HyPer: A Hybrid OLTP&OLAP Main Memory Database System

Presenter: Lavanya Subramanian

Need for Online Analytics

- Business intelligence today demands fresh data

- Business analytics of yesterday
  - Transactions are run on an OLTP database
  - OLTP database state extracted periodically
  - Analytics performed on the extracted state

- **The "perform analytics offline" model too stale and slow for today's business intelligence**

How To Perform Online Analytics?

- Run transactions (OLTP queries) and analytics (OLAP queries) on the same machines

- **Problem:** ***Long running analytics queries interfere with transactions***

HyPer: Key Idea

- In-memory database runs transactions & analytics
- Transactions are run on the main database
- Snapshots are created for analytics
  - by forking the OLTP process
- Properties of snapshots created on a fork()
  - Data is not duplicated rightaway
  - ***A page is duplicated only when modified (copy-on-write)***

Basic Transaction Processing Model in HyPer

- Builds on prior work on in-memory transaction processing

- Single-threaded execution is effective enough
  - ***No IO wait times***

- Short transactions
  - No interactive transactions

Analytical Processing in HyPer

Image Credit: Alfons Kemper

How Does Copy on Write Work?

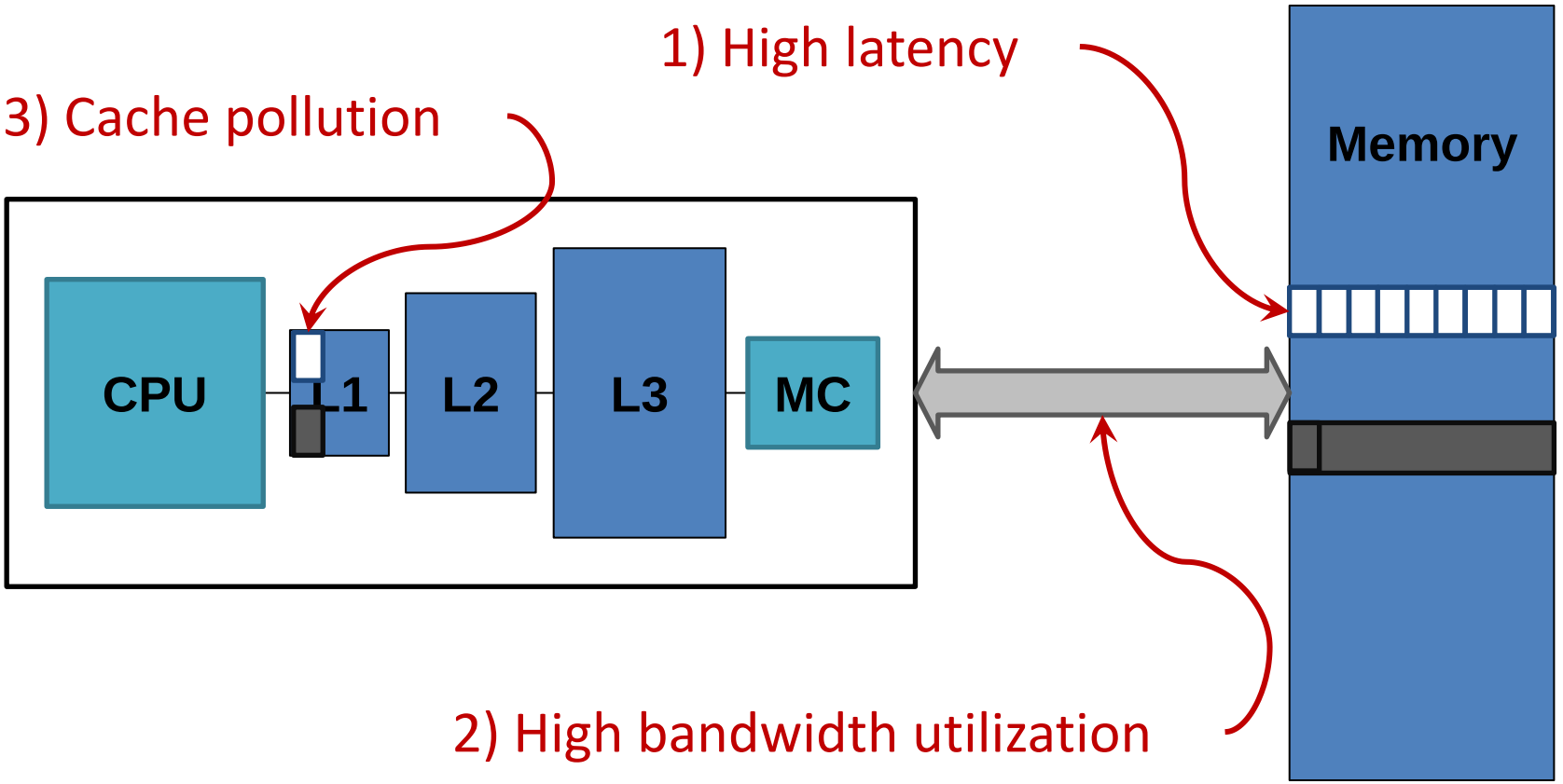


Image Credit: Vivek Seshadri

Hardware Support For Fast Copy-On-Write

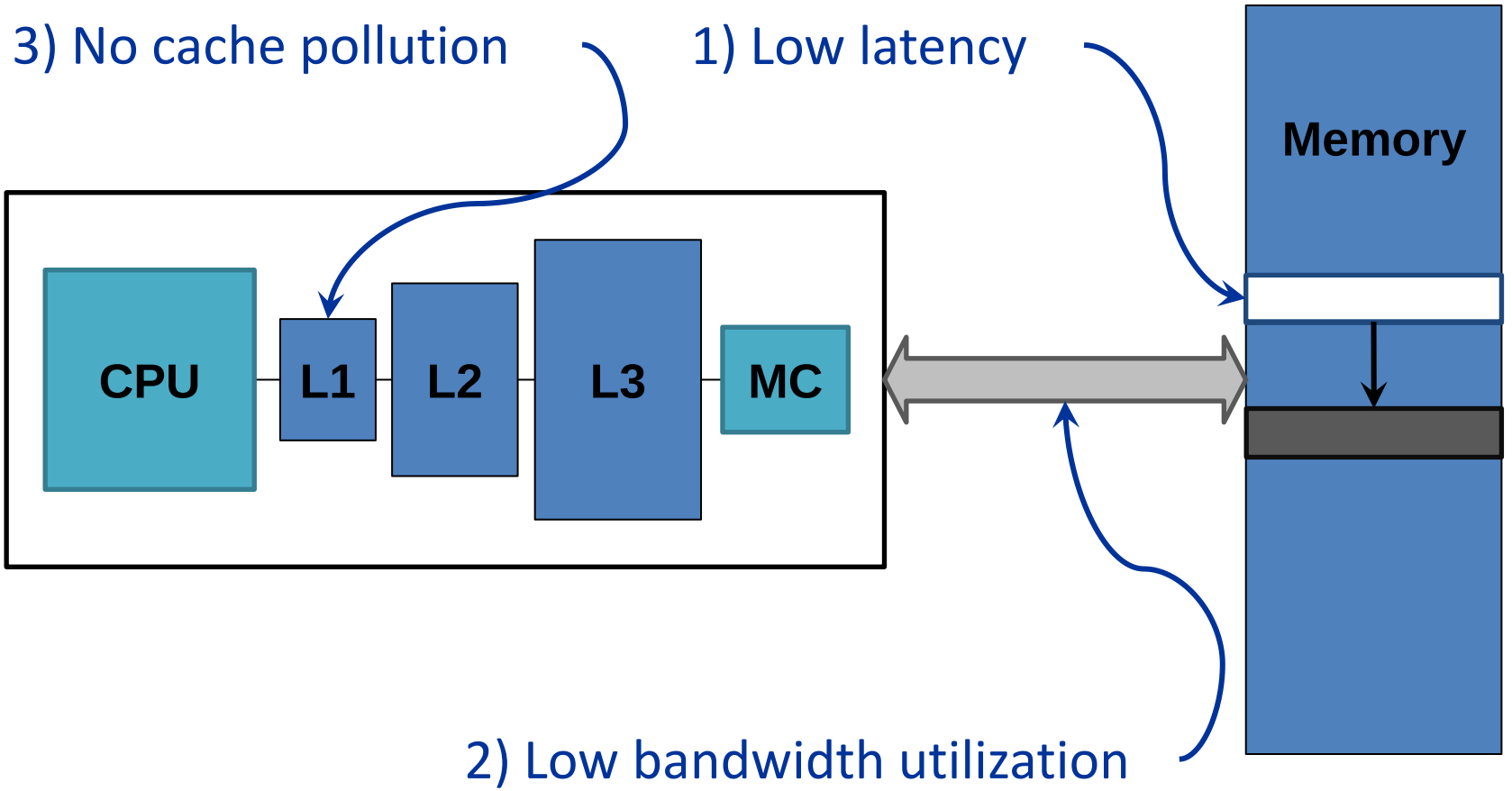


Image Credit: Vivek Seshadri

Parallelizing Analytics and Transactions

Multiple OLAP Sessions

- Snapshots for OLAP
  - Do not consume much space
  - Can be created easily using fork()
- Parallelize OLAP query execution
  - Using multiple snapshots
  - Executing on idle CPU cores
- Snapshot deleted after last query of a session

Multi-Threaded Transaction Processing

- Execute multiple read-only queries in parallel
- Execute read-write queries in parallel
  - Scenarios where data can be partitioned
  - Transactions confined to partitions
- Only one transaction per partition
- Cross-partition transactions run single threaded

More Discussion on Transactions

- Snapshot Isolation
- Durability
- Transaction Consistency

Snapshot Isolation

- Roll-back
  - Roll back when an older query needs older data
- Versioning
  - Create a new object version on every update
  - Retrieve youngest version before query start time
- Shadowing
  - Write updates to a shadow copy
  - Update main copy upon commit
- Virtual memory snapshots

Durability

- On failure recovery, all effects of committed transactions should be restored

- Solution: Logical redo logging
  - Apply log to database after failure recovery

- Redo log can be used to feed a secondary server
  - Potential uses: standby, analytics processing

Transaction Consistency

- Perform Undo logging to obtain a transaction consistent snapshot

- Applied to a snapshot created from a fork()
  - To undo effects of current transactions

Methodology

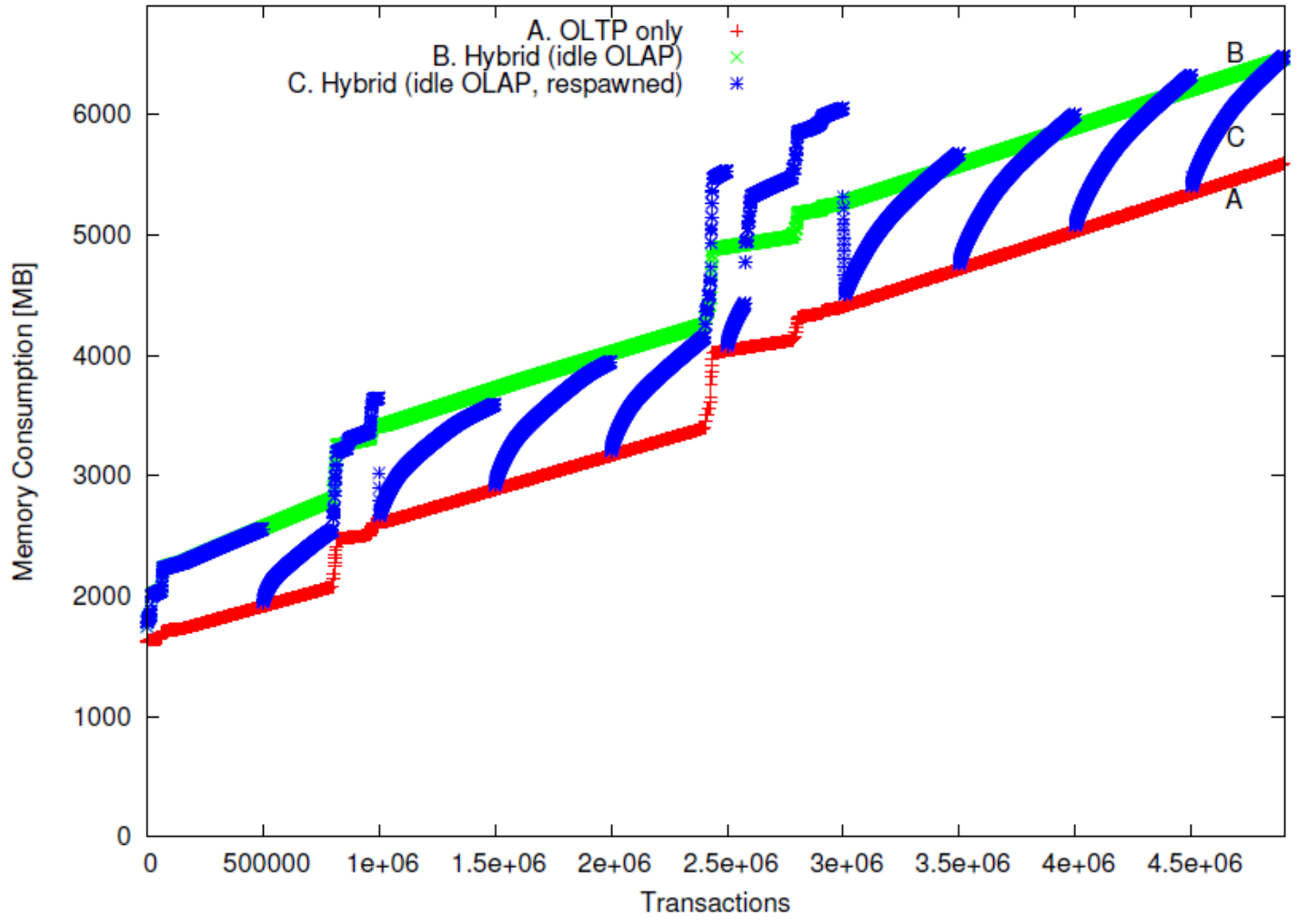
- Benchmark
  - TPC-C scheme
  - Additional three relations from TPC-H
- Hardware
  - Intel X5570 – Quad Core CPU
  - 64 GB DRAM
- Comparison Points
  - MonetDB (for analytics)
  - VoltDB (for transactions)

Results - Performance and Memory Consumption

Query No.	HyPer configurations: one query session (stream) single threaded OLTP — OLTP throughput	Query resp. times (ms)	HyPer configurations: 8 query sessions (streams) single threaded OLTP — OLTP throughput	Query resp. times (ms)	HyPer configurations: 3 query sessions (streams) 5 OLTP threads — OLTP throughput	Query resp. times (ms)	MonetDB no OLTP 1 query stream Query resp. times (ms)	VoltDB no OLAP only OLTP results from [18]
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		46		50		50	329	
Q22		7		9		9	141	

Fig. 9. Performance Comparison: HyPer OLTP&OLAP, MonetDB only OLAP, VoltDB only OLTP

Memory Consumption



Discussion

- Simple mechanism that exploits an existing feature of virtual memory management
- How would memory consumption increase with multiple snapshots?
- Is their OLTP performance evaluation fair?