Intel® Optane™ Persistent Memory Accelerates Mars Performance for More Efficient Scientific Computing

"Mars is a distributed computing framework based on tensors. It can efficiently accelerate scientific computing tasks through parallelization and distribution. In the process of computing, Mars mainly stores data in memory. When memory is not enough, the temporarily-not-used data is stored in disk. Scientific computing tasks often include a large number of intermediate processes with hard disk I/O, leading to performance degradation. Intel® Optane™ Persistent Memory[1] can accelerate data I/O process, significantly improving computing efficiency of distributed scientific computing."

— Li Ruibo, Senior Technical Specialist, Alibaba

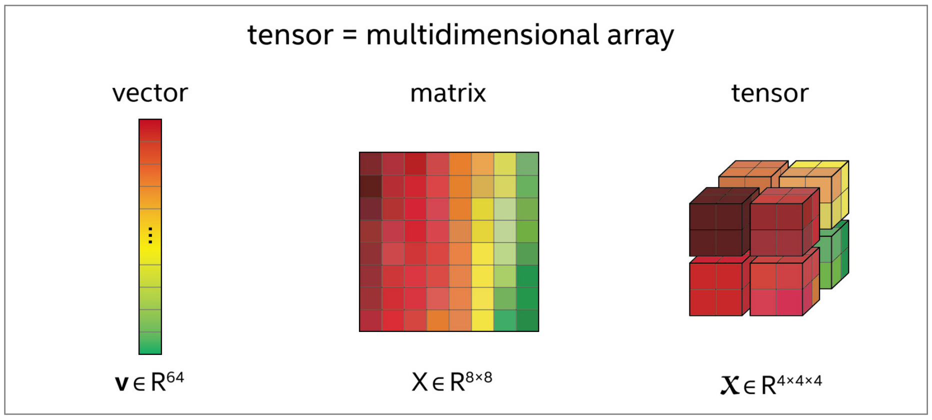
Table of Contents

Scientific Computing

Scientific computing is using a computer to address mathematical calculation problems in scientific research and engineering technology. In modern scientific research and engineering technology, massive complex mathematical calculations are often involved, such as in climate research, gene sequencing, financial engineering, astronomical simulation, and high energy physics, etc.

Such scientific computing is often characterized by massive data and high computing workloads. It requires abstraction and rigor in mathematical theory, as well as practicality and practice on program design. The laws of natural science are usually expressed in various types of mathematical equations, so there will be linear or non-linear equations of several variables and complex calculus formulas. These complex mathematical problems and massive computing workloads require the support of computers with powerful capacities and computer languages and libraries that support scientific computing.

Among many computer languages and libraries, Numpy has stood out with its easy-to-use syntax and powerful performance, and a large technology stack has been created based on it. Numpy is the core of the entire technology stack, and a large number of upper-level tools are using the data structure and calculations of Numpy. Numpy supports calculation of massive multi-dimensional arrays and matrices. Multi-dimensional array is called tensor, which is the basis of deep learning. Tensor creates higher-dimensional matrix and vector, therefore having stronger expression capability.



tensor = multidimensional array
vector
matrix
tensor
$\mathbf{v} \in R^{64}$
$X \in R^{8\times8}$
$\mathcal{X} \in R^{4\times4\times4}$

Figure 1. Tensor definition

1. https://www.intel.cn/content/www/cn/zh/architecture-and-technology/optane-dc-persistent-memory.html

With the upsurge of machine learning and deep learning, the concept of tensor has been gradually known by the public, and the demand for tensor calculation is also on the rise. However, the reality is that in a scientific computing library like Numpy, most functions cannot be parallelized, and the multi-core processing capability of CPU cannot be utilized to improve computing efficiency; the parallelization of Numpy requires the use of multi-threaded, distributed writing tasks. This manual writing method is difficult and inefficient, especially for computing involving massive data sets; yet current popular distributed computing engine is not designed for scientific computing. The mismatch of upper interface makes it difficult to write scientific computing tasks with traditional SQL/MapReduce. The engine itself is not optimized for scientific computing, which makes computing efficiency unsatisfactory.

Based on the status quo of scientific computing, after more than a year of R&D, Mars was launched by the R&D team of MaxCompute, a big data computing platform of Alibaba.

What is Mars

Mars is a unified distributed computing framework based on tensors. It breaks through the existing big data computing engine, which mainly based on relational algebra model, and introduces distributed technology into the fields of scientific and numerical computing, thus significantly increasing the scale and efficiency of scientific computing. Using Mars for scientific computing not only reduces the number of codes required to complete large-scale scientific computing tasks from thousands of lines on MapReduce to several lines on Mars, but also greatly improves performance. All of these are enabled with the unique advantages and features of Mars.

• Familiar and easy-to-use interface

So far, 70% of common Numpy interfaces have been implemented on Mars, and the built-in tensor modules are compatible with Numpy interfaces; users can move existing Numpy-based codes into Mars to realize tens of thousands of times larger scale and dozens of times higher processing power. The ease of use of Mars is described through a simple example as follows.

First, we use Numpy to write codes for matrix multiplication:

```
import numpy as np
↵
a = np.random.rand(50, 1_000_000)
b = np.random.rand(1_000_000, 100)
a.dot(b)
```

Import Mars' tensor module by importing mars.tensor:

```
import mars.tensor as mt
↵
a = mt.random.rand(50, 1_000_000)
b = mt.random.rand(1_000_000, 100)
a.dot(b).execute()
```

In this example, only a few changes are made in the code; the first one is "import numpy" is changed into "import mars.tensor"; the second is what should be particularly noted, that is the execution should be triggered by adding "execute" in Mars. The advantage is that the whole intermediate processes can be optimized as much as possible.

• Support GPU acceleration

One of the major advantages of Mars is hardware acceleration. It accelerates computing task execution with hardware such as GPU. It is also easy to set up GPU acceleration in Mars. Simply adding a line "gpu=True" when creating a tensor, the subsequent calculations can be executed on GPU.

• Support sparse matrix

Mars supports two-dimensional sparse matrix. This can save storage, and improve computing efficiency. A sparse matrix can be created by adding a line "sparse=True".

• Scalability

Mars can be scaled inward to a single machine, or outward to a server cluster consisting of thousands of computers. Both local and distributed versions share the same codes. As data grows, it is easy to move from a single machine to a cluster.

On a single machine, you can run thread-based scheduling or local cluster scheduling that bundles the whole distributed components. Mars can easily scale out to a cluster by starting different components of Mars distributed runtime on the different machines in the cluster.

• Memory computing supports OOM control to ensure computing security

Mars is a memory-based distributed computing engine. During the execution of a computing task, in order to avoid failures caused by insufficient memory in a single machine or cluster, Mars uses data overflow control to spill the data temporarily not used to the disk for storage. This is a way of controlling OOM (out of memory) to ensure computing security.

As a new generation of scientific computing engine of ultra-large scale, Mars brings scientific computing into the distributed era, and makes it possible for big data to be efficiently and scientifically calculated. Now, Mars has been widely used in the businesses and production of Alibaba and its cloud customers as well.

Parallel and Distributed Principle of Mars

So how does Mars enable powerful computing performance through parallel and distributed computing?

Like all Dataflow frameworks, Mars also has computational graphs, which include coarse-grained as well as fine-grained graphs. On client end, Mars does not perform any computing operation. When a user writes codes, the operation is recorded in memory with a graph (coarse-grained graph). When the user calls the command "execute", the graph is submitted to the distributed execution environment of Mars; essentially, Mars is an execution scheduling system for fine-grained graphs. Upon receiving coarse-grained graphs from client end, it tries to convert them into fine-grained ones to ensure that each chunk and its input are stored in memory. This process is called "tile".

For example, if a tensor is given, Mars automatically slices it at each dimension into smaller chunks and process them separately. Then, Mars schedules these small chunks and executes them in multiple cores or distributed clusters.

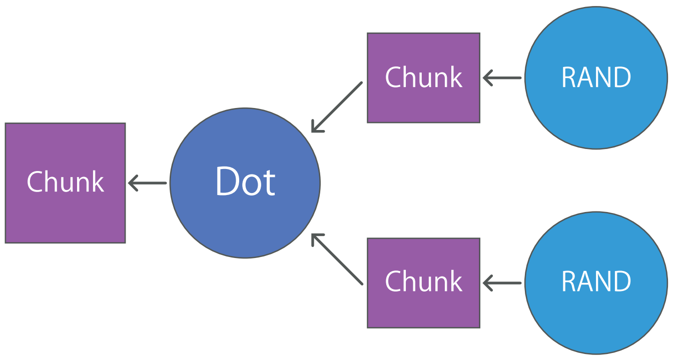


Figure 2. Coarse-grained graph

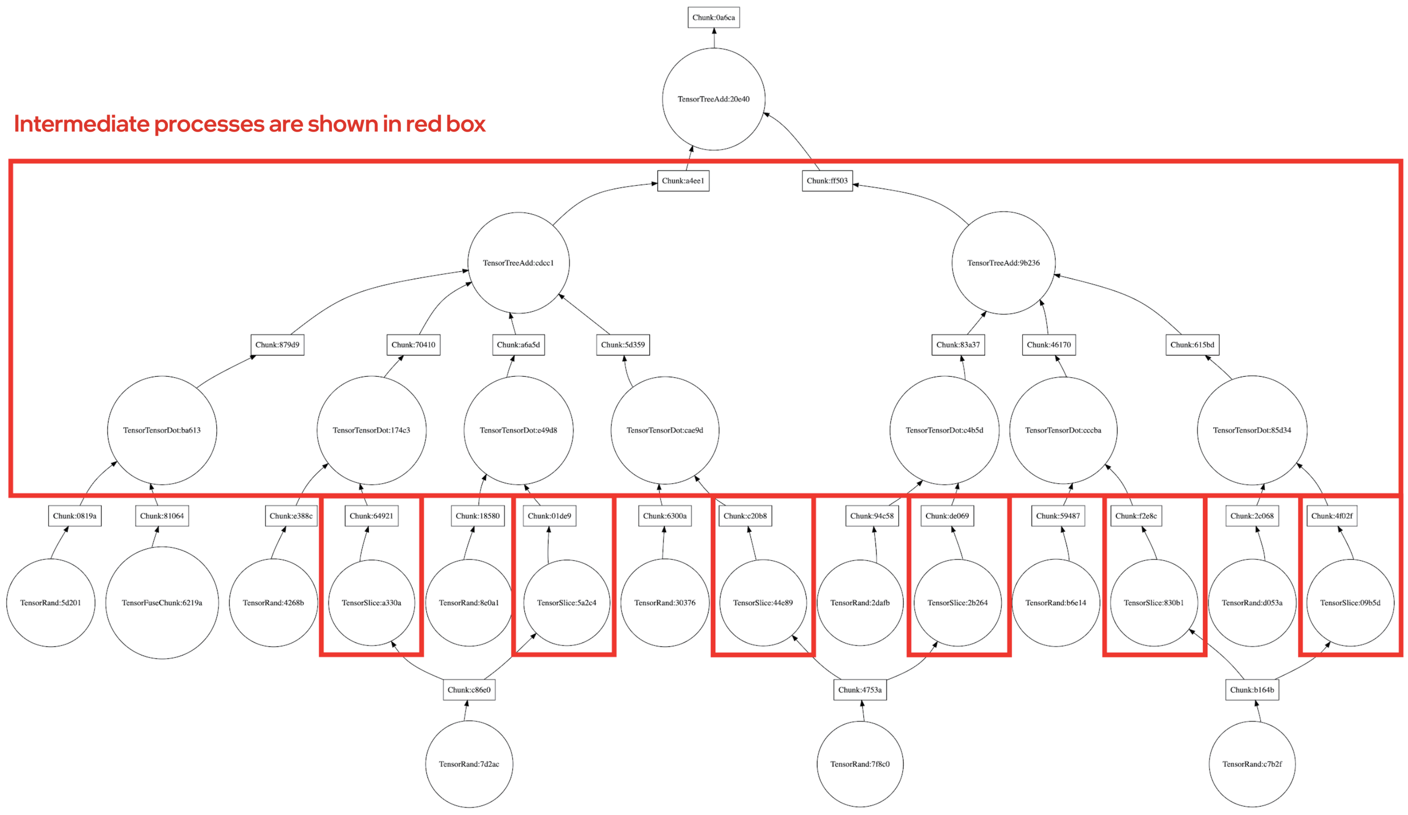


Figure 3. Execution map of Mars at chunk level

Taking the above matrix multiplication as an example, Mars automatically generates a chunk-level execution map as shown above in Figure 3.

Computing Bottlenecks of Mars

In the chunk-level execution map of Mars, we find out that data I/O volume of matrix operation is not that large, yet many intermediate processes are involved during the operation, as shown in the red boxes in Figure 3. As a memory-based distributed scientific computing engine, this means that a large amount of data will use the limited and valuable memory resources during execution.

When Mars uses multi-process scheduling or distributed scheduling, each process transfers data through the shared memory of a single machine or a cluster. In the worker structure of Mars as shown in Figure 4 (Mars shares memory through Plasma, a component of Apache Arrow), we can see that: after a chunk is executed, Mars worker puts the results in shared memory. When there are too many data in shared memory, Mars Worker will start data overflow control and automatically spill the infrequently-used data to disk or cloud storage. In this way we can avoid OOM, which leads to the failure of computing tasks.

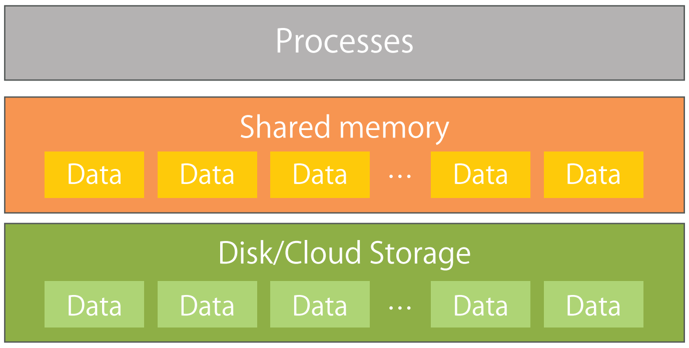


Figure 4. Mars Worker structure

In common scientific computing scenarios such as matrix multiplication, broadcasting and linear algebra, a large amount of data is generated in intermediate processes. As a result, the memory of a server or cluster must meet very high requirements. The data overflow control of Mars spills the infrequently-used data during execution to disk or cloud storage device, which effectively reduces unnecessary waste of memory resources. However, this also brings new problems.

Traditional HDD (hard drive disk) provides large storage space at a lower price, but this causes TCO (total cost of ownership) issues including factors such as reliability, physical space requirement and cooling. In addition, data I/O operations generate huge overheads. Accessing data from traditional HDD may cause serious access latency, making it much slower than from DRAM. For DRAM, the latency is usually tens of nanoseconds, while that for traditional HDD may reach tens of microseconds or milliseconds. This means a latency gap of at least 1,000x, which dramatically reduces the computing performance of Mars. To ensure operational speed, more memory is needed in server, which will, undoubtedly, largely increase the cost of a server or cluster.

With the same TCO, how can we improve the computing performance of Mars? In order to address this issue, Alibaba has worked closely with Intel to install and configure Intel® Optane™ Persistent Memory on servers running Mars. With the benefits of large capacity and low latency, Intel® Optane™ Persistent Memory provides shared memory of large capacity for Mars computing, and enhances scientific computing performance of Mars.

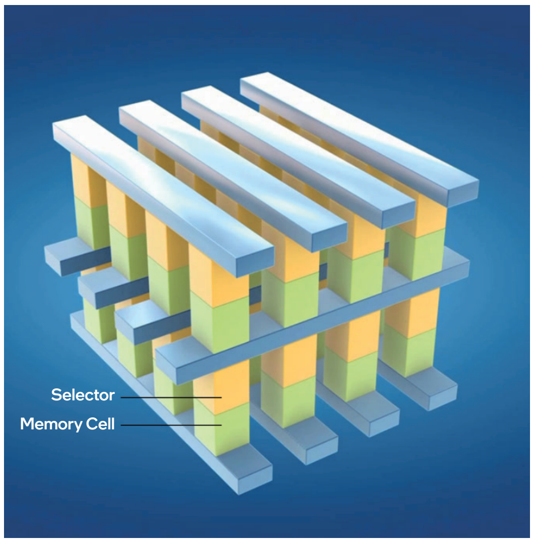


Figure 5.
Intel® Optane™ memory and storage media offers properties of both memory and storage by using a revolutionary 3D structure that provides high density, low latency, and persistence

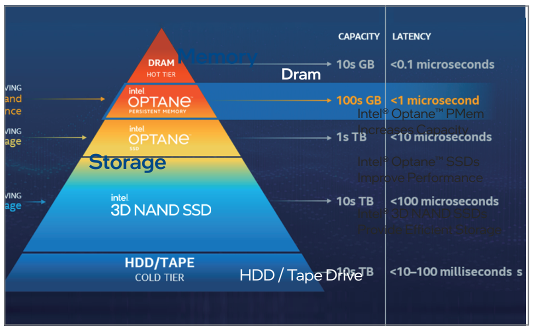


Figure 6.
In the storage hierarchy, Intel® Optane™ persistent memory is located below DRAM

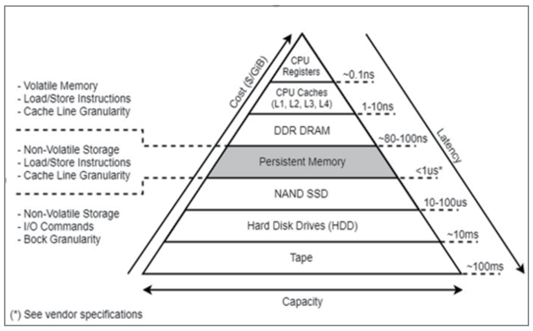


Figure 7.
Capacity, price and performance of the storage hierarchy

Break Computing Bottleneck with Intel® Optane™ Persistent Memory

• What is Intel® Optane™ Persistent Memory?

Intel® Optane™ Persistent Memory (PMem) is a revolutionary memory product based on 3D Xpoint medium, and it has advantages of high speed, low latency, high cost-effectiveness, high capacity, persistent data protection and advanced encryption. Its core is the ability to stack memory grids in a 3D matrix to improve density, increase performance, and provide persistence (Figure 5). This architecture allows Intel® Optane™ technology act as DRAM (byte addressability, high endurance, write in place) or traditional storage (block addressability, persistence)2.

Intel® Optane™ Persistent Memory changes the original storage hierarchy (Figure 6). Combined with 2nd Gen Intel® Xeon® Scalable processor, it provides performance similar to DDR memory (DRAM), and stores data persistently like SSD. In addition, persistent memory offers larger capacity and is cheaper than DRAM. Figure 7 shows a comparison of capacity, price, and performance in the new storage hierarchy.

• Advantages of Intel® Optane™ Persistent Memory

Intel® Optane™ Persistent Memory is available in 128 GB, 256 GB and 512 GB, and is a much larger alternative to DIMM with only 128 GB of SDRAM DIMM; compared to traditional DRAM DIMM, Intel® Optane™ Persistent Memory module features a lower cost per GB of memory. This affordable, flexible, large-capacity non-volatile memory meets the demands of Mars.

Persistent memory is directly connected to processor through memory bus. In App Direct-based storage mode (AD mode), applications can bypass operation system and access data on persistent memory from user space, without requiring device drivers, system calls, interrupts or context switches. Figure 8 compares the overheads of hardware and software on different storage devices. With direct access to persistent memory, the system has almost no software overheads, and the latency is in hundreds of nanoseconds. In this case, persistent memory is at the same latency level as DRAM.

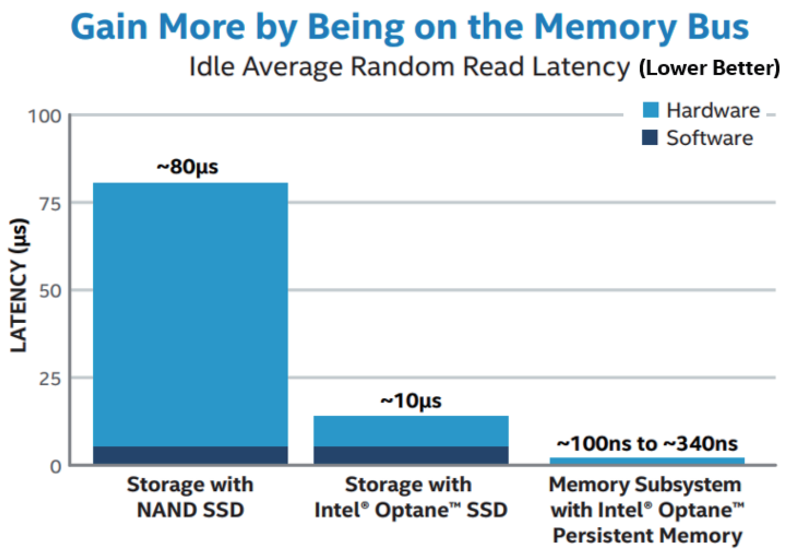


Figure 8. Software and hardware overhead for accessing various storage devices

2. Intel® Optane™ Technology: Memory or Storage? Both. https://cdrdv2.intel.com/v1/dl/getContent/615396?wapkw=%E6%8C%81%E4%B9%85%E5%86%85%E5%AD%98 2019.8.4

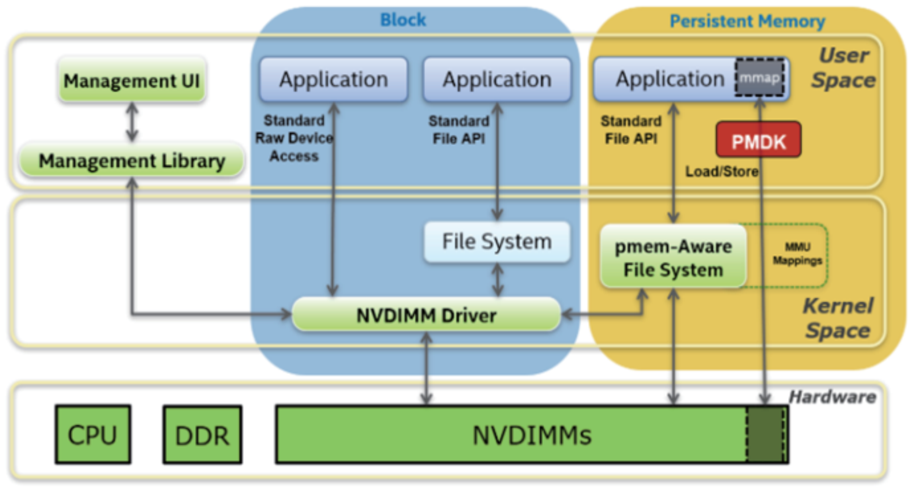


Figure 9. Persistent memory SNIA programming model and PMDKs

Persistent memory devices follow SNIA programming model, and Intel provides a set of PMDKs (persistent memory development kits) for it (Figure 9). PMDKs help applications directly access persistent memory devices without going through page cache system, system calls or drivers related to file system, which reduces many processes.

Therefore, in scientific computing, with Intel® Optane™ Persistent Memory, Mars eliminates huge overheads of data I/O, reduces data latency, and greatly improves computing efficiency.

• Tuning 1 - Persistent memory enables Snoop for AD mode

In order for Intel® Optane™ Persistent Memory to ensure data coherence across multiple caches, the server platform that supports multiple processors follows MESIF protocol3. This protocol uses snoop to sniff the status of data to be accessed in the cache of each processor to ensure coherence of multi-processor cache.

Directory is used to record the state of data to be accessed on the cache of each processor. These states can reduce unnecessary snoop (For example, if Directory records that the data to be accessed does not exist in the cache of processor, snoop messages are not needed. Directory records will be stored in the space for data access).

The write bandwidth of persistent memory is comparatively low, so Directory update has a significant impact on the performance of persistent memory. Enabling Snoopy for AD mode will disable Directory update, which is caused by access of local persistent memory from a remote processer. When Snoopy for AD mode is set to enabled, Directory can still be used for DRAM access. This feature benefits applications not opted for NUMA binding.

In AD mode, this effect can be proved by running a test that let MLC4 read persistent memory, and allow access to persistent memory on CPU1 from threads running on CPU0. When Snoopy for AD mode is disabled, the write activity can be observed in persistent memory even if MLC only remotely reads and access its modules. When running the test for the first time, read performance is only a small part of the actual bandwidth of persistent memory. Upon continuous tests, the highest bandwidth of persistent memory is reached, and the span depends on the capacity of the persistent memory. When all spaces have been accessed, the Directory of each address has been updated to E@D1. When they are accessed again, the Directory will not be updated, so performance can reach the highest bandwidth of the device. If CPU1 accesses its local persistent memory, then the Directory will be updated to E@D2 and the performance will be affected.

When Snoopy for AD mode is set to enabled, write bandwidth is not observed in persistent memory when running the test. Snoop for AD mode should be enabled for any application not optimized for NUMA binding, as shown in Figure 10.

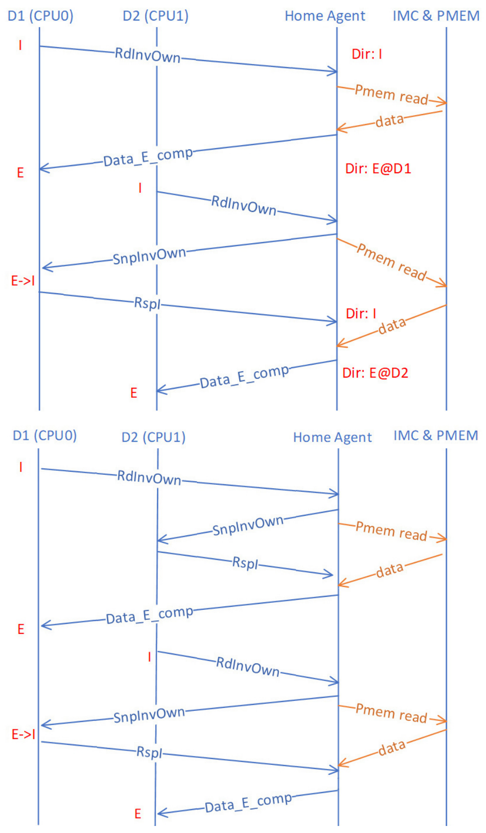
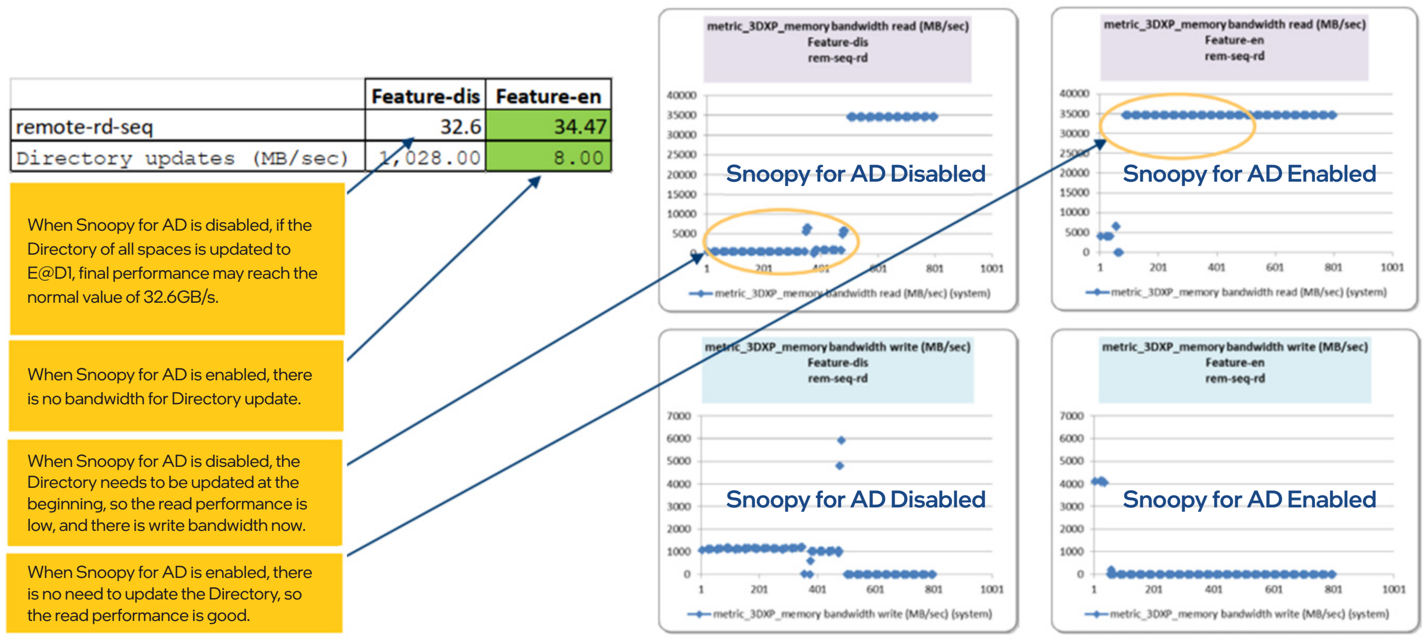


Figure 7.
Directory is updated when Snoopy for AD is disabled, and Directory is not updated when it is enabled

3. https://en.wikipedia.org/wiki/MESIF_protocol
4. https://software.intel.com/content/www/us/en/develop/articles/intelr-memory-latency-checker.html



	Feature-dis	Feature-en
remote-rd-seq	32.6	34.47
Directory updates (MB/sec)	1,028.00	8.00

When Snoopy for AD is disabled, if the Directory of all spaces is updated to E@DI, final performance may reach the normal value of 32.6GB/s.

When Snoopy for AD is enabled, there is no bandwidth for Directory update.

When Snoopy for AD is disabled, the Directory needs to be updated at the beginning, so the read performance is low, and there is write bandwidth now.

When Snoopy for AD is enabled, there is no need to update the Directory, so the read performance is good.

metric_3DXP_memory bandwidth read (MB/sec) Feature-dis rem-seq-rd
Snoopy for AD Disabled
metric_3DXP_memory bandwidth read (MB/sec) Feature-en rem-seq-rd
Snoopy for AD Enabled
metric_3DXP_memory bandwidth write (MB/sec) Feature-dis rem-seq-rd
Snoopy for AD Disabled
metric_3DXP_memory bandwidth write (MB/sec) Feature-en rem-seq-rd
Snoopy for AD Enabled
metric_3DXP_memory bandwidth read (MB/sec) (system)
metric_3DXP_memory bandwidth write (MB/sec) (system)

Figure 11. Results of enabling and disabling Snoopy for AD

Figure 11 shows the results of tests with MLC. When Snoopy for AD is enabled, remote persistent memory is read, and Directory is not updated (no persistent memory write), so read performance is high. If Snoopy for AD is disabled, read performance is quite low because Directory needs to be updated at the beginning.

• Tuning 2 - Plasma enables shared memory in persistent memory

As mentioned earlier, when Mars utilizes multi-process or distributed scheduling, it uses Plasma, a component of Apache Arrow, to implement shared memory on a single machine or cluster (Figure 12).

Plasma is a high-performance object storage based on shared memory. It was firstly developed by Ray, and was later contributed to Apache Arrow community. Plasma provides a way for storing and sharing data with Zero-copy for multiple processes on a single node, which avoids data duplication and expensive serialization and deserialization operations.

With memory map, Plasma maps relevant areas of memory (/dev/shm as the shared memory by default) on the server and multiple clients into its own address space, and all data accesses are operated in memory to avoid additional overheads.

Usually, we use memory not only as shared memory, but as an important computing resource as well. In an environment where memory is relatively limited, using Plasma does not bring many benefits; however, Intel® Optane™ Persistent memory changes this situation. As Persistent memory bypasses page cache in AD (App Direct) mode, mmap can have direct access to the address space of persistent memory. Its large capacity is also ideal for shared memory, so Plasma can build shared memory on it and release expensive memory resources for other tasks.

On a machine equipped with multiple NUMA nodes, multiple persistent memory devices can be bound together with device mapper to provide the maximum shared memory; together with Snoopy for AD mode, the highest cross-NUMA access performance can be realized.

The use of large-capacity persistent memory allows Plasma to build shared memory on it, without having to take up expensive memory resources, which can be released to the most-needed processes, thereby effectively improving computing performance of Mars.

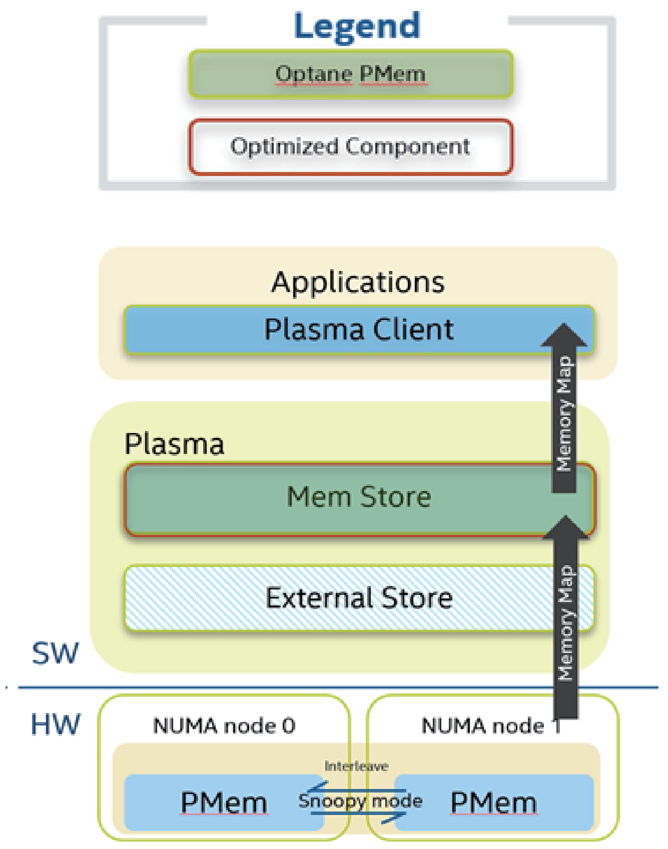


Figure 12. PMEM Enabled Plasma architecture

Scenario verification: Intel® Optane™ Persistent Memory improves scientific computing capability of Mars

With the same TCO, in order to verify whether Intel® Optane™ Persistent Memory as shared memory can improve performance of scientific computing for Mars, we use two scientific computing workloads of matrix multiplication, and execute the two tasks on a server installed with Intel® Optane™ Persistent Memory and the other one with DRAM to verify the performance difference between the two.

First, Intel® Optane™ Persistent Memory is installed on 2nd Gen Intel® Xeon® Scalable platform. This processor not only provides enhanced computing performance, but also has memory controller that can respond to Intel® Optane™ Persistent Memory to take full advantage of hardware capabilities. Then persistent memory is set to AD mode and set Snoopy with AD in BIOS, and the test environment is set up according to the configuration list (Table 1). On the test server, three Mars core components, Mars Web, Mars Scheduler and Mars Worker, are spined up at the same time.

Platform	Memory-based	Based on Intel® Optane™ Persistent Memory
Working node	1	
CPU	Intel® Xeon® Gold 6252 CPU @2.10GHz	
Memory	768GB (24x DDR4 32GB @2666MHz)	192GB (12x DDR4 16GB @2666MHz)
Persistent Memory (PMem)	N/A	1TB (8x 128GB Intel® Optane™ memory)
Storage	2x 1.2TB SATA SSD	
Network	10GbE	
OS	CentOS 7.6	
conda	Miniconda3-py37_4.8.3-Linux-x86_64	
Pymars	0.4.5	
Pyarrow	1.0.0	
Python	3.7.7	
numpy	1.19.1	
Scipy	1.5.0	
Pandas	1.0.5	

Table 1. Software and hardware configuration list

After testing workloads of different data sets on Mars, we find out from the test results (Figure 13) that when the data set is relatively small, the data can be cached by DRAM completely, so the performance of persistent memory is close to DRAM; but as the size of data set increases, the advantages of persistent memory begin to appear. When the size of data set increases to SF = 0.5, persistent memory has 1.11x more performance vs. DRAM.

The performance improvement is largely benefited from using Intel® Optane™ Persistent Memory featuring large capacity. When Mars performs scientific computing, the size of data set increases as computing progresses. When the size of data set can be cached completely by memory, the performance gap of the two is relatively small; when the size of data set is larger than that of the cache of DRAM, Mars starts to overflow some of the data. The server with DRAM stored the spilled data in SSD, while the other stored them in persistent memory, whose capacity can completely buffer the spilled data, thus significantly reducing data latency and improving performance of Mars.

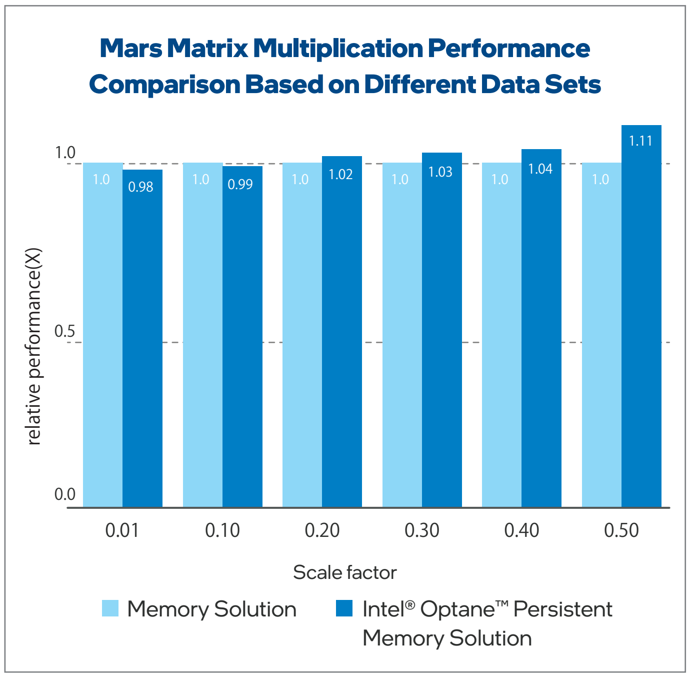


Figure 13. Performance test comparison

Conclusion

In summary, Intel® Optane™ Persistent Memory addresses the issues of increased I/O overheads and decreased computing performance as data spill to traditional HDD during scientific computing of Mars; the test proves that with the same TCO, Intel® Optane™ Persistent Memory can effectively improve scientific computing performance of Mars and make scientific computing more efficient.