**Contents**

Revision Record.................................................................................................................. 2

1. **Introduction** .................................................................................................................. 3
   1.1. Introduction to RocksDB*............................................................................................ 3
   1.2. Introduction to Persistent Memory............................................................................... 3

2. **Hardware Tuning** ........................................................................................................... 5
   2.1. BIOS Setting ............................................................................................................... 5
   2.2. Intel® Optane™ Persistent Memory 200 Series Hardware Settings............................ 7
   2.3. Memory Configuration/Settings.................................................................................. 12
   2.4. Storage/Disk Configuration/Settings ......................................................................... 12
   2.5. Network Configuration.............................................................................................. 12

3. **Software Tuning** .......................................................................................................... 13
   3.1. Linux® Kernel Optimization Settings ......................................................................... 13
   3.2. RocksDB Architecture............................................................................................... 13
   3.3. RocksDB Tuning.......................................................................................................... 14

4. **The RocksDB Engine** .................................................................................................... 21
   4.1. Kvrocks* .................................................................................................................... 21
   4.2. Basic kvrocks Performance......................................................................................... 21
   4.3. kvrocks Write Performance ....................................................................................... 21
   4.4. kvrocks Read Performance ....................................................................................... 21

5. **Related Tools** ............................................................................................................... 22
   5.1. Performance Monitoring Tool - PCM ......................................................................... 22

6. **Conclusion** .................................................................................................................... 22

7. **Feedback** ....................................................................................................................... 22
## Revision Record

<table>
<thead>
<tr>
<th>Date</th>
<th>Rev.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/10/2021</td>
<td>1.2</td>
<td>Fixed error in max PMem memory supported</td>
</tr>
<tr>
<td>08/15/2021</td>
<td>1.0</td>
<td>Initial Public Release</td>
</tr>
</tbody>
</table>
1. Introduction

This guide is targeted towards users who are already familiar with RocksDB* and provides pointers and system setting for hardware and software that will provide the best performance for most situations. However, please note that we rely on the users to carefully consider these settings for their specific scenarios, since RocksDB* can be deployed in multiple ways and this is a reference to one such use-case.

3rd Gen Intel® Xeon® Scalable processors deliver industry-leading, workload-optimized platforms with built-in AI acceleration, providing a seamless performance foundation to help speed data's transformative impact, from the multi-cloud to the intelligent edge and back. Improvements of particular interest to this workload applications are:

- Enhanced Performance
- More Intel® Ultra Path Interconnect
- Increased DDR4 Memory Speed & Capacity
- Intel® Security Essentials and Intel® Security Libraries for Data Center
- Intel® Speed Select Technology
- Support for Intel® Optane™ persistent memory 200 series

1.1. Introduction to RocksDB*

RocksDB* started as an experimental Facebook* project intended to develop a database product for high-workload services, which is comparable in data storage performance to a fast data storage system, flash memory in particular. RocksDB utilizes major source code from the open source project, LevelDB*, as well as foundational concepts from the Apache* HBase* project. The source code can be sourced back to LevelDB 1.5 fork.

RocksDB supports various configurations and can be tuned for different production environments (memory only, Flash, hard disks, or HDFS). It also supports a range of data compression algorithms and tools for debugging in production environments. RocksDB is mainly designed for excellent performance in fast storage applications and high workload services. Therefore, this database system requires sufficient utilization of the read and write speeds of flash memory and RAM. RocksDB needs to support efficient "point lookup" and "range scan" operations, and configuration of various parameters for random read and write operations in high-workload services or for performance tuning in case of heavy read and write traffic.

For more information on RocksDB, visit the RocksDB wiki page.

1.2. Introduction to Persistent Memory

Persistent memory has changed the existing memory/storage hierarchy. Intel® Optane™ persistent memory (PMem), which works with Intel® Xeon® Scalable processors, performs similarly to DRAM and stores data persistently like SSDs. It also offers larger memory capacity than DDR and is cheaper. Figure 1 shows a comparison of capacity, price, and performance in the new memory/storage hierarchy.
PMem comes in 128GB, 256GB, and 512GB capacities. A maximum of 4TB of PMem is supported by certain Intel processors. PMem mainly operates in two modes, Memory Mode and App-Direct Mode, and the three application modes used with PMem are as follows:

<table>
<thead>
<tr>
<th>Applications modes</th>
<th>Memory Mode</th>
<th>App-Direct Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The system DRAM (near memory) acts as the cache for PMem (far memory), and the system is similar to DRAM in performance when there is a data hit in DRAM. However, if the data is not in DRAM (known as &quot;near memory cache miss&quot;), the data will be read from PMem and the data is cached in DRAM. No changes need to be made to applications when using Memory Mode. Under this mode, the DRAM space is invisible to the system. Memory Mode may not be able to meet the requirements for some applications which only rely on DRAM speed to meet some performance requirements.</td>
<td>under this mode, both PMem and DRAM space can be managed by the system. Therefore, the application is required to manage the data distribution by itself. Intel's Persistent Memory Development Kit (PMDK), can be used for the persistency feature. The App-Direct Mode of PMem is usually achieved by mmap via the PMem-aware file system. In this scenario, there will be no system calls, no context switching, and no interrupts. Therefore, copy-on-write is also not supported for anonymous pages in the kernel. In Linux* Kernel 5.1 and later versions, the management of persistent memory can be passed to the MMU in the App-Direct mode. This allows the system to view the PMem as an anonymous space in a similar way to DRAM and treat the PMem as a slower and larger memory node.</td>
</tr>
</tbody>
</table>

Intel Optane PMem has now developed to the second generation. For more information about the product, please refer to the Intel Optane PMem page on the Intel website.
2. Hardware Tuning

This guide targets usage of RocksDB on 3rd Gen Intel Xeon Scalable processor with Intel Optane PMem 200 series. Parts of content might work on 2nd Gen Intel® Xeon® Scalable processor with Intel Optane PMem 100 series.

On Intel’s website you will find a list of OSs that support different Intel Optane PMem modes (Memory Mode [MM], App-Direct Mode [AD], and Dual Mode [DM]). The Linux® kernel must be v4.19 or above. If you want to use KMEM DAX-based Linux Kernel 5.1 or above is required.

2.1. BIOS Setting

Begin by resetting your BIOS to default setting, then change the default values to the below suggestions:

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Recommended Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyper-Threading</td>
<td>Disable</td>
</tr>
<tr>
<td>Hardware Prefetcher</td>
<td>Enable</td>
</tr>
<tr>
<td>L2 RFO Prefetch</td>
<td>Disable</td>
</tr>
<tr>
<td>Adjacent Cache Prefetch</td>
<td>Enable</td>
</tr>
<tr>
<td>DCU Streamer Prefetch</td>
<td>Enable</td>
</tr>
<tr>
<td>DCU IP Prefetch</td>
<td>Enable</td>
</tr>
<tr>
<td>LLC Prefetch</td>
<td>Enable</td>
</tr>
<tr>
<td>Total Memory Encryption (TME)</td>
<td>Disable</td>
</tr>
<tr>
<td>SNC (Sub NUMA)</td>
<td>Disable</td>
</tr>
<tr>
<td>UMA-Based Clustering</td>
<td>Hemisphere (2-clusters)</td>
</tr>
<tr>
<td>Configuration Item</td>
<td>Recommended Value</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Memory Bandwidth Boost Average Power (mW)</td>
<td>15000</td>
</tr>
<tr>
<td>Memory Bandwidth Boost Feature Enable</td>
<td>Enable</td>
</tr>
<tr>
<td>Memory Bandwidth Boost Average Power Limits (mW)</td>
<td>18000</td>
</tr>
<tr>
<td>Memory Bandwidth Boost Average Power Time Constant (ms)</td>
<td>15000</td>
</tr>
<tr>
<td>PMem QoS</td>
<td>Enable</td>
</tr>
<tr>
<td>DDR vs. DDRT Policy</td>
<td>BW Optimized (AD mode)</td>
</tr>
<tr>
<td>FastGo Configuration</td>
<td>Disable</td>
</tr>
<tr>
<td>Snoop mode for AD</td>
<td>Enable</td>
</tr>
<tr>
<td>Snoopy mode for 2LM</td>
<td>Enable</td>
</tr>
</tbody>
</table>
QoS Table 1: Intel® Optane™ persistent memory 100 series Platform

<table>
<thead>
<tr>
<th>Profile</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCPMM QoS Disabled</td>
<td>Feature disabled</td>
<td>Disabling &quot;run&quot; is recommended. Test whether enabling this option improves performance. Disabled by default from 536.D14</td>
</tr>
<tr>
<td>DCPMM QoS Profile 1</td>
<td>6 modules, 4 modules per socket (optimized)</td>
<td></td>
</tr>
<tr>
<td>DCPMM QoS Profile 2</td>
<td>2 modules per socket (optimized)</td>
<td></td>
</tr>
<tr>
<td>DCPMM QoS Profile 3</td>
<td>1 module per socket (optimized)</td>
<td>Available starting in 534.D01</td>
</tr>
</tbody>
</table>

QoS Table 2: Intel® Optane™ persistent memory 200 series Platform

<table>
<thead>
<tr>
<th>Profile</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable the PMem QoS Feature</td>
<td>Feature disabled</td>
<td>Disabling &quot;run&quot; is recommended. Test whether enabling this option improves your application.</td>
</tr>
<tr>
<td>Recipe 1</td>
<td>8+8 (DRAM + PMEM)</td>
<td></td>
</tr>
<tr>
<td>Recipe 2</td>
<td>8+4, 8+2, 8+1 (DRAM + PMEM)</td>
<td></td>
</tr>
</tbody>
</table>

2.2. Intel® Optane™ Persistent Memory 200 Series Hardware Settings

2.2.1. Non-Volatile Device Control (ndctl) and daxctl

The ndctl program is used to manage the Non-volatile memory device (libnvdimm) sub-system and namespace in the Linux kernel. The daxctl program provides enumeration and configuration commands to any devicedax namespace created. daxctl is required when using the devicedax namespace. Refer to the ndctl user guide for specific commands.

Several methods for installing ndctl and daxctl are described below:

- **Install using package manager**
  
  ```bash
  $ yum install ndctl
  $ yum install daxctl
  ```

- **Install from source code**
  
  Download source code.
Dependent package:

```
kmod-devel libudev-devel libuuid-devel json-c-devel
bash-completion libtool ruby asciidoc xmlto
```

Installation steps:

```
$ ./autogen.sh
$ ./configure CFLAGS=' -g -O2' --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib64
$ make -j & make install
```

2.2.2. IPMCTL

IPMCTL is a platform utility used to configure and manage Intel Optane PMem. It supports the following features:

- Discovery
- Configuration
- Firmware management
- Management of security features
- Health monitoring
- Performance tracking
- Debugging and troubleshooting

For more information on how to use IPMCTL, please refer to Persistent Memory Provisioning Introduction guide on the web. IPMCTL requires libsapec as a dependent component. Use the repo files for libsapec and IPMCTL.

Install using package manager:

```
$ yum install ipmctl
```

Install from source code:

Download source code.

Recommended platforms and versions:

Recommended version when installing on a platform using Intel Optane PMem 100 series: v1.X.X
Recommended version when installing on a platform using Intel Optane PMem 100 series: v2.X.X

2.2.3. Installing PMDK

Download PMDK from github

Install from source code:

```
$ git clone https://github.com/pmem/pmdk
```
2.2.4. Memory Mode Settings

Step 1: create goal

$ ipmctl create -f -goal memorymode=100

Step 2: restart server

$ shutdown -r now

After creation, you can use the numactl command to view the current memory size.

![Figure 2 Memory size]

2.2.5. App-Direct (AD) Mode Settings

Filesystem-DAX (fsdax)

Filesystem-DAX (fsdax) is the default namespace mode. If no other options are configured when using `ndctl create-namespace` to create a new namespace, a block device `/dev/pmemX[,Y]` will be created, which supports the DAX functionality of the Linux file system (xfs and ext4 currently supported). DAX deletes the page cache from the I/O path, and allows `mmap(2)` to establish direct mapping in the PMem medium. The DAX function enables workloads or working sets that overflowed from the page cache to be extended to PMem. Workloads suitable for page caching or ones that have batch data transfers might not benefit from using DAX.
Step 1: create goal

$ ipmctl create -f -goal (default is AD mode if no parameter is added)
$ ipmctl create -f -goal persistentmemorytype=appdirect

Step 2: restart server

$ shutdown -r now

Step 3: create namespace, with fsdax mode by default

$ ndctl create-namespace (When running the command once, one PMem will be generated)

Step 4: format device

$ mkfs.ext4 /dev/pmem0
$ mkfs.ext4 /dev/pmem1

Step 5: mount device

$ mount -o dax /dev/pmem0 /mnt/pmem0
$ mount -o dax /dev/pmem1 /mnt/pmem1
KMEM DAX

KMEM DAX is supported in Linux Kernel 5.1 and above. The KMEM DAX patch can also be ported to previous versions of the kernel. However, this requires additional configuration. A newer kernel can be used directly to quickly support the KMEM DAX feature. The steps to check whether the system supports KMEM DAX is shown below. If the system does not support it, you can upgrade the kernel or use the KMEM DAX kernel patch and add it to the existing kernel.

```bash
$ cat /boot/config-5.4.32 | grep KMEM
```

The steps to creating KMEM DAX is as follows:

**Step 1: create goal**

```bash
$ ipmctl create -f -goal persistentmemorytype=appdirect
```

**Step 2: restart server**

```bash
$ shutdown -r now
```

**Step 3: create namespace and set it as devdax mode**

```bash
$ ndctl create-namespace --mode=devdax --map=mem -r 0
```

**Step 4: set NUMA node**

```bash
$ daxctl reconfigure-device dax0.0 --mode=system-ram
```
Figure 6 Creating a NUMA node in KMEM DAX

**Note 1:** *dax_pmem_compat* is disabled in *modprobe*, if it exists. The *dax_pmem* module is deployed to convert it to the */sys/bus/dax* model, which might not be supported in kernel versions before v5.1. In this case, the command will return "nop" if the kernel has not been updated, and */sys/class/dax* changes to */sys/bus/dax* because it enables the alternative driver for the device dax instance, and in particular, the *dax_kmem* driver. Then, restart the system to allow the configurations to take effect. Refer to the documentation on *ndctl* and *daxctl*.

If *dax_pmem_core* is not *dax_pmem*, the following command can be used for conversion:

```
$ daxctl migrate-device-model
```

**Note 2:** the namespace can only be removed after the newly created node is taken offline.

```
$ daxctl offline-memory dax0.0
```

### 2.3. Memory Configuration/Settings

At least 1 DIMM DRAM & 1 DIMM PMem per memory channel needs to be populated. Refer 2.2.

### 2.4. Storage/Disk Configuration/Settings

<No specific workload setting for this topic>

### 2.5. Network Configuration

In the application scenario of RocksDB, since performance is usually limited by the bandwidth of the network rather than the performance of memory and persistent memory, running Redis across the network requires a large network card bandwidth (the larger, the better), which is recommended to be above 10GB/s.
3. Software Tuning

Software configuration tuning is essential. From the Operating System to RocksDB configuration settings, they are all designed for general purpose applications and default settings are almost never tuned for best performance.

3.1. Linux® Kernel Optimization Settings

3.1.1. CPU Configuration

- Configure the CPU to the performance mode

```
cpupower -c <cpulist> frequency-set --governor performance
```

- Configure the energy/performance bias

```
x86_energy_perf_policy performance
```

- Configure the minimum value for the processor P-State

```
echo 100 > /sys/device/system/cpu/intel_pstate/min_perf_pct
```

3.1.2. Kernel Settings

- Configure the corresponding CPU core parameters

```
sysctl -w kernel.sched_domain.cpu<x>.domain0.max_newidle_lb_cost=0
sysctl -w kernel.sched_domain.cpu<x>.domain1.max_newidle_lb_cost=0
```

- Configure the scheduling granularity

```
sysctl -w kernel.sched_min_granularity_ns=10000000
sysctl -w kernel.sched_wakeup_granularity_ns=1500000
```

- Configure the virtual memory parameters

```
sysctl -w vm.dirty_ratio = 40
sysctl -w vm.swappiness = 10
sysctl -w vm.dirty_background_ratio=10
```

3.2. RocksDB Architecture

RocksDB is implemented based on the LSM-Tree algorithm for SSD- and HDD-friendly workloads. RocksDB write process: data is added to Write-Ahead Log (WAL) first for every insert or update, then it is inserted into Memtable for sorting. If the Memtable is already full, it will be converted to an immutable Memtable, and the data is refreshed as the SST file to level 0 in the back-end. Similarly, when one level is full, "compaction" is triggered in the back-end, which takes the data in the SST file and merges it with a higher level SST file with overlapping key-range. The merged data takes the new SST file and writes it to a higher level, while the expired data is discarded. Since the higher level is 10 times larger than the lower level, compaction will cause serious write amplification and occupy a large amount of storage bandwidth. This is the main performance issue with LSM trees.
A Read operation starts with a Memtable search. Then a level-by-level search is performed in the SST files until the block data is found. If the block data is already in the block cache, the data is read directly from the cache (cache hit). Otherwise, the data is loaded onto the block cache from the SST file and read (cache miss). Block data is the smallest unit of I/O for a read operation and is usually larger than a key-value pair. Therefore, there will be a certain degree of read amplification.

3.3. RocksDB Tuning

3.3.1. Compiling RocksDB

Compilation Environment

Install g++, and ensure that the g++ version is recent and meets the requirements.

```bash
yum install gcc48-c++
```

Dependent Libraries

- **gflags**

  ```bash
git clone https://github.com/gflags/gflags.git
git checkout v2.0
cd gflags
./configure && make && make install
```

The updated version of gflags requires cmake for compilation:

```bash
mkdir build
cmake ..
make && make install
```
RocksDB Tuning Guide for 3rd Generation Intel® Xeon® Scalable Processors Based Platforms

- **snappy**
  
  `yum install snappy snappy-devel`

- **zlib**
  
  `yum install zlib zlib-devel`

- **bzip2**
  
  `yum install bzip2 bzip2-devel`

- **lz4**
  
  `yum install lz4-devel`

- **zstandard:**
  
  ```
  wget https://github.com/facebook/zstd/archive/v1.1.3.tar.gz
  mv v1.1.3.tar.gz zstd-1.1.3.tar.gz
  tar zxf zstd-1.1.3.tar.gz
  cd zstd-1.1.3
  make && make install
  ```

- **memkind**
  
  memkind v1.10.0 or above is required to build the block cache allocator. For more information on memkind, refer to the [GitHub memkind page](https://github.com/memkind/memkind).

  ```
  yum --enablerepo=PowerTools install memkind-devel
  ```

### Compiling RocksDB

**Clone RocksDB**

```
git clone https://github.com/facebook/rocksdb.git
```

Compile the static library, release mode, and obtain **librocksdb.a**.

```
make static_lib
```

Compile the dynamic library, obtain **librocksdb.so**, release mode, and obtain **lbrocksdb.so**.

```
make shared_lib
```

### 3.3.2. Optimizing RocksDB in PMEM

**Storage Over AD**

The App Direct mode of PMem supports reading and writing of files directly from an application. Hence, an SST file can be stored directly in the PMem. The default IO mode for the operating system is Cache IO (page cache). The PMem Aware File System is supported in Kernel v4.19 and above. This enables a file system to use the block device drivers of the I/O sub-system, or to directly use PMem for byte-addressable load/store memory bypassing the I/O sub-system. This is the fastest and shortest access to data in PMem. This approach does not require I/O operations, which also
enables better performance than traditional block devices by allowing for small data writes at faster speeds. Traditionally, a block storage device requires the file system to access the raw block size of the device and make changes to the database, before writing the entire data block back to the device.

Based on the principles mentioned above, storing Rocksdb data files directly in PMem will yield direct benefits. As shown in the figure below, the paths for db and WAL just need to be redirected to PMem and no source code needs to be changed.

In order to improve read performance, block cache is configured in Rocksdb. At the same time, direct I/O can be configured to access storage devices directly instead of using the system's page cache. When the required data can be found in the block cache, this is known as "cache hit". Otherwise, a "cache miss" is triggered. Therefore, the block cache hit rate greatly affects read performance.

**RocksDB Write Optimization**

The performance of RocksDB in PMem can be further optimized using functions from the PMDK library.

**WAL**

Each update in RocksDB will be written to two locations:
1. A memory data structure called Memtable (which is then written to the SST file)

2. The WAL log on the disk. When a crash occurs, the WAL log can be used to completely restore the data in the Memtable to ensure the database is recovered to its original state.

With the default configurations, RocksDB calls `fflush` on WAL after each write operation to ensure consistency. When using PMem, the `mmap` function can be used to map the WAL file to the corresponding virtual memory space for reading and writing, and the NT-Store command can be used to increase write efficiency.

KVS

The Key-value separate (KVS) solution can be used to effectively improve the overhead generated by compaction in the RocksDB back-end. In the KVS solution, the SST file forms a key-value data structure, and its value is stored as a pointer that points to the actual value/data residing in the PMem. Therefore, during the compaction process, it is no longer necessary to merge and rewrite a large number of values, and you only need to work with the pointer.

For KVS, in RocksDB, big data is stored in PMem using the `libpmemobj` library as a part of PMDK. The following features are used:

- `pmemobj_create`: used to create pool space to store values in PMem
- `pmem_memcpy_persist`: used to copy and refresh data to PMem
- `pmemobj_free`: used to release pool space

The KVS solution is very good for operations that involve a lot of write and compression operations, as well as for applications with large values. The drawback is that the read performance is not ideal.

Read Optimization in RocksDB

Layered

The overall data set can be huge in some client applications. Therefore, data at low levels in RocksDB can be stored in the PMem, and the data at high levels can be stored on Non-Volatile Memory Express* (NVMe*). The benefit of this solution is that data at lower levels are newer and more frequently used, so PMem has advantages over SSDs, and increases the overall performance of RocksDB. It also provides users with a "cold data" option using cheap storage media.
Dynamic LRU

Based on the layered solution mentioned in the previous section, the priority levels for the data in the LRU queue in the block cache can also be changed to increase the cache hit rate. As discussed earlier in the paper, data access to PMem is relatively faster than SSDs. Therefore, when there is a block cache miss in an SSD, the data needs to be read from the data file in the SSD, which reduces overall performance.

Therefore, cache data loaded in the LRU can be fine-tuned for increased performance. The solution is to set the priority of the most frequently-used meta block as the highest, and the priority for data in SSDs as the next highest, while data in PMem is set to the lowest priority. The advantage of this method is that it improves the cache hit rate of data in SSD. For more information refer to the GitHub pmem-rocksdb webpage.

Fine Grained

Traditional file systems use a block size of 4k for reading and writing, while byte-addressable configurations are available in PMem. Therefore, a smaller block size and mmap can be used to bring improvements to read performance.
This solution reduces read amplification and also bypasses the block cache. Index metadata can also move from PMem to DRAM.

Our experiments were run on a machine equipped with Intel® Xeon® Platinum 8269C CPU, all experiments running on one socket with 24 cores, 64GB memory, and four interleaved 128GB AEP devices, read and write performance with DBBench (the default benchmark tool of RocksDB). We issue read and write requests under random key distribution, and fix key size to 16 bytes. We use different value size with 128, 512 and 1024 bytes. 16 threads for background compactions, and set 8 sub compaction threads (concurrent compaction in level 0) to avoid write stall on level 0. Other unmentioned configurations are as default of DBBench. Read performance improves many times faster when compared to NVMe SSDs since block cache is not required. This also saves DRAM usage by a significant amount.

**Block Cache on PMem**

The block cache is by default allocated on DRAM. It is possible to allocate the block cache on PMem using the MemkindKmemAllocator introduced with [PR #6214](#), merged in RocksDB since v6.10. Placing the block cache on PMem allows allocation of large caches (up to TBs in size), which is useful for application with large working sets. PMem is used in volatile mode and the cache is not persistent across restarts (same behavior as for a block cache on DRAM).

The allocator expects PMem to be set up to support KMEM DAX mode, and it uses the memkind library internally for allocation. Version 1.10.0 of the memkind library is required for KMEM DAX support. The allocator needs to be specified when creating an LRU cache (if no allocator is specified, the default DRAM allocator is used).

```c
#include "rocksdb/cache.h"
#include "memory/memkind_kmem_allocator.h"

NewLRUCache(
    capacity /*size_t*/,
    6 /*cache_numshardbits*/,
    false /*strict_capacity_limit*/,
    false /*cache_high_pri_pool_ratio*/,
    std::make_shared<MemkindKmemAllocator>());
```
3.3.3. PlainTable Optimization

PlainTable is a RocksDB SST file format which is optimized for low-latency memory only or very low-latency mediums.

**Benefits:**
- An index is established in the memory, and a "binary + hash" method is used to replace direct binary searching
- Block cache is bypassed to prevent resource waste caused by block copying and the LRU caching mechanism
- All memory copying (mmap) is avoided during searching

**Limitations:**
- File size needs to be smaller than 31 bits integer
- Data compression is not supported
- Delta encoding is not supported
- `Iterator::Prev()` is not supported
- Non prefix-based seek is not supported
- Loading a table more slowly than building an index
- mmap mode is supported only

The common features of PlainTable makes it very suitable for PMem. PlainTable uses mmap (memory mapping file) to prevent page caching, and also to avoid context switching in the kernel. In this solution, `libpmem` is used to optimize the normal table in the `io_posix.c` file. `pmem_memcpy_nodrain` is used to optimize write and flush performance. Instead of flushing it, the `pmem_memcpy_nodrain` based on the non-temporary (NT) write instruction sets can bypass the cache. For more information refer to the GitHub `pmem-rocksdb` webpage.
4. The RocksDB Engine

4.1. Kvrocks*

kvrocks is an open source key-value database. It is based on Rocksdb and is compatible with the Redis* protocol. Compared to Redis, it is intended to reduce memory cost and improve function. The design of the copy and storage functions are inspired by rocksplicator and blackwidow. kvrocks has the following main features:

- Redis protocol: users can use a Redis client to access kvrocks
- Namespace: similar to redis db, but each namespace uses tokens
- Copy: uses MySQL and other binlog for asynchronous copying
- High availability: when there is a fault in the main or slave server, failover of redis sentinel is supported
- Codis protocol: users can use the Codis proxy and dashboard to manage kvrock

4.2. Basic kvrocks Performance

Basic performance of kvrocks in various scenarios, which can be used as reference.

- NVMe: using NVMe SSDs for data storage
- PMem: using PMem for data storage
- Ramfs: using the memory file system for data storage
- dummy: The RocksDB engine is set as null in the read and write function to measure the performance of the upper-layer structure

Normally, that NVMe SSD performs the worst, and that the performance of ramfs is about twice that of PMem and NVMe SSD, while the performance of "dummy" is multiple times that of NVMe SSD and PMem. By observing the performance of "dummy", to find out where the performance optimization bottleneck lies.

4.3. kvrocks Write Performance

4.3.1. WAL mmap Optimization

After optimizing WAL mmap, the write performance of PMem is about double that of NVMe SSD. After optimizing WAL, "write stress" is reduced

4.3.2. KVM Optimization Solution

KVM results that there are significant improvement for large values. The latency is greatly reduced and is very stable. Therefore, KVM is suitable for applications with larger values.

4.4. kvrocks Read Performance

Fine Grained Optimization Solution

The fine grained solution sets the block size to 256 bytes, which is exactly the granularity used for writing in the first and second generation PMem. Compared to the original granularity of 4k, the read performance improves multiple times faster.
5. Related Tools

5.1. Performance Monitoring Tool - PCM

PCM (Processor Counter Monitor) can be used to monitor the performance indicators of the Intel CPU core. PCM is often used to monitor the bandwidth of the PMem.

$ pcm-memory.x -pmm

6. Conclusion

The combination of RocksDB, 3rd Generation Intel Xeon Scalable processor, and Intel Optane PMem is a very practical way to use RocksDB. The large storage capacity of persistent memory supports the RocksDB application to use larger data sets, and through optimization, it can get very good performance.

7. Feedback

We value your feedback. If you have comments (positive or negative) on this guide or are seeking something that is not part of this guide, please reach out and let us know what you think.

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.