Enabling Intel® Optane™ DC Persistent Memory for OmniSciDB

Zhiqiang Ma

zhiqiang.ma@intel.com

IAGS/OISA/DSE

Abstract

Intel® Optane^{TM} DC Persistent Memory Modules (DCPMM) is byte-addressable non-volatile memory providing comparable performance to and much larger capacity than DRAM and much better performance than traditional storage media. OmniSciDB without any source or binary code change can take advantage of DCPMM capacity to run big workload it is not able to run before with DRAM. With a new memory level for DCPMM added in the Data Manager, OmniSciDB is able to run in AppDirect mode with tuned performance by placing hot columns in DRAM and cold columns in DCPMM. Further, hot columns can be automatically discovered through a profiler built in the Data Manger.

Keywords: Intel® Optane™ DC Persistent Memory, OmniSciDB, Memory mode, hot columns, AppDirect mode, performance

Enabling Intel® Optane™ DC Persistent Memory for OmniSciDB

Introduction

Modern computer systems with multi-tier or heterogenous memory architectures are trending to replace traditional systems with single tier of volatile DRAM memory. Combining memory tiers with different capacity and performance characteristics, these systems can provide better or equal performance with lower cost. Of all commercially available systems the platforms with the recently released Intel® Xeon® Scalable processor and Intel® Optane^{TM} DC Persistent Memory are an exemplary ones.

Intel® Optane^{TM} DC Persistent Memory Modules (DCPMM) is byte-addressable non-volatile memory providing comparable performance to and much larger capacity than DRAM and much better performance than traditional storage media. Tiering with faster, but lower-capacity DRAM, Intel® Optane^{TM} DCPMM, together with Intel® Xeon® processors, can provide breakthrough performance for big data analytic challenges.

Intel® Optane^{TM} DCPMM can operate in three modes: Memory, AppDirect and Mixed. In Memory mode, the DCPMM operates as volatile main memory and the DRAM is used as hardware cache for DCPMM. Whenever data is requested by the CPU, the hardware first checks the DRAM cache. If the data is present in DRAM (DRAM cache hit), the data is fetched from DRAM with DRAM latency. If the data is not present in DRAM (DRAM cache miss), with longer latency, the data requested is first fetched from DCPMM into DRAM cache before it is transferred to CPU caches. In AppDirect mode, the DCPMM operates as a separate memory pool which can be directly byte-addressed and accessed. The software sees 2 different memory pools: DRAM pool and DCPMM pool. It is software's responsibility to allocate memory from and place data in the 2 different pools. Mixed Mode is a mix of Memory mode and AppDirect mode. It allows for part of Intel® Optane^{TM} DCPMM to be configured to operate in Memory mode and the rest in AppDirect mode.

All three operating modes provide large capacity. While Memory mode does not require any software change to take capacity advantage, AppDirect and Mixed modes give software completely control of data placement. Though data written to Intel® Optane^{TM} DCPMM in AppDirect mode is persistent, it is perfectly ok for software to not make use of or rely on the persistency feature but simply use DCPMM as volatile memory.

Memory capacity is critical to in-memory computing, for example, big data analytics. This paper studies how DCPMM can be used in an in-memory database, namely OmniSciDB.

OmniSciDB is a columnar relational in-memory database targeted for big-data analytics, utilizing massive parallelism of modern GPU and CPU processors. In this study, we focus our study on Intel® CPU processors only. Further, we study the performance of OmniSciDB in Memory mode and AppDirect mode of Intel® Optane^{TM} DCPMM.

In both modes, thanking the large capacity of DCPMM, on a single node OmniSciDB is now able to run large workload that it is not able to run before using only DRAM. We will study which mode renders the best performance.

We leave the performance study of Mixed mode for the future.

Memory Mode

In Memory mode, the DRAM is the last level cache and the DCPMM is the main memory. There is no OmniSciDB source or binary code change required to exploit the big capacity of DCPMM. The DCPMM is the RAM in the following architecture diagram and real

DRAM is the last level hardware cache which is not visible to or cannot be controlled by OmniSciDB.

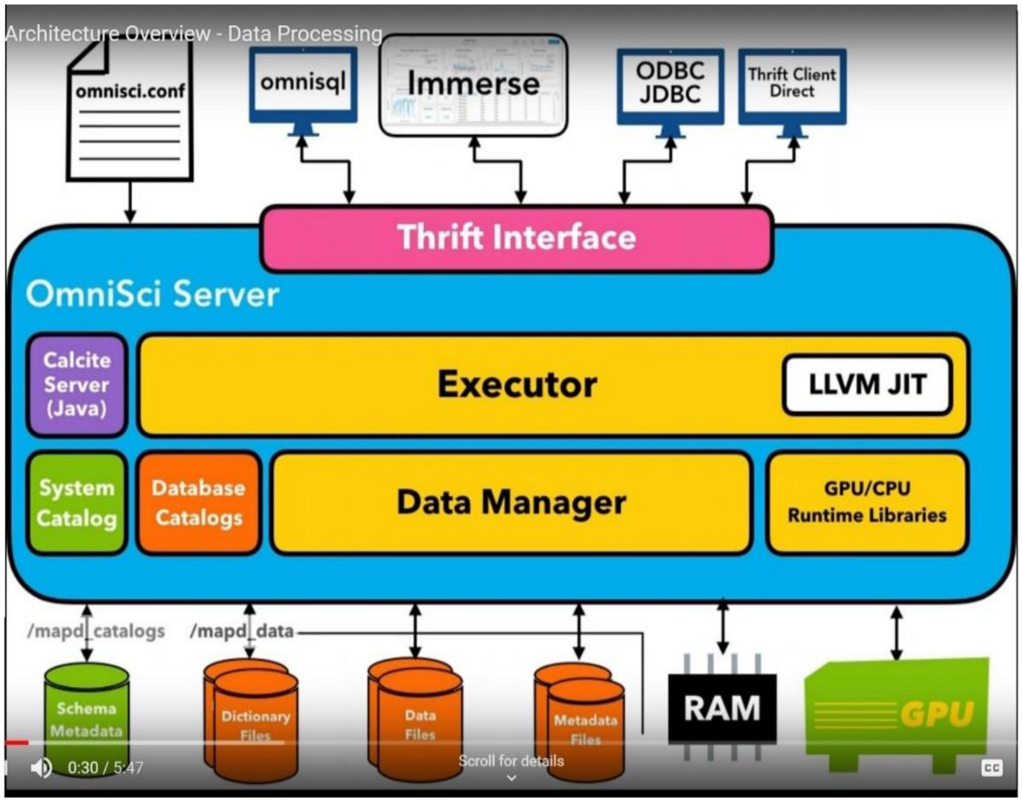

Figure 1 OmniSciDB architecture overview (courtesy of https://www.youtube.com/watch?v=hNjZtUijGU0)

The performance of DRAM cache is determined by the cache hit rate. The higher the hit rate, usually the better the performance. A DRAM cache miss is expensive, and the penalty is high: the data has to be fetched from DCPMM, which has longer latency and lower bandwidth than DRAM.

Different from DRAM, DCPMM performs better with sequential read or write access than random read/write access. Fetching from or evicting to DCPMM, for example, 4 consecutive cache lines is cheaper than fetching from or evicting to DCPMM 4 random non-consecutive cache lines. Though what cache lines are fetched or evicted, in what order the cache lines are fetched or evicted and when a specific cache line is fetched or evicted are out of control of the software, they are directly affected by the application's or OmniSciDB's behavior.

Enabling Big Workloads

The Memory Mode of DCPMM enables OmniSciDB server to be able to run much larger workload that it is not able to run before on traditional systems with only DRAM (we refer this traditional DRAM only mode as DRAM mode) without any source or binary code change.

We use the NYC taxi ride database for performance evaluation. We measure the execution time of the following 4 queries in DRAM mode and DCPMM Memory mode:

- `Q1: SELECT cab_type, count(cab_type)  FROM trips GROUP BY cab_type;`
- `Q2: SELECT passenger_count, avg(total_amount) FROM trips GROUP BY passenger_count;`
- `Q3: SELECT passenger_count, EXTRACT(year from pickup_datetime) as year, count(*) FROM trips GROUP BY passenger_count, year;`
- `Q4: SELECT passenger_count, EXTRACT(year from pickup_datetime) as year, round(trip_distance) distance, count(*) trips FROM trips GROUP BY passenger_count, year, distance ORDER BY year, trips;`

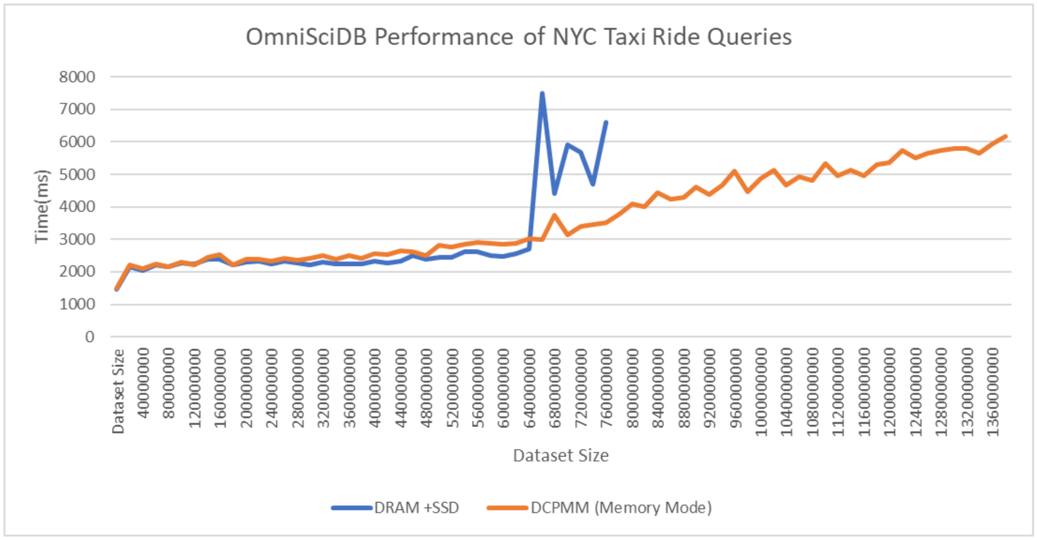

Figure 2 Performance of NYC taxi ride queries in DRAM mode and DCPMM Memory mode

The performance of OmniSciDB server in Memory mode is comparable to that in DRAM mode if the workload can fit in the DRAM. If the workload grows too big to fit in DRAM, however, the performance of DRAM mode starts to drop quickly due to disk swapping before memory is running out. On the other hand, OmniSciDB in Memory mode continues to run.

As the workload continues to grow, more memory for work area or data structures, for example, hash tables, are needed. These data structures are usually randomly accessed with less optimal locality, potentially resulting more cache misses. Therefore, the performance trends to get hurt as the workload grows.

AppDirect Mode

OmniSciDB is a database management system that stores data in columns. Not every column is equally accessed or used. Some columns are hot and some columns are cold. In AppDirect mode, hot columns are placed in DRAM and cold columns in DCPMM. Placing cold columns in DCPMM also guarantees these columns do not contend with working areas or data structures created by the Executor during query execution, for example, hash tables, for DRAM space.

A new memory level for DCPMM and buffer manager must be created in the Data Manager to support AppDirect mode and column placement, as shown in Figure 3.

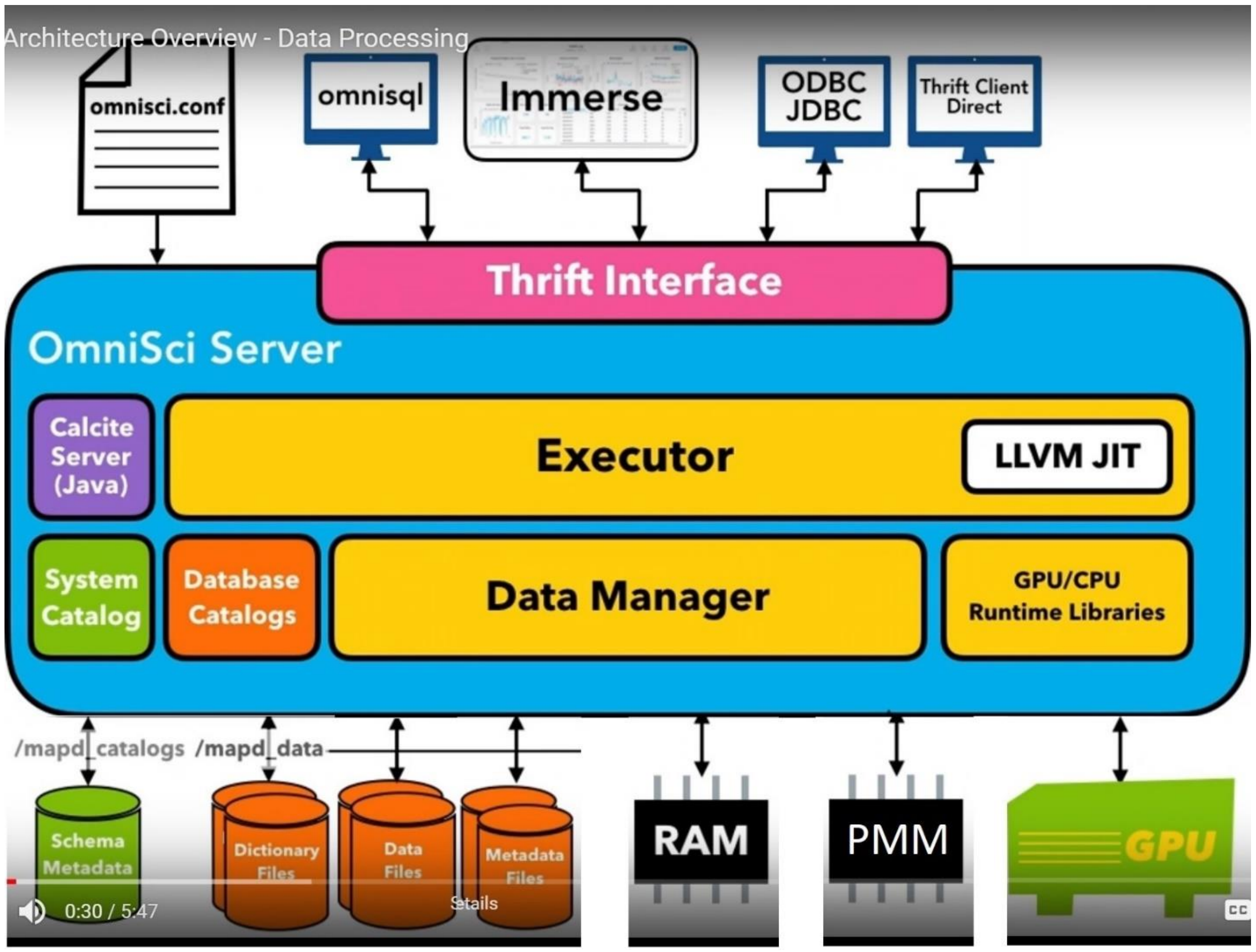

Figure 3 OmniSciDB architecture (DCPMM AppDirect mode)

Start OmniSciDB in AppDirect Mode

To start OmniSciDB in AppDirect mode, the -pmm flag needs to be specified in the omnisci_server startup command or script, for example:

```
startomnisc --data /omnisci/data –pmm ./vpmem.conf
```

Hard Hot Column vs. Soft Hot Column

A hard hot column is always placed in DRAM regardless of DRAM size whenever the column is fetched into memory (if not enough DRAM is available, other columns may be evicted or the server may stop running). Initially, all columns are cold when a table is created. The user can use omnisql front end to set a column from cold to hot or set it back to cold, for example:

```
omnisql> \heat_column trips trip_id
Column trip_id in table trips is set to hot.
omnisql> \cool_column trips trip_id
Column trip_id in table trips is set to cold.
omnisql>
```

The data base catalogs are extended to have a hard hotness attribute for each column.

Different from hard hot column, whether a soft hot column is placed in DRAM depends on available DRAM size when OmniSciDB is started. OmniSciDB in AppDirect mode uses statistics and heuristics (discussed later) to determine if a column should be placed in DRAM based on size of DRAM in the system. This is done automatically by OmniSciDB and there is no front end commands for the user to intervene.

A column placed in DRAM will not be migrated or moved to DCPMM in the same OmniSciDB running instance. A column placed in DRAM in one OmniSciDB instance may be placed in DCPMM in a different instance.

AppDirect mode does not support partial hot columns. A column will never be placed partially in DRAM and partially in DCPMM. A column will never be placed wholly in both DRAM and DCPMM either.

Profiling Workloads

If he/she knows in advance what columns should be placed in DRAM and what columns should be placed in DCPMM, the user can use the front-end commands to hard set hot columns. In most cases, however, users lack this knowledge and determining hot columns is a real challenge. It is very desirable to have OmniSciDB automatically discover hot columns without user's intervening for best performance.

We have built a profiler in OmniSciDB server to solve this challenge. The profiler collects a few metrics including time cost of each query, columns used in each query, amount of data used in each query etc. when the workload is running. The OmniSciDB server later uses these metrics for auto column placement.

The workload for profiling does not have to be a production workload, but it has to be representative and can fit in the available DRAM.

The profiler profiles the workloads in 2 steps. In each step. The user can use the front-end commands to start and stop the profiling.

```
omnisql> \start_profiling     //before workload starts
Data manager profiling on.
omnisql>                      //run workload
omnisql> \stop_profiling
Data manager profiling off.   //after workload completes
```

The first step runs and profiles the workload with all columns placed in DCPMM. The profiler collects the time cost of each query, columns used by each query and amount of column data accessed by each query etc. In addition, the peak work area memory size of the workload is also collected. All statistics are stored in catalogs.

After the first step completes, the OmniSciDB server needs to be shutdown and restarted for the second step. The second step runs and profiles the workload with all columns placed in DRAM. Again, the profiler collects the time cost of each query, columns used by each queries and amount of column data accessed by each query etc. All statistics are stored in catalogs.

Column Placement

Once the workload is profiled. OmniSciDB is able to place the columns automatically based on the statistics collected when the workload is run again later. All hard hot columns will be placed in DRAM. The rest are placed in either DRAM as soft hot columns or DCPMM as cold columns using a knapsack algorithm with size of the available DRAM in the system as the constraint.

Scale Factor

The workload to be profiled needs to be representative but does not have to be a production workload. It is usually smaller than the production workload for efficiency. For example, if a production workload is 10TB, a reduced and representative workload of 10GB may be big enough for profiling. The database server takes this into account when soft hot columns are calculated if a scaling factor is specified, for example:

```
      startomnisci --data /home/omnisci/data --pmm ./vpmem.conf -
-profile-scale-factor 1000
```

The time cost of each query and memory size of each column is linearly scaled up by the scale factor when soft hot columns are calculated. If time cost of queries and size of columns are not linearly related, multiple workloads with different sizes need to be profiled and a scale factor that is approximate to the real non-linear factor should be specified, for example, 1600 instead of 1000.

Performance

We use the same dataset and same queries to evaluate the performance with columns used in the queries placed in DRAM and other columns placed in DCPMM. Figure 4 shows performance in AppDirect mode is significantly improved.

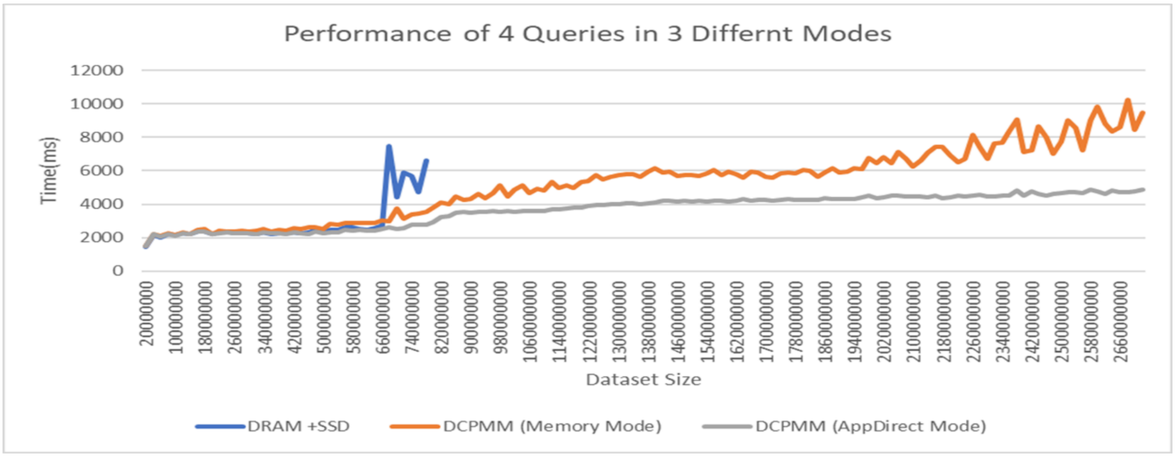

Figure 4 OmniSciDB performance of NYC taxi ride queries in 3 different modes

Estimating Amount of DRAM Needed to Meet a Performance Target

So far, we have focused on answering the question of what columns to place in DRAM with a fixed DRAM size as the constraint. An orthogonal question is: how much DRAM is needed to meet a performance target, for example 90% of DRAM performance (or 90% of the performance when DRAM is unlimited)

We use the profiled statistics and the scale factor specified to calculate the DRAM size needed given a performance target, for example, 95% of DRAM performance:

```
omnisql> \estimate_dram_size 95
120GB of DRAM recommended.
omnisql>
```

The size reported is approximate. There is no guarantee that the value is the minimal size requirement.

Conclusions

Both DCPMM Memory mode and AppDirect mode enable OmniSciDB to run big workload it is not able to run in DRAM mode before. Memory mode does not require any source or binary code change, but AppDirect mode gives OmniSciDB the opportunity to place hot columns in DRAM and cold columns in DCPMM for better performance and hot columns can be found automatically by the data profiler, which profiles a representative workload in 2 steps.