Fast and Scalable In-network Lock Management Using Lock Fission

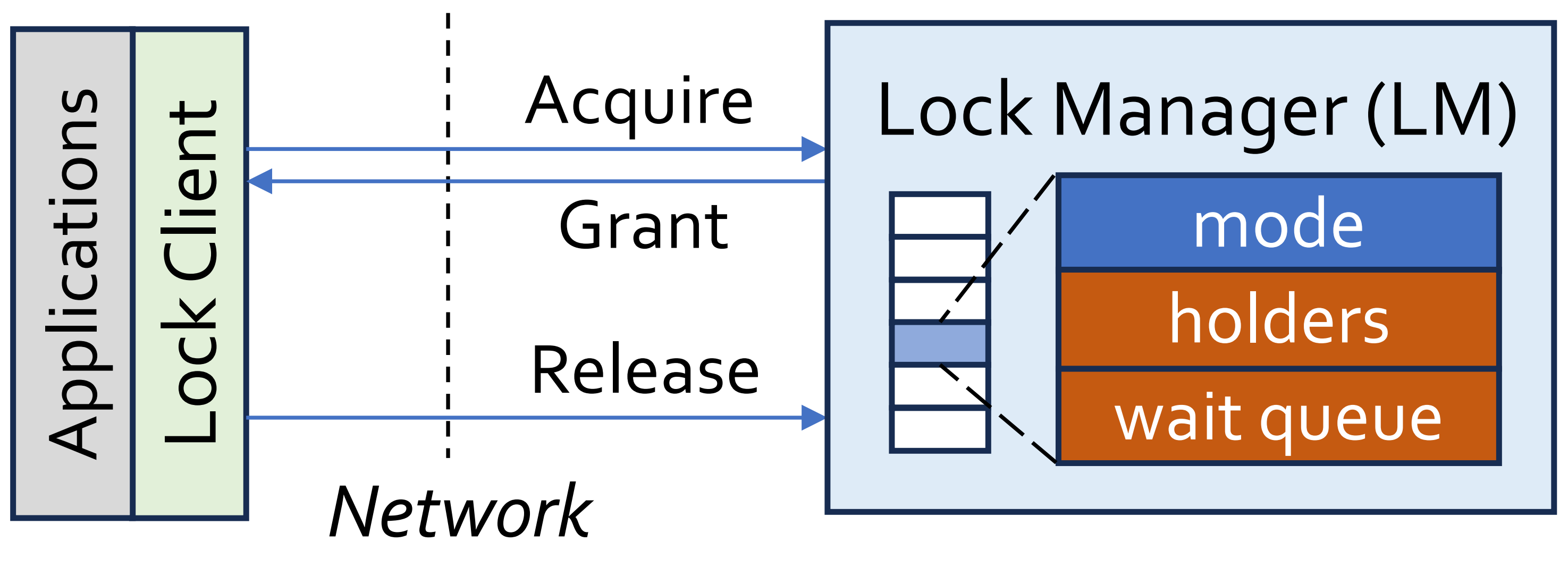
Hanze Zhang, 3rd year Ph.D. candidate from IPADS, SJTU

Background

Distributed workloads tends to have **low execution time** and **large data scale**:

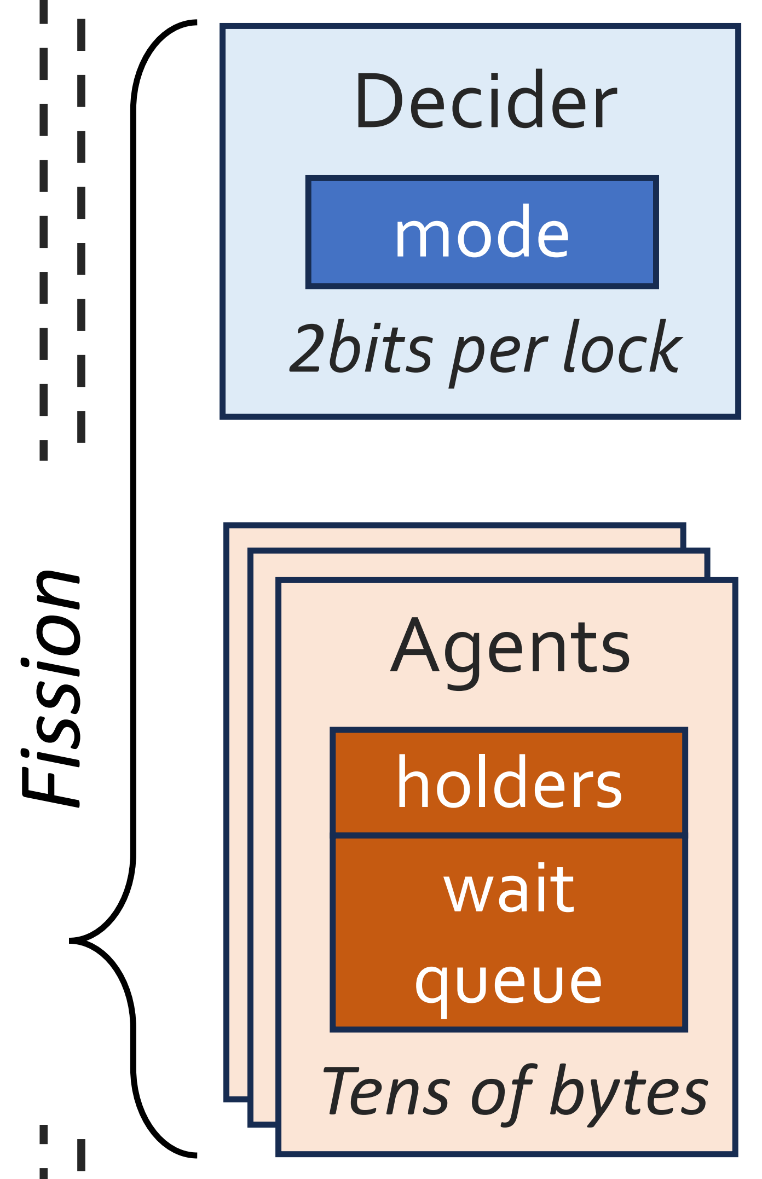
Txn. Processing	7/2.8 µs	160M rows
File System	1/10/20 µs	10B files
Key-value Store	8/15 µs	250M keys

Therefore, distributed lock services need to be **fast** and **scalable**.



Key technique

We propose ***lock fission*** that decouples lock grant **decision** and metadata **maintenance**:



Synchronous decision

- *On grant critical path, must be **fast***
- *Depends on **small, fixed-size** data*

Suitable for programmable switch

Asynchronous maintenance

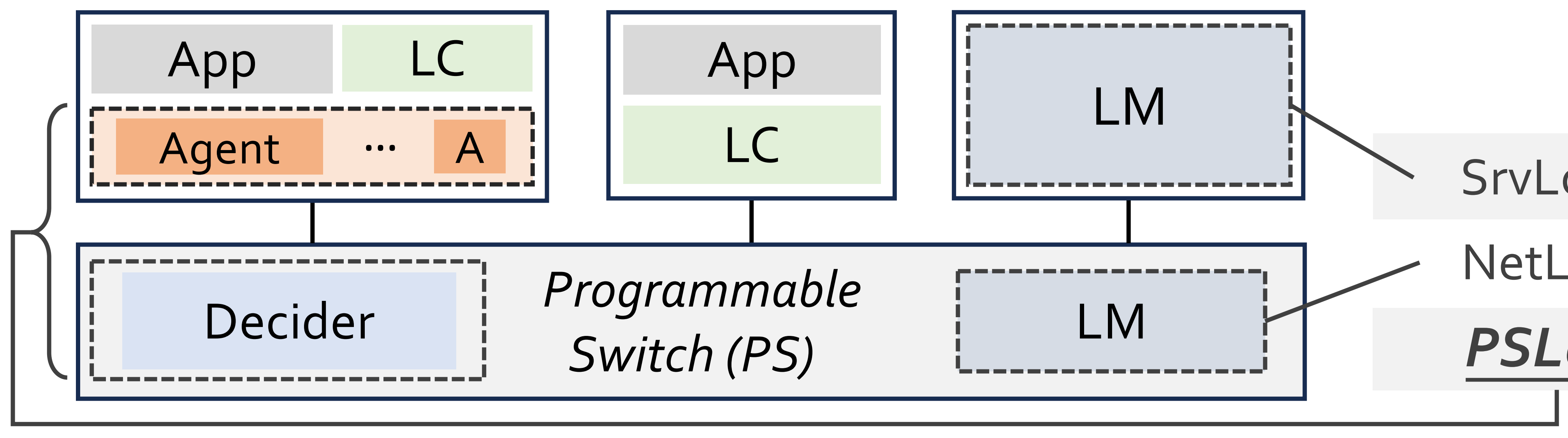
- *Off grant critical path, can be **slow***
- *Depends on **large, variable-size** data*

Suitable for servers

- *On release critical path, require locality*

Per-lock, migrate with lock ownership

Comparison of LM forms

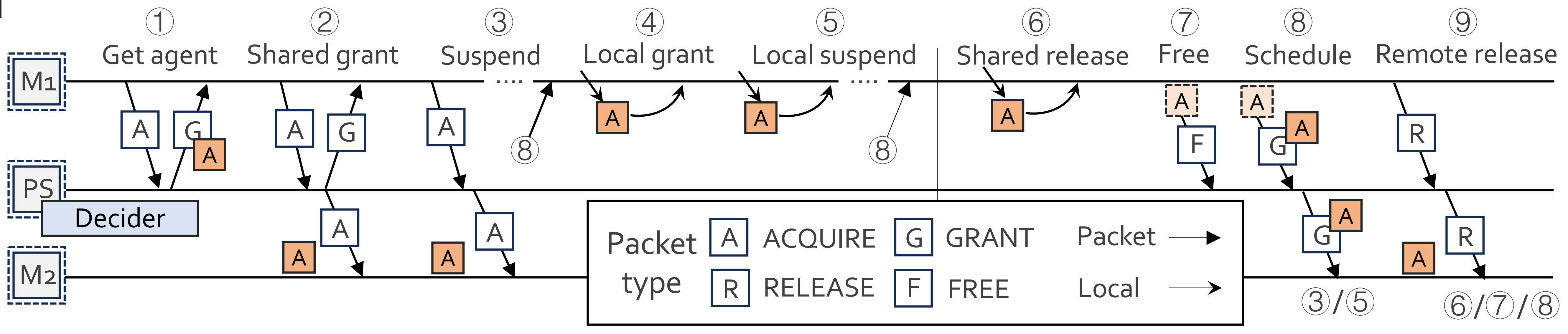


	Queueing delay	Lock number
SrvLock	Up to 100µs	>Million-scale
NetLock	<1µs	Thousand-scale
PSLock	***<1µs***	***Million-scale***

Lock fission protocol

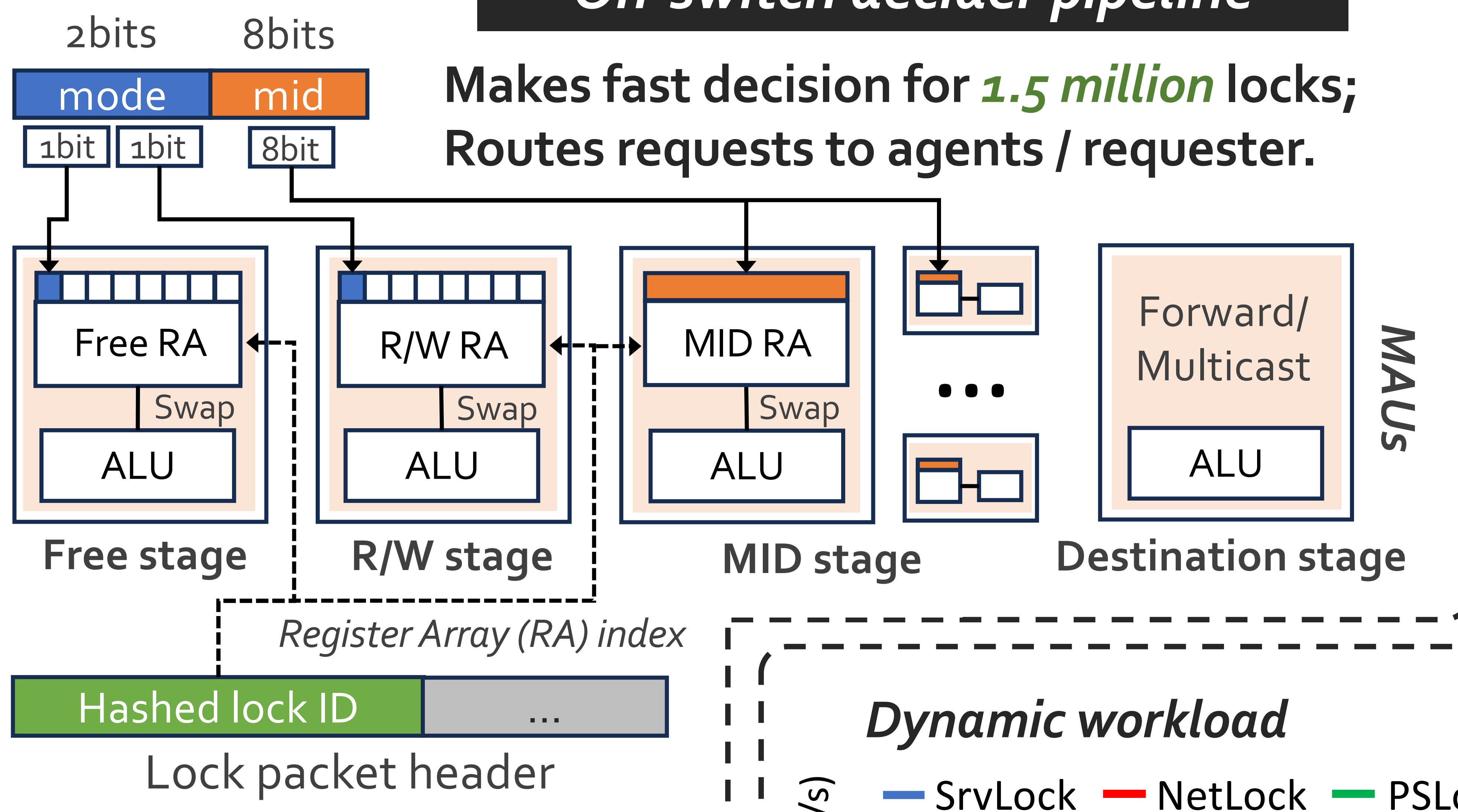
- *Read/Write* semantics
- Network exception
- Failure recovery

Acquire: ①-⑤ Release: ⑥-⑨



On-switch decider pipeline

Makes fast decision for *1.5 million* locks;
Routes requests to agents / requester.

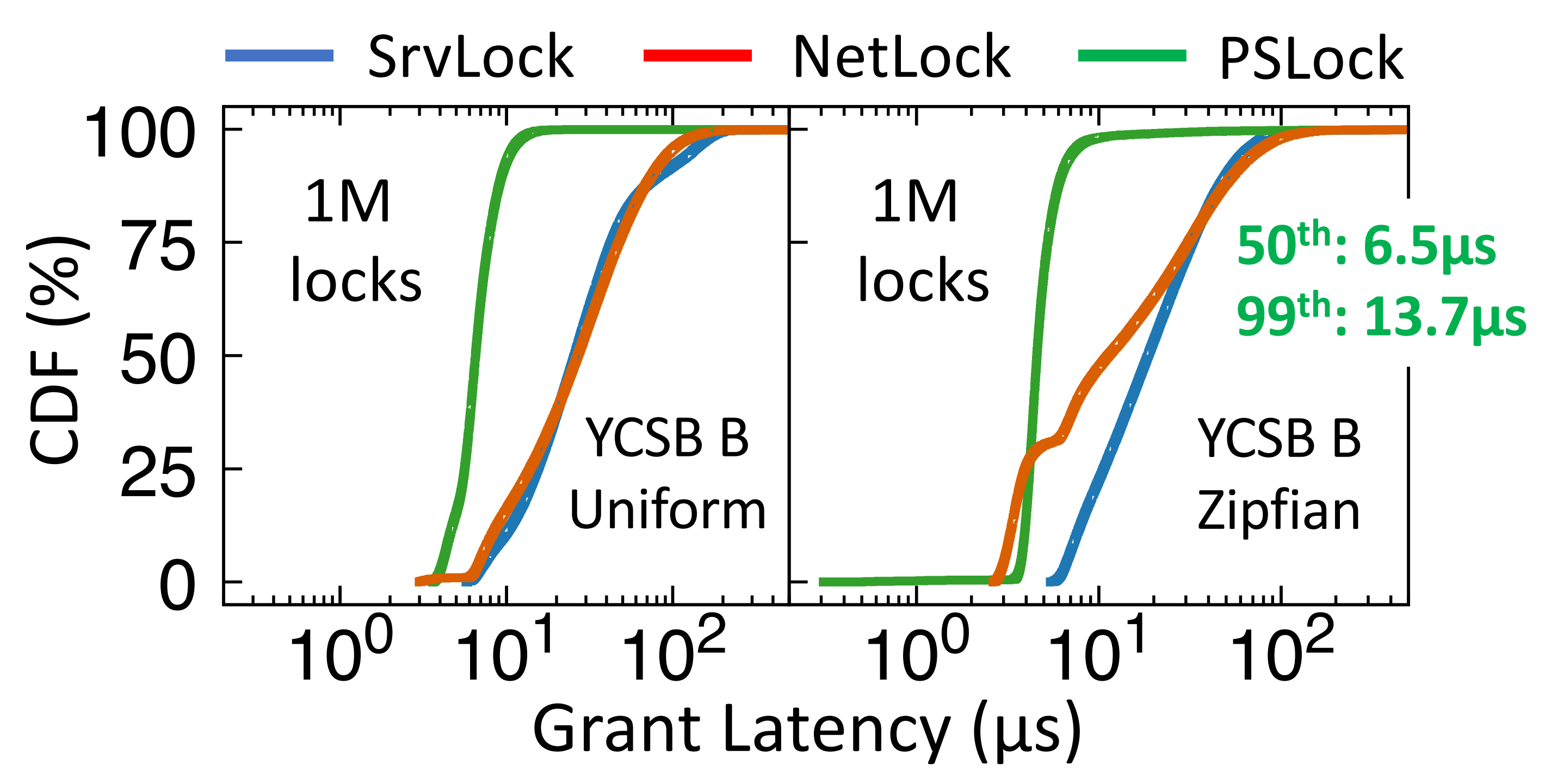


Switch memory challenges:

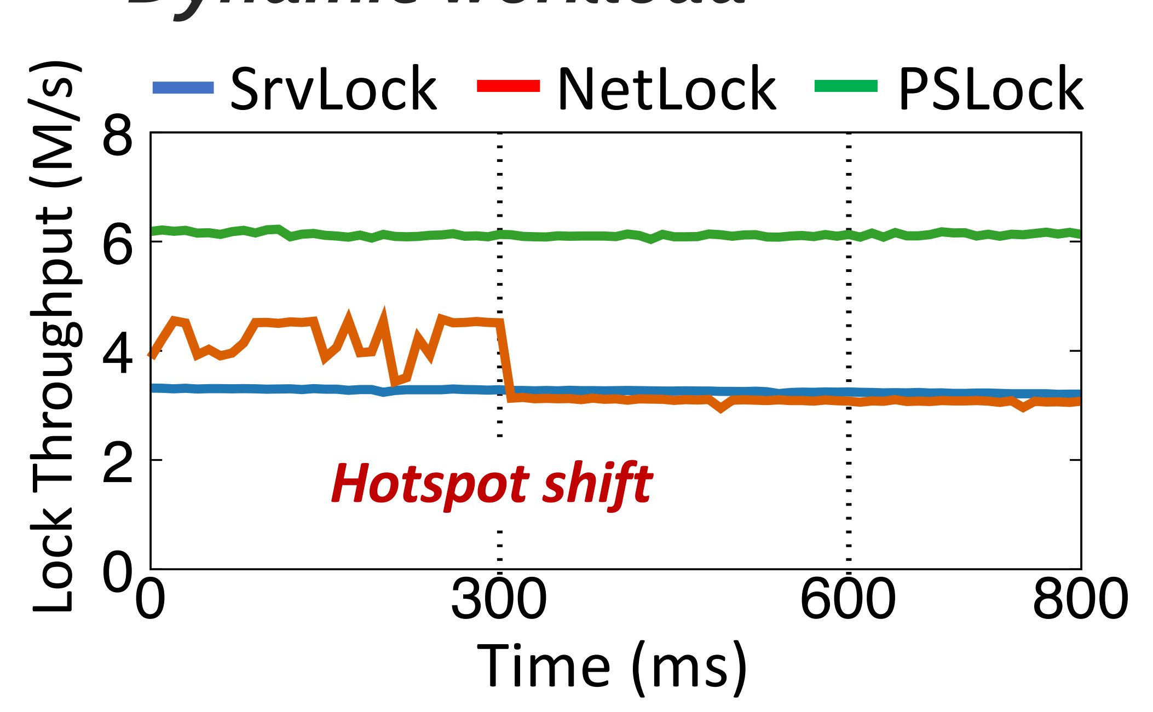
- Restricted layout (Arrays of registers)
- Limited access (Once per MAU)
- Scattered distribution (Limited per-MAU capacity)

Evaluation

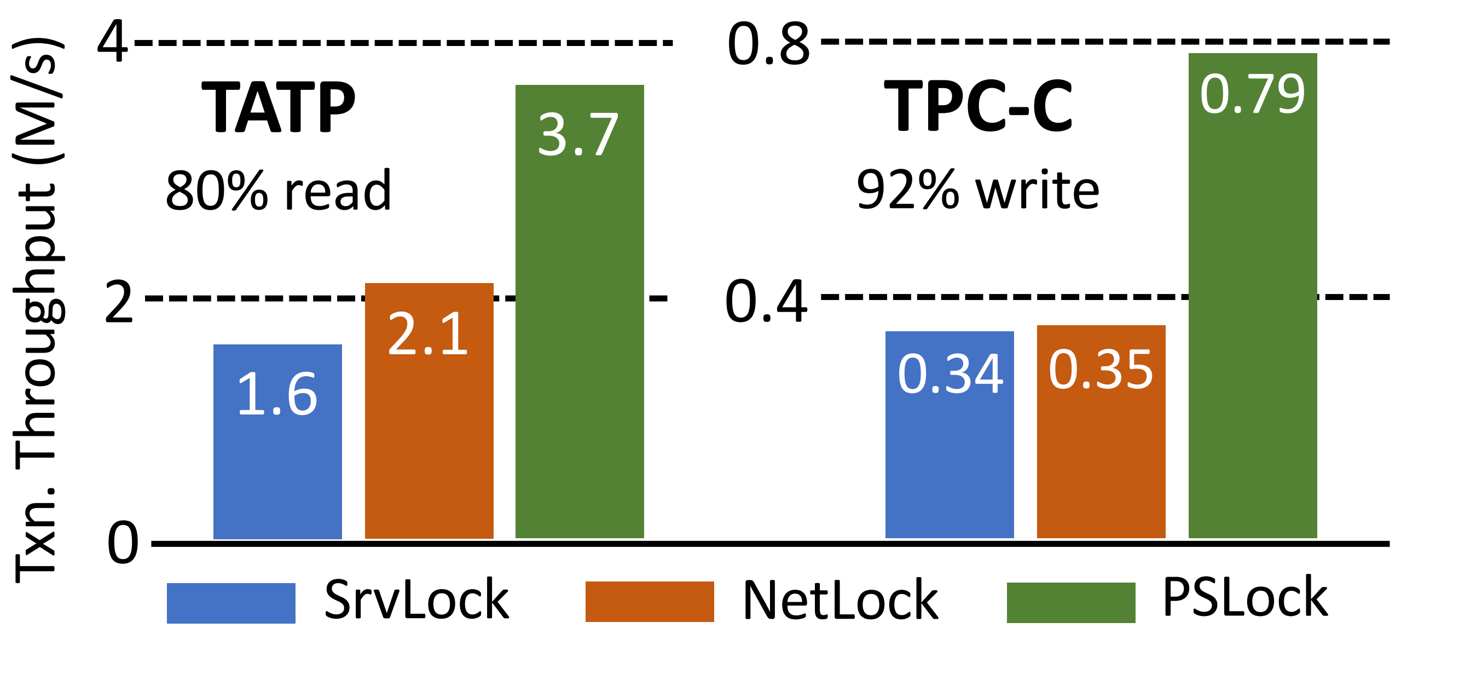
Microbenchmark: *Key-value store*



Dynamic workload



Application benchmark: *Transaction*


Txn. Throughput (M/s)
TATP
80% read
TPC-C
92% write
SrvLock
NetLock
PSLock