Large-Scale and Multi-Structured Databases
In Memory Databases

Prof Pietro Ducange

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB
Innovation for industry 4.0

Not Only Big Data and Scalable Solutions

The design of a database solution is driven by the application *requirements*.

Thus, if our application ***does not need*** to handle huge amount of data and to scale, both vertically and horizontally, we can think about adopting ***different solutions*** than the ones discussed so far.

In-memory databases may be suitable for applications whose handled data may ***fit all*** in the memory of a single server.

Moreover, there also can be databases that can reside in the memory capacity of a cluster (***if replication is needed***).

How Much Does Memory Cost?

Source: http://www.jcmit.com/memoryprice.htm
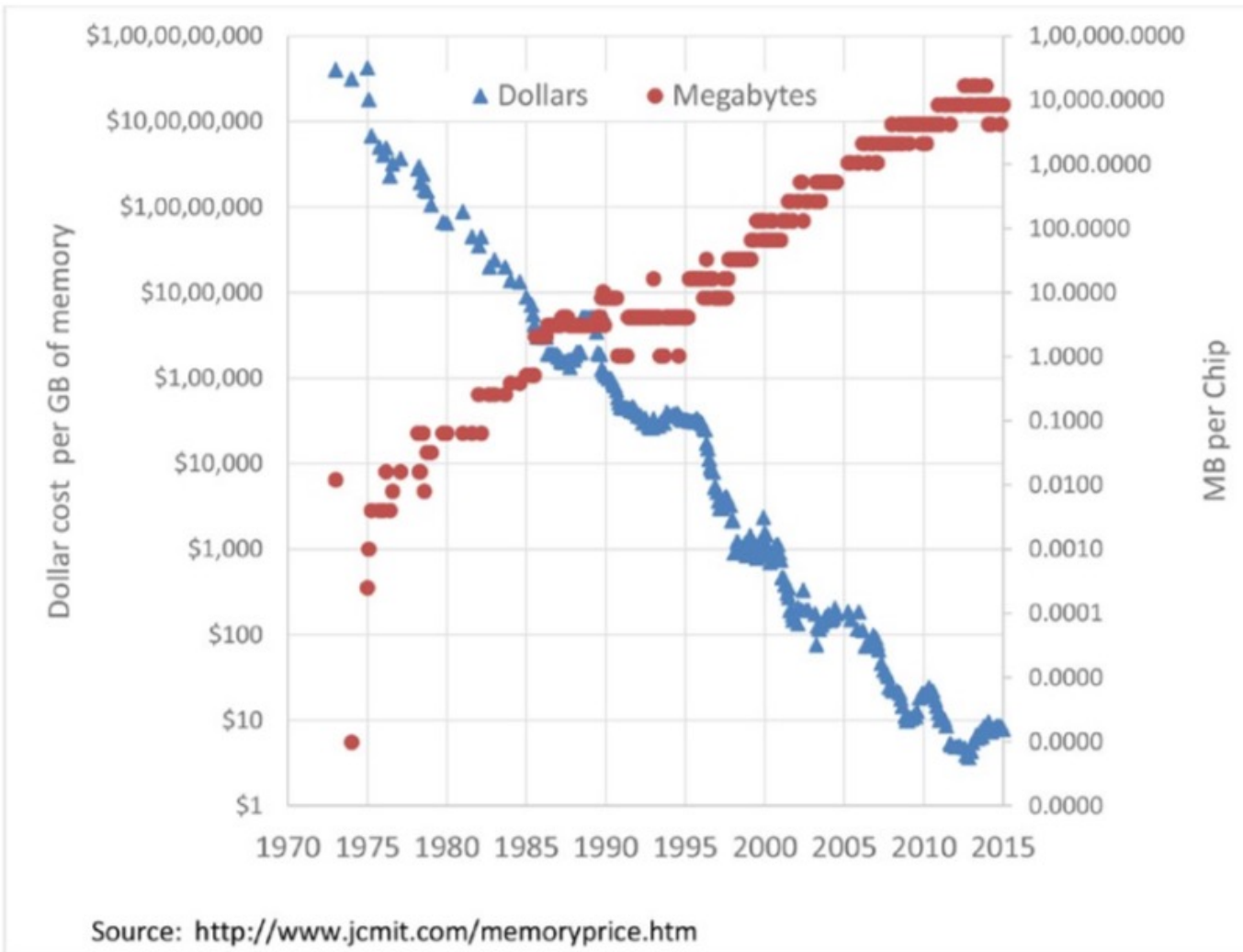
Features of In-Memory Databases

Very *fast access* to the data, avoiding the bottleneck of the I/O transfers.

Suitable for application that do not need *to write continuously* to a persistent medium.

Ad-hoc architecture for being aware that its data are resident in memory and for *avoiding the data loss* in the event of a power failure (*alternative persistency model*).

Cache-less architecture: despite traditional disk-based databases, a memory caching systems is not need (everything is already in memory!!!).

Solutions for Data Persistency

In the following, we show some solutions for avoiding data loss in case of *system failure*:

- Writing complete database *images* (called snapshots or checkpoints) to disk files.

- Writing out transaction/operation records to an append-only disk file (called a *transaction log* or *journal*).

- *Replicating data* to other members of a cluster.

TimesTen (Oracle Solution)

TimesTen implements a fairly familiar **SQL-based** relational model.

All data is **memory resident.**

Persistence is achieved by writing:

- **periodic snapshots** of memory to disk

- **transaction log** following transaction commits.

By default, the disk writes are asynchronous! To ensure ACID transactions the database can be configured for synchronous writes to a transaction log after each write (performance loss).

# TimesTen Architecture
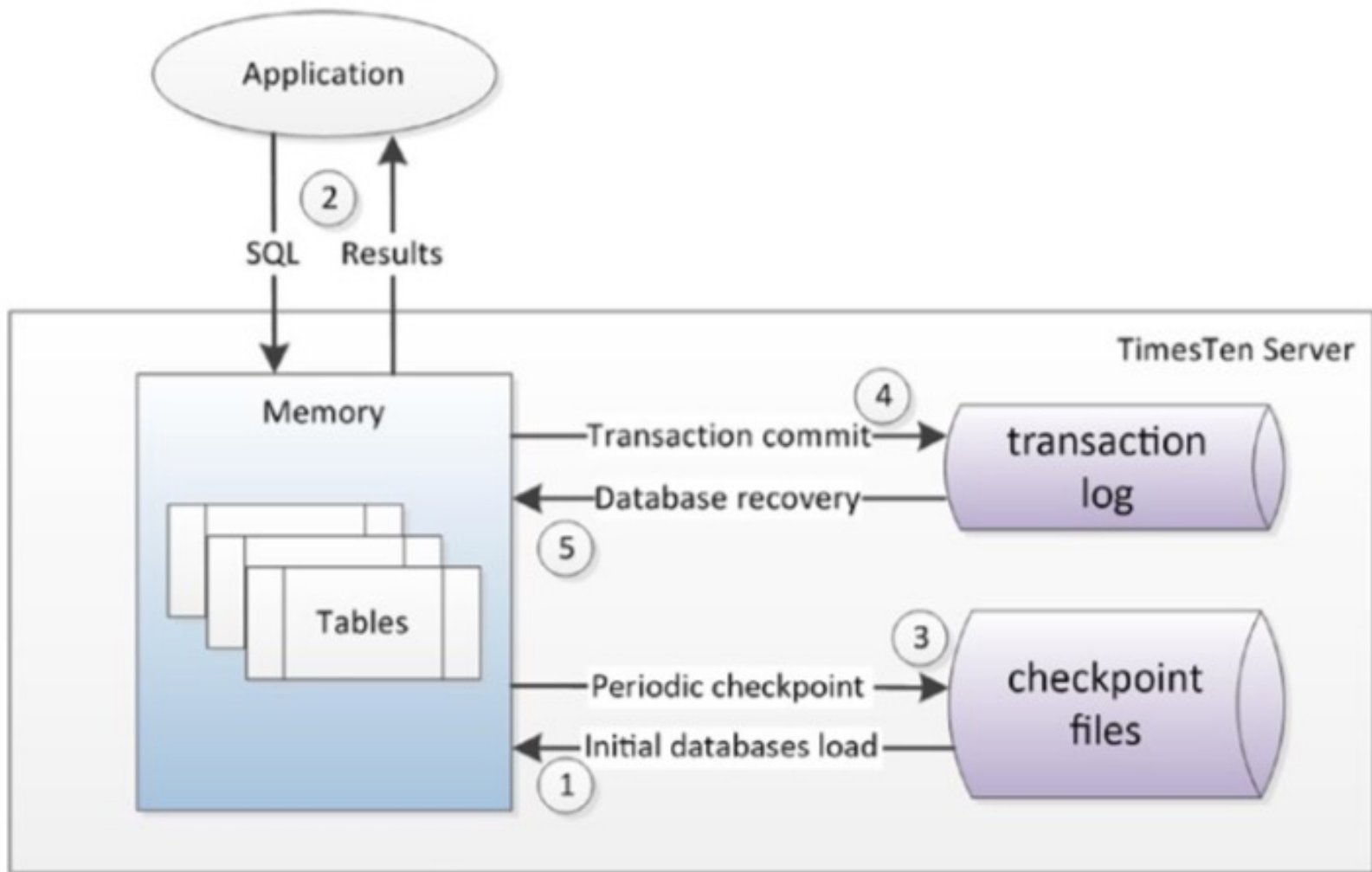
Redis: Main Features

Redis is an in-memory **key-value** store.

Objects consist mainly of strings and various types of collections of strings, such as **lists** and **hash maps**.

Only **primary key** lookups are supported (**special attention should be paid for managing secondary indexing**).

Redis may exploit a sort of "**virtual memory**" (**deprecated**) mechanism for storing an amount of data larger than the server memory. Old keys are **swapped out** to the disk and recovered if needed (**lost of performance**).

Redis: Architecture
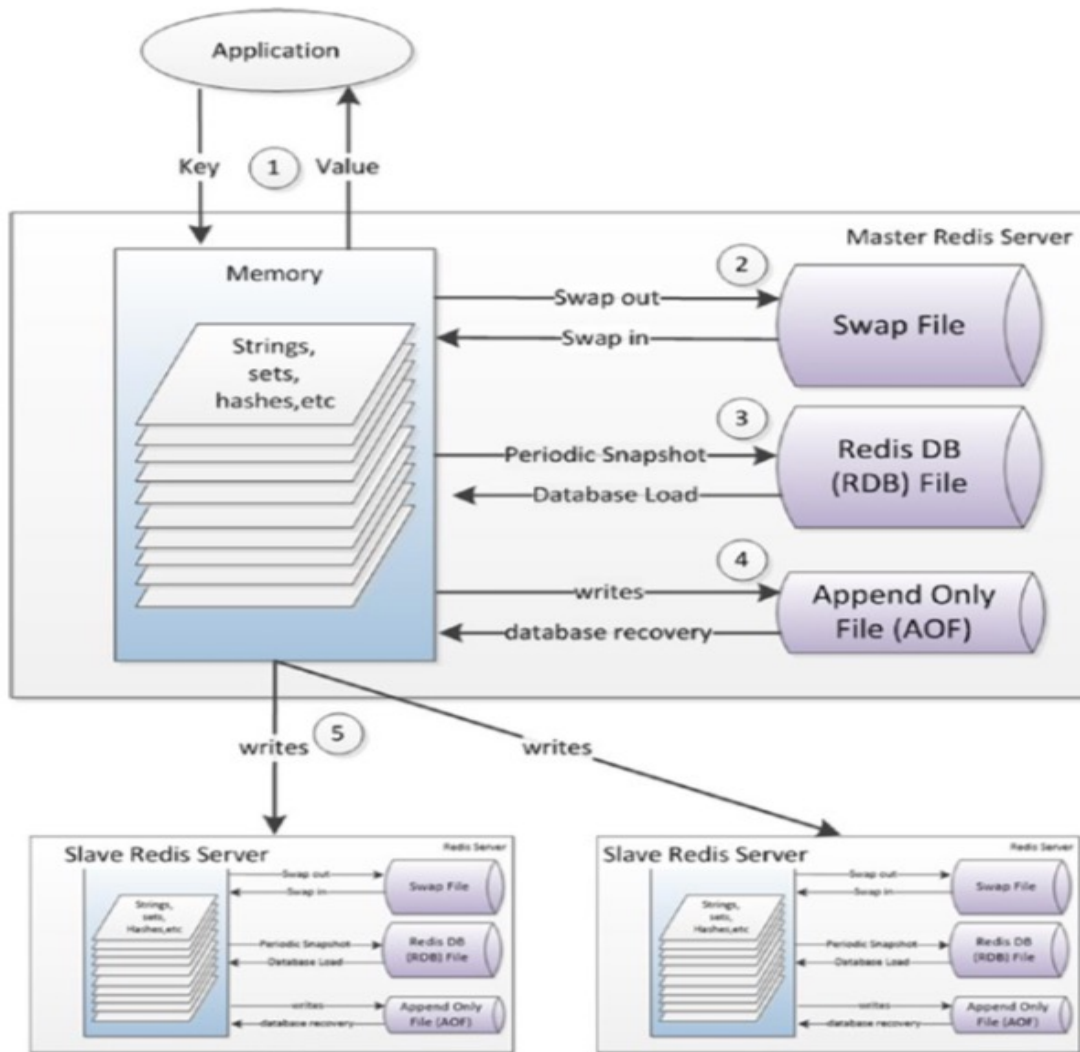
Redis supports asynchronous master/slave **replication**.

If **performance is very critical** and **some data loss** is acceptable, then a replica can be used as a backup database

In this case, the master is configured with **minimal disk-based persistence**.

Redis may replicate the state of the master database asynchronously to slave Redis servers.

HANA DB (an SAP product)

SAP HANA is *a relational database* which combines an *in-memory* technology with a *columnar* storage solution.

It must be installed on an *optimized hardware configuration*.

Classical *row tables* are exploited for *OLTP* operations and are stored *in memory*.

Columnar tables are often used for also *OLAP* operations. By default, they are *loaded in memory by demand*. However, the database may be configured to load in memory a set of columns or entire tables since the beginning.

HANA DB: Architecture
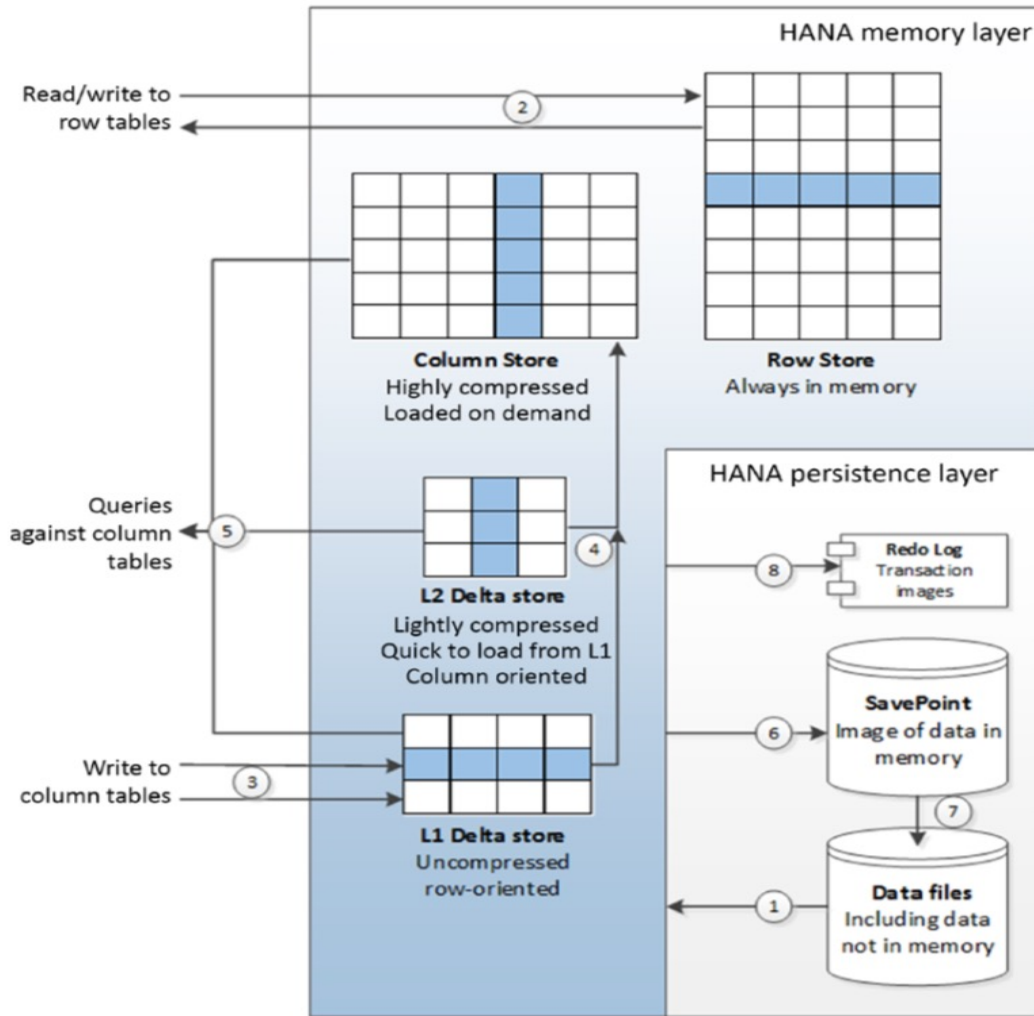
Persistence:

- Images of memory are copied to save points periodically (6)-

- The save points are merged with the data files in due course (7).

- When a commit occurs, a transaction record is written to the redo log (8) (on fast SSD).

Oracle 12c: Architecture

Data is maintained in disk files (1), but *cached in memory* (2).

An *OLTP* application primarily reads and writes from memory (3),

Any *committed transactions* are written immediately to the transaction log on disk (4).

Row data is loaded into a *columnar representation* for use by *analytic applications* (5).

Any transactions that are committed once the data is loaded into columnar format are *recorded in a journal* (6),

Analytic queries will *consult the journal* to determine if they need to read updated data from the row store (7) or to rebuild the columnar structure.
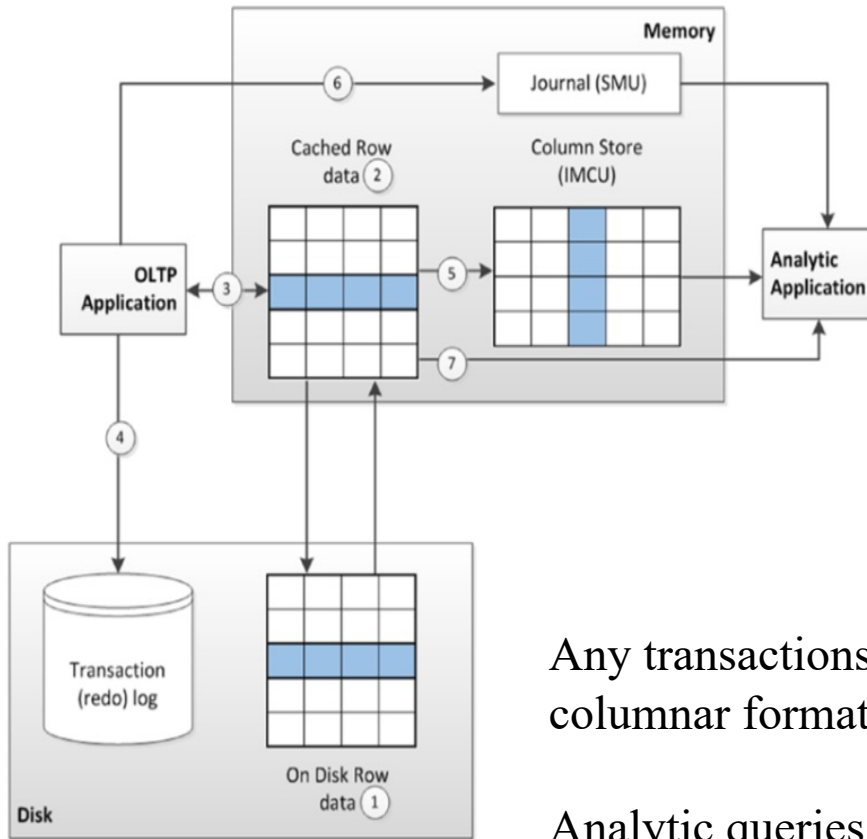
Suggested Readings

Chapter 7 of the book "*Guy Harrison, Next Generation Databases, Apress, 2015*"

Images

All the images used in this presentation have been extracted from:

Chapter 7 of the book "*Guy Harrison, Next Generation Databases, Apress, 2015*"