CSCI317 Database Performance Tuning

In-Memory, GPU, and NVM Database Systems

Dr Janusz R. Getta

School of Computing and Information Technology - University of Wollongong

In-Memory, GPU, and NVM Database Systems

Outline

In-Memory Database System ? What it it ?

The cost of transient memory has been continuously decreasing and the amount of transient memory that can be used by a database server has been continuously increasing

Many databases of modest size can now be comfortably be stored within the transient memory of a single server

An In-Memory database (`IMDB`) stores data in a computer's transient memory instead of persistent storage to produce quicker response times

A central problem is how to make data stored for a long time in transient memory safe

In a traditional database system, `COMMIT` operation the writes the modifications performed on a database to a transaction log in persistent storage

Taking full advantage of a large transient memory system requires an architecture of database system that makes data persistent despite a power failure

In-Memory Database System ? What it it ?

In-Memory database systems must address the following architectural problems

- Cache-less architecture because there is no point caching in memory what is already stored in memory
- Alternative persistence model to ensure that no data loss occurs due to power failure

In-memory databases use the following techniques to ensure that data kept in transient memory is not lost

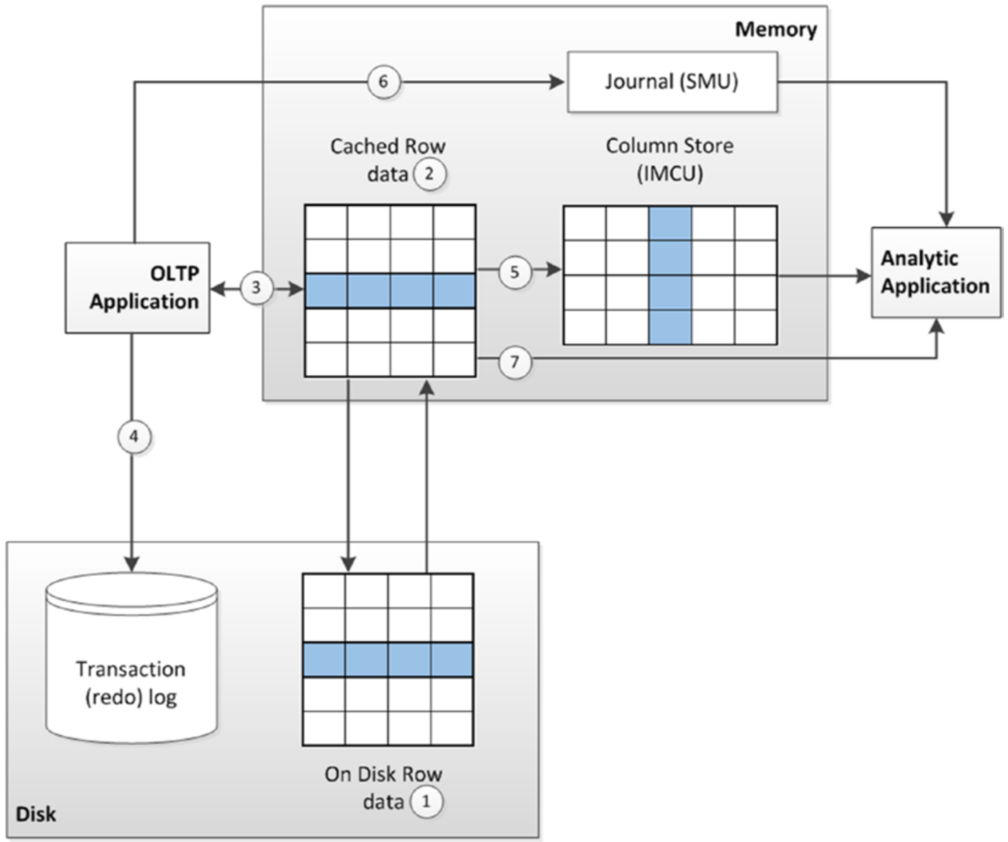
- Replicating data on other systems
- Writing from time to time complete database images to persistent storage
- Writing out the outcomes of transactions to an append-only persistent storage

In-Memory, GPU, and NVM Database Systems

Outline

In-Memory Database Oracle 19c

`Oracle 19c` In-Memory architecture



In-Memory Database Oracle 19c

Persistence is achieved through the following actions

- (1) Data is maintained in disk files (1) and ...
- (2) ... it is cached in memory
- (3) An OLTP application primarily reads and writes from memory
- (4) Any committed transactions are written immediately to the transaction log on disk
- (5) When required or as configured, row data is loaded into a columnar representation for use by analytic applications
- (6) Any transactions that are committed once the data is loaded into columnar format are recorded in a journal
- (7) Analytic queries access the journal to determine if they need to read updated data from the row store or possibly rebuild the column

In-Memory Database Oracle 19c

In-Memory Database uses In-Memory Area located in `System Global Area` (`SGA`)

In-Memory Areais subdivided into 2 pools: `1MB` pool and `64K` pool

`1MB` pool stores columnar formatted data, `64K` pool stores metadata

Listing In-Memory Area parameters

```
SELECT POOL, ALLOC_BYTES, USED_BYTES, POPULATE_STATUS
FROM V$INMEMORY_AREA;
```

The size of In-Memory Area is controlled by the system initialization parameter `INMEMORY_SIZE` (default `0`)

In-Memory Area must have a minimum size of `100MB`

It is possible to increase In-Memory Area by increasing `INMEMORY_SIZE` parameter with `ALTER SYSTEM` command (assuming that there is spare memory in `SGA`

Changing INMEMORY_SIZE parameter

```
ALTER SYSTEM SET INMEMORY_SIZE=300M SCOPE=SPFILE;
```

In-Memory Database Oracle 19c

Note, that the system must be shutdown and restarted when an option SCOPE=SPFILE is used !

Values of In-Memory Area parameters

```
POOL                          ALLOC_BYTES USED_BYTES
----------------------------- ----------- ----------
1MB POOL                        217055232          0
64KB POOL                        79691776          0
```

INMEMORY attribute can be specified for tablespace, table, partition, subpartition, or materialized view

Enabling INMEMORY attribute for a tablespace

```
ALTER TABLESPACE USERS DEFAULT INMEMORY;
```

Populating a table LINEITEM in In-Memory Area

Enabling INMEMORY attribute for a tble

```
ALTER TABLE LINEITEM INMEMORY;
```

In-Memory Database Oracle 19c

Explaining query processing plan of SELECT statement that uses LINEITEM table

Query processing plan for In-Memory table

```
EXPLAIN PLAN FOR SELECT L_EXTENDEDPRICE FROM LINEITEM;
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows   | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |  1800K|    10M| 12148   (1)| 00:00:01 |
|   1 |  TABLE ACCESS INMEMORY FULL| LINEITEM |  1800K|    10M| 12148   (1)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Excluding a column O_TOTALPRICE in a table ORDERS from In-Memory Area

Excluding a column from In-Memory Area

```
ALTER TABLE ORDERS INMEMORY NO INMEMORY(O_TOTALPRICE);
```

Excluding a table LINEITEM from In-Memory Area

Excluding a table from In-Memory Area

```
ALTER TABLE LINEITEM NO INMEMORY;
```

In-Memory, GPU, and NVM Database Systems

Outline

GPU Database systems

Graphical Processing Unit (GPU) database systems use GPU chipset technology for parallel processing of data analytics queries mainly computed at Data Warehousing systems

GPU database systems have been on the market for a few years and slowly became adopted in the high-performance computing areas

Most GPU databases have a master-slave architecture where a central CPU-based node submits the subqueries in parallel to an array of GPU-accelerated database instances, each on a separate server

The individual servers execute their subqueries in parallel, send and result sets back to the master node

The master node then combines them and sends the final result back to the client

GPU database system works best when you have data analytics applications that can be easily broken down into parallelized tasks

GPU Database systems

The existing systems

- `Brytlyt`: built on PostgreSQL, can cost-effectively query multibillion row datasets in seconds, easily scalable, supports additions and removals of GPU-acceleration nodes, optimized for real-time insights on large and streaming data sets
- `BlazingDB`: processes simple SQL queries on massive data sets, optimized for data warehousing workloads, its SQL engine runs on a cluster of distributed GPU servers
- `Kinetica`: uses in-memory storage and distributed processing, up to 6,000 cores in parallel, processes SQL queries on billions of rows in microseconds, provides native support for geospatial objects and it comes with a suite of geospatial functions
- `MapD`: uses in-memory storage, compiles SQL queries with a just-in-time `LLVM`-based compiler into machine code that can run on Nvidia GPUs as well as X86 or Power CPUs, parallelizes computation across multiple GPUs and CPUs
- `SQream`: SQL based column-oriented database system, processes trillions of rows in near real-time, uses CPUs as well as massively parallel multi-core GPU nodes, uses automatic and transparent partitioning, allows for selective access to the required subset of columns

In-Memory, GPU, and NVM Database Systems

Outline

Non-Volatile Memory Database System ? What is it ?

Non-Volatile Memory (NVM) is a name for a group of technologies such as Phase-Change RAM (PCRAM), Magnetic RAM (MRAM) and Resistive RAM (RRAM) that enable memory chips that are non-volatile (persistent), require low energy, and have density and latency closer to current DRAM chips

A commercial implementation of NVM is the Intel® OptaneTM DC Persistent Memory Module (simply called as Optane Memory, March 2019)

NVM has 4 times faster Input/Output Operations Per Second (IOPs)than SSD and seek time for data is ten times faster

NVM supports byte-addressable loads and stores with low latency and it can be used for efficient architecture of In-Memory database systems

NVM Database System is an In-Memory Database System implemented in Non-Volatile Memory

In-Memory, GPU, and NVM Database Systems

Outline

Non-Volatile Memory

Important properties of Non-Volatile Memory

- Byte-Addressability: NVM supports byte-addressable loads and stores, no need to transfer data in blocks
- High Write Throughput: NVM delivers more than an order of magnitude higher write throughput compared to SSD
- Read-Write Asymmetry: in certain NVM technologies, write take longer to complete when compared to read and excessive writes to a single memory cell can destroy it

Attribute	NVM					
	DRAM	PCM	RRAM	MRAM	SSD	HDD
Read latency	60 ns	50 ns	100 ns	20 ns	25 μs	10 ms
Write latency	60 ns	150 ns	100 ns	20 ns	300 μs	10 ms
Sequential bandwidth	60 GB/s	10 GB/s	10 GB/s	5 GB/s	1 GB/s	0.1 GB/s
$/GB	10	1	1	1	0.25	0.02
Addressability	Byte	Byte	Byte	Byte	Block	Block
Persistent	No	Yes	Yes	Yes	Yes	Yes
Endurance	$>10^{16}$	$>10^{10}$	10^{8}	10^{15}	10^{5}	$>10^{16}$

In-Memory, GPU, and NVM Database Systems

Outline

Architecture of NVM Database Systems

NVM-based architectures have several key advantages over the current in-memory systems

- NVM-based architecture adopts a logging and recovery protocol that improves the availability of the DBMS by 100 times when compared with the write-ahead logging protocol
- NVM-based storage engines utilize the durability and byte-addressability properties of NVM to assure better persistent storage utilization
- NVM-based storage engines use the index tailored for NVM
- Data buffer management policy uses the direct-addressability property of NVM to reduce data migration
- NVM-based architecture improves the runtime performance, availability, operational cost, and development cost of DBMS

Architecture of NVM Database Systems

There are two possible applications of NVM in a database system

- (1) Database system only has NVM storage with no DRAM (NVM-only architecture)
- (2) NVM is added as another level of the storage hierarchy between DRAM and SSD(NVM+DRAM architecture)

Both approaches can be applied to in-memory oriented database systems and storage (SSD/HDD) oriented database systems

The experiments show that in-memory oriented systems are better suited to take advantage of NVM and outperform storage oriented systems

Both in-memory oriented and storage oriented systems are doing unnecessary work in their storage and recovery methods, since all writes to NVM are durable

It means that neither system is ideally suited for NVM

Instead, a new system is needed with lightweight recovery processes designed to utilize the non-volatile property of NVM

References

G. Harrison, Next Generation Databases NoSQL, NewSQL, and Big Data, Apress, 2015 Chapter 6, Column Databases

Oracle Database In-Memory with Oracle Database 19c, Technical Overview, February 2019

J. Arulraj, A. Pavlo, Non-Volatile Memory Database Management Systems, Synthesis Lectures on Data Management, Morgan & Claypool, 2019