Redis* Tuning Guide on 3rd Generation Intel® Xeon® Scalable Processors & Intel® Optane™ Persistent Memory

Revision Record........................................................................................................................................... 2
1. Introduction ............................................................................................................................................... 3
2. Intel Persistent Memory Configuration .................................................................................................. 4
  2.1. Intel PMem DIMM Slot Configuration ............................................................................................... 4
  2.2. Memory mode Settings ....................................................................................................................... 5
3. BIOS Settings ........................................................................................................................................... 5
4. Linux Optimizations ............................................................................................................................... 6
  4.1. Network Configuration ...................................................................................................................... 6
  4.2. Disk Configuration ............................................................................................................................ 6
  4.3. CPU Configuration ............................................................................................................................ 6
  4.4. Kernel Configuration ......................................................................................................................... 6
5. Redis Settings .......................................................................................................................................... 7
  5.1. Redis on Memory Mode Setting ....................................................................................................... 7
6. Tools ......................................................................................................................................................... 7
  6.1. ipmctl .................................................................................................................................................. 7
  6.2. numactl .............................................................................................................................................. 8
  6.3. Performance monitoring tool - PCM ................................................................................................. 9
## Revision Record

<table>
<thead>
<tr>
<th>Date</th>
<th>Rev.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/11/2021</td>
<td>1.2</td>
<td>Corrected an error on max PMem</td>
</tr>
<tr>
<td>04/06/2021</td>
<td>1.0</td>
<td>Initial public release.</td>
</tr>
</tbody>
</table>
**1. Introduction**

Redis* (Remote Dictionary Server) is an open source (under BSD licenses) memory-based data structure storage, and can serve as a database, cache, and more. Redis supports a variety of data structures, such as strings, hash tables, lists, sets, sorted sets with range queries, bitmaps, HLLs, and geospatial indexes and streams. Redis provides functions such as built-in replications, Lua scripts, LRU evictions, transactions, and disk persistence at different levels. Redis features a high performance and a low latency, and is widely used in gaming, videos, news, location, finance, and other fields. For more information on Redis, please visit: https://redis.io/.

![Redis Storage Data Diagram](image1)

**Figure 1 Redis Storage Data Diagram**

Intel® Optane™ persistent memory (PMem) is an innovative memory technology and is a new category between memory and storage. Intel® persistent memory is similar to DRAM in performance and can store data persistently like an SSD. Intel persistent memory has three single capacities of 128 GB, 256 GB and 512 GB, and up to 4TB Intel persistent memory capacity can be supported per processor per server, which is much larger than that of DDR, and at a lower price.

![Storage Hierarchy](image2)

**Figure 2 Storage Hierarchy**

Intel persistent memory supports the following two modes:

1) **Memory Mode**: the operating system views Intel persistent memory as the system’s primary memory, DRAM as the cache for Intel persistent memory. It does not require making any changes to your applications before they run in this large volatile memory pool.

2) **App-Direct Mode**: Available for both DRAM and Intel persistent memory. In this case, Intel persistent memory is a low latency, byte addressable memory pool, requiring your applications to manage hierarchical data storage, and thus making changes to your applications.
In Memory Mode, your applications can run without any changes required. As it is easy to use, Redis which uses Memory Mode has been widely used.

This document provides suggestions for configuring various performance optimizations when Redis uses Memory Mode.

This document is applicable to 2-socket 3rd Generation Intel® Xeon® Scalable Processor systems & Intel® Optane™ persistent memory 200 series.

2. Intel Persistent Memory Configuration

2.1. Intel PMem DIMM Slot Configuration

- 3rd Gen Intel Xeon Scalable platforms (codename Ice Lake/Whitley) have 16 DDR4 Slots per CPU
- Intel Optane persistent memory 200 series can be configured up to 16 per CPU
- General ratio (DRAM : PMem) is between 1:4 to 1:16, with minimum 1:2
2.2. Memory mode Settings

1. Please install ipmctl according to your OS version.
2. Check dimm information

   $ ipmctl show -dimm

3. Configure Memory Mode

   $ ipmctl create -f -goal MemoryMode=100

4. Reboot server

   $ shutdown -r now

5. Check current memory size

   $ numactl -H

![Figure 5 Check current memory size](image)

3. BIOS Settings

The configuration items and recommended values that can be optimized for BIOS are as follows:

<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 Disable</td>
<td>Disable</td>
</tr>
<tr>
<td>Adjacent Cache Prefetch</td>
<td>Enable</td>
</tr>
<tr>
<td>DCU Streamer Prefetcher</td>
<td>Enable</td>
</tr>
<tr>
<td>DCU IP Prefetcher</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>200 Series PMem Memory Bandwidth Boost (MBB)</td>
<td>18000</td>
</tr>
<tr>
<td>200 Series PMem Turbo/Memory Bandwidth Boost Feature (MBB) Power Limit (in MW)</td>
<td>18000</td>
</tr>
<tr>
<td>PMem QoS</td>
<td>Disable the PMem QoS Feature</td>
</tr>
<tr>
<td>PMem Performance Setting</td>
<td>BW Optimized</td>
</tr>
<tr>
<td>PMem FastGo Configuration</td>
<td>Auto</td>
</tr>
<tr>
<td>Snoop mode for AD</td>
<td>Disable</td>
</tr>
<tr>
<td>Snoopy mode for 2LM</td>
<td>Disable</td>
</tr>
<tr>
<td>Boot performance mode</td>
<td>Max Performance</td>
</tr>
<tr>
<td>Turbo Mode</td>
<td>Enable</td>
</tr>
<tr>
<td>Hardware P-State</td>
<td>Native Mode</td>
</tr>
<tr>
<td>Local/Remote Threshold</td>
<td>Auto</td>
</tr>
</tbody>
</table>

4. Linux Optimizations

4.1. Network Configuration

In the Redis application scenario, performance is usually restricted more by the bandwidth of the network than the performance of memory and Intel persistent memory. Therefore, when you run Redis across networks, you need an NIC with a highest possible network bandwidth. It is recommended that the value is above 10GB/s.

4.2. Disk Configuration

- Configure disk readahead size
  
  ```bash
  echo 4096 > /sys/block/sd<x>/queue/read_ahead_kb
  ```

4.3. CPU Configuration

- Configure performance mode for the CPU

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

- Configure the power-performance gap

  ```bash
  x86_energy_perf_policy performance
  ```

- Configure the minimum P-State value for the processor

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

4.4. Kernel Configuration
Redis Tuning Guide on 3rd Generation Intel® Xeon® Scalable Processors and Intel® Optane™ persistent memory 200 Series

- Configure kernel parameters for the CPU
  
  ```
  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 scheduling granularity
  
  ```
  sysctl -w kernel.sched_min_granularity_ns=10000000
  sysctl -w kernel.sched_wakeup_granularity_ns=15000000
  ```

- Configure virtual memory parameters
  
  ```
  sysctl -w vm.dirty_ratio = 40
  sysctl -w vm.swappiness = 10
  sysctl -w vm.dirty_background_ratio=10
  ```

5. Redis Settings

5.1. Redis on Memory Mode Setting

In Memory Mode, your application runs without any changes required, so simply download and compile the open source Redis, and run it.

1) Download source code
   
   Github address: https://github.com/redis/redis

2) Compile command (for more details, see https://github.com/redis/redis/README.md)

   ```
   $ make
   ```

In addition, for each running Redis instance, in order to reduce access across the Numa node which affects performance, different Redis instances can be bound to different sockets and cores using numactl and taskset.

   ```
   numactl -C <cpu>/redis-server ./redis.conf > ./redis.log
   ```

6. Tools

6.1. ipmctl

ipmctl is a platform utility program used to configure and manage Intel Optane PMem, and it supports the following functions:

- Discovery
- Configuration
- Firmware management
- Security management
- Health monitoring
- Performance tracking
- Debugging and troubleshooting
1. For specific usages, refer to:

2. ipmctl requires libsafecl as a dependent component. The following are repo files for libsafecl and ipmctl

3. Installation method:
   - Package manager installation:

   ```bash
   $ yum install ipmctl
   ```

   - Source code installation:

   Source code download: https://github.com/intel/ipmctl

6.2. `numactl`

The `numactl` tool can be used to check the configuration and status of the NUMA node of the current server. It binds a process to the specified CPU core which runs the corresponding process.

1. Installation method:

   ```bash
   $ yum -y install numactl numastat
   ```

2. Command usage examples:

   1) Check NUMA configurations of the current server using `numactl`

   The `numactl` performance results show that the example server is divided into four NUMA nodes. Each node contains 16 CPU cores, and the memory size for each node is about 64 GB. At the same time, this command also provides the distance between different nodes. The farther the distance, the greater latency the memory access across-NUMA nodes. When an application program is running, you should have less memory access across-NUMA nodes.
2) Bind a process to the specified CPU core using `numactl`

Using the "numactl -C 0-15 top" command, you can bind the process "top" to the "0-15 CPU core" that will run the process.

```
[root@localhost ~]# numactl -H
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
node 0 size: 64794 MB
node 0 free: 55404 MB
node 1 cpus: 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
node 1 size: 65404 MB
node 1 free: 58642 MB
node 2 cpus: 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
node 2 size: 65404 MB
node 2 free: 61181 MB
node 3 cpus: 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
node 3 size: 65404 MB
node 3 free: 55592 MB
node distances:
node 0 1 2 3
  0: 10 15 20 20
  1: 15 10 20 20
  2: 20 20 10 15
  3: 20 20 15 10
```

3) Check the memory access hit rate for the current NUMA node using `numastat`

The `numastat` command observes the states of various NUMA nodes.

```
[root@localhost test]# numactl -C 0-15 top
```

```
2276 root 20 0 118400 8448 3712 S 1.3 3.0 1:51.20 top
22003 root 20 0 161.9q 8.4q 86400 S 1.0 3.3 322:53.26 impalad
```

- `numa_hit` indicates the number of times the CPU core on the node accesses local memory.
- `numa_miss` indicates the number of times the CPU core on the node accesses the memory for other nodes. Cross-node memory access has a high latency, thereby degrading performance. As a result, the lower the `numa_miss` value the better. And if it is too high, core binding should be considered.

### 6.3. 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 Intel persistent memory. Download here: [https://github.com/opcm/pcm](https://github.com/opcm/pcm)

```
$ pcm-memory.x -pmm
```
Redis Tuning Guide on 3rd Generation Intel® Xeon® Scalable Processors and Intel® Optane™ persistent memory 200 Series

Figure 1 PCM monitoring Persistent Memory bandwidth

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.