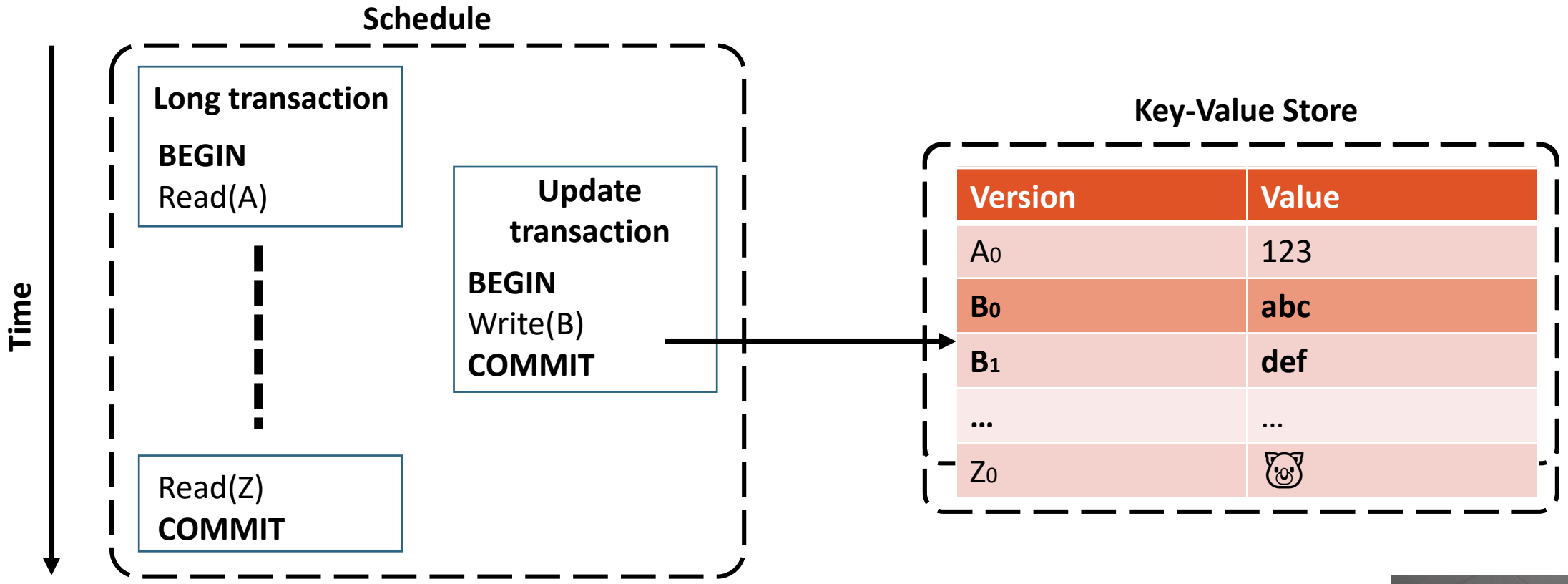
Kvell+: Snapshot Isolation without Snapshots

Baptiste Lepers
Oana Balmau
Karan Gupta
Willy Zwaenepoel

THE UNIVERSITY OF SYDNEY

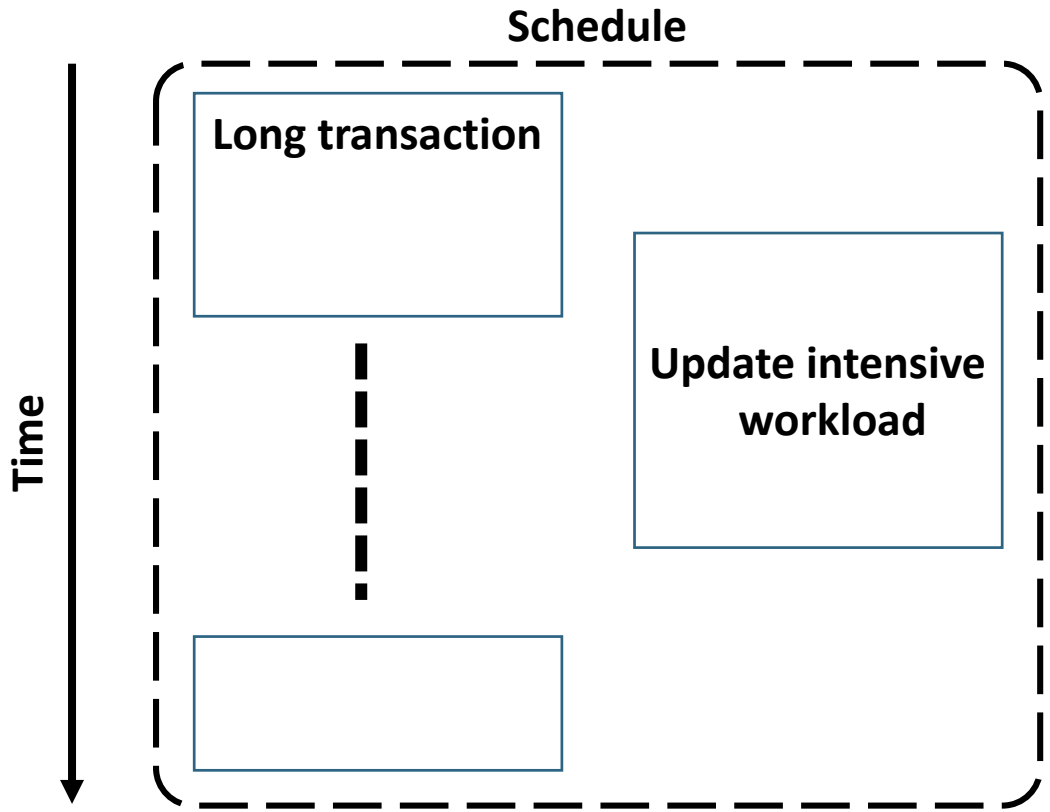
NUTANIX

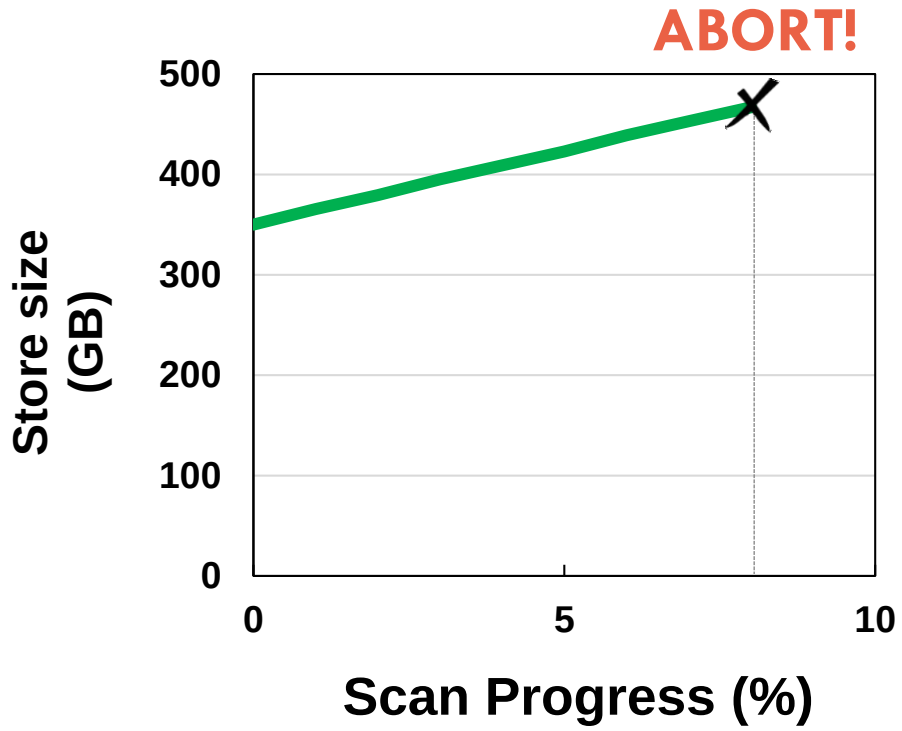
Snapshot Isolation



Versions are kept if they belong to a snapshot

Old versions ➔ Space amplification



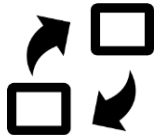


Long transactions cause massive space amplification

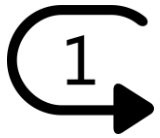
Store may run out of space

Do we really need to keep versions?

Long transactions are usually analytics

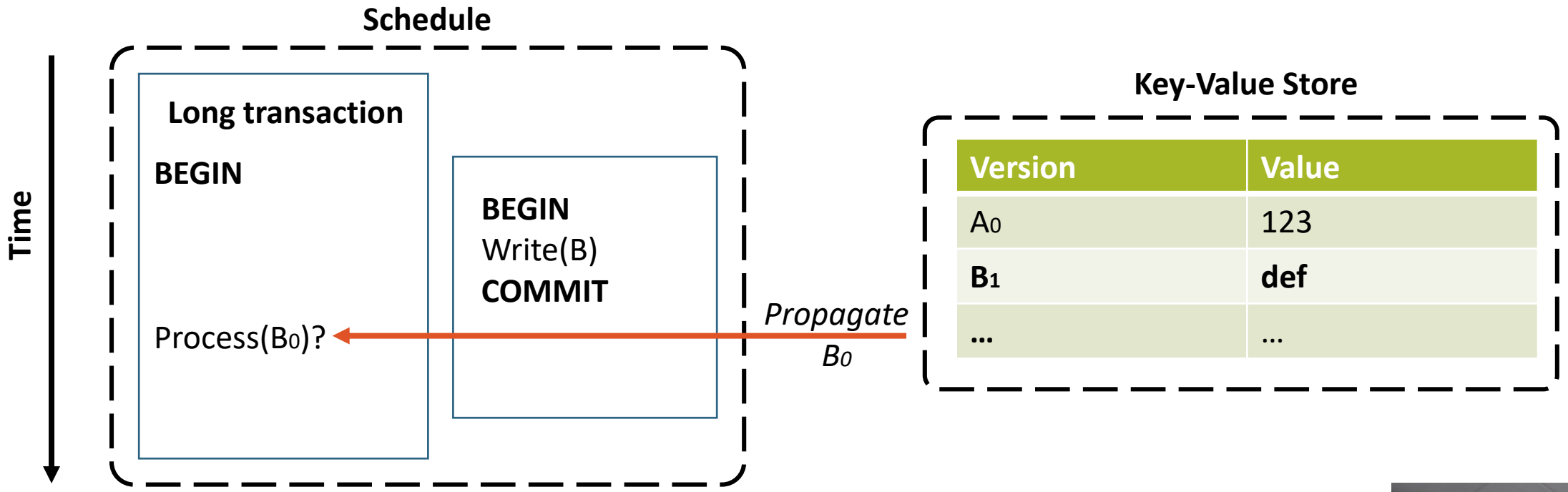


Oblivious to the order of reads



Only read items once

Solution: propagate & delete old versions

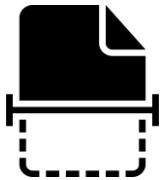


Idea: get a chance to process old versions before they are deleted

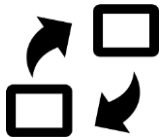
Solves space amplification

New class of analytics

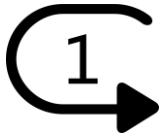
OLCP: OnLine *Commutative* Processing



Scan portions of the store



Oblivious to the order of reads



Only read items once

OLCP interface

OLCP: OnLine *Commutative* Processing

olcp_query(map, payload, scan ranges, point ranges)

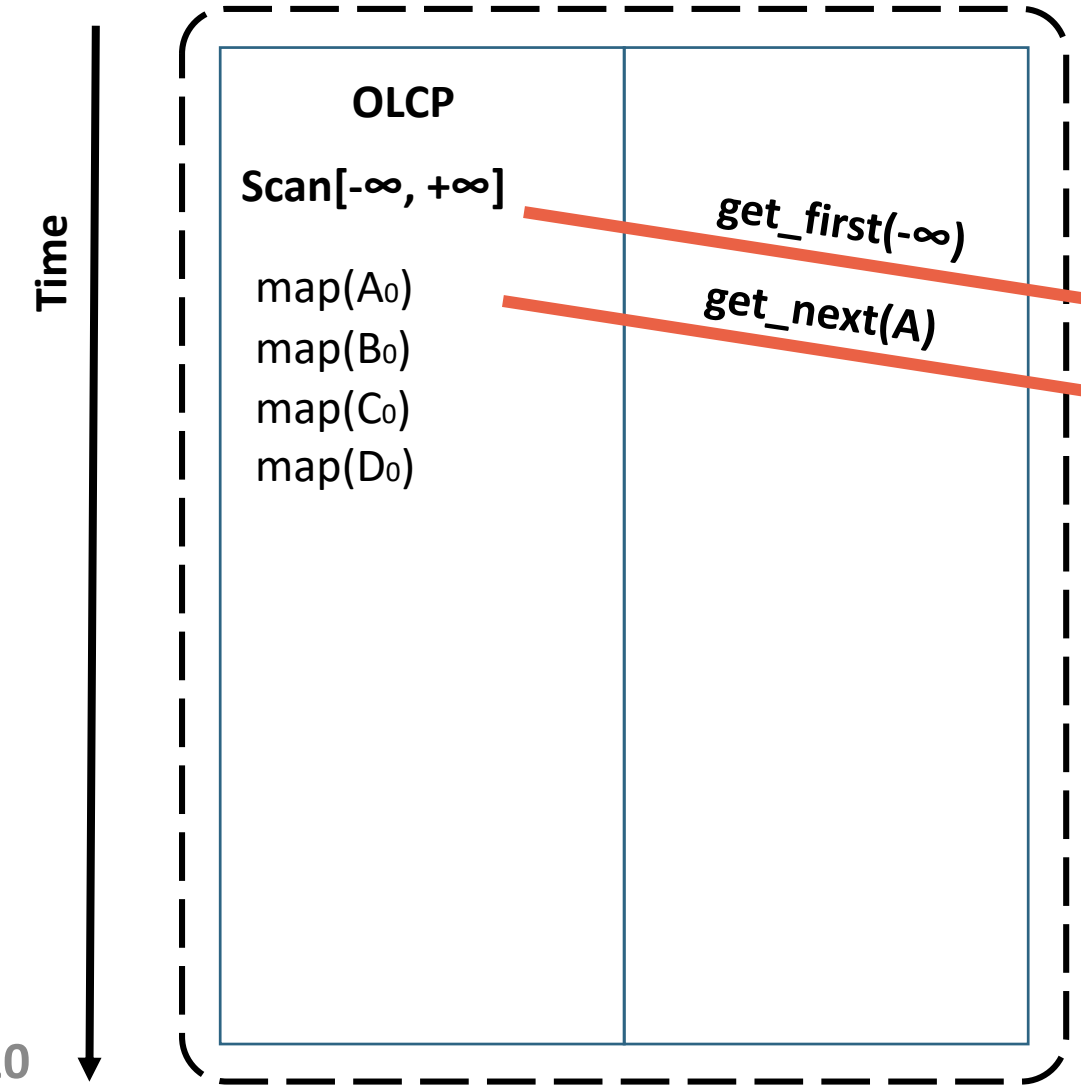
No versioning (but propagations)
For commutative operations

Versioned items
For ordered operations

Called once on every item of the scan ranges: map(item, payload)

OLCP example, no concurrent transaction

Schedule - olcp_query(map, NULL, [-∞, +∞], NULL)

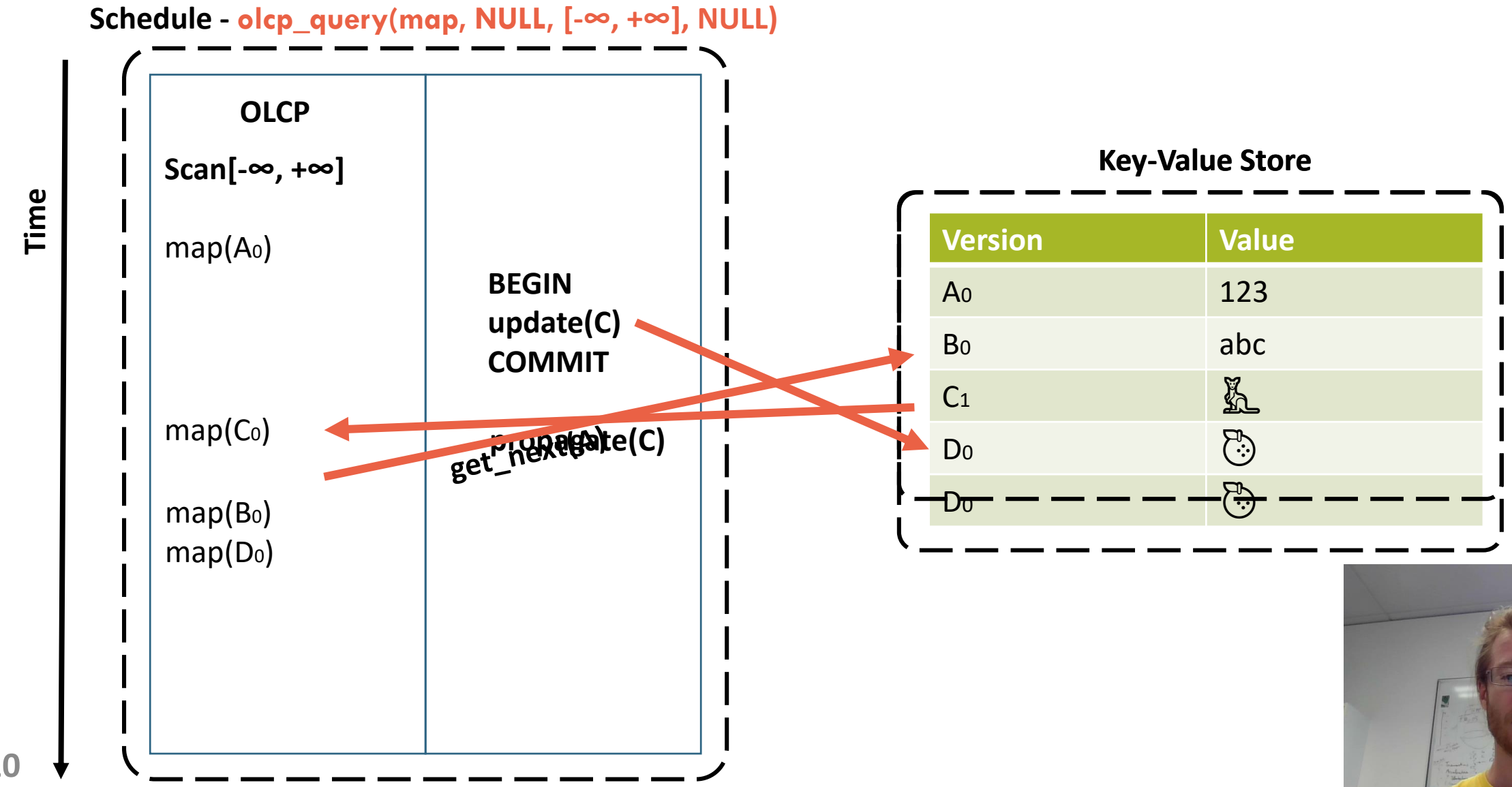


Key-Value Store

Version	Value
A_0	123
B_0	abc
C_0	
D_0	

Scan in lexicographic order

OLCP example, concurrent transactions



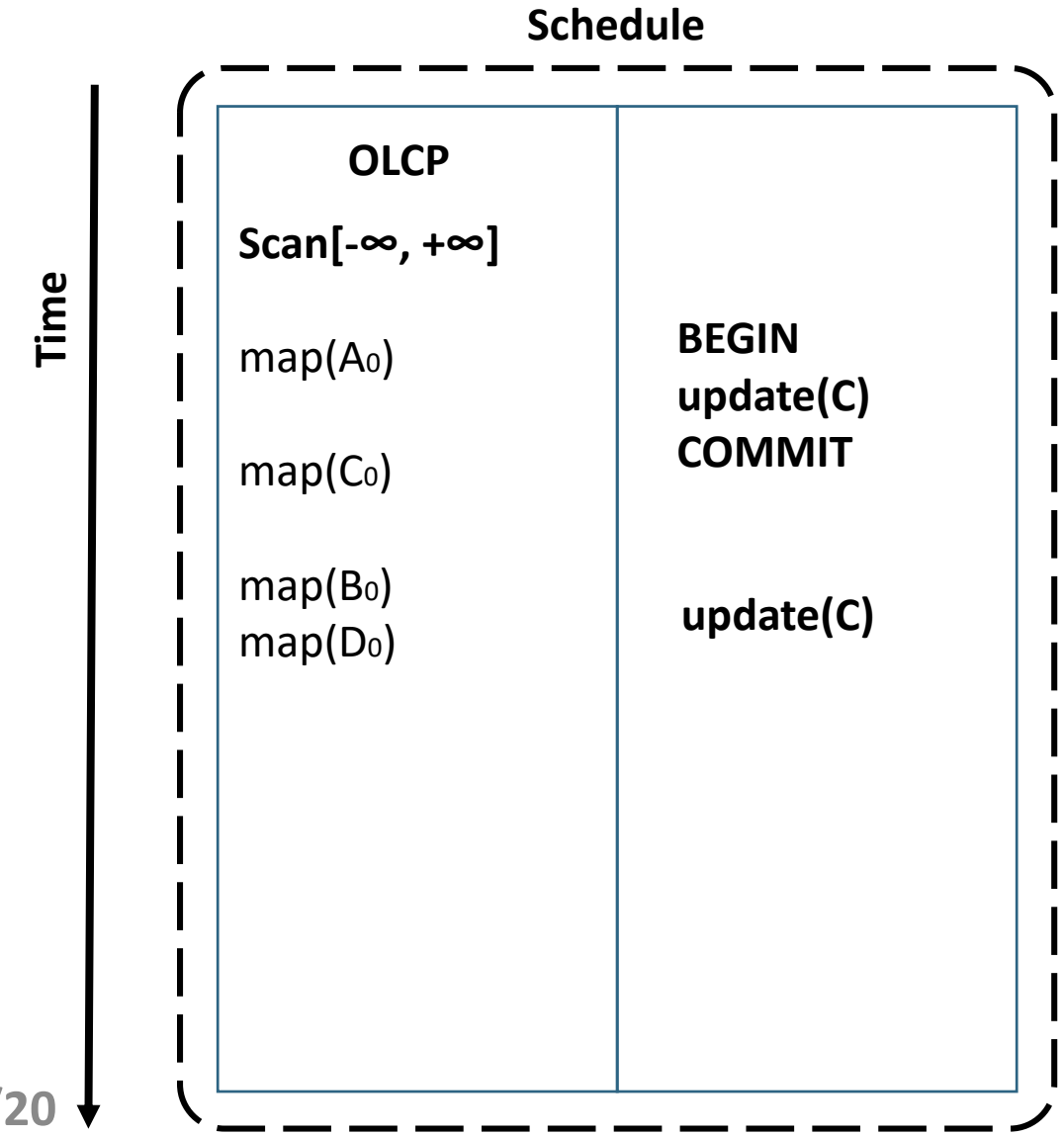
How to call map() exactly once?

Snapshot timestamp

Last scanned item

➔ **Low memory overhead**

Snapshot timestamp usage

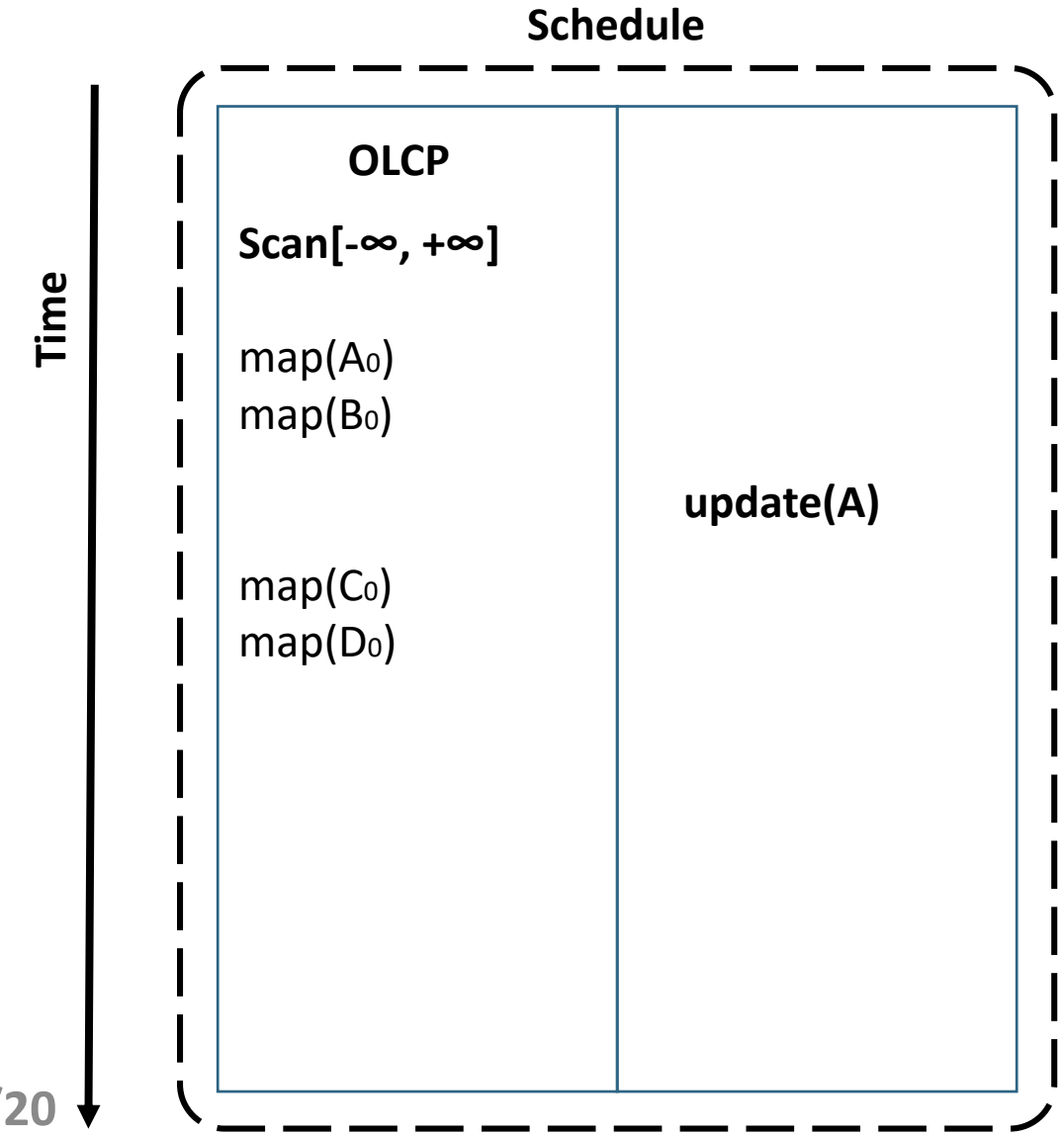


Too recent!

Key-Value Store

Version	Value
A_0	123
B_0	abc
~~C_1~~	
C_2	
D_0	

Last scanned item usage

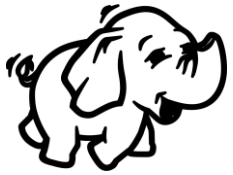


Already Scanned (A < B)

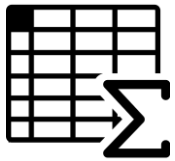
Key-Value Store

Version	Value
~~A_0~~	~~123~~
A_1	456
B_0	abc
C_0	
D_0	

OLCP applicability

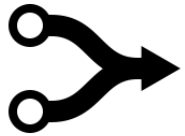


Map Reduce



SUM, COUNT

GROUP BY, CUBE, ROLLUP



Joins

OLCP example, TPC-H

SQL

```
select  sum(l_quantity) as sum_qty
from  lineitem
where l_shipdate <= '1998-09-04'
group by l_returnflag , l_linestatus
```

OLCP

```
map(item *i, payload *p) {
        if( i->l_shipdate <=  "1998-09-04")
                string k = i->l_returnflag+"|"+i->l_linestatus;
                p->sum_qty[k] += i->l_quantity;
        }
}
olcp_query(map, &p, [lineitems], NULL);
```

No space amplification with OLCP

Evaluation

Machine:

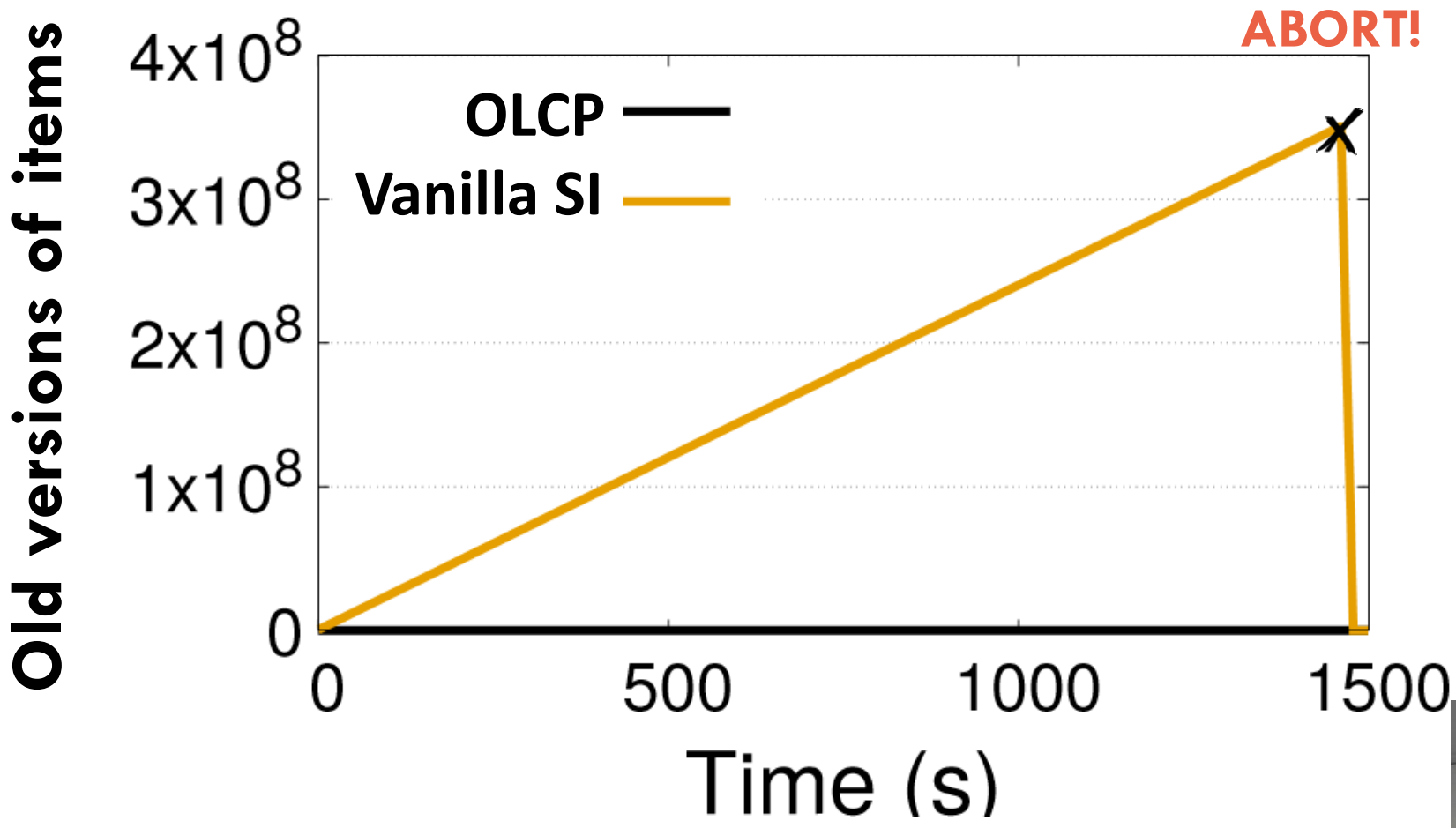
4 cores, 32GB RAM, Optane 905P drive (500K IOPS, 2GB/s)

YCSB-T Benchmark:

1KB items, 100M elements (100GB)

Update heavy + 1 Vanilla SI/OLCP scan

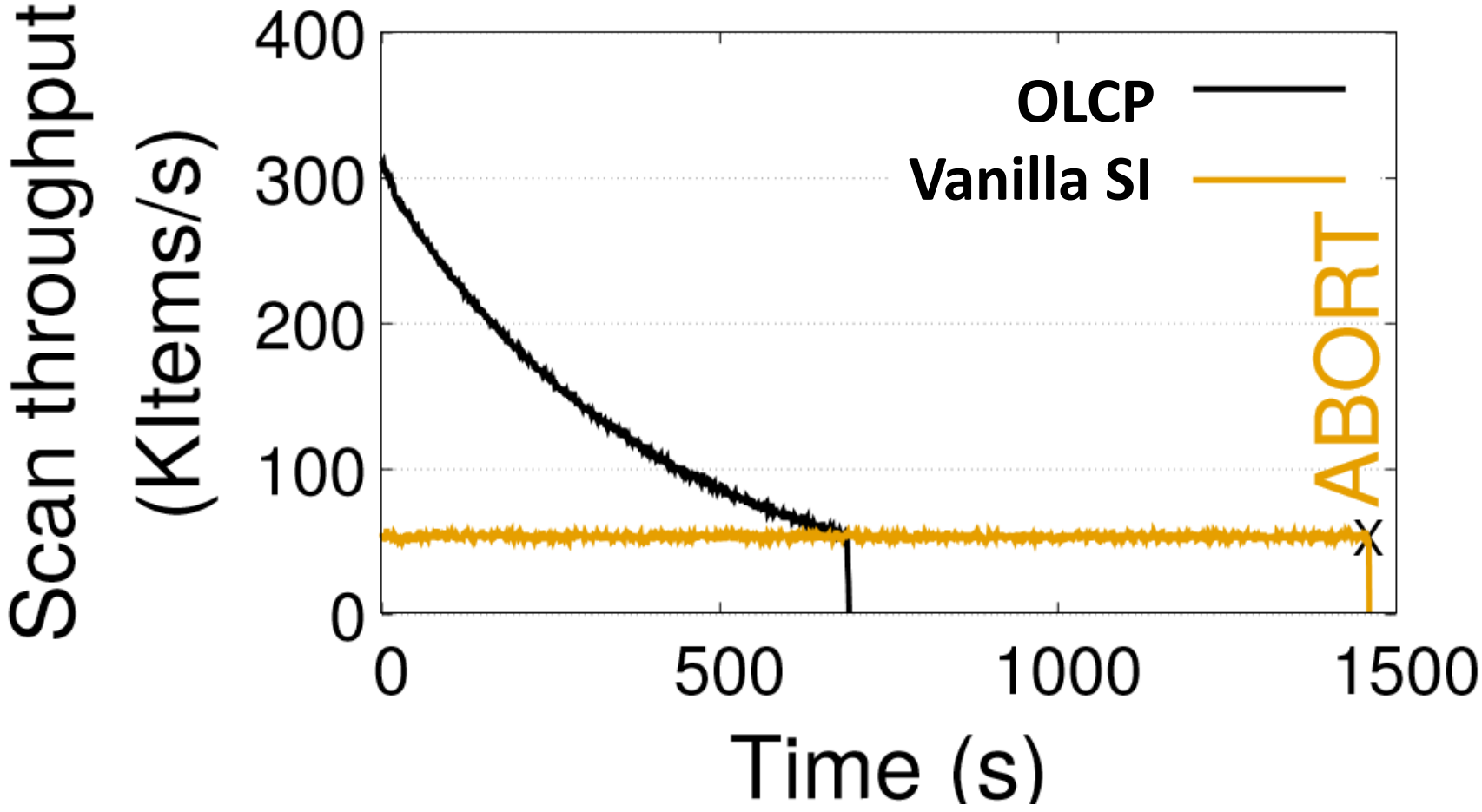
Evaluation – YCSB-T + 1 scan



Vanille SI scan aborts after 1500s

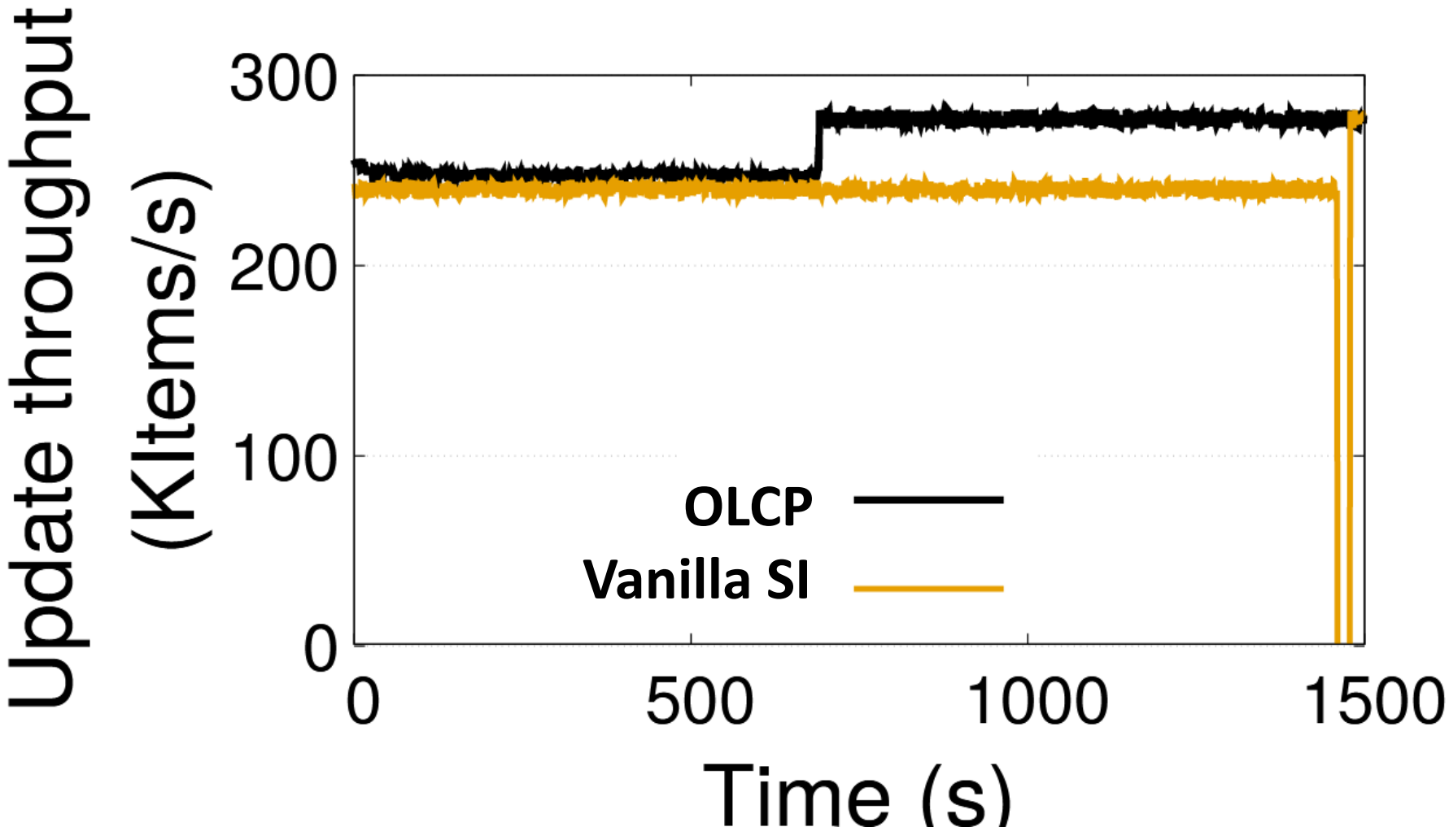
No space amplification with OLCP

Evaluation – YCSB-T + 1 scan



OLCP scan is faster
(propagated items are mostly in memory)

Evaluation – YCSB-T + 1 scan



Propagations have little/no overhead

More in the paper

Snapshot Isolation without Snapshots

Baptiste Lepers, *University of Sydney*
Oana Balmau, *University of Sydney*
Karan Gupta, *Nutanix*
Willy Zwaenepoel, *University of Sydney*

Abstract

Snapshot Isolation (SI) enables online analytical processing (OLAP) to run on a coherent view of the data despite concurrent updates performed by online transactional processing transactions (OLTP). SI keeps stale versions of updated items for as long as they might be read by OLAP transactions. Thus, long-running OLAP transactions cause significant space amplification. Space amplification is especially problematic on NVMe drives which are expensive, have limited storage capacity, and can persist millions of updates per second. In our experiments, we found that running standard analytical processing, such as a TPC-H OLAP query along with TPC-C update queries, causes the datastore to run out of space, unless disks are over-provisioned by more than 4x the dataset size. Moreover, garbage collection for snapshots cleanup creates stalls in the order of tens of seconds.

We propose a new model for analytical queries that we call Online Commutative Processing (OLCP). OLCP transactions process stale items *as they are created*, to avoid storing them in snapshots, while providing the same consistency guarantees as SI. OLCP is expressive and limits space amplification. We show how analytics workloads used in industry and TPC-H queries can be modeled through OLCP.

We extend KVell –a key-value store for NVMe SSDs– with both OLAP+SI and OLCP models. Using synthetic benchmarks and production workloads from Nutanix, we show that OLCP allows running a wide range of analytics with no space amplification, no garbage collection overhead, and speeds up range queries used for analytics by up to 2x.

1 Introduction

The desire to run frequent analytics on fresh data has led to the recent development of databases that allow running online transaction processing (OLTP) and online analytical processing (OLAP) concurrently. To provide a consistent view of the data to OLAP transactions, databases typically rely on Snapshot Isolation (SI). SI is implemented at the storage layer, usually a Key-Value store (KV), using multi-version control (MVCC). Under SI, every update generates a new version of the data, and previous versions are kept in the KV for as long as they belong to a transaction's snapshot. Long transactions may thus cause the store to grow—a phenomenon known as *space amplification*.

Minimizing disk usage is important in production systems. Facebook found that "storage space is the bottleneck" for workloads running on SSDs and switched from InnoDB to RocksDB to reduce space amplification by 2x [23]. Alibaba Group reported having to prioritize space reclamation by running expensive garbage collections on their system tiered over DRAM, Optane, and SSDs [32]. Space amplification is particularly problematic when the dataset is stored on expensive NVMe SSDs. KVs running on NVMe drives can persist millions of items per second [45] and snapshots grow fast on mediums with limited space. Because NVMe disks have similar random and sequential bandwidth, scanning data is no longer faster than performing point queries. An analytical query running concurrently with write-intensive transactions may cause the store size to increase manyfold. For instance, we found that running a single TPC-H analytical query, along with TPC-C transactions, causes our store to quadruple in size. When disks are not sufficiently over-provisioned, the TPC-H query systematically aborts. To avoid the space explosion problem, many current analytics systems are forced to run datastores in read-committed mode, trading off SI consistency guarantees for not having to over-provision space.

The observation that SI causes problematic space amplification has long been present in in-memory datastores. Prior work proposes memory-specific solutions, such as executing transactions sequentially to avoid the need for snapshots [28], or running OLAP transactions in batches on a replica of the store [39,51]. These solutions do not benefit NVMe drives because NVMes are highly parallel and because storing replicas on disk is expensive.

In this paper we argue that a new design is necessary to support analytical processing with SI consistency on NVMe drives. We make two key observations. First, most OLAPs

1

- How to do efficient propagations
- Concurrency issues
- TPC-H examples
- Detailed evaluation

Conclusion

- Snapshot Isolation induces space amplification
- OCLP Propagations → avoid space amplification
- Fits many analytics queries, is efficient

To kvell: to feel happy and proud

https://github.com/BLepers/KVell

baptiste.lepers@sydney.edu.au