SVL: Storage Virtualization Engine Leveraging DBMS Technology

Presentation of a paper by L. Qiao, B. Iyer, D. Agrawal and A. El Abbadi from the International Conference on Data Engineering (ICDE'05)

Ken Salem

David R. Cheriton School of Computer Science
University of Waterloo

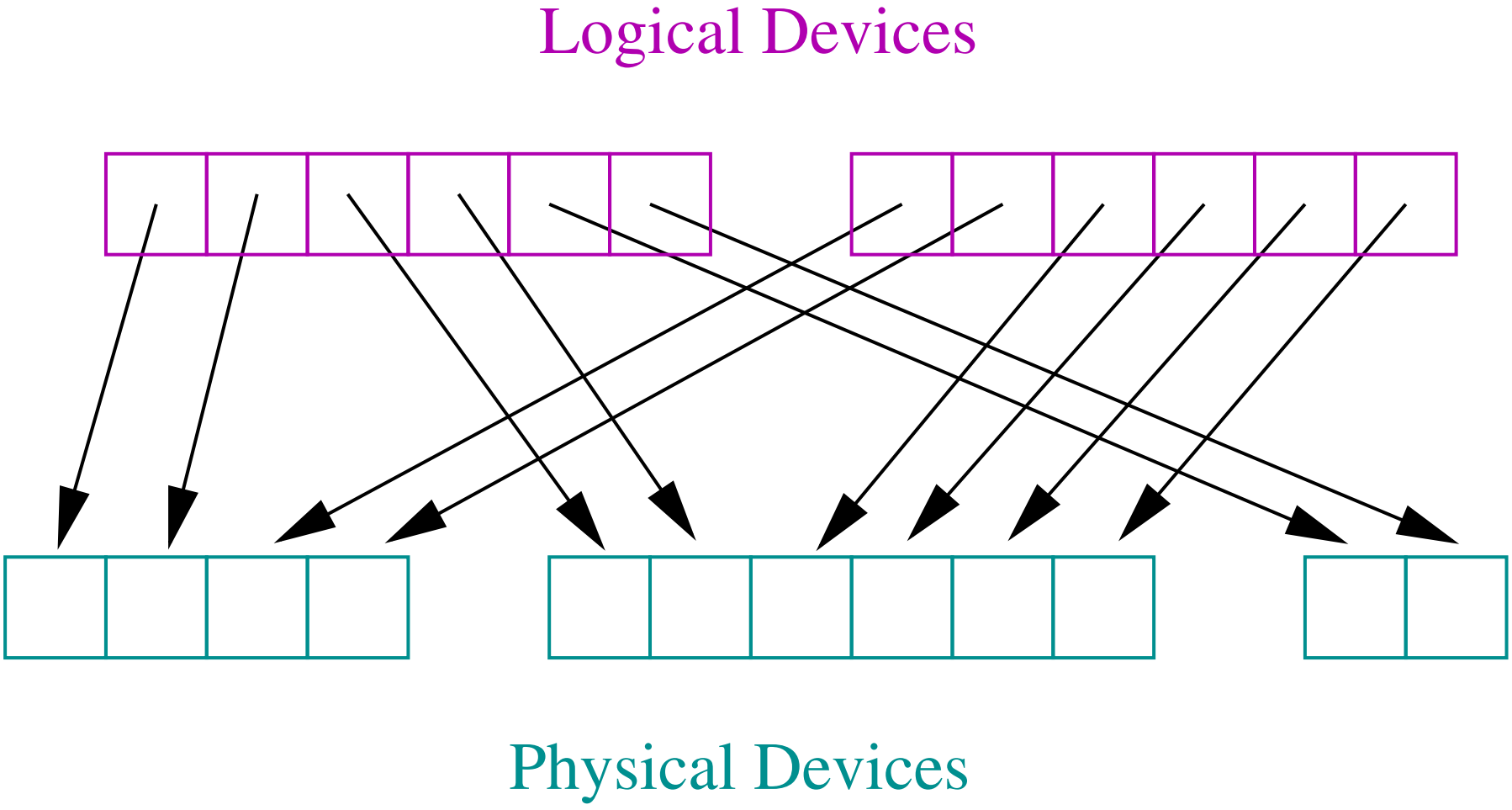
September 27, 2006

Storage Virtualization

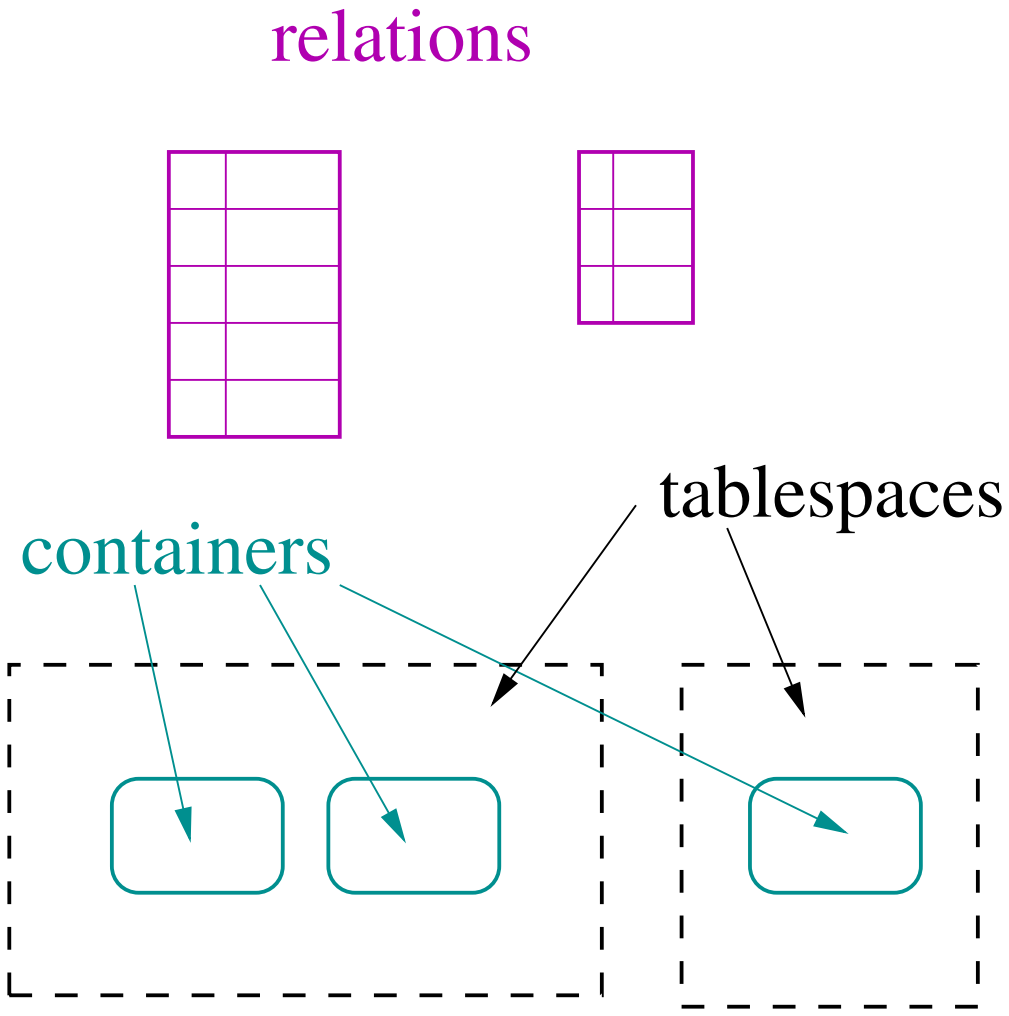
Logical Devices

Physical Devices

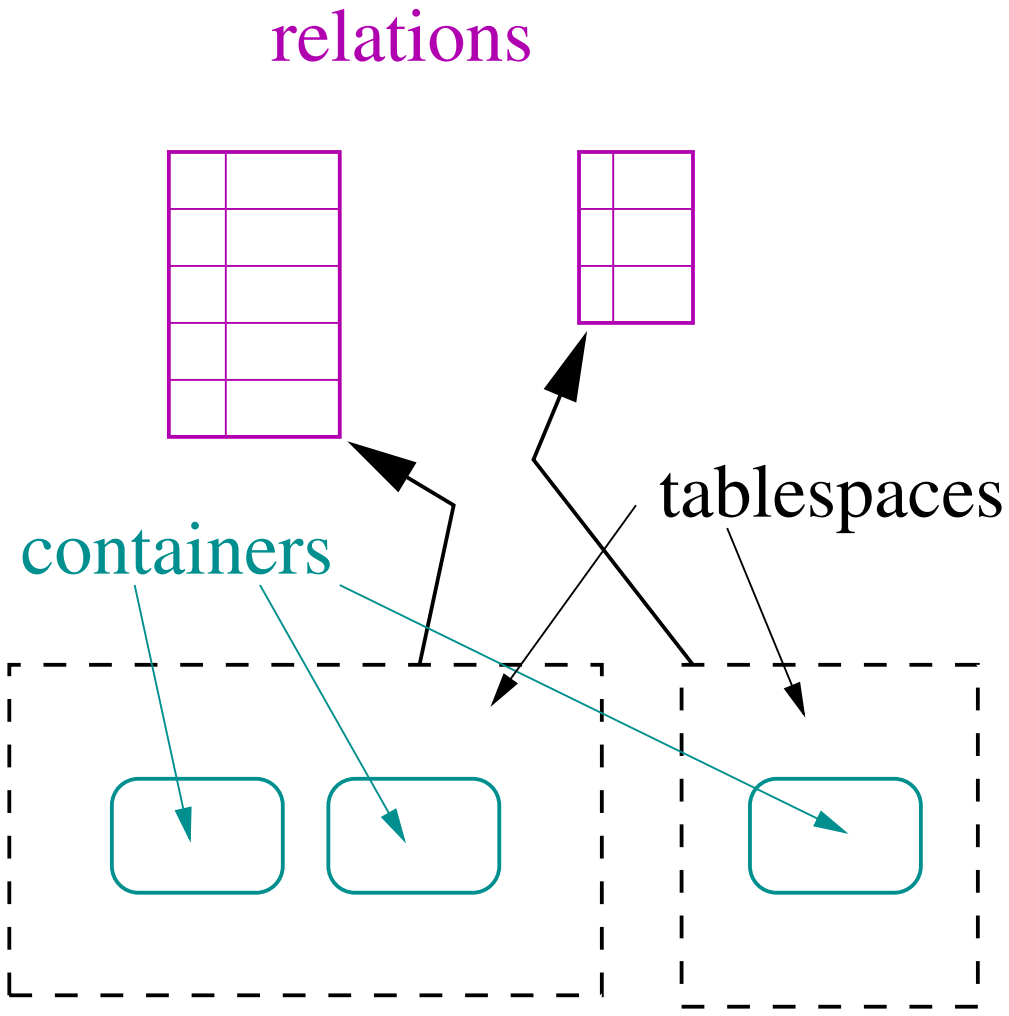
Storage Virtualization



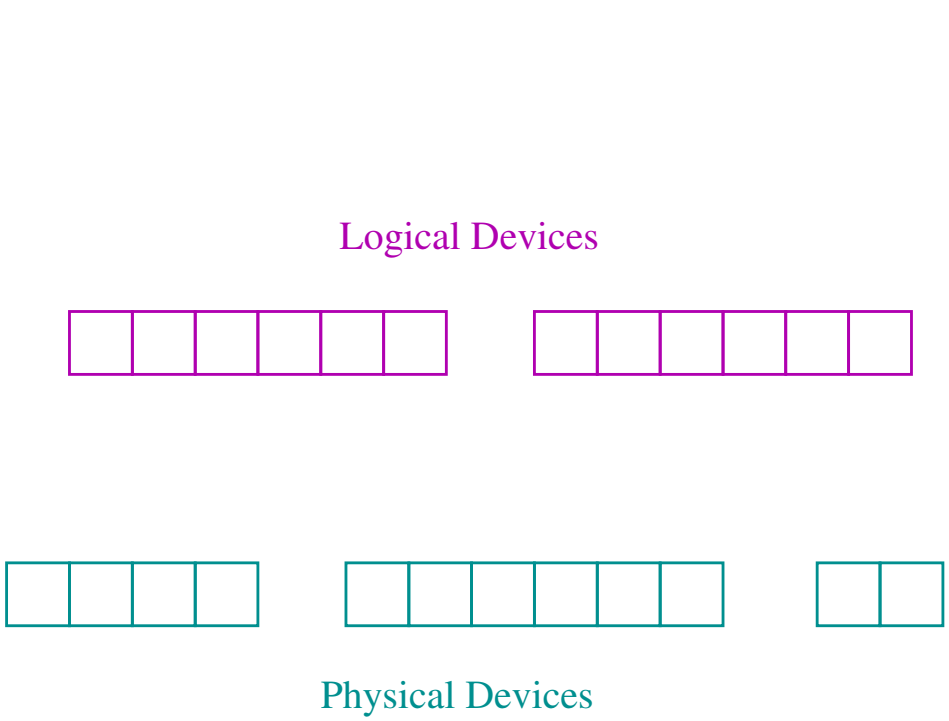
DBMS Storage Mapping

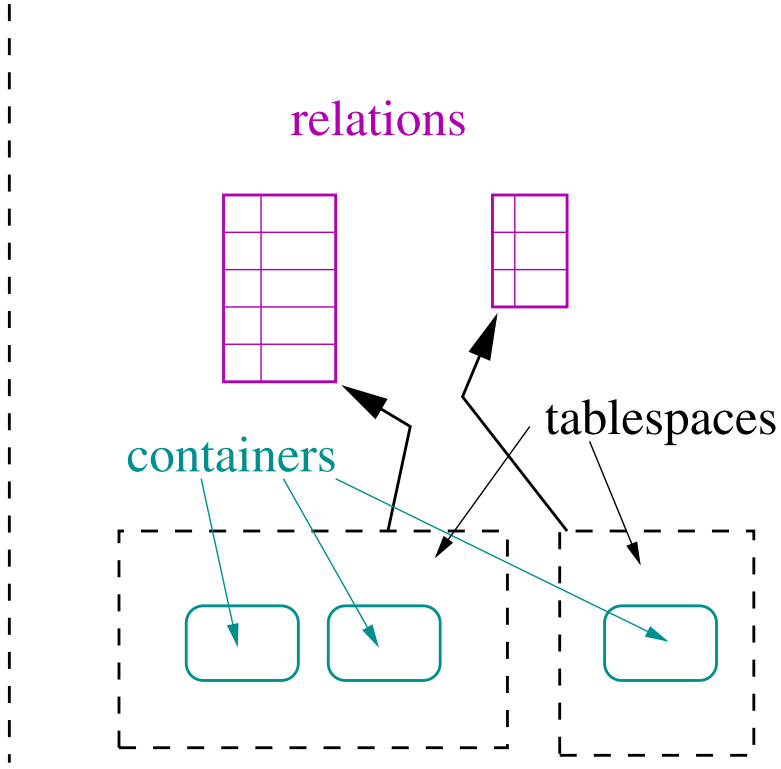


DBMS Storage Mapping

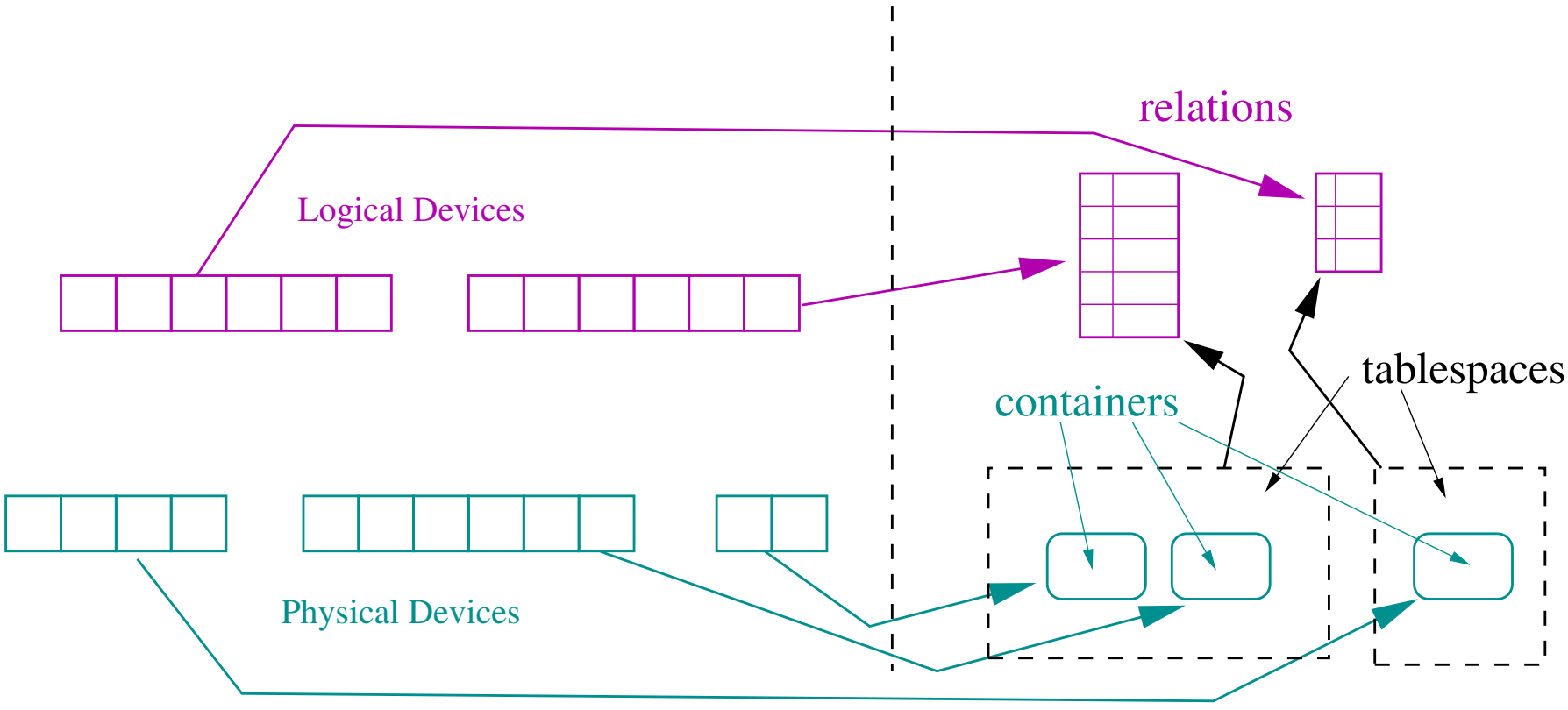


Virtualization Mapping in SVN





Virtualization Mapping in SVN



SVN Table Block Vectors

number	data
1	1 2 3 4 5
6	6 7 8 9 10
11	11 12 13 14 15
16	16 17 18 19 20

- k logical blocks per tuple ($k = 5$ in example)

SVN Table Block Vectors

number	data
1	1 2 3 4 5
6	6 7 8 9 10
11	11 12 13 14 15
16	16 17 18 19 20

- k logical blocks per tuple ($k = 5$ in example)
- k determined by number of logical blocks that fit into one DBMS page

SVN Table Block Vectors

number	data
1	1 2 3 4 5
6	6 7 8 9 10
11	11 12 13 14 15
16	16 17 18 19 20

- k logical blocks per tuple ($k = 5$ in example)
- k determined by number of logical blocks that fit into one DBMS page
- clustered index is created on the block number attribute

SVN Table Block Vectors

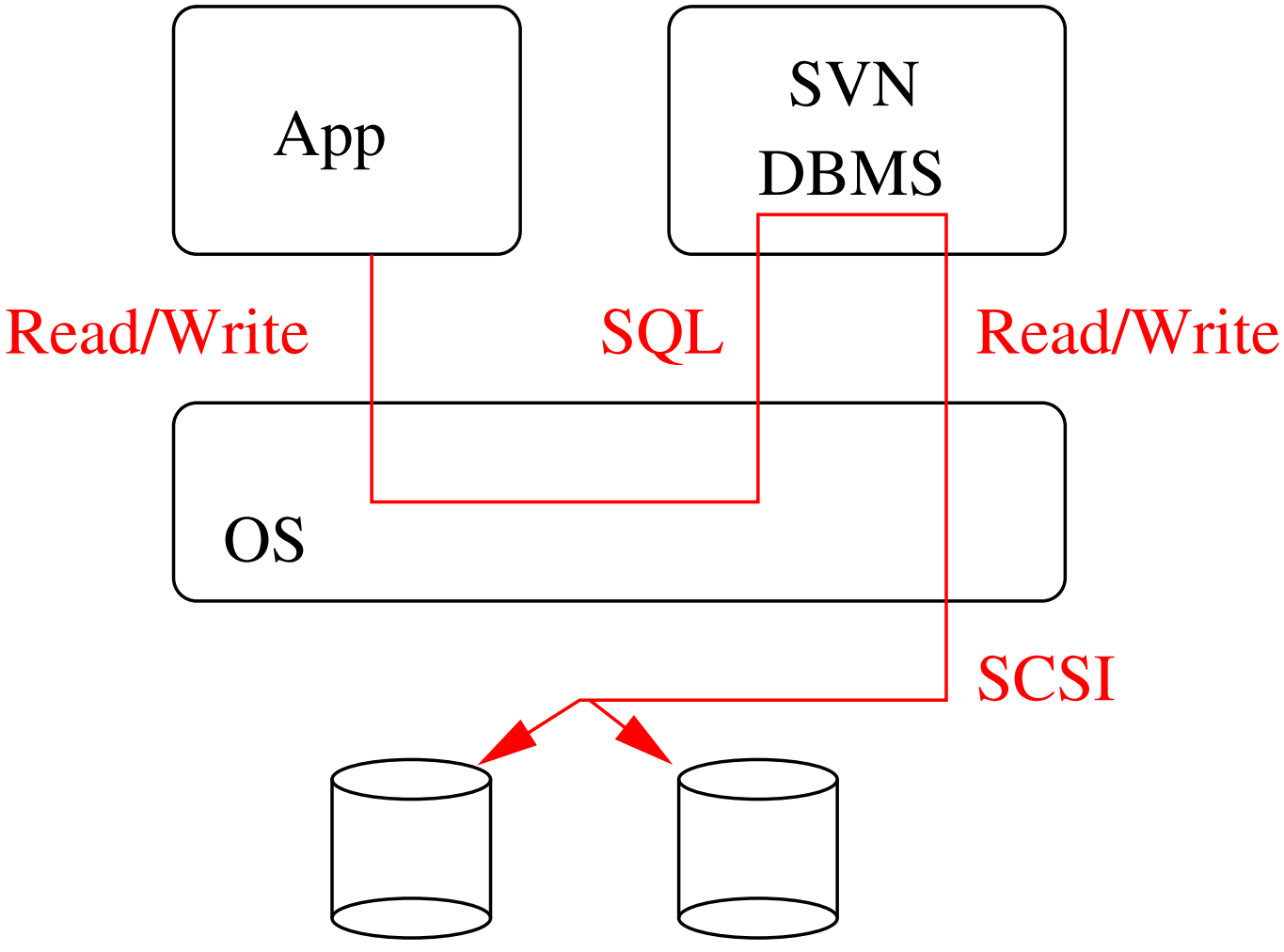
number	data
1	1 2 3 4 5
6	6 7 8 9 10
11	11 12 13 14 15
16	16 17 18 19 20

- k logical blocks per tuple ($k = 5$ in example)
- k determined by number of logical blocks that fit into one DBMS page
- clustered index is created on the block number attribute

Block Read Procedure

Read block x from logical device $i \Rightarrow$
SELECT data FROM T_i WHERE number = x/k

SVN Control Path



What About Performance?

static SQL: pre-compile SQL for block read and write commands

What About Performance?

static SQL: pre-compile SQL for block read and write commands

Table Direct Access: eliminate B-tree on `number` in favour of direct computation of tuple ID.

What About Performance?

static SQL: pre-compile SQL for block read and write commands

Table Direct Access: eliminate B-tree on `number` in favour of direct computation of tuple ID.

CONCAT SQL aggregation function:

- aggregates logical blocks from multiple tuples
- returns data through a shared memory side channel

What About Performance?

static SQL: pre-compile SQL for block read and write commands

Table Direct Access: eliminate B-tree on `number` in favour of direct computation of tuple ID.

CONCAT SQL aggregation function:

- aggregates logical blocks from multiple tuples
- returns data through a shared memory side channel

DECONCAT SQL scalar function: like CONCAT, but for Writes

Bonus Features

encryption: apply built-in ENCRYPT/DECRYPT DBMS functions to data in each tuple

Bonus Features

encryption: apply built-in ENCRYPT/DECRYPT DBMS functions to data in each tuple

compression: apply built-in COMPRESS/UNCOMPRESS DBMS functions to data in each tuple

Bonus Features

encryption: apply built-in ENCRYPT/DECRYPT DBMS functions to data in each tuple

compression: apply built-in COMPRESS/UNCOMPRESS DBMS functions to data in each tuple

Warning

These transformations turn fixed-length data into variable-length data. This breaks Table Direct Access.

Experimental Results

- experiments compared SVM to “virtual shared disk” (VSD) in AIX
- workload provided by several block I/O request traces
- SVM had 10GB tablespace and two raw device containiners. VSD had one 10GB logical device from two physical devices.

Experimental Results

- experiments compared SVM to “virtual shared disk” (VSD) in AIX
- workload provided by several block I/O request traces
- SVM had 10GB tablespace and two raw device containiners. VSD had one 10GB logical device from two physical devices.
- Without optimizations, SVM required 30 times more CPU time than VSD on reads and writes.

Experimental Results

- experiments compared SVM to “virtual shared disk” (VSD) in AIX
- workload provided by several block I/O request traces
- SVM had 10GB tablespace and two raw device containiners. VSD had one 10GB logical device from two physical devices.
- Without optimizations, SVM required 30 times more CPU time than VSD on reads and writes.
- With all optimizations, SVM and VSD CPU times were comparable (12% slowdown on writes). SVM had about a 40% latency penalty on writes, 0-10% for reads.

So What?

- Driving tacks with a sledgehammer?

So What?

- Driving tacks with a sledgehammer?
- Storage virtualization is more than a mechanism for logical-to-physical mapping. (Provisioning, management/admin tools...) Does DBMS implementation help with these?

So What?

- Driving tacks with a sledgehammer?
- Storage virtualization is more than a mechanism for logical-to-physical mapping. (Provisioning, management/admin tools...) Does DBMS implementation help with these?
- How flexible are tablespace mappings in DBMS?