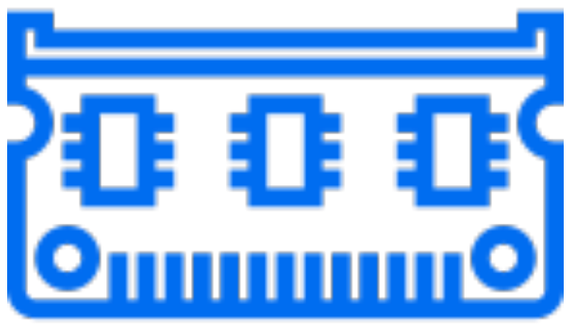
- A cache line is evicted to memory atomically
- Embed the log inside **the same cache line** as modified node
- No explicit write-back

In Cache Line Log

- A cache line is evicted to memory atomically
- Embed the log inside **the same cache line** as modified node
- No explicit write-back

How In Cache Line Log Works

Cache



No write back

Implicit write back

How In Cache Line Log Works

In Cache Line Log

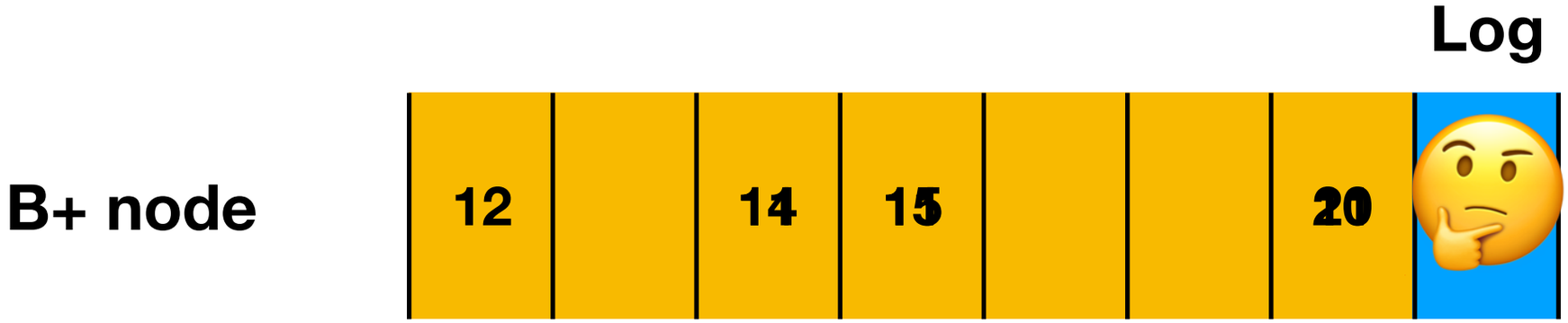
enables recovery

without explicit write backs

11

In Cache Line Log: Drawback

- Capacity is very limited

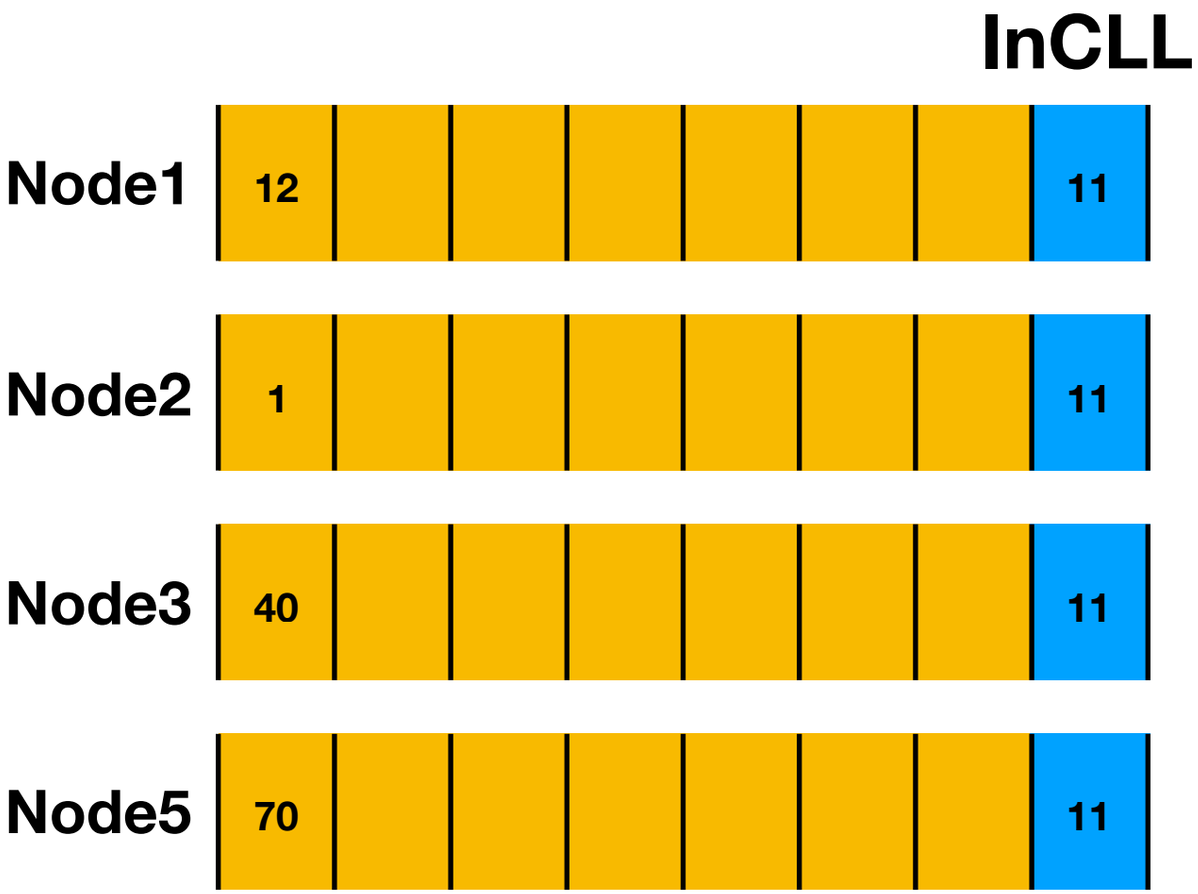


External Undo Log at Node Granularity

- Node is modified two times
    - Probably it will be modified again during the epoch
- Log entire node, **explicit write back**
    - Subsequent modifications (during same epoch) **do not require logging**

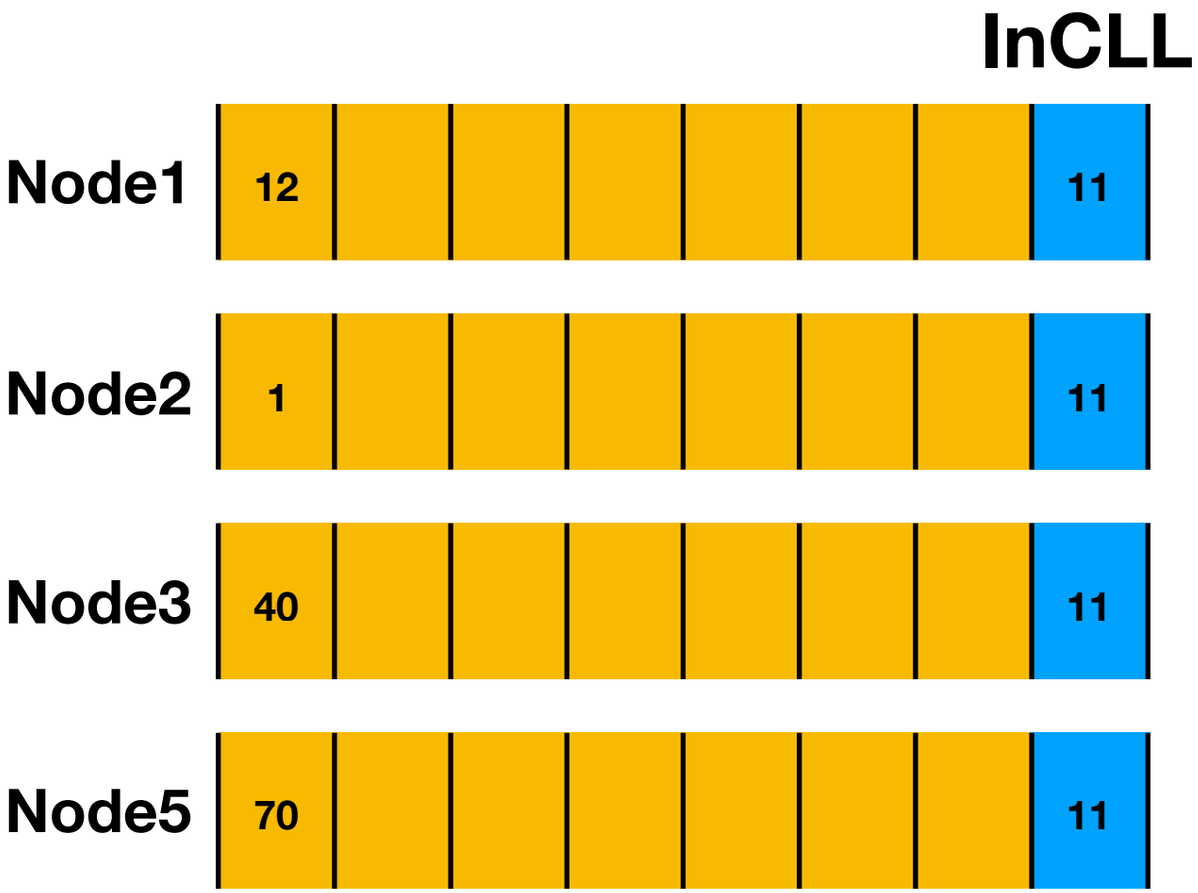
External Undo Log + In Cache Line Log

- First modification: use InCLL



External Undo Log + In Cache Line Log

- First modification: use InCLL
- 2+ modifications: use external log



External log

On average, 1/#modifications explicit write backs

External Undo Log + In Cache Line Log

- First modification: use InCLL
- 2+ modifications: use external log

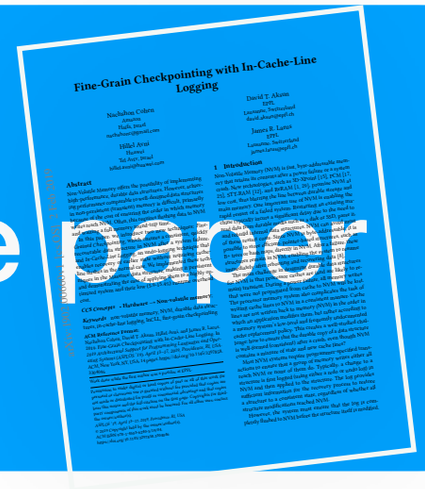
Effective when
modifications are sparse
- Data structure is large
- Key distribution is uniform

Effective when
modifications are dense
- Splitting a B+ node
- Modify a range of values

External Undo Log + In Cache Line Log

- Best case:
  - A single popular key
  - Key distribution is skewed
- Worst case:
  - Two keys modified exactly once
  - One explicit write back per two modifications

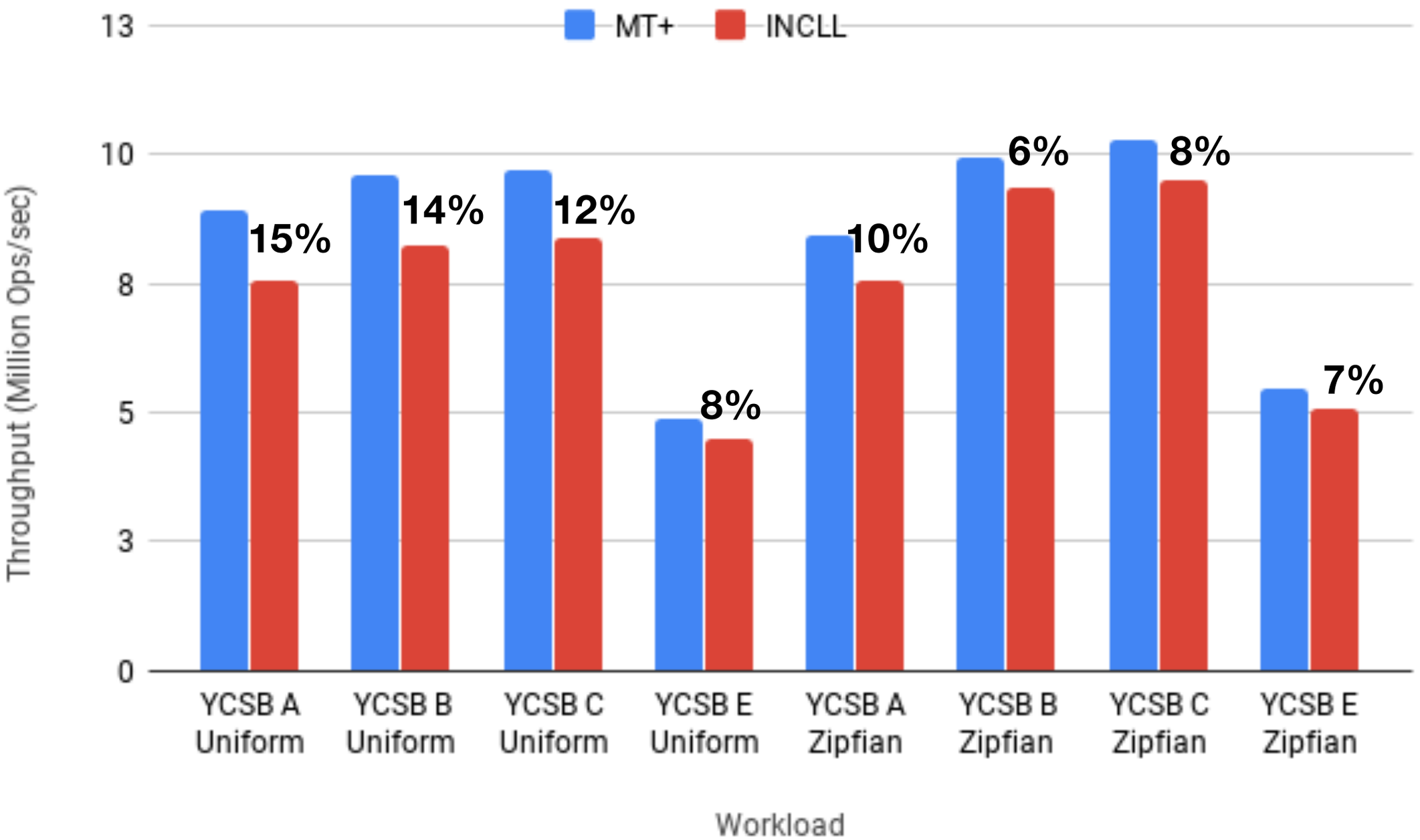
Many additional details, see PDL



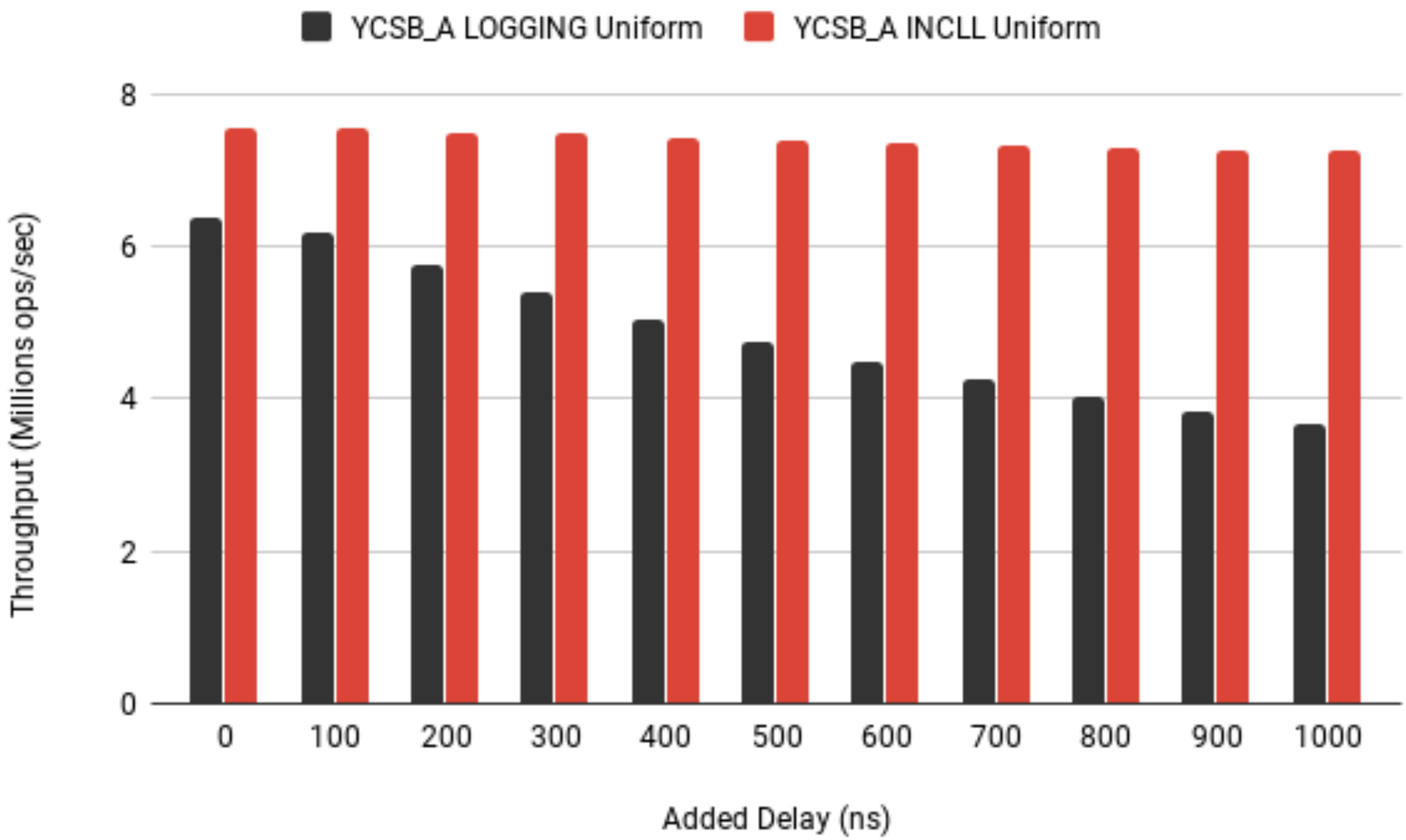
Implementation and Evaluation

- Incorporated into MassTree [Mao, Kohler, Morris, EuroSys'12]
  - B+ Tree/Trie with excellent performance
- Also made MassTree's allocator durable with InCLL
  - Avoid **dangling pointers** and **durable memory leaks**
- Workloads
  - Ycsb A (50% writes), B (5% writes), C (0% writes), E (scans)
  - Key distribution: Uniform and Zipfian

Performance vs. Workload



Performance vs. NVM Latency



Conclusion

- **Explicit write backs (cache line flushes)** are expensive
- Use **In Cache Line Log**
    - Place log inside cache line and avoid explicit write backs
- Plus: Periodic persistence, External log for second modification
- **Durability** with **small overhead**

Conclusion

- **Explicit write backs** are expensive
- Use **In Cache Line Log**
    - Place log inside cache line and avoid explicit write backs
- Plus: Periodic persistence, External log f
  modification
- **Durability** with **small overhead**