Motor: Enabling Multi-Versioning for Distributed Transactions on Disaggregated Memory

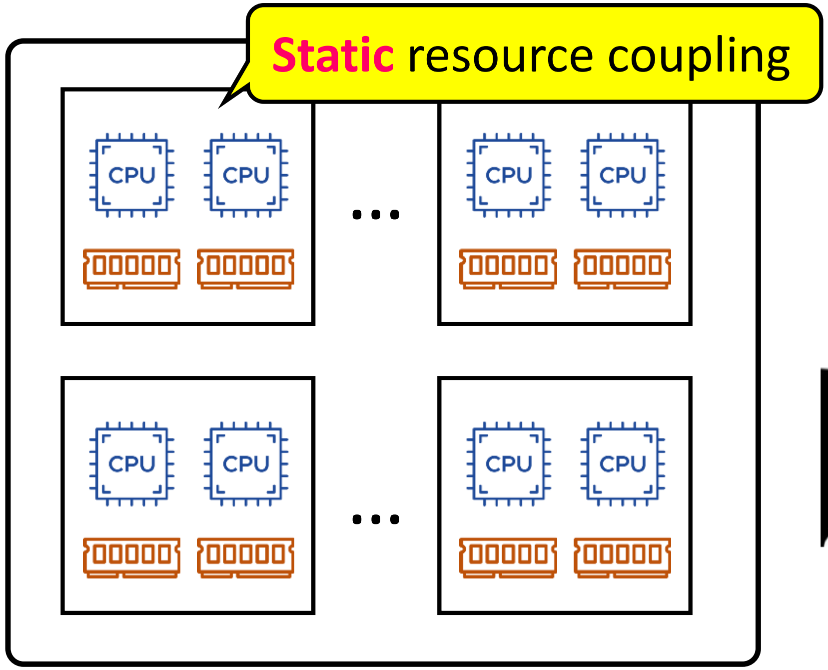
Ming Zhang, Yu Hua, Zhijun Yang

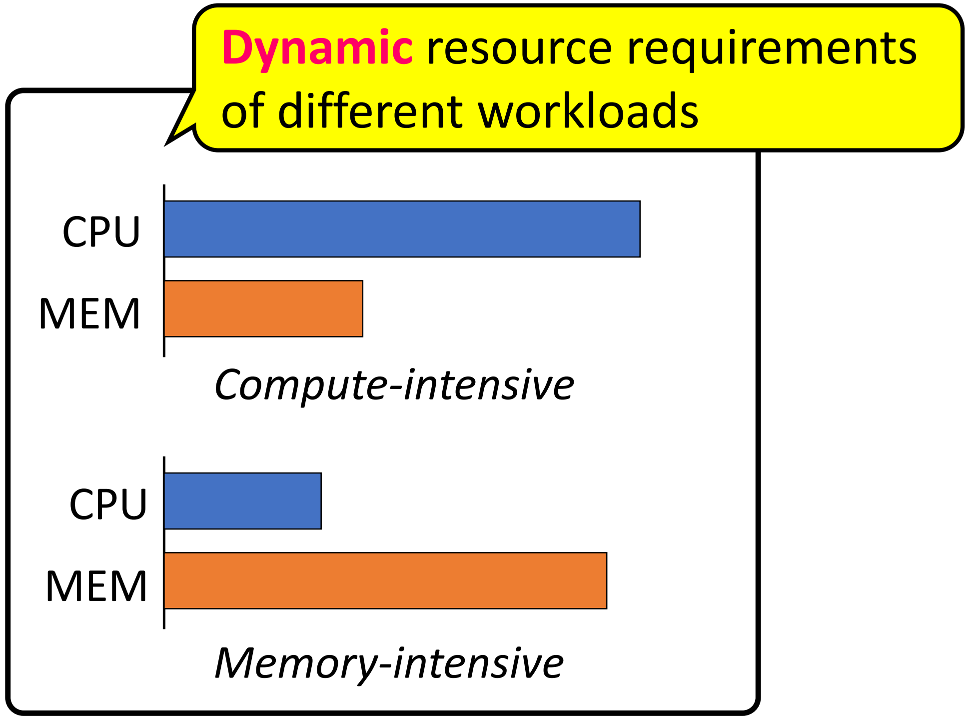
Huazhong University of Science and Technology, China

18th USENIX Symposium on Operating Systems Design and Implementation (**OSDI**), 2024

Low Memory Utilization in the Cloud

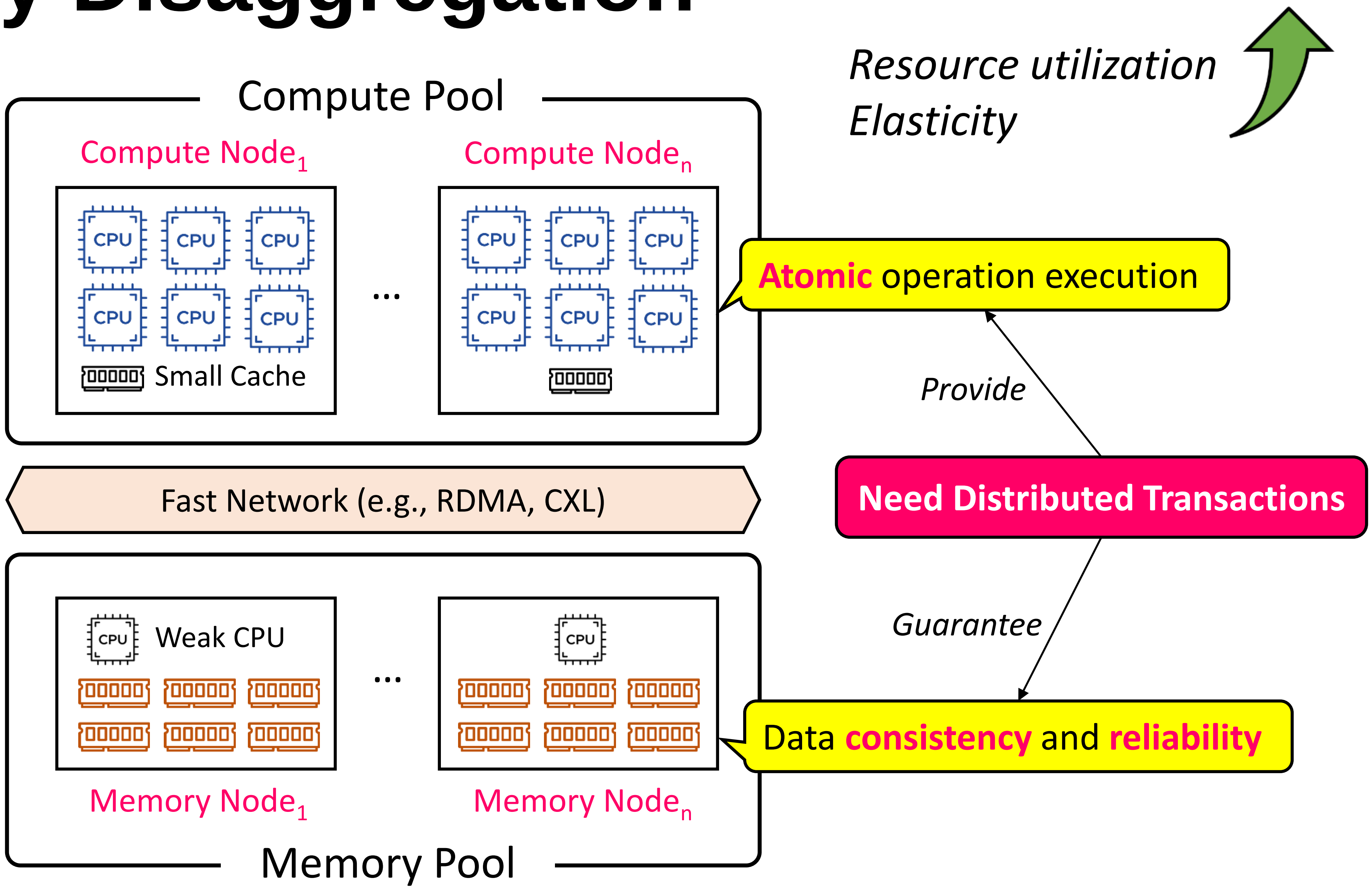
- Below ~60% [1-4]
- One major reason: monolithic server



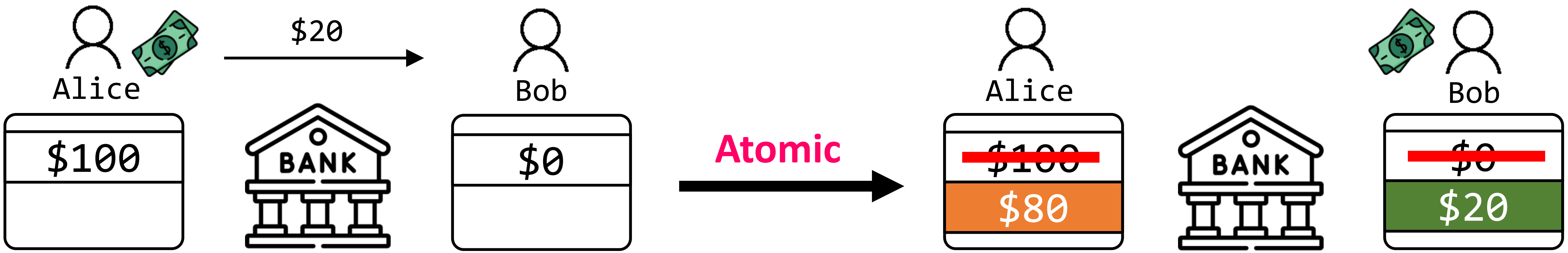


[1] MemTrade@SIGMETRICS'23, Borg@EuroSys'20, LegoOS@OSDI'18
[2] Google Production Cluster Trace. https://github.com/google/cluster-data
[3] Alibaba Production Cluster Trace. https://github.com/alibaba/clusterdata
[4] Snowflake Dataset. https://github.com/resource-disaggregation/snowset

Memory Disaggregation



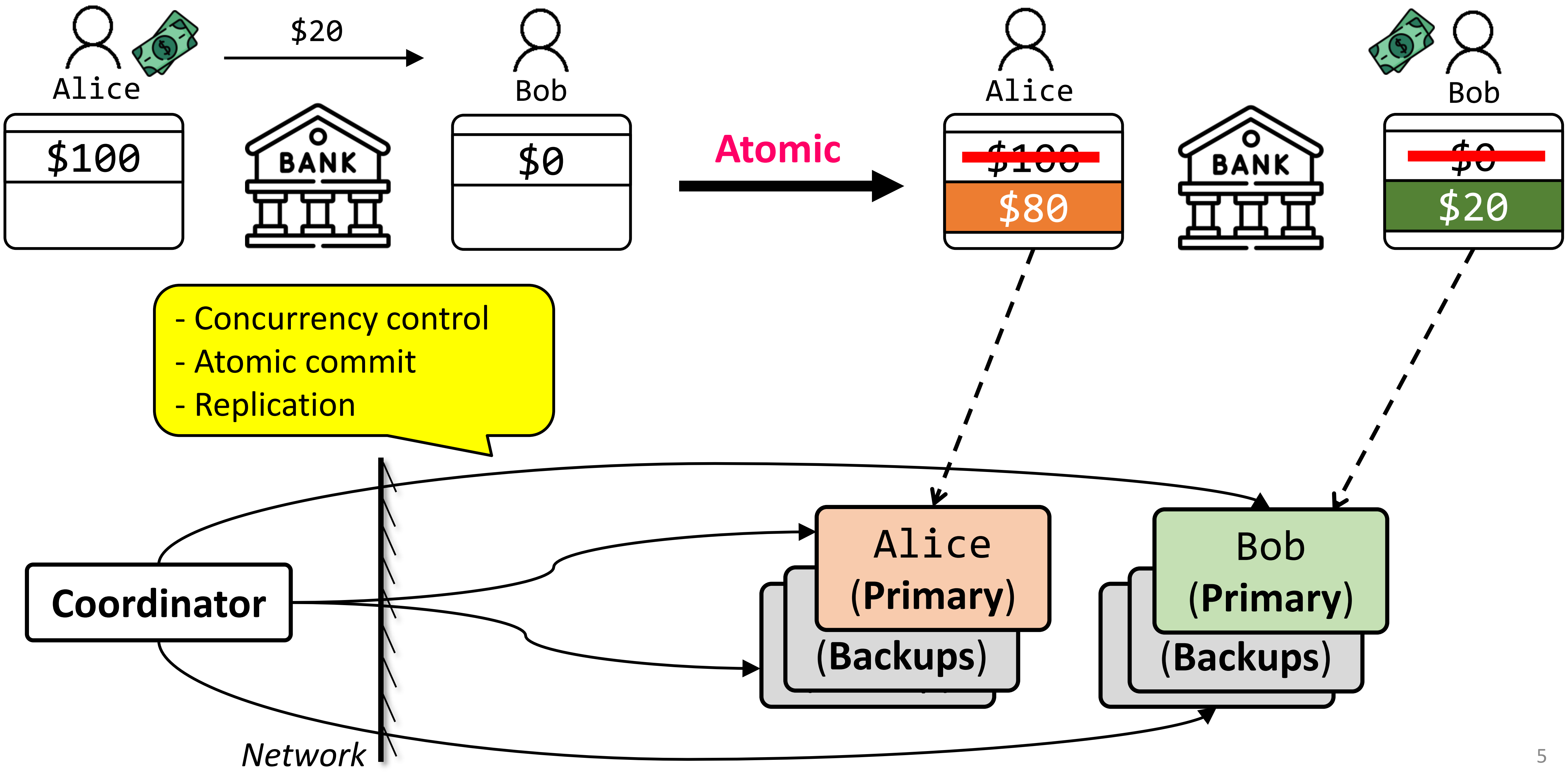
Transaction



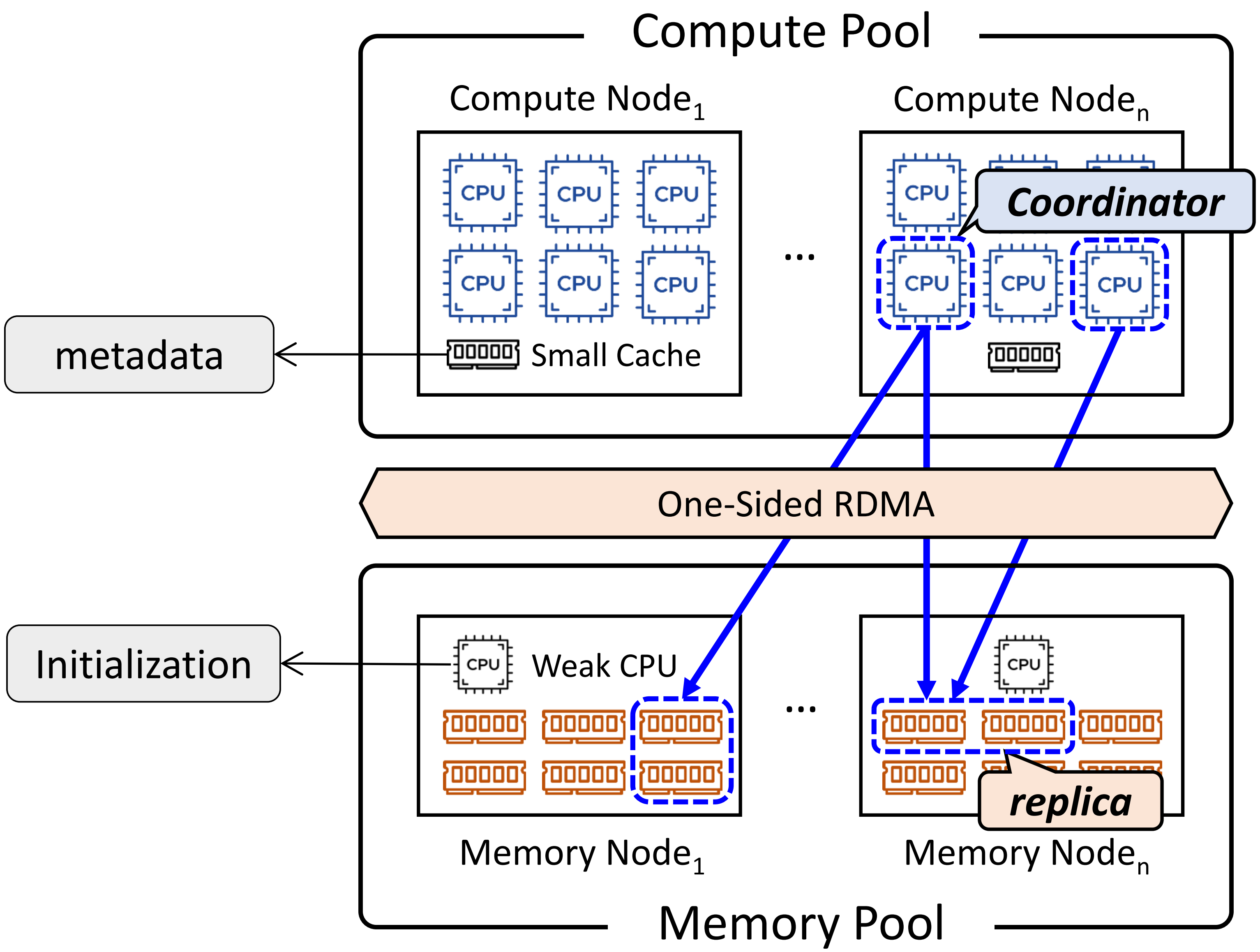

```
Txn begin
   Alice: $100 -> $80
     Bob:    $0 -> $20
Txn end
```

Transaction

Distributed Transaction



Execution Model on Disaggregated Memory

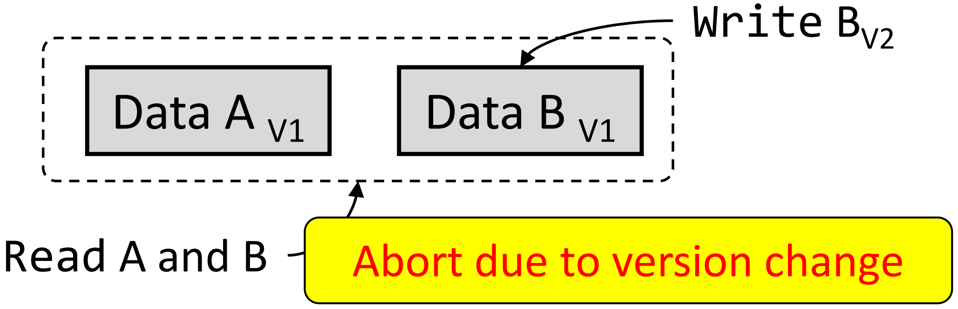


State-of-The-Art Studies

- Single-versioning txn system
  - Based on disaggregated memory[1]

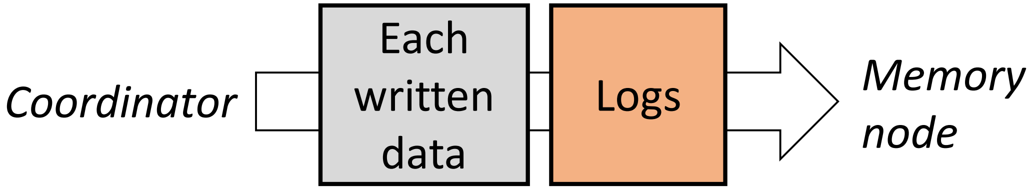
☹ Write interrupts read

  - Hamper throughput



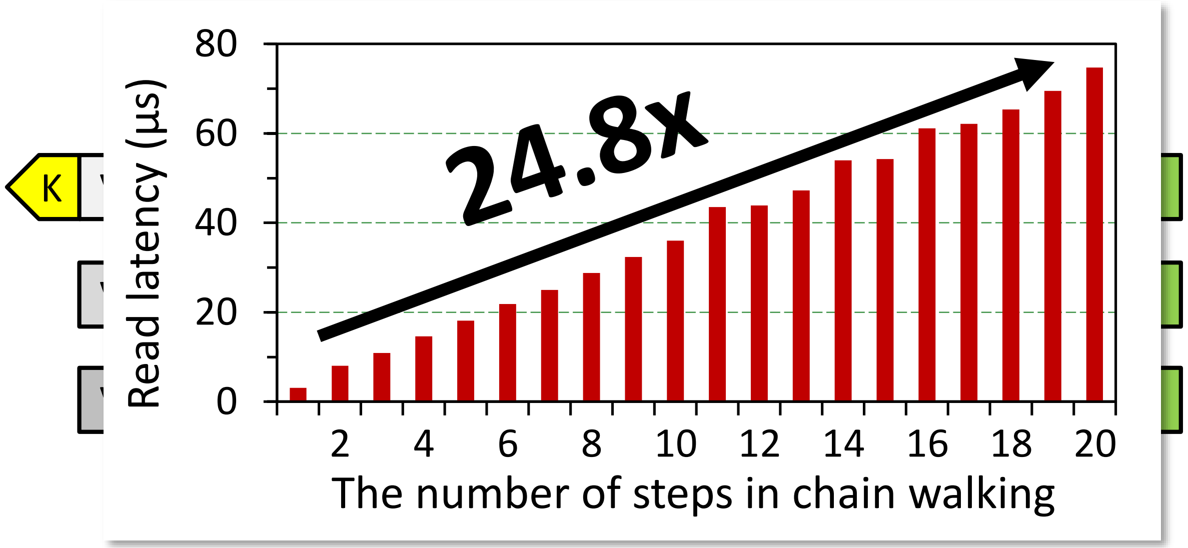
☹ Substantial write-ahead logs

  - Consume network resources



- Multi-versioning txn systems
  - Based on monolithic architecture

☹ Inefficient linked version chain



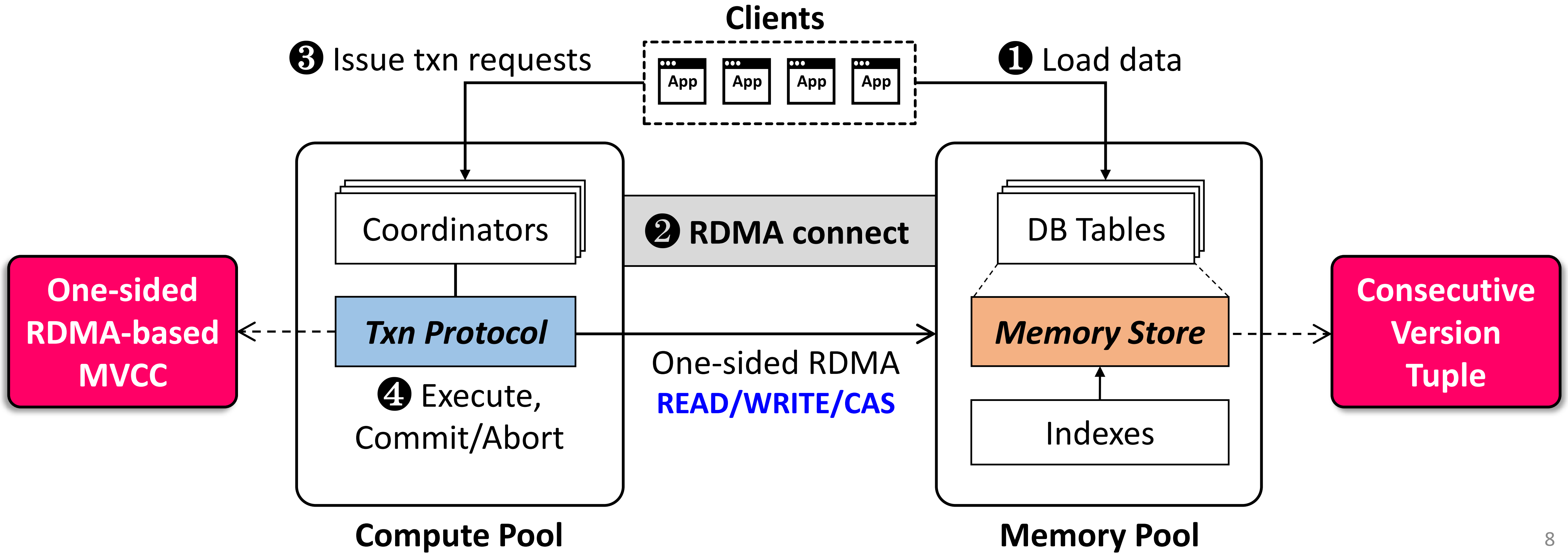
☹ Incompatible transaction protocol

  - Frequent CPU involvement on each data node: timestamp calculation[2], locking[6], validation[7]

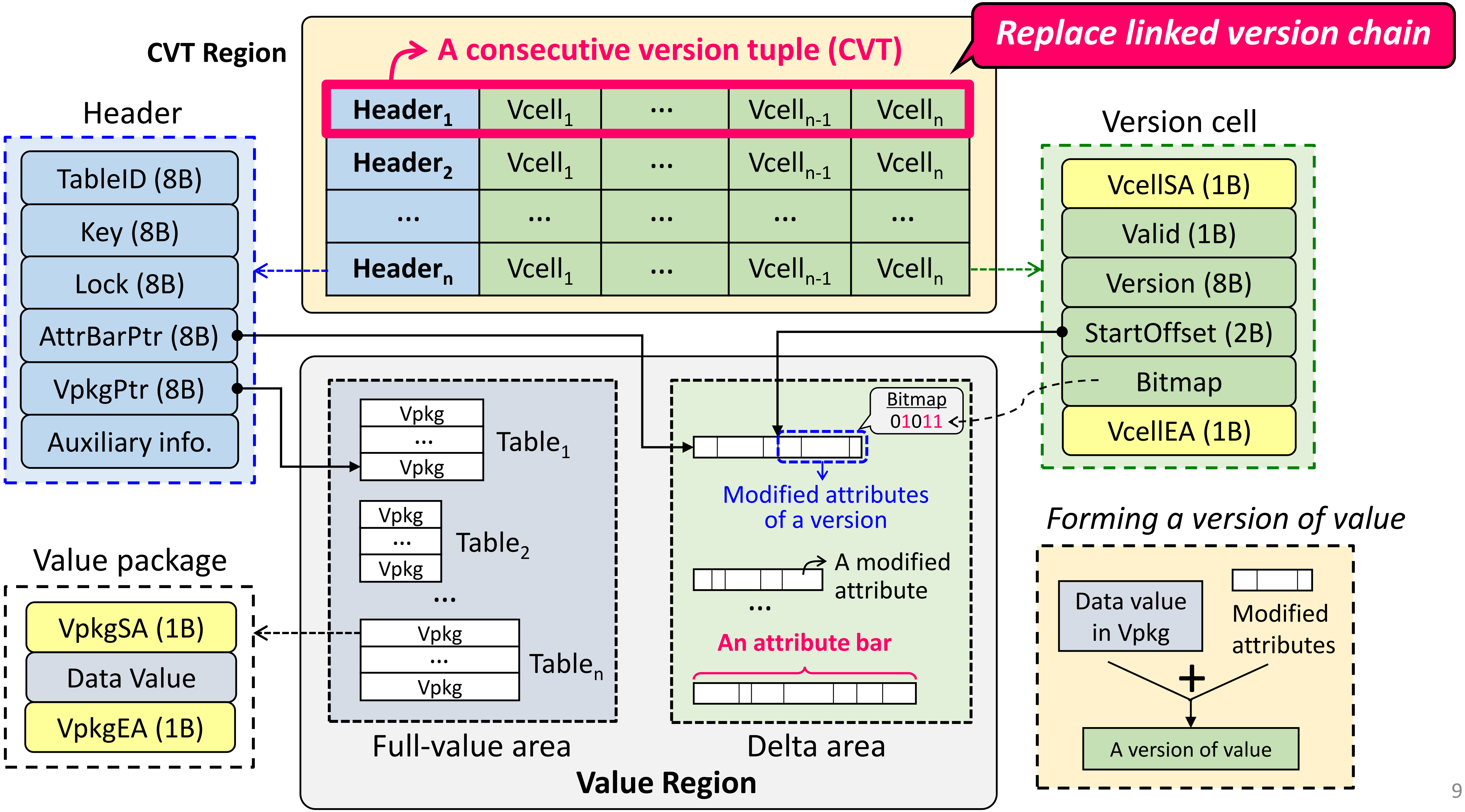
[1] FORD@FAST'22 [2] DST@NSDI'21 [3] Postgres [4] Hekaton@SIGMOD'13 [5] Aurogon@FAST'22
[6] FaRMv2@SIGMOD'19 [7] Neumann et al.@SIGMOD'15 [8] MySQL [9] NAM-DB@VLDB'17

Motor: Overview

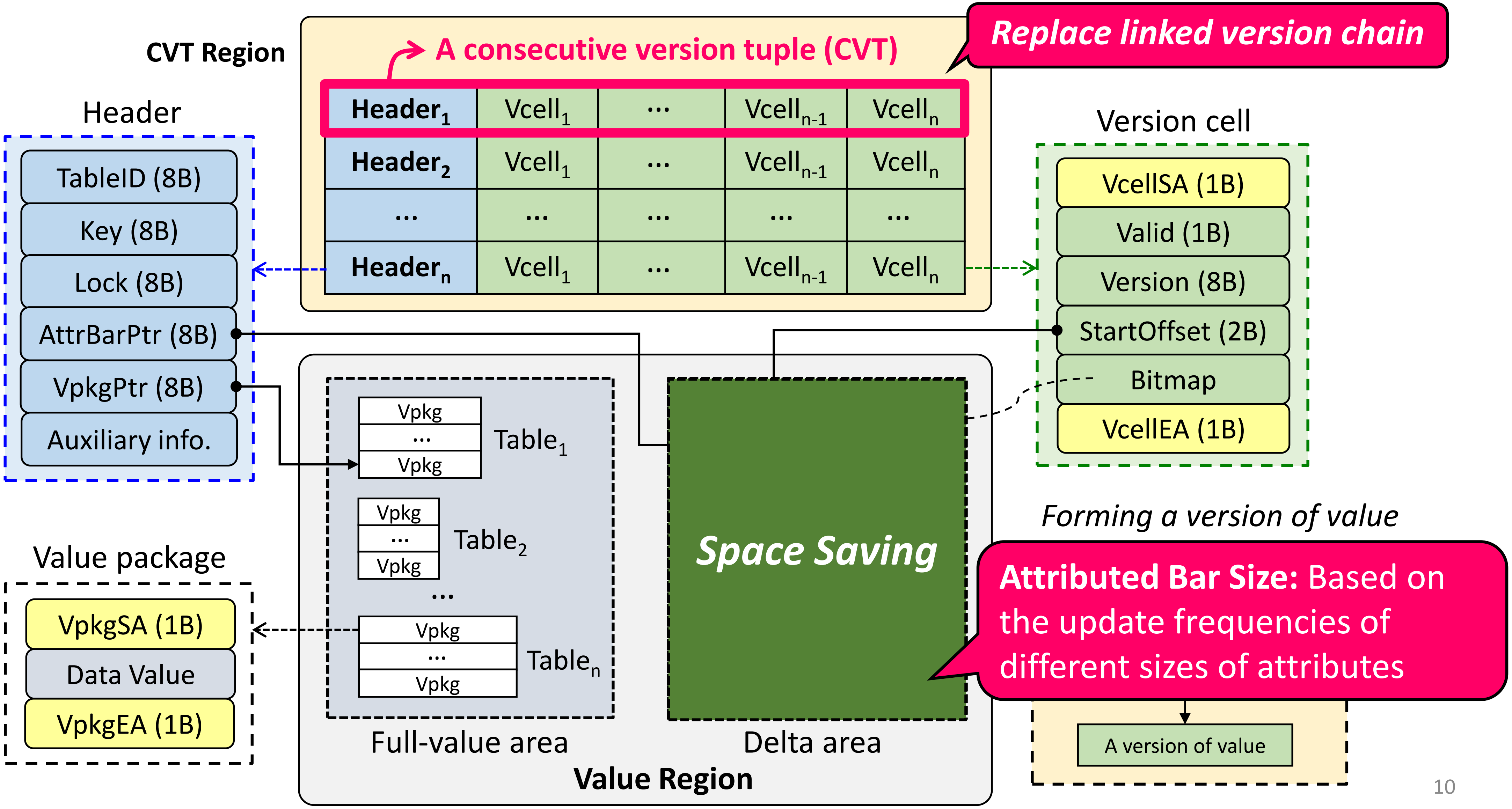
- Holistic new design working in harmony
  - *Version structure* in memory pool
  - *MVCC protocol* in compute pool



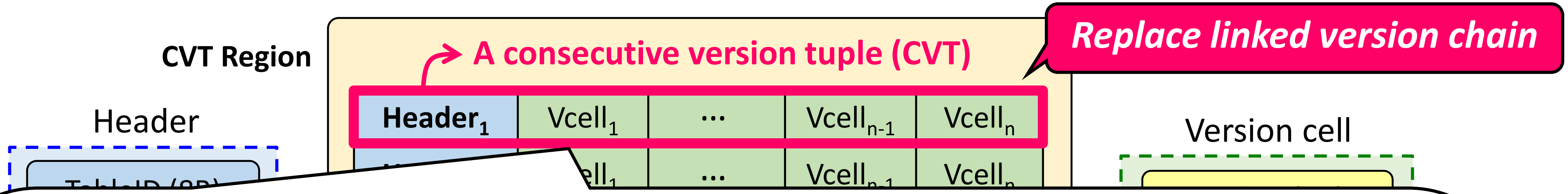
Consecutive Version Tuple



Consecutive Version Tuple

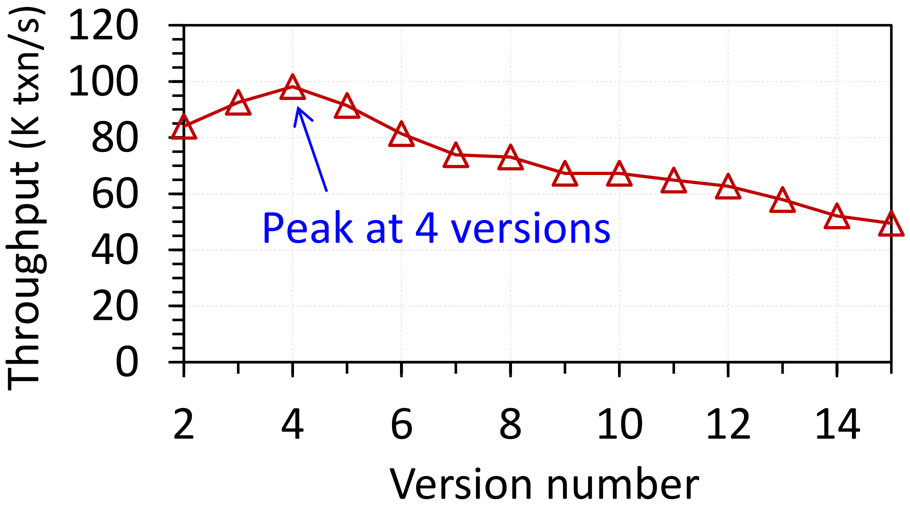


Consecutive Version Tuple

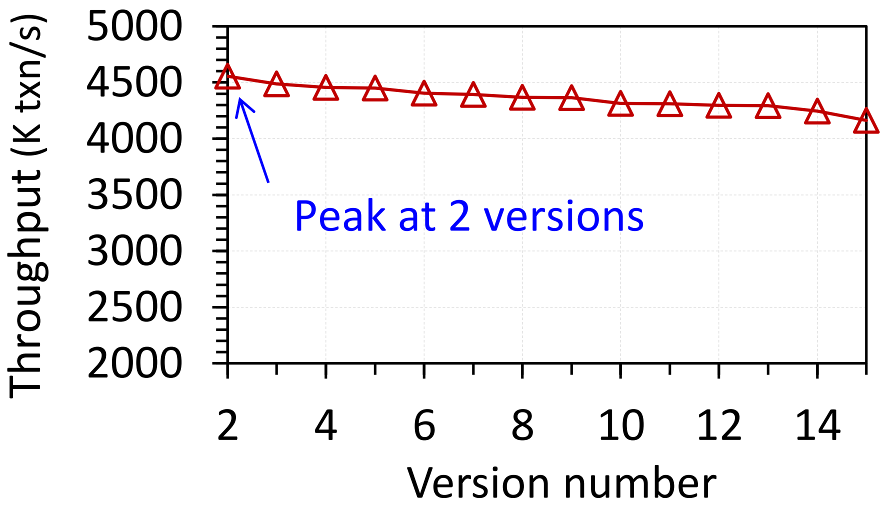


Number of Versions: depending on workload characteristics

- Read-write contention
- Number of accessed records



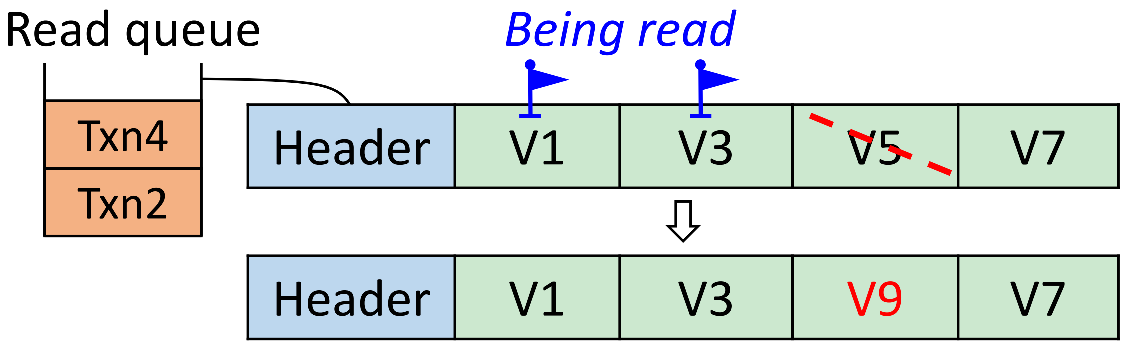
TPCC (high contention, long txns)



TATP (low contention, short txns)

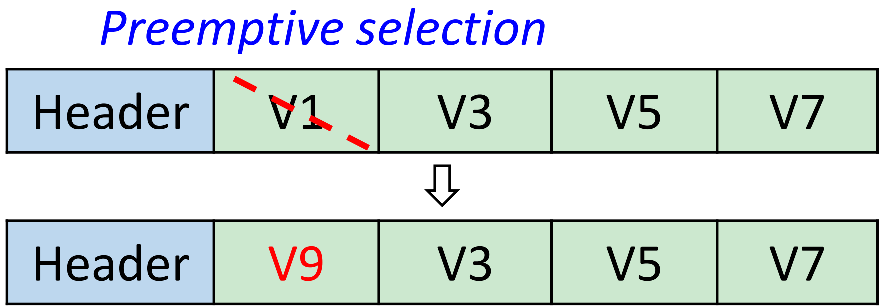
Coordinator-Active Garbage Collection

- A CVT runs out of space – GC required
- Prior systems track transaction states[1-2]
  - CPU in memory nodes is too weak to frequently track



Skip the versions being read

High overhead for compute nodes to maintain states



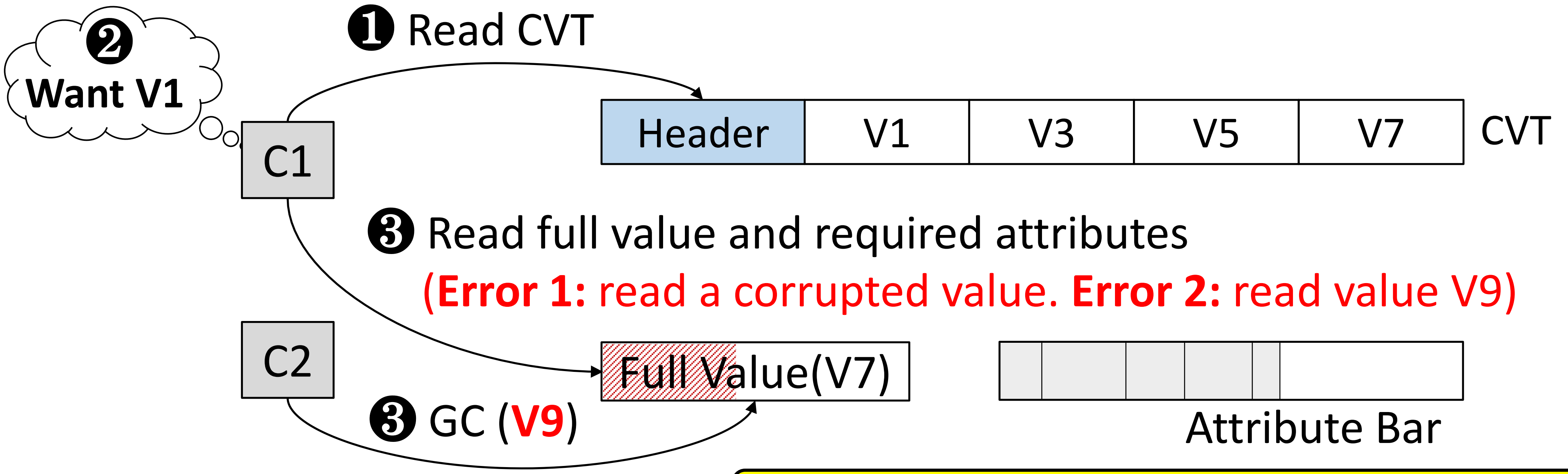
Overwrite the oldest version

Simple, no tracking

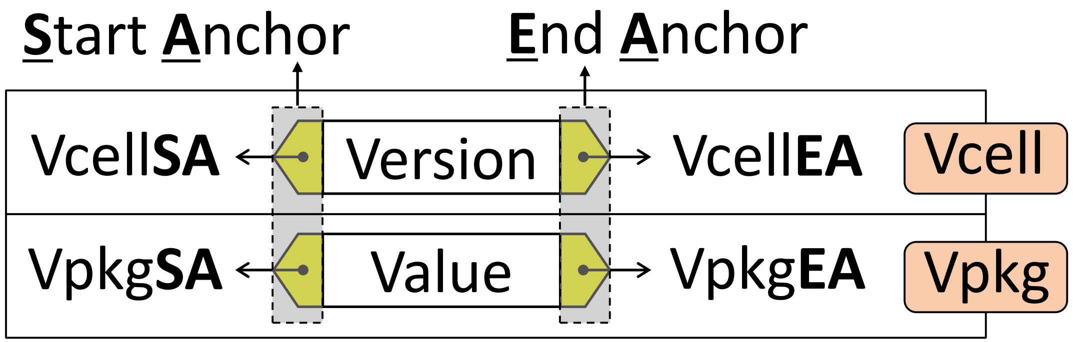
Low abort rate with fast RDMA

[1] Steam@VLDB'19 [2] FaRMv2@SIGMOD'19

Anchor-Assisted Read

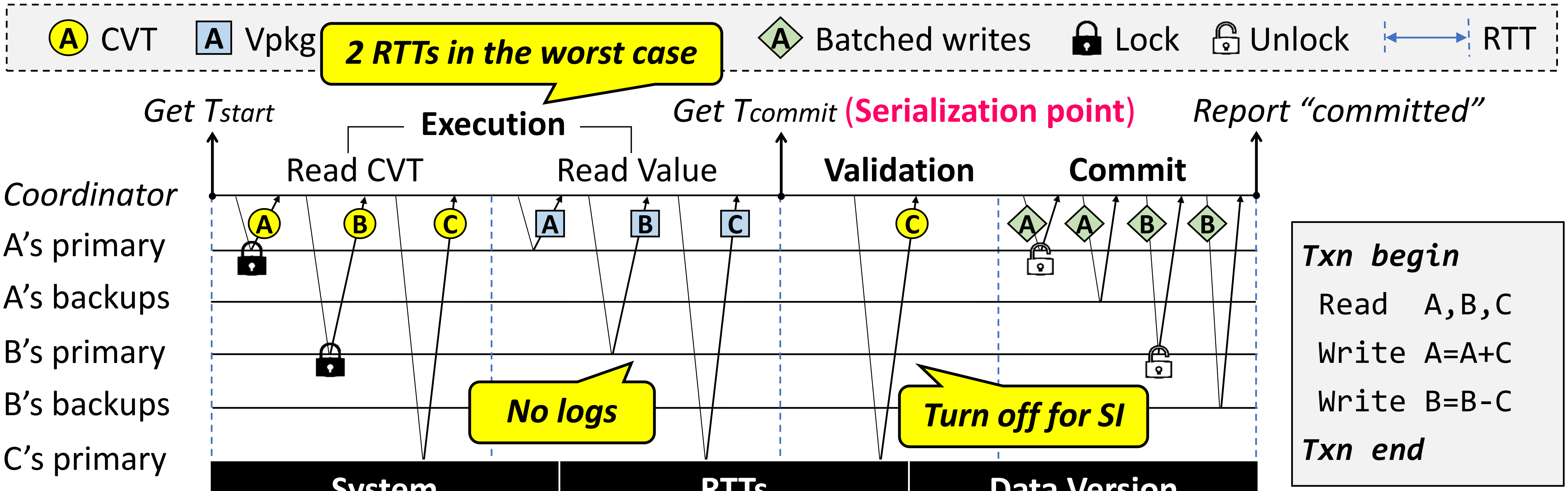


Root cause: version and value are not **“atomically”** read



- **Writer:** Vpkg → attributes → Vcell
- **Reader:** check “all anchors are equal”
  VcellSA = VcellEA = VpkgSA = VpkgEA ✓

One-Sided RDMA-Based MVCC



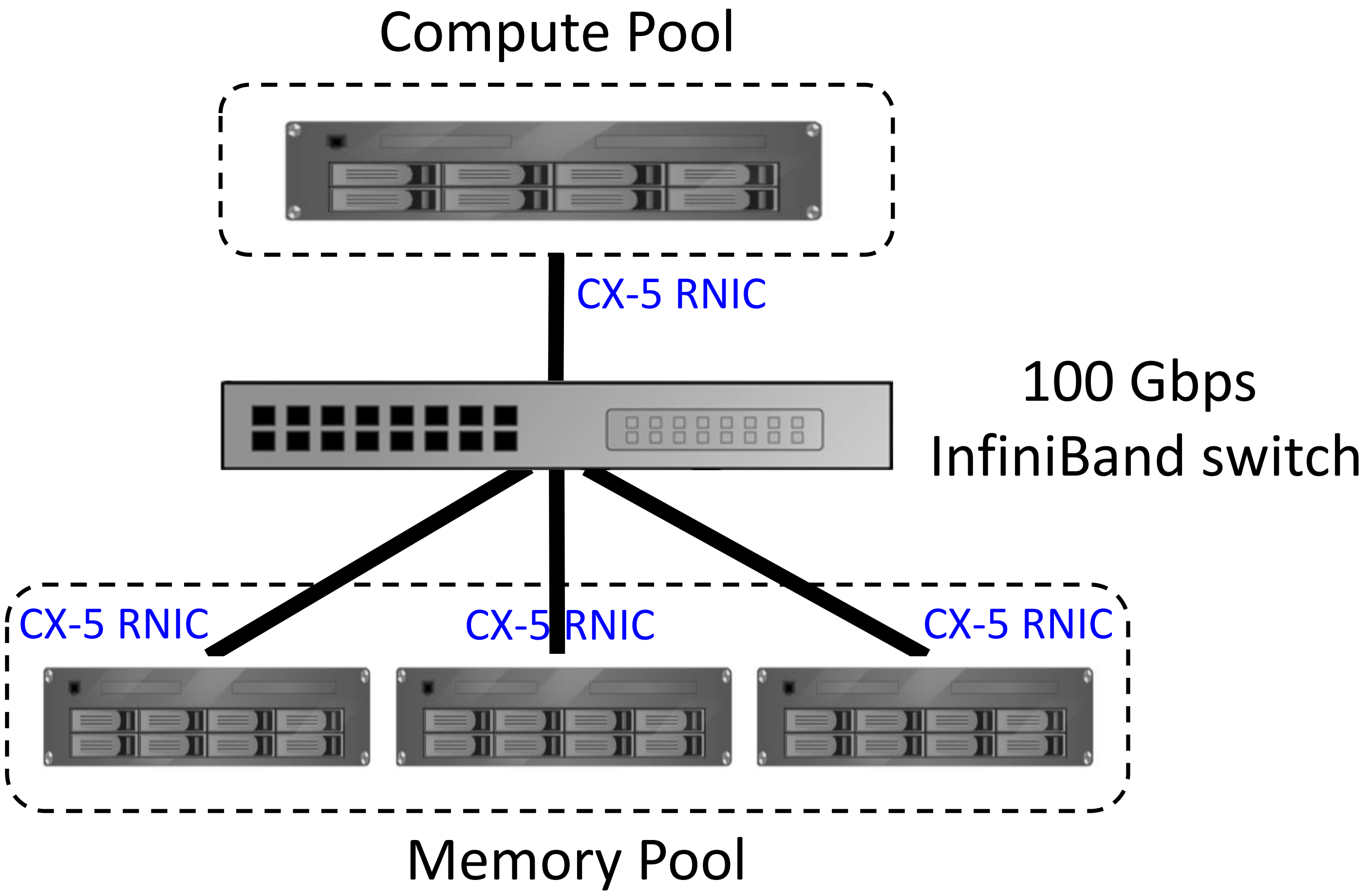
System	RTTs	Data Version
DrTM+H@OSDI'18	4	Single Version
FORD@FAST'22	3	Single Version
FaRMv2@SIGMOD'19	≥ 5	Multiple Version
Motor	**≤ 4**	**Multiple Version**

Deliver MV benefits without long RTTs

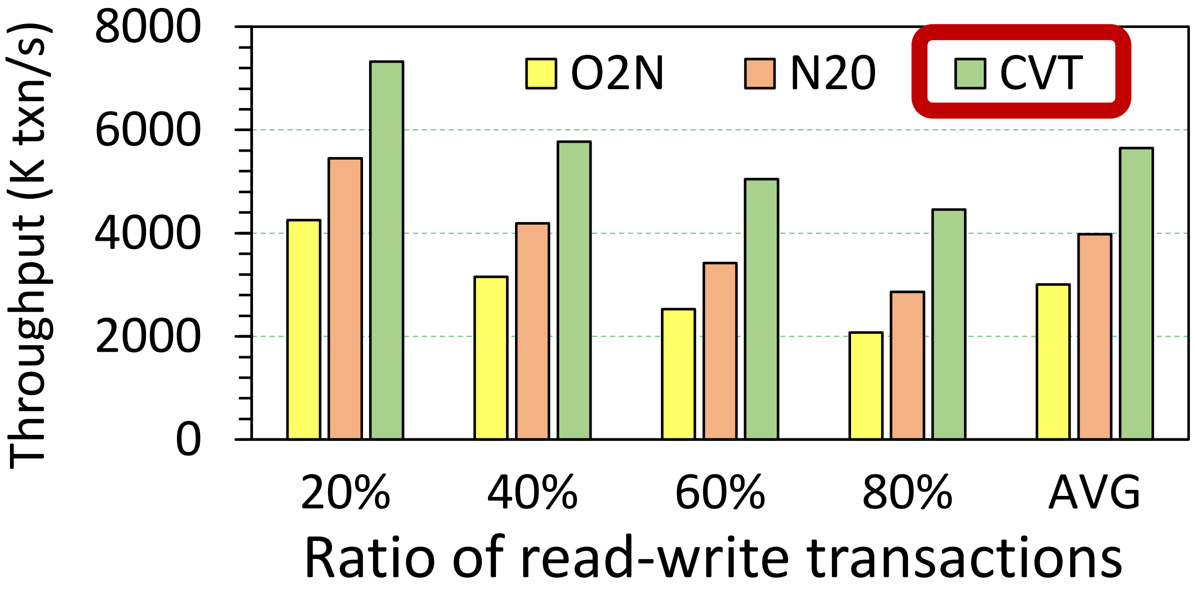
* Tstart/Tcommit stands for start/commit timestamp

Evaluation

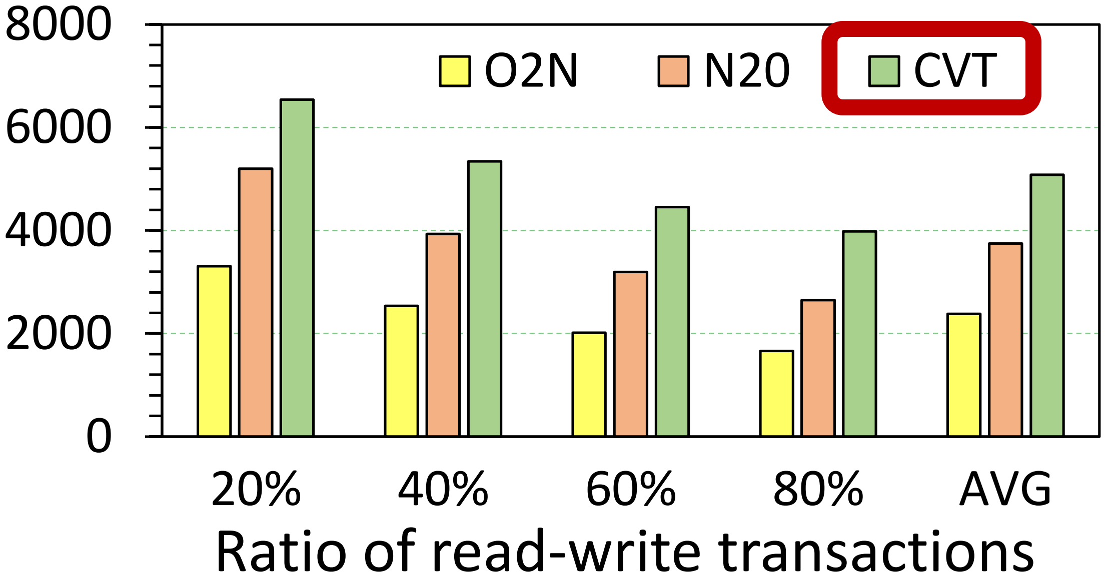
- **Benchmarks**
  - KV store
    - 8B key + 40B value
    - Skewed (skewness tunable)
  - TATP
    - RO/RW: 80%/20%, max 48B
  - SmallBank
    - RO/RW: 15%/85%, 16B
  - TPCC
    - RO/RW: 8%/92%, max 672B
- **Comparisons**
  - FaRMv2@SIGMOD’19
  - FORD@FAST’22



Performance of Version Structures



Version number = 3, skewness = 0.7



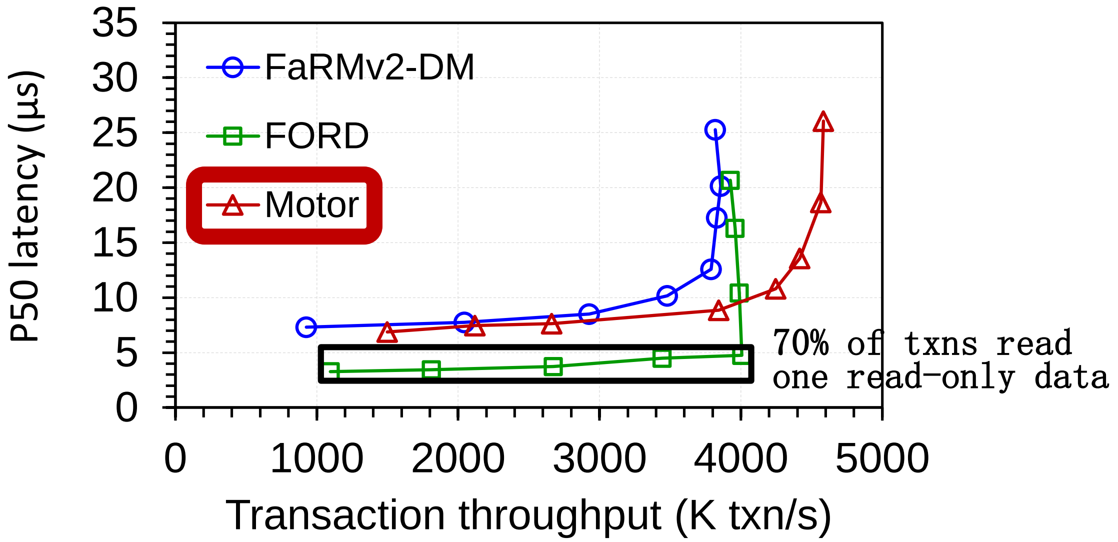
Version number = 4, skewness = 0.99

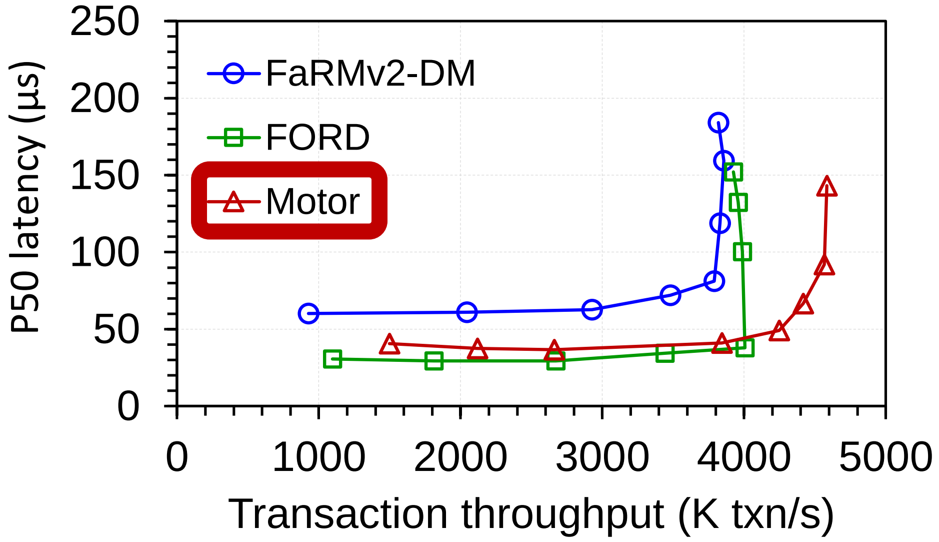
CVT improves throughput by

- 1.7 - 2.4x over O2N
- 1.3 - 1.6x over

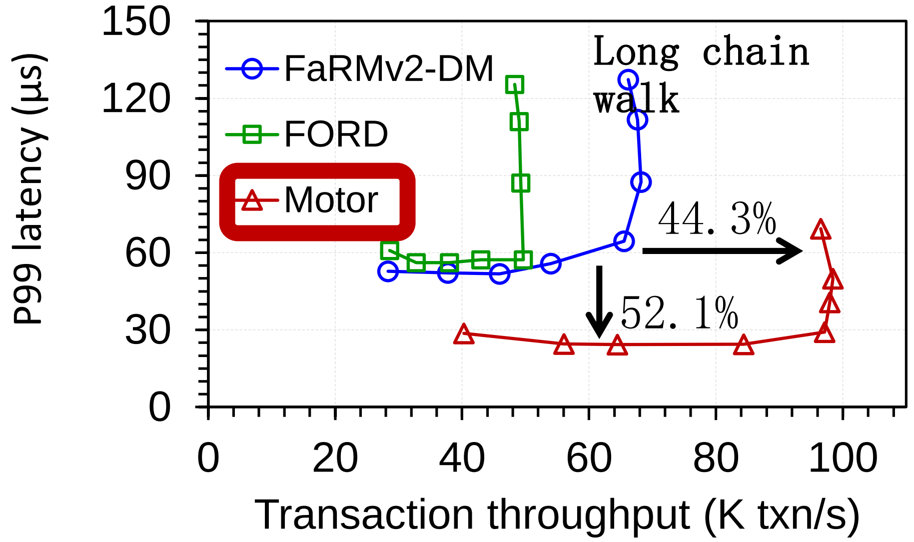
End-to-End Performance

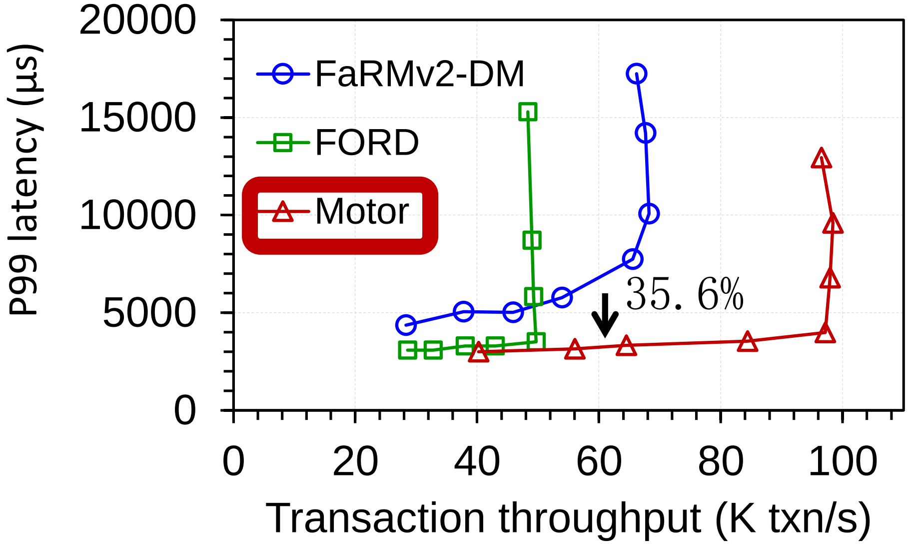
TATP (read-intensive)



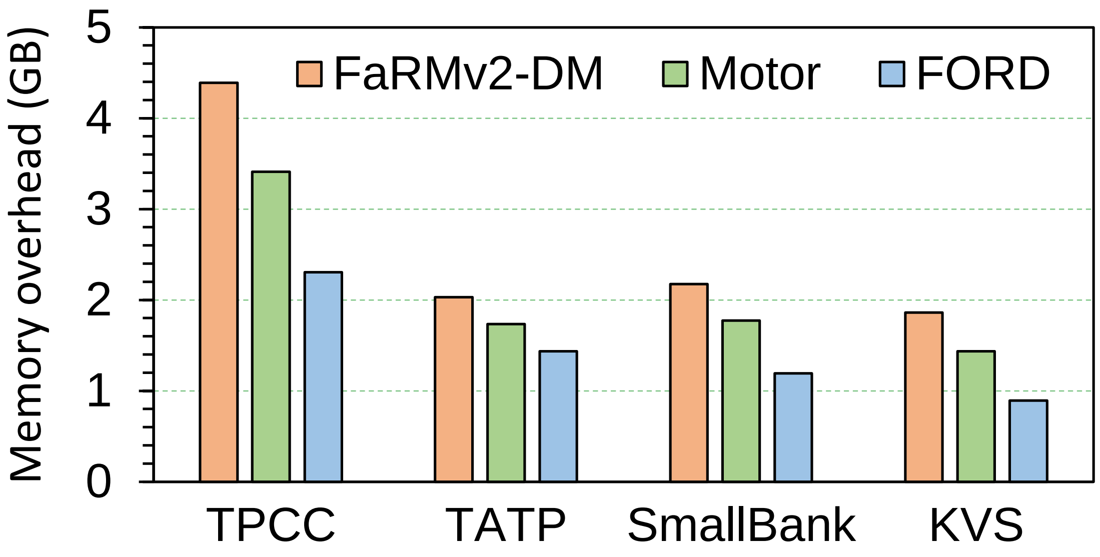


TPCC (write-intensive)

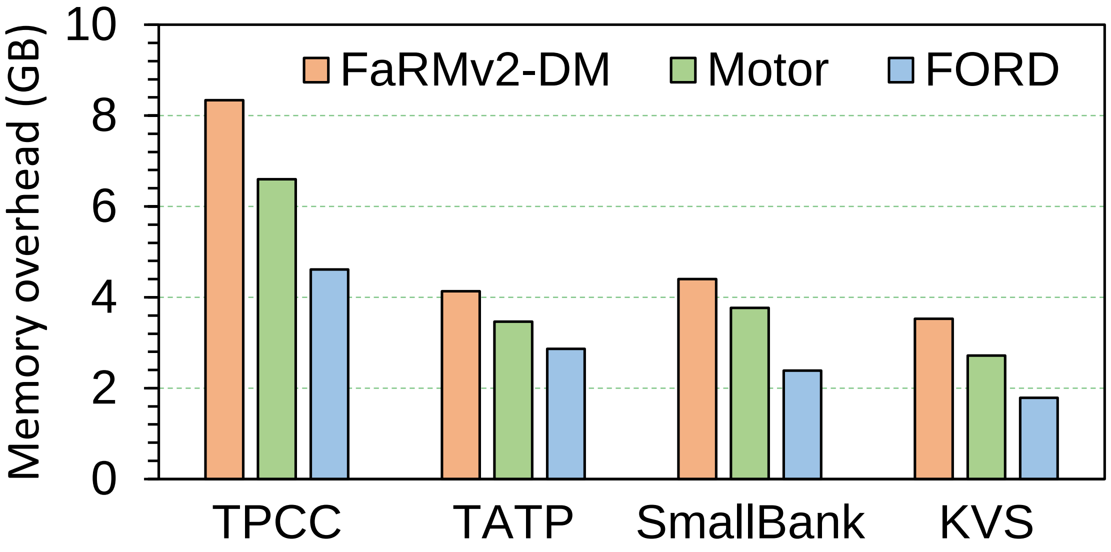




Memory Overhead



Memory consumption on workload scale1*



Memory consumption on workload scale2*

Motor's full-delta value storage design

- VS. FORD (single-versioning): 1.45x, not 4x, on TPCC with 4 versions
- VS. FaRMv2 (multi-versioning): save 14.6%~22.8% of memory space
- Stable advantages on larger scale of workloads

* Workload scale1: TPCC 24 warehouses; TATP 2M subscribers; SmallBank 10M accounts; KVS 10M objects

* Workload scale2: TPCC 48 warehouses; TATP 4M subscribers; SmallBank 20M accounts; KVS 20M objects

Conclusion

- Distributed transaction is a key pillar for disaggregated memory
- Limitations of existing systems
  - Single-versioning: limited concurrency, high logging overhead
  - Multi-versioning: inefficient linked chain, incompatible txn protocol
- ***Motor:*** a holistic multi-versioning design
  - Memory pool: *consecutive version tuple*
  - Compute pool: *one-sided RDMA-based MVCC*
- **Benefits**

High Throughput | **Low Latency** | **Low Memory Overhead**

https://github.com/minghust/motor

Thank you! Q&A