



CS 839: Design the Next-Generation Database

Lecture 24: HTAP

Xiangyao Yu
4/16/2020

Announcements

Vote on the topic of the last lecture

Option 1: Streaming

- [required] Discretized Streams: Fault-Tolerant Streaming Computation at Scale
- [optional] Apache Flink™: Stream and Batch Processing in a Single Engine
- [optional] The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

Option 2: Time series

- [required] Gorilla: A Fast, Scalable, In-Memory Time Series Database
- [optional] Time Series Management Systems: A Survey

Discussion Highlights

FaaS vs. BaaS for databases

- BaaS advantages: simplifies communication and state sharing, caching
- BaaS disadvantages: potentially lower CPU and memory utilization
- FaaS advantages: fine-granularity pricing model, auto-scaling
- FaaS disadvantages: overhead of inter-function coordination, functions have limited resources and execution time, communication through S3, inherently designed for small functions

What can BaaS (e.g., Snowflake) borrow from FaaS?

- Auto-scaling: Dynamically resource allocation and fine-grained pricing

Benefits and limiting factors of running OLTP on serverless computing?

- Benefits: Elastic scaling based on demand, transactions are inherently short-lived
- Limiting factors: S3 has no read-after-write consistency, concurrency control is hard due to lack of communication

Today's Paper

HyPer: A Hybrid OLTP&OLAP Main Memory Database System Based on Virtual Memory Snapshots

Alfons Kemper¹, Thomas Neumann²

*Fakultät für Informatik
Technische Universität München
Boltzmannstraße 3, D-85748 Garching*

¹kemper@in.tum.de

²neumann@in.tum.de

Abstract—The two areas of online transaction processing (OLTP) and online analytical processing (OLAP) present different challenges for database architectures. Currently, customers with high rates of mission-critical transactions have split their data into two separate systems, one database for OLTP and one so-called *data warehouse* for OLAP. While allowing for decent transaction rates, this separation has many disadvantages including data freshness issues due to the delay caused by only pe-

database system. In addition, a separate Data Warehouse system is installed for business intelligence query processing. Periodically, e.g., during the night, the OLTP database changes are extracted, transformed to the layout of the data warehouse schema, and loaded into the data warehouse. This data staging and its associated ETL (Extract–Transform–Load) obviously incurs the problem of *data staleness* as the ETL process can

HTAP: Hybrid Transactional/Analytical Processing

Hybrid transactional/analytical processing (HTAP), a term created by Gartner Inc in 2014:

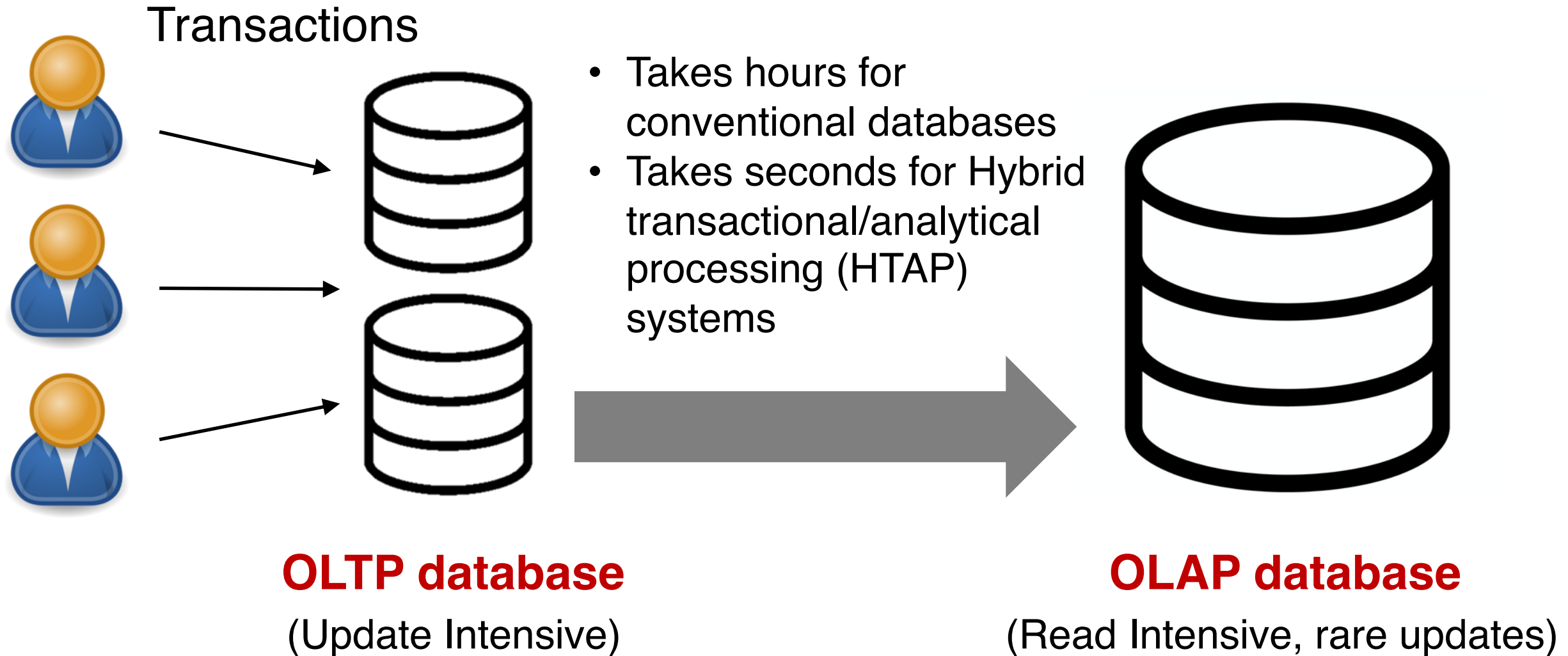
“

Hybrid transactional/analytical processing (HTAP) is an emerging application architecture that "breaks the wall" between transaction processing and analytics. It enables more informed and "in business real time" decision making.

”

Key advantage: **reducing time to insight**

OLTP vs. OLAP (Slide from L2)



HTAP Design Options [1]

Single System for OLTP and OLAP

- *Using Separate Data Organization for OLTP and OLAP*
- *Same Data Organization for both OLTP and OLAP*



Separate OLTP and OLAP Systems

- *Decoupling the Storage for OLTP and OLAP*
- *Using the Same Storage for OLTP and OLAP*

Background: Through the Looking Glass [2]

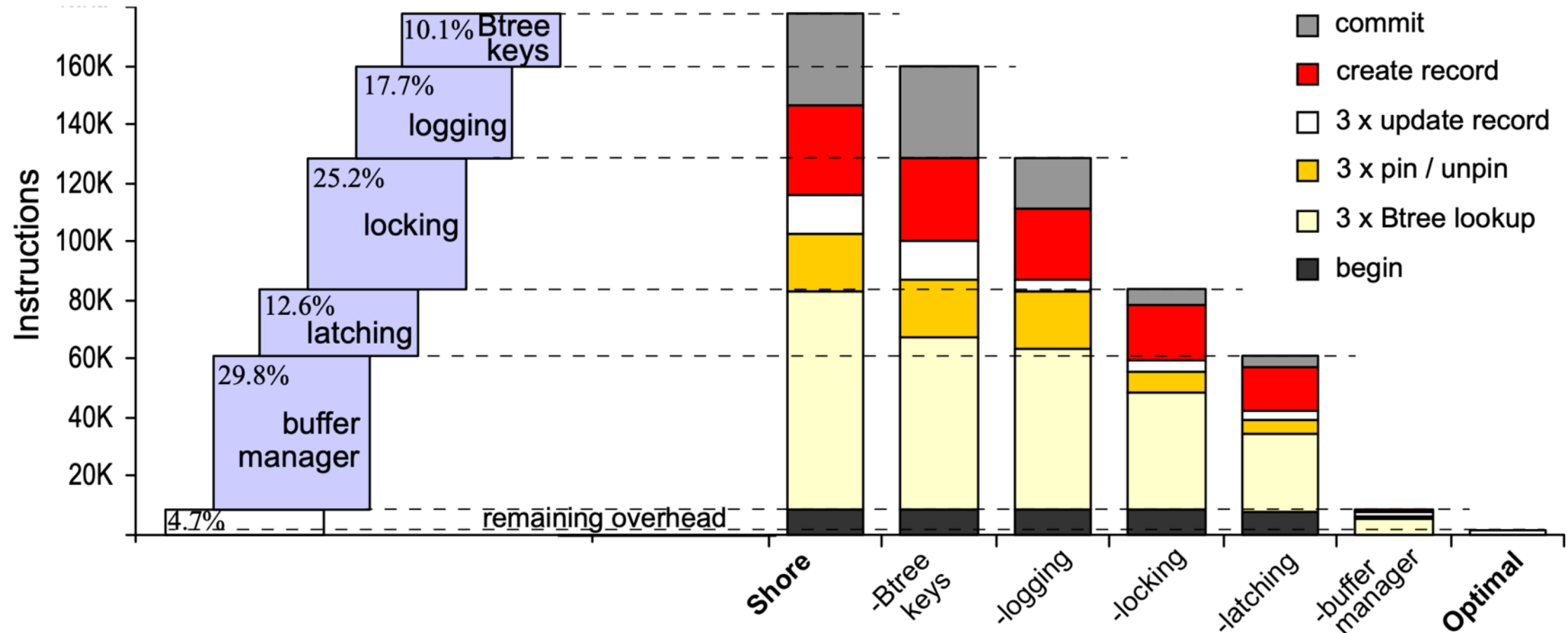
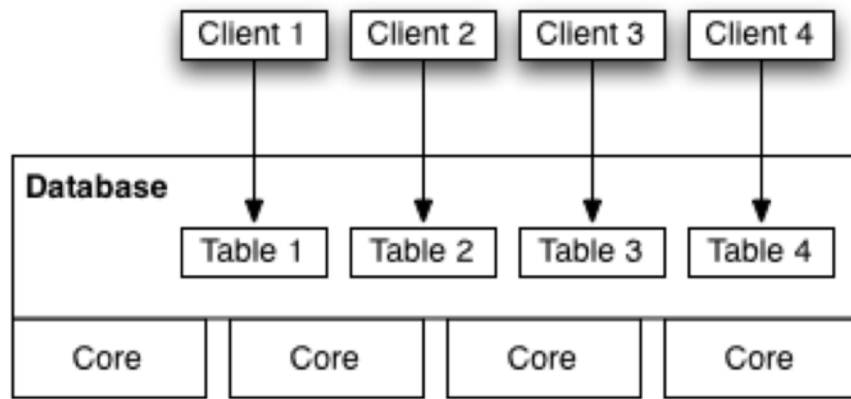
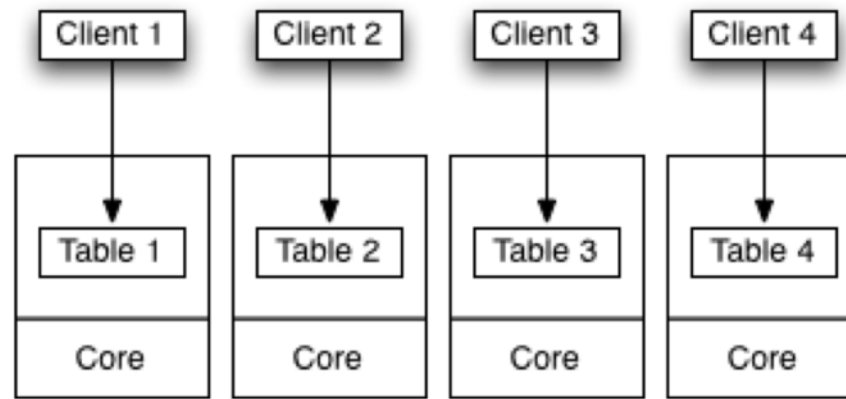


Figure 5. Detailed instruction count breakdown for Payment transaction.

Background: H-STORE [3]



Threads



Partitions



- Single partition transactions are sequentially executed
- Multi-partition transactions lock entire partitions
- Support short, stored-procedure transactions

Background: VoltDB

H-Store is commercialized into VoltDB

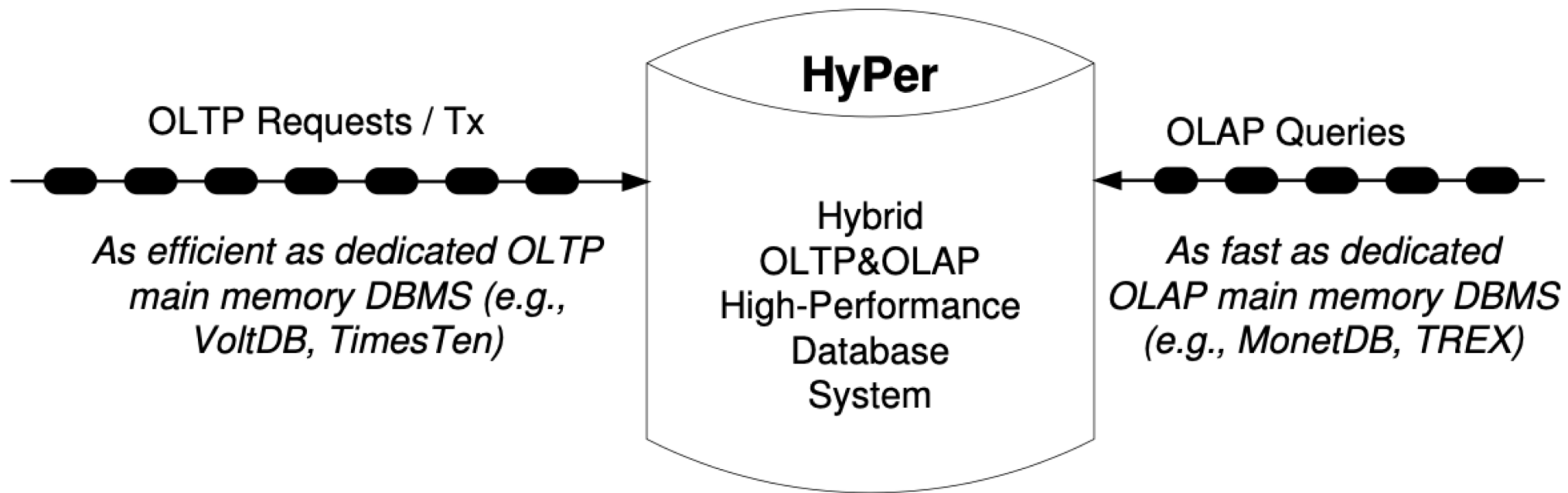


VoltDB has some cool features

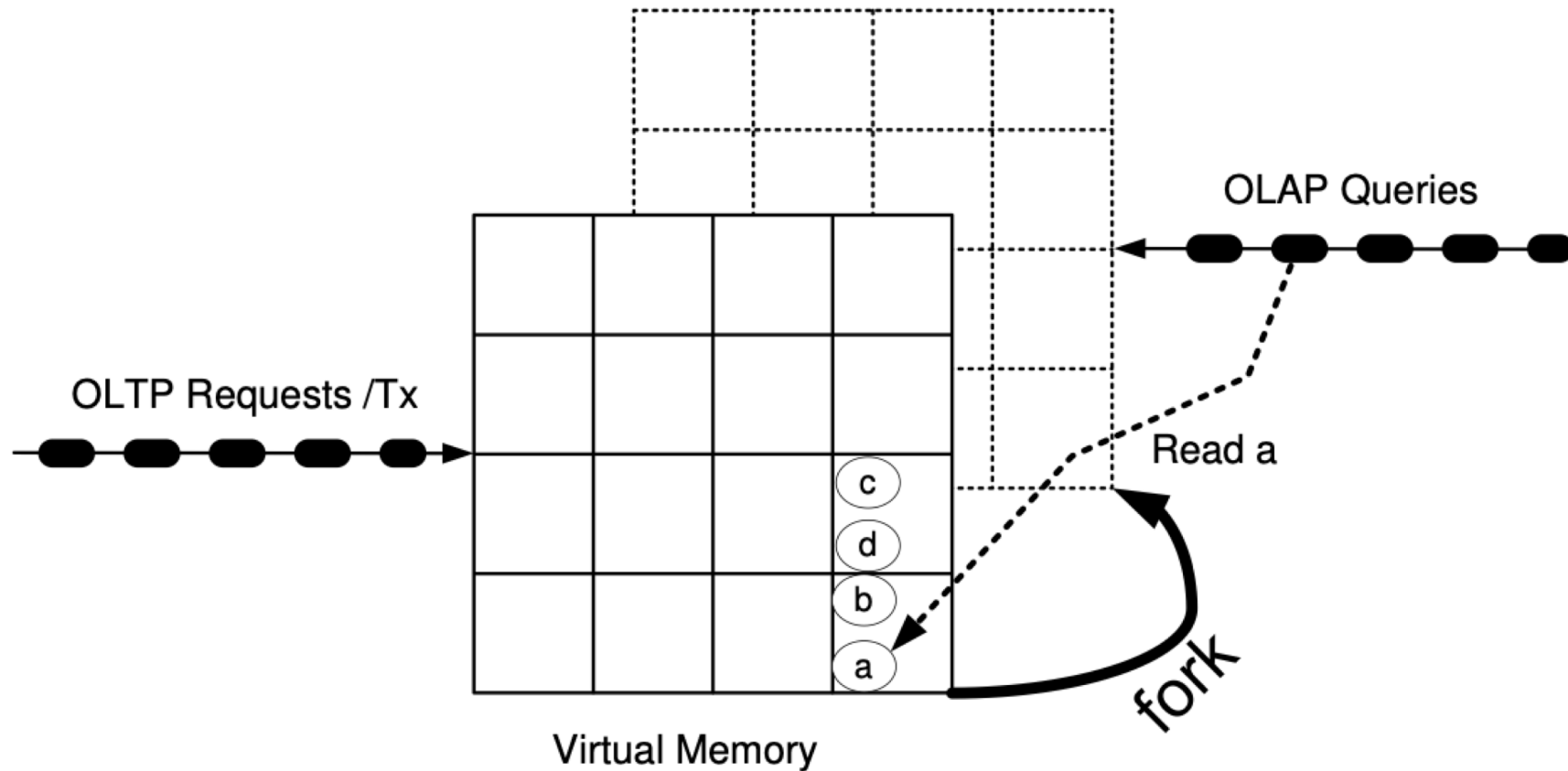
- Active-active replication (deterministic execution)
- Command logging

Hyper

Execute analytical queries without blocking transactions



Virtual Memory Snapshots



Create consistent database snapshot for OLAP queries to read
Transactions run with copy-on-write to avoid polluting the snapshots

Fork()

Linux Programmer's Manual

fork() creates a new process by duplicating the calling process. The new process is referred to as the *child* process. The calling process is referred to as the *parent* process.

Does **not** copy all the memory pages

Does copy the parent's page table (all pages set to readonly mode)

Copy-on-write (COW)

- If any page is modified by either parent or child process, a new page is created for the corresponding process

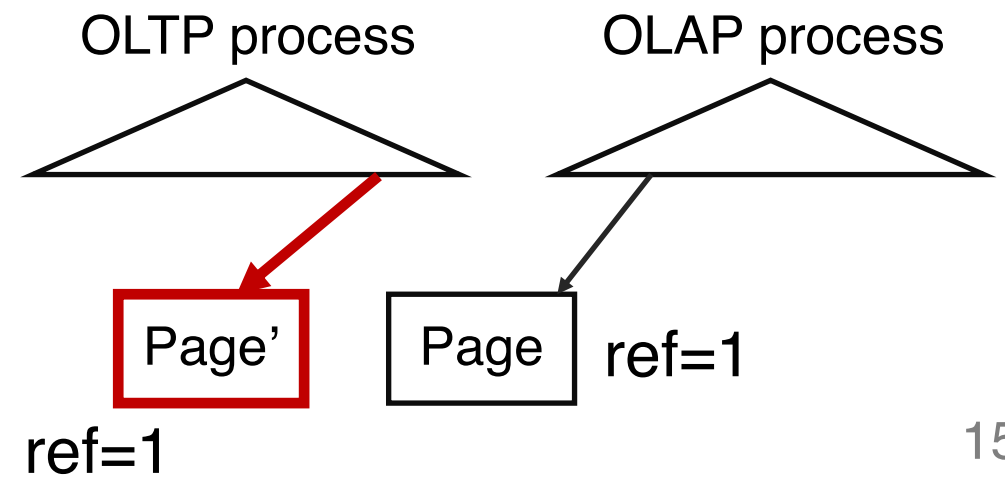
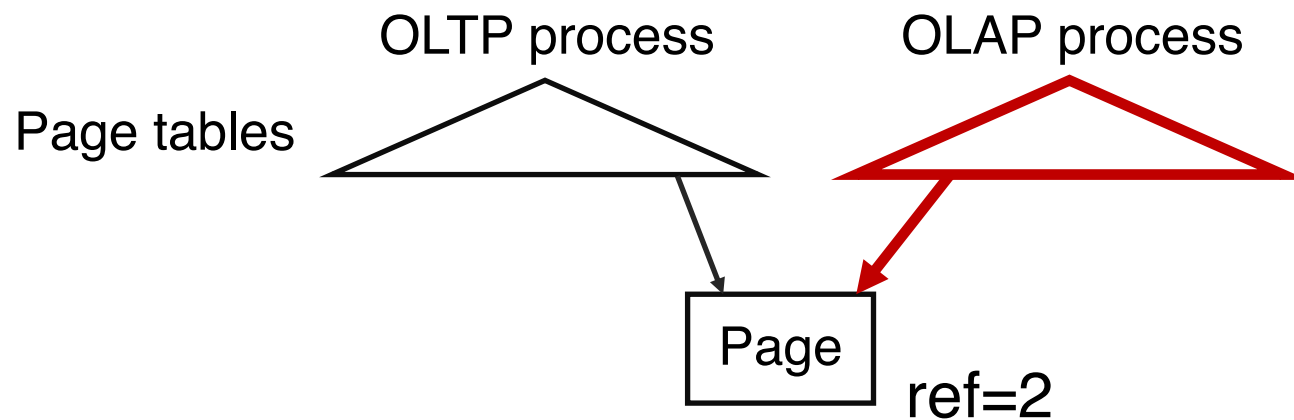
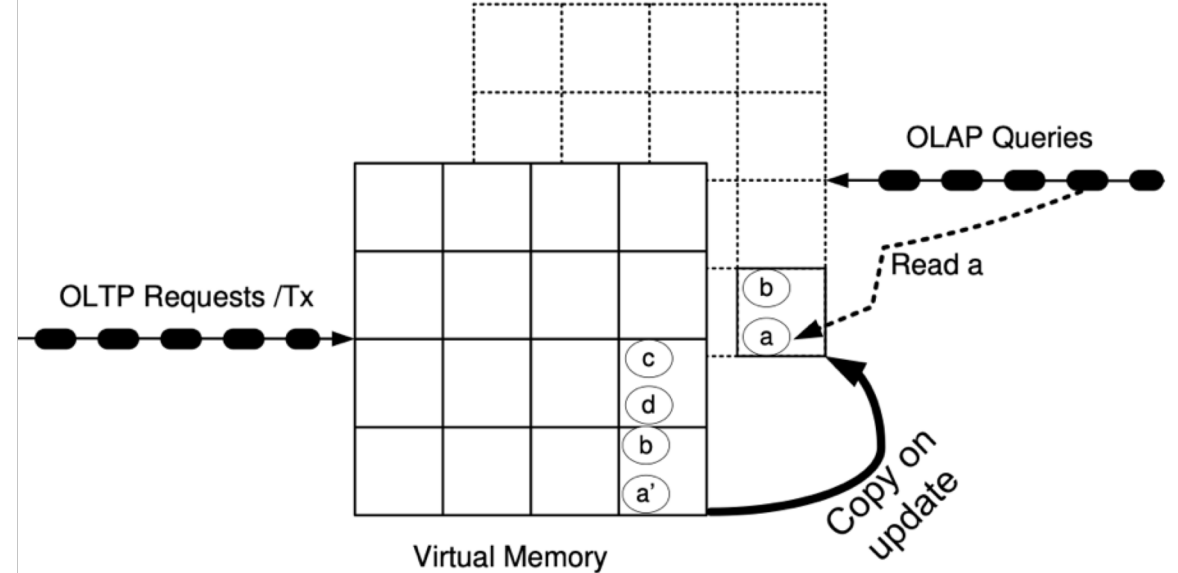
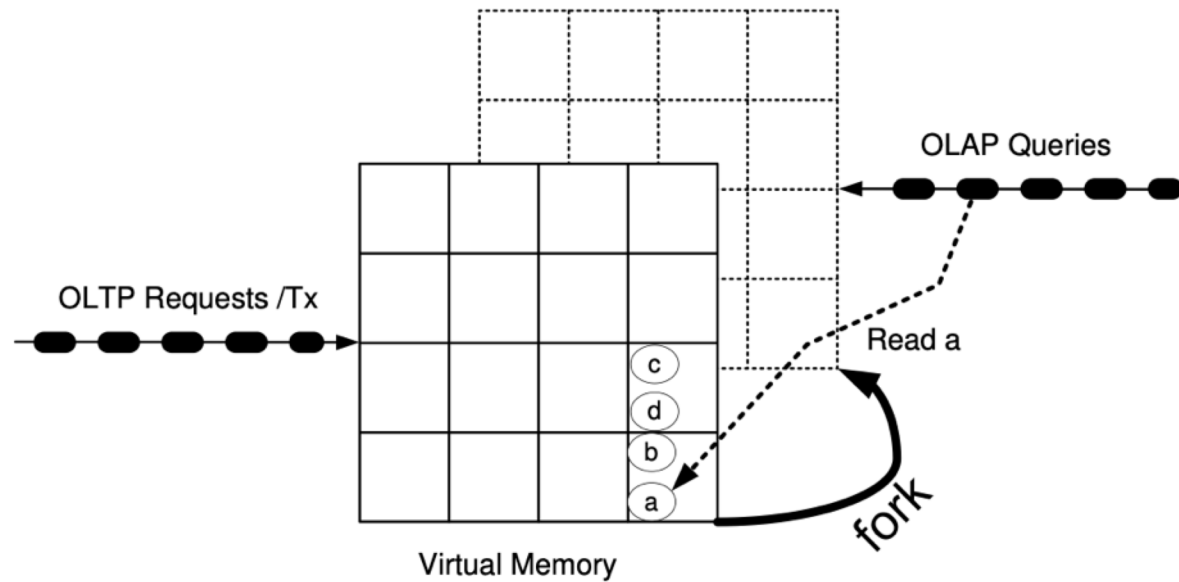
Cost of Fork()

DB size in MB	small pages (4 KB)		large pages (2 MB)	
	fork duration	... per 1 MB DB	fork duration	... per 1 MB DB
409.6	7ms	17 μ s	0.087ms	0.21 μ s
819.2	14ms	17 μ s	0.119ms	0.15 μ s
1638.4	28ms	17 μ s	0.165ms	0.10 μ s
4096	34ms	8 μ s	0.300ms	0.07 μ s
8192	69ms	14 μ s	0.529ms	0.06 μ s
16384	136ms	8 μ s	0.958ms	0.06 μ s
32768	271ms	8 μ s	1.863ms	0.06 μ s
40960	344ms	8 μ s	2.702ms	0.06 μ s

Cost of fork() is proportional to the page table size, which depends on

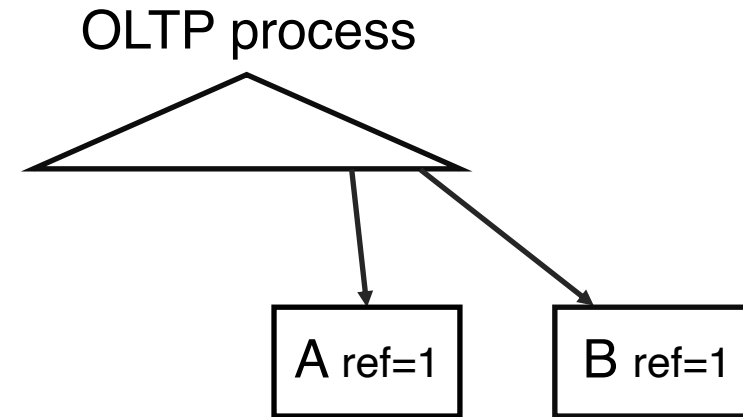
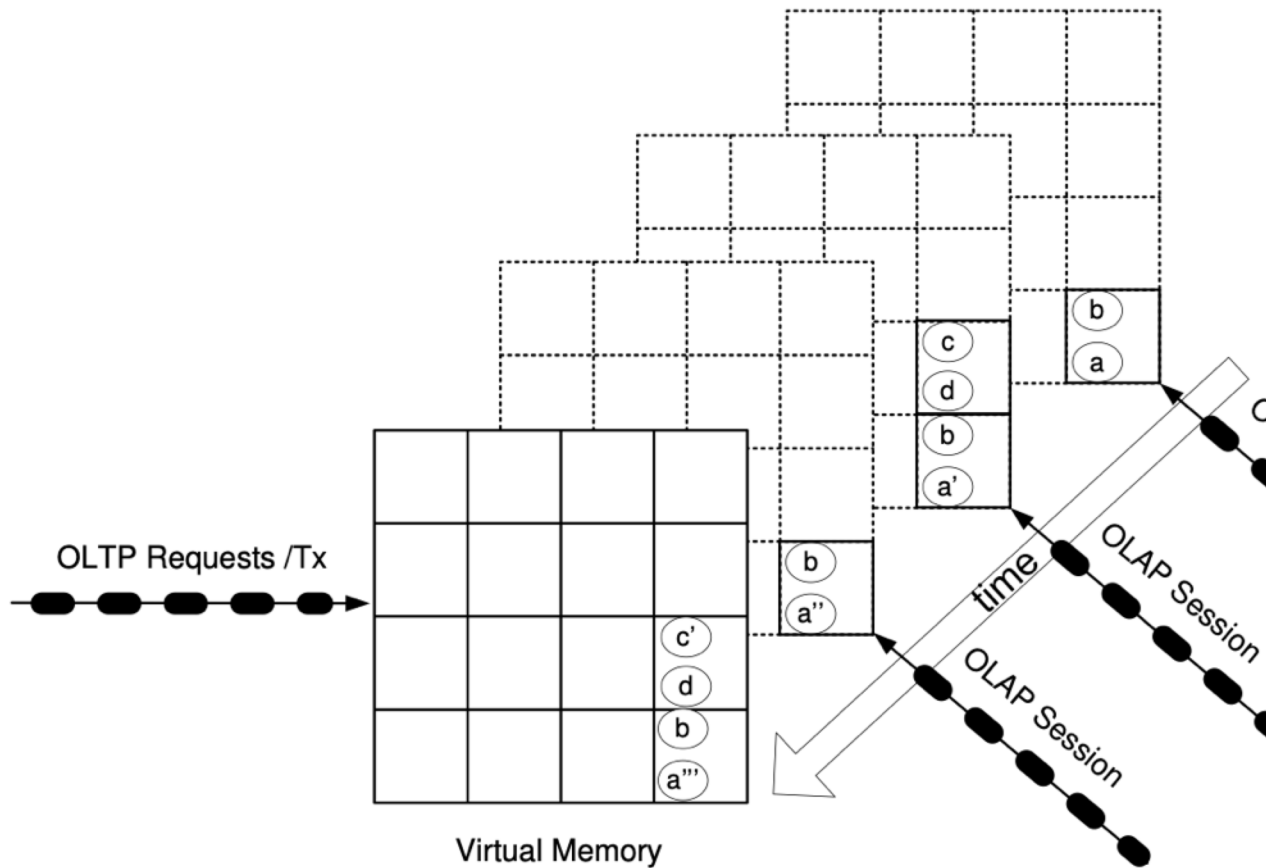
- Database size
- Page size

Fork-Based Virtual Snapshots



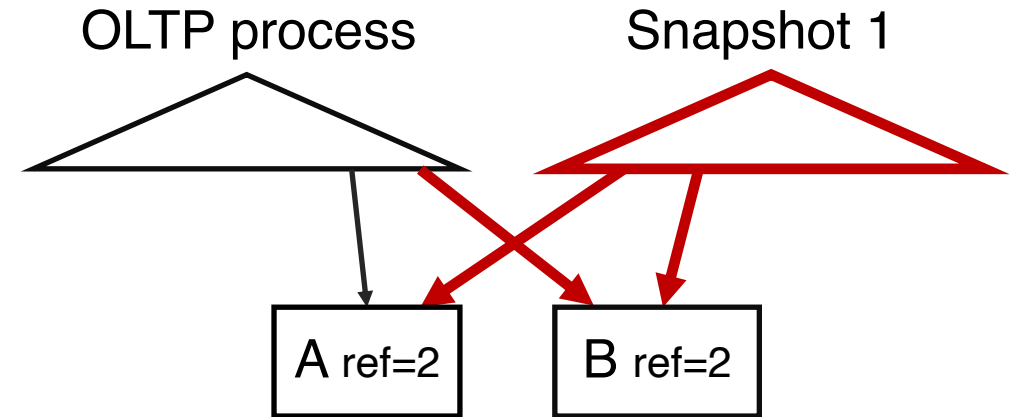
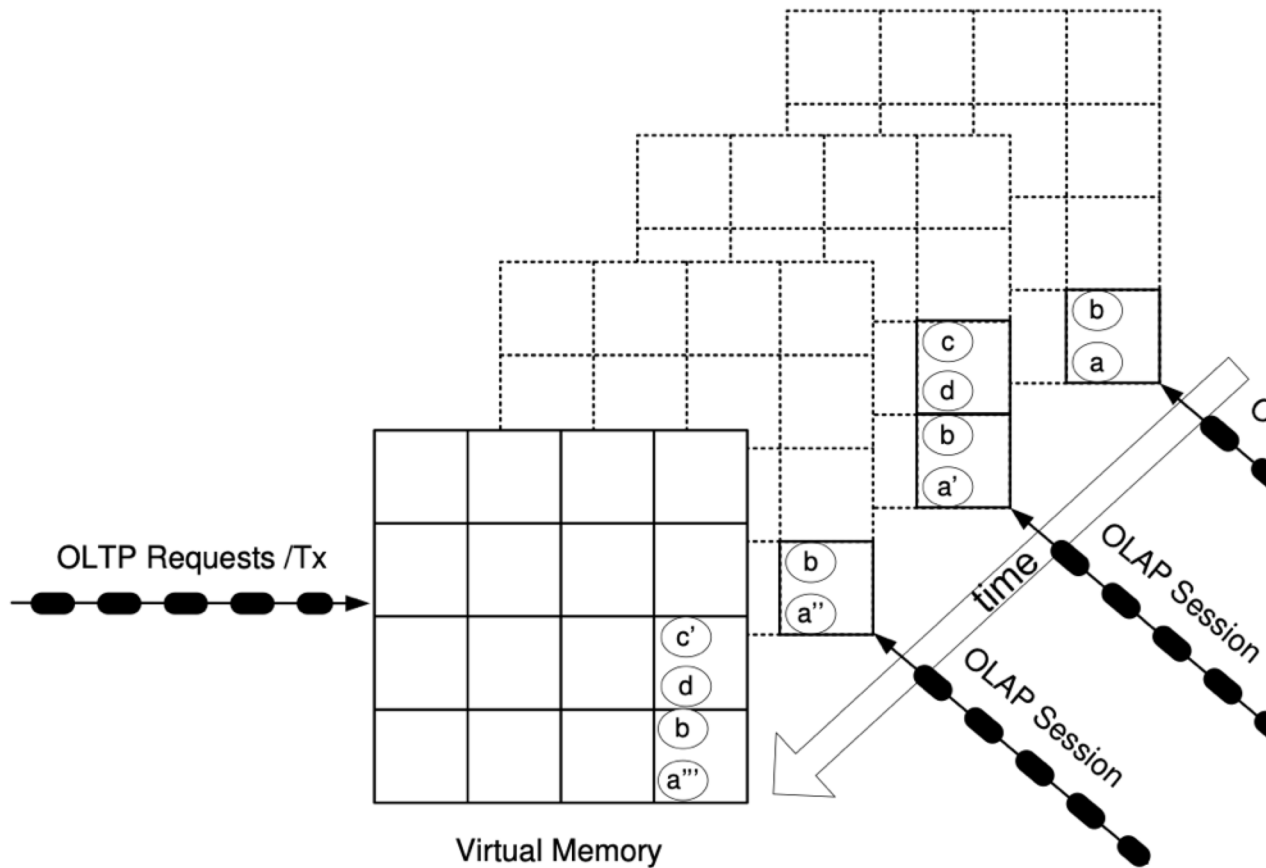
Multiple OLAP Session

OLAP Session: Group of OLAP queries that access the same snapshot



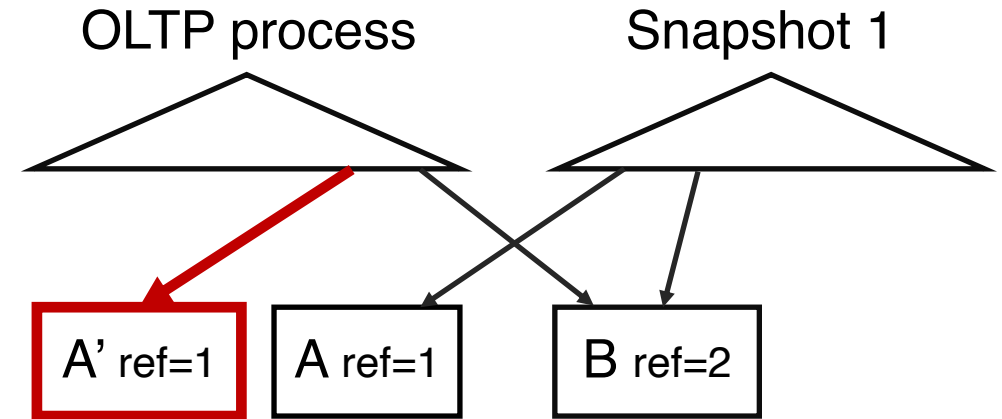
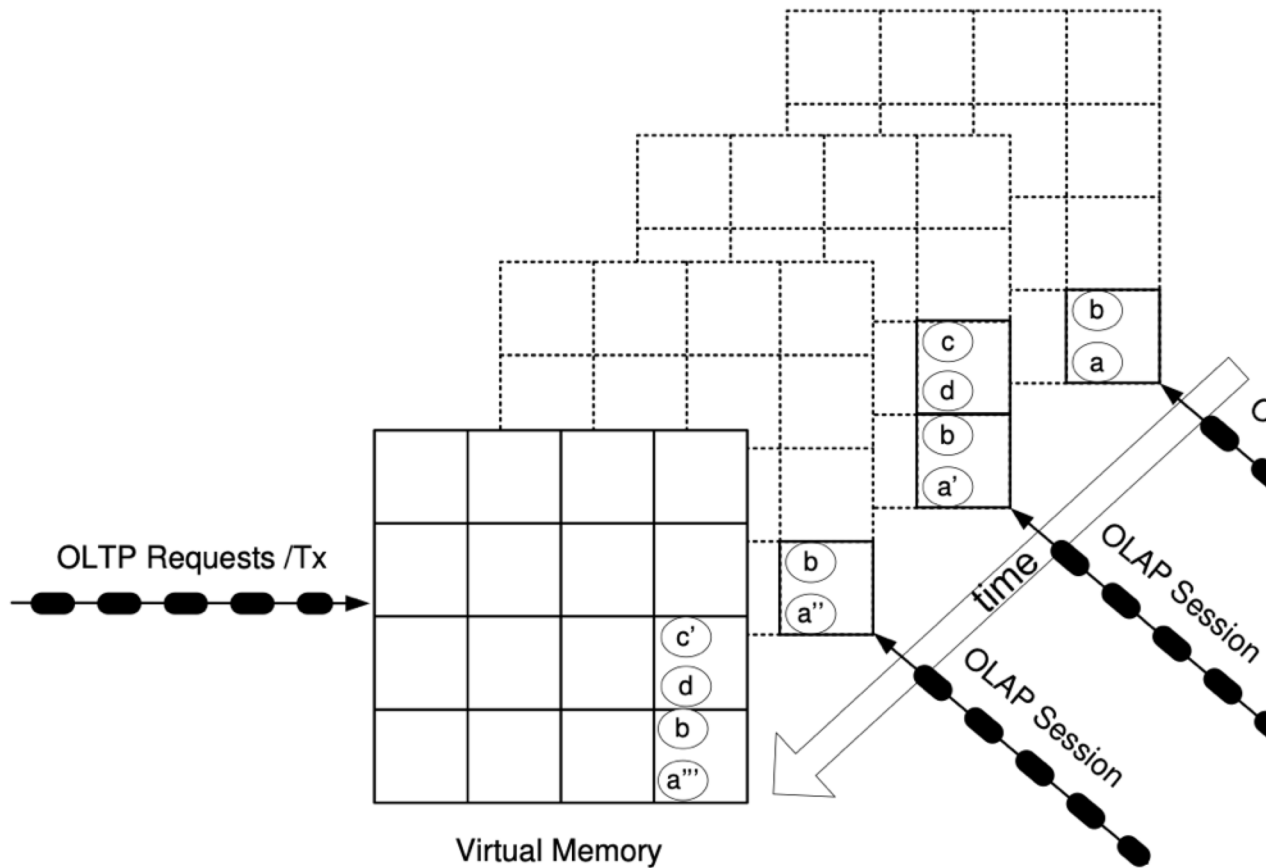
Multiple OLAP Session

OLAP Session: Group of OLAP queries that access the same snapshot



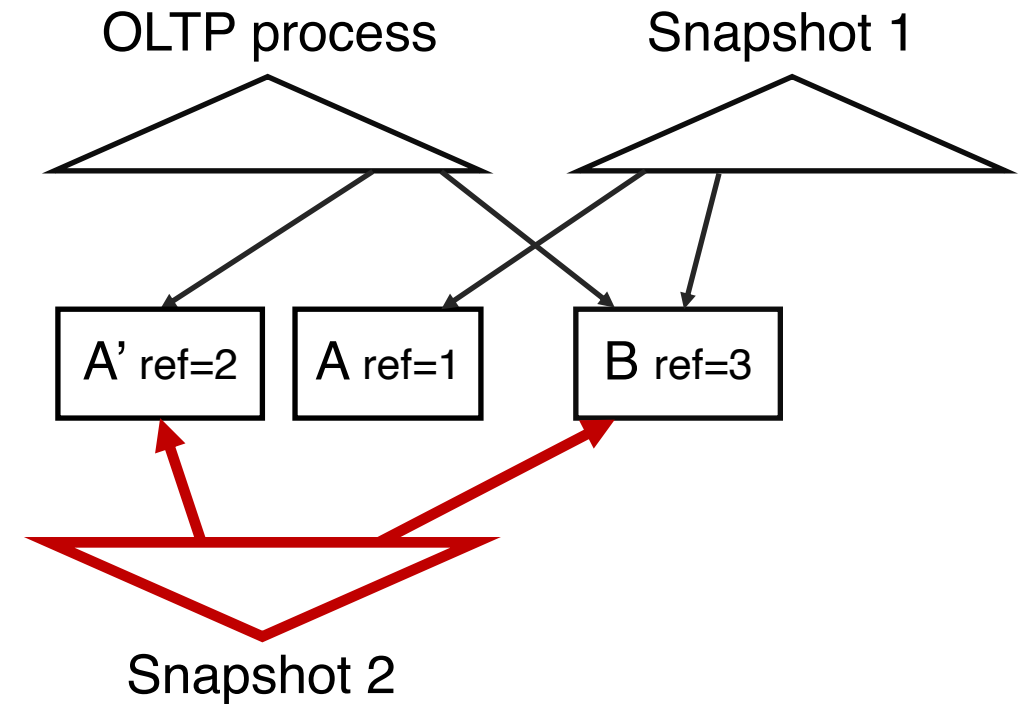
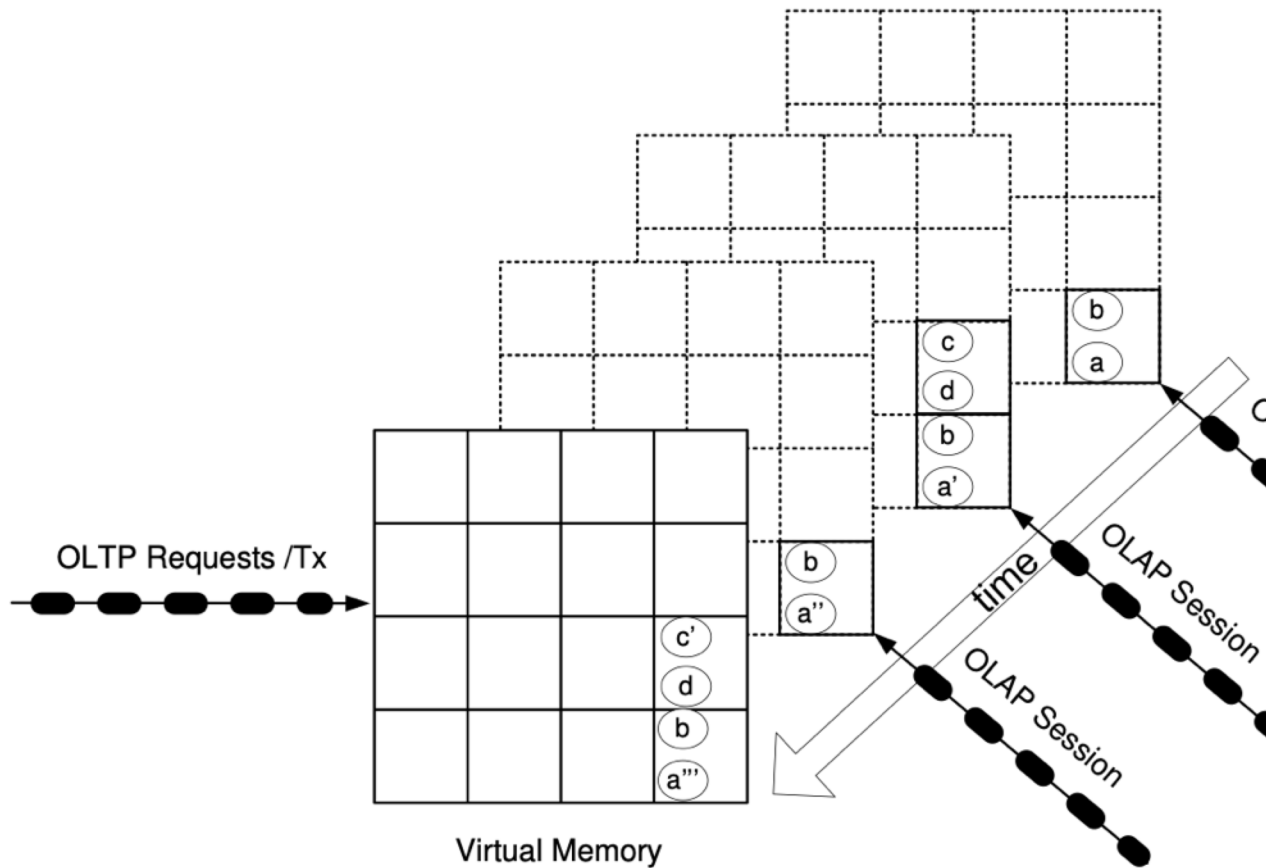
Multiple OLAP Session

OLAP Session: Group of OLAP queries that access the same snapshot



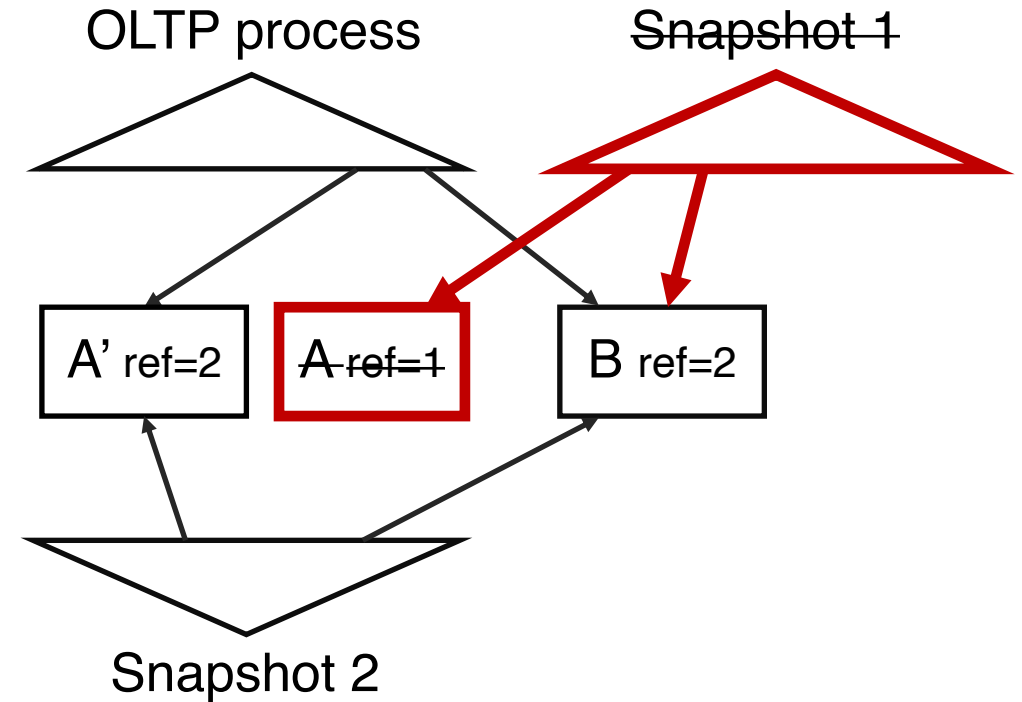
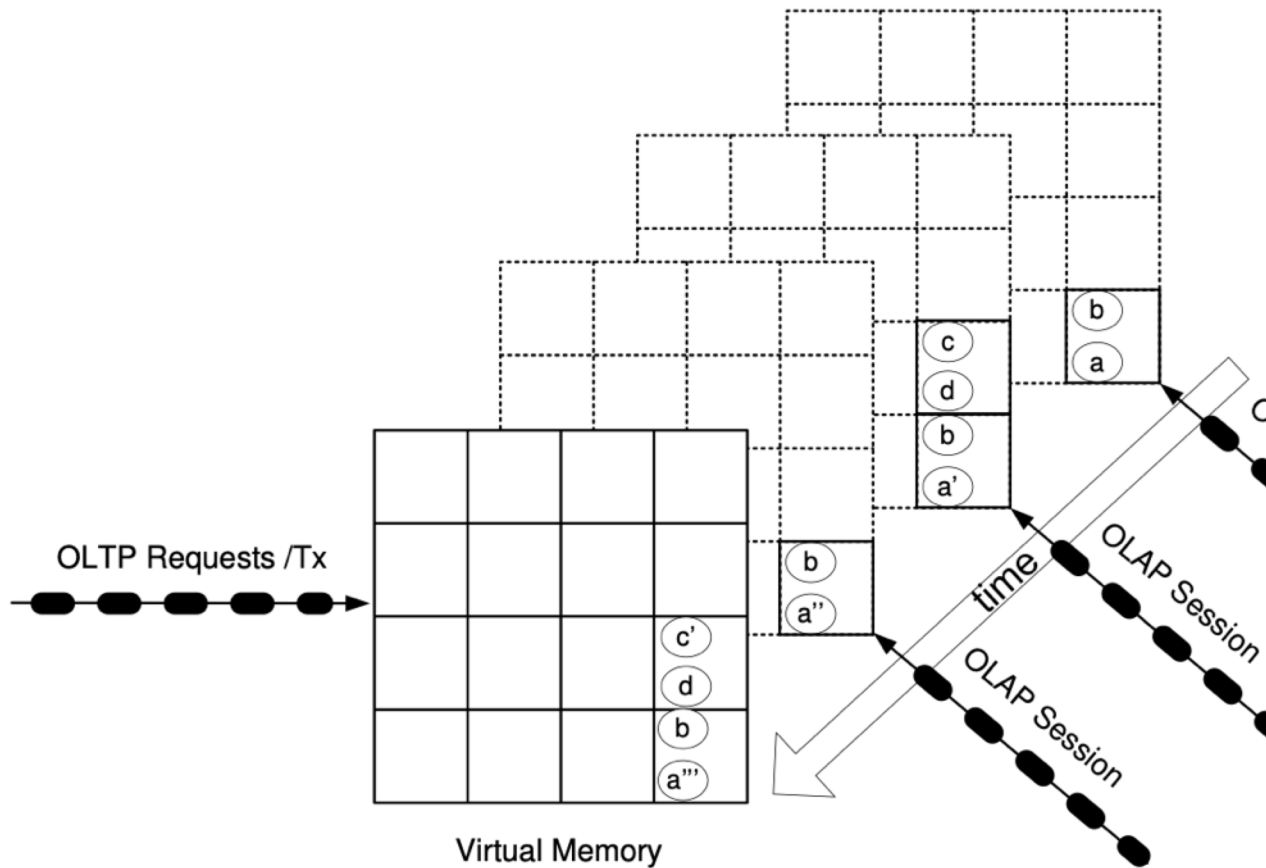
Multiple OLAP Session

OLAP Session: Group of OLAP queries that access the same snapshot

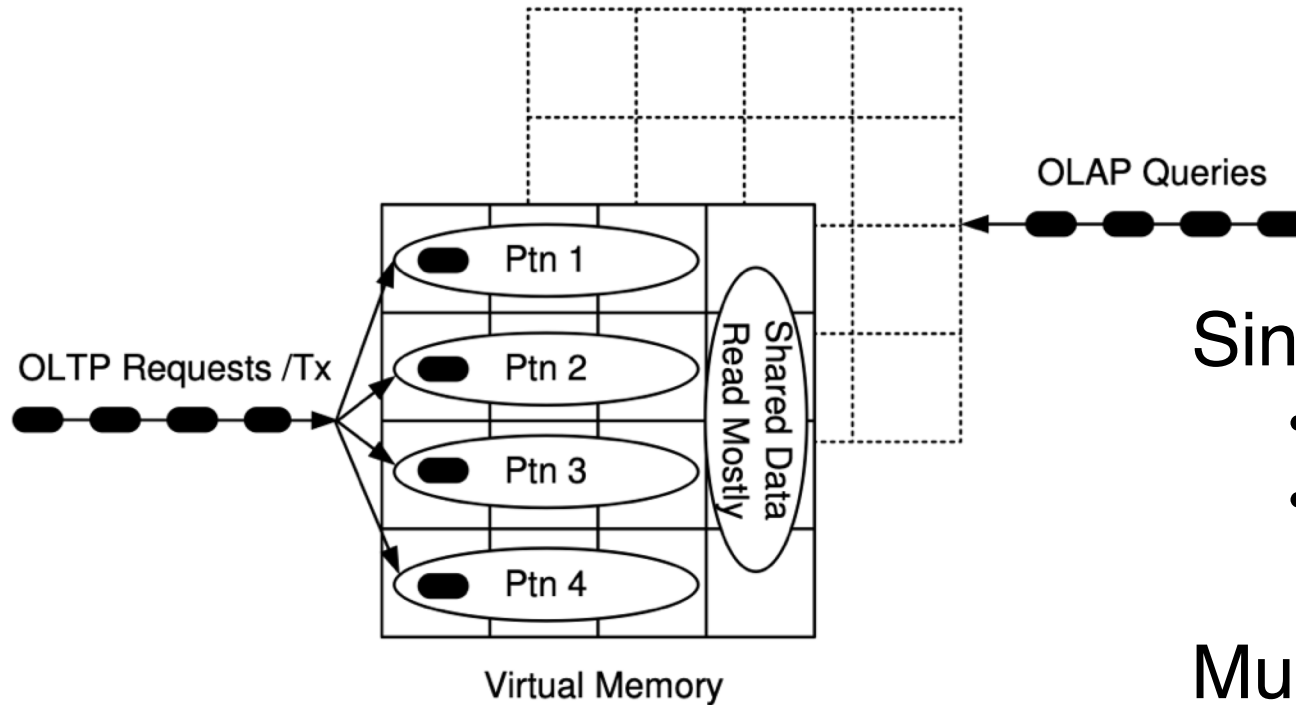


Multiple OLAP Session

OLAP Session: Group of OLAP queries that access the same snapshot



Multi-Threaded OLTP Processing



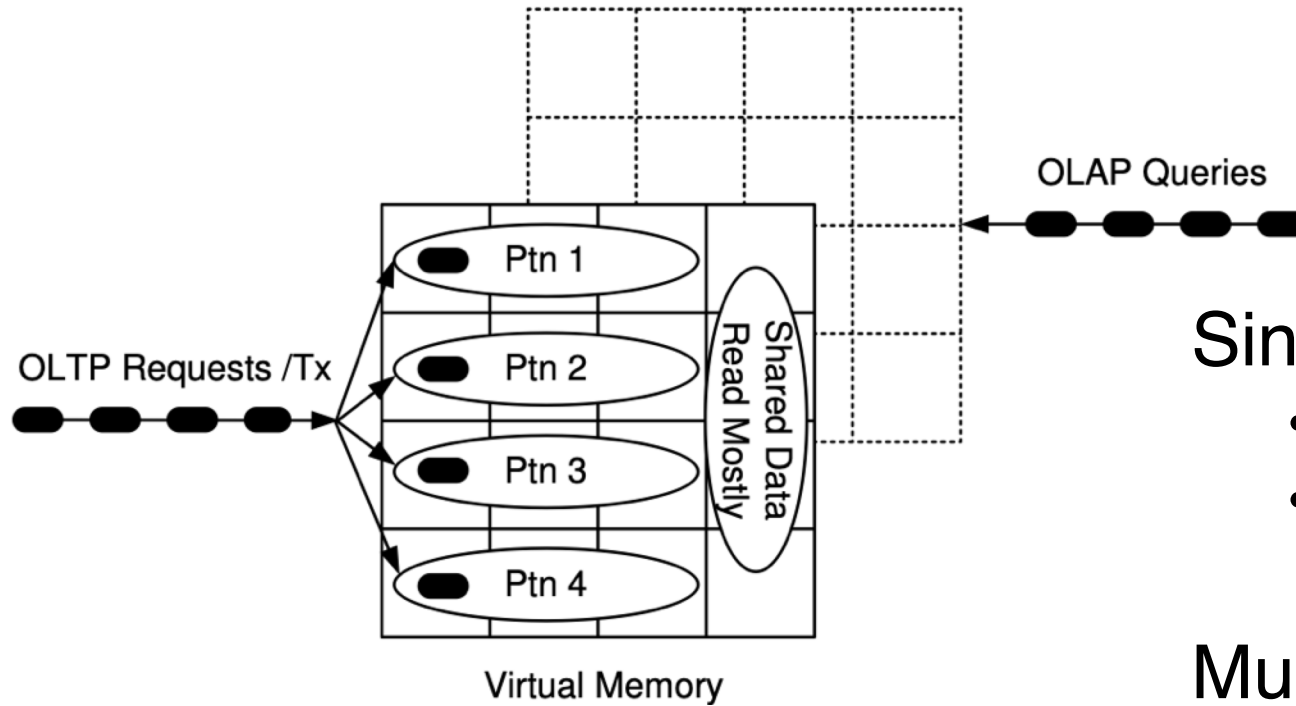
Single-partition transaction

- Sequential execution within partition
- Different partitions run in parallel

Multi-partition transaction

- System-wide sequential execution

Multi-Threaded OLTP Processing



Single-partition transaction

- Sequential execution within partition
- Different partitions run in parallel

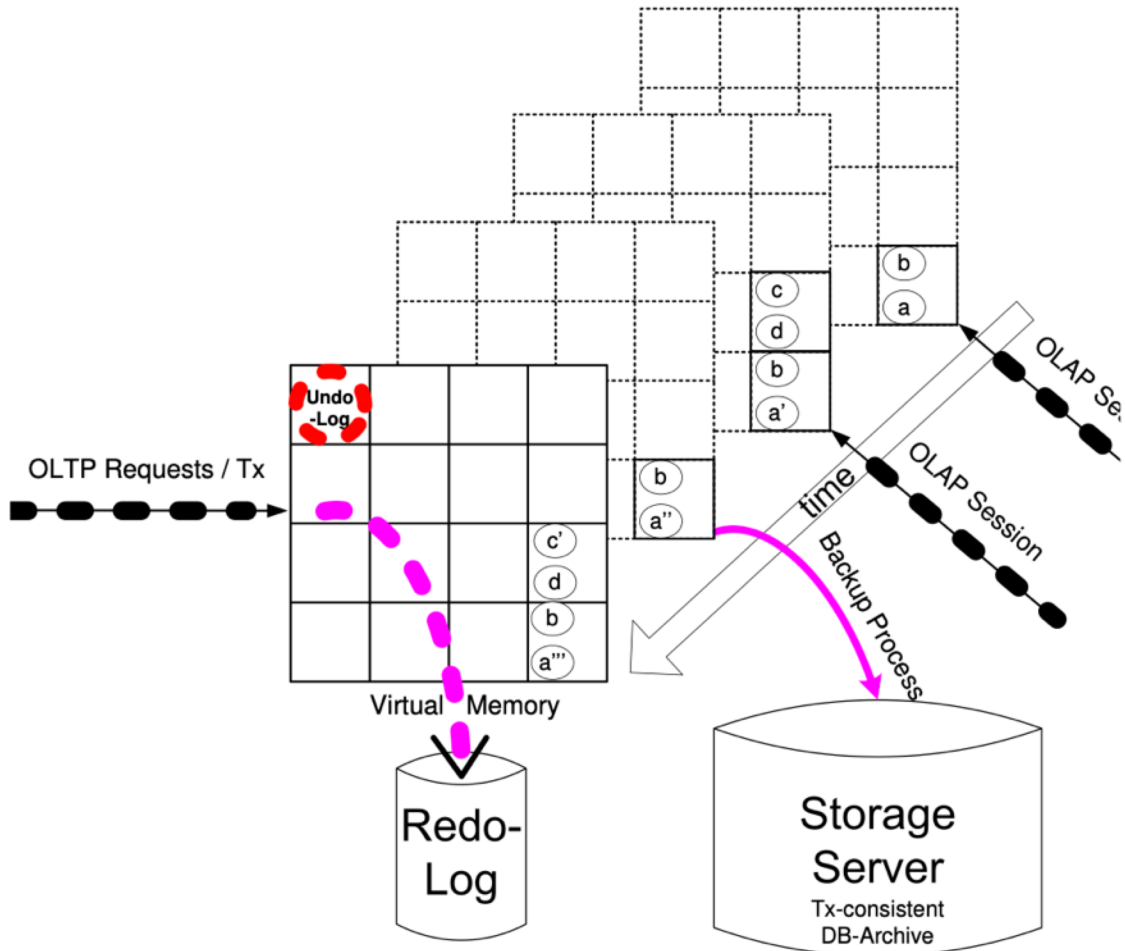
Multi-partition transaction

- System-wide sequential execution

When to fork()?

- Option 1: Fork after quiescing all threads
- Option 2: Fork in the middle of a transaction and then undo the transaction's changes

Logging and Checkpointing



Logging

- Logical redo logging

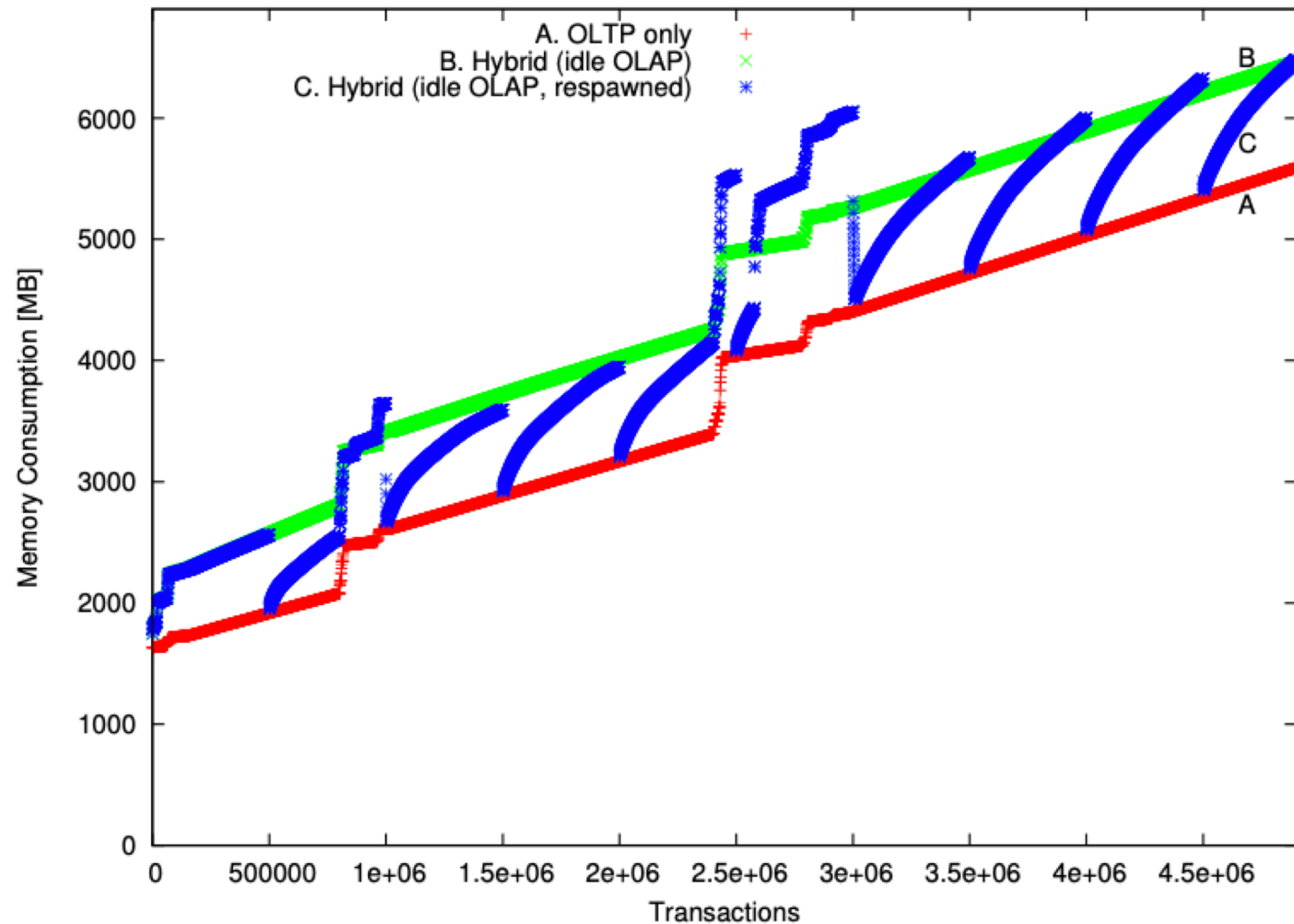
Checkpointing

- Based on the same VM snapshot mechanism

Evaluation – Performance Comparison



Query No.	one query session (stream) single threaded OLTP		HyPer configurations 8 query sessions (streams) single threaded OLTP		3 query sessions (streams) 5 OLTP threads		MonetDB no OLTP 1 query stream	VoltDB no OLAP only OLTP results from [18]
	OLTP throughput	Query resp. times (ms)	OLTP throughput	Query resp. times (ms)	OLTP throughput	Query resp. times (ms)	Query resp. times (ms)	
Q1	new order: 56961 tps; total: 126576 tps	67	new order: 29359 tps; total: 65269 tps	71	new order: 171384 tps; total: 380868 tps	71	63	55000 tps on single node; 300000 tps on 6 nodes
Q2		163		233		212	210	
Q3		66		78		73	75	
Q4		194		257		226	6003	
Q5		1276		1768		1564	5930	
Q6		9		19		17	123	
Q7		1151		1611		1466	1713	
Q8		399		680		593	172	
Q9		206		269		249	208	
Q10		1871		2490		2260	6209	
Q11		33		38		35	35	
Q12		156		195		170	192	
Q13		185		272		229	284	
Q14		122		210		156	722	
Q15		528		1002		792	533	
Q16		1353		1584		1500	3562	
Q17		159		171		168	342	
Q18		108		133		119	2505	
Q19		103		219		183	1698	
Q20		114		230		197	750	
Q21	Config 1	46	Config 2	50	Config 3	50	329	
Q22		7		9		9	141	

Evaluation – Memory Consumption



Hyper Today?

[HyPer](#) [Home](#) [Highlights](#) [Team](#) [Publications](#) [Summary](#) [Contact](#) [WebInterface](#) [Blog](#)

 **HyPer** – A Hybrid OLTP&OLAP High Performance DBMS 

HyPer is a main-memory-based relational DBMS for mixed OLTP and OLAP workloads. It is a so-called all-in-one New-SQL database system that entirely deviates from classical disk-based DBMS architectures by introducing many innovative ideas including **machine code generation** for data-centric query processing and **multi-version concurrency control**, leading to exceptional performance. HyPer's OLTP throughput is comparable or superior to dedicated transaction processing systems and its OLAP performance matches the best query processing engines — however, HyPer achieves this **OLTP and OLAP** performance simultaneously on the **same database** state. Current research focuses on extending HyPer's functionality beyond OLTP and OLAP processing to exploratory workflows that are deeply integrated into the database kernel by utilizing HyPer's pioneering compilation infrastructure.

[Learn more »](#) [WebInterface »](#)

Tableau Acquires HyPer

Innovative High Performance Database System to Integrate with Tableau Products

PUBLISH DATE: MARCH 10, 2016 - 3:00AM

HTAP – Q/A

State-of-the-art in HTAP?

Overhead of Hyper?

Row-format has the same performance as column-format for OLTP?

Really necessary to do real-time analytical work?

What if data does not fit in memory? (Anti-caching)

Why not using shared memory and a concurrency control?

Why logical logging is a problem in conventional system?

Evaluation is weak

Analytical data no longer fits in memory in 2020

Topic of the Last Lecture

Option 1: Streaming

- [required] Discretized Streams: Fault-Tolerant Streaming Computation at Scale
- [optional] Apache Flink™: Stream and Batch Processing in a Single Engine
- [optional] The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

Option 2: Time series

- [required] Gorilla: A Fast, Scalable, In-Memory Time Series Database
- [optional] Time Series Management Systems: A Survey

Group Discussion

What are the challenges of applying the VM-snapshot idea to a shared-memory OLTP system?

Fork() replicates the page table, which is expensive when the database is large. Can you think of any approach to reduce this cost?

Given the four possible designs of HTAP ({single system, separate system} x {shared data, separate data}), which one is the most promising in your opinion? What if you have infinite number of machines?