Hewlett Packard Enterprise

Technical white paper

Using NVDIMM Persistent Memory Server Technology with Linux

Contents

Prerequisites

To use NVDIMMs as non-volatile memory a supported OS is needed. NVDIMMs are currently enabled in these distro releases:

- SLES 12 SP2, SP3
- RHEL 7.3, 7.4
- Fedora 24, 25, 26

Future releases of SLES and RHEL can be expected to continue having full NVDIMM support. Developers can enable NVDIMM support by building an upstream kernel, e.g. 4.14, for testing. If an older kernel or OS is used the NVDIMMs will be treated as regular volatile memory.

NVDIMMs supported modes

NVDIMMs can be configured in four distinct modes each with their own advantages and disadvantages: raw, sector, memory, and DAX. To configure an NVDIMM into different modes use your distro's package manager to install—ndctl. It's a utility for managing non-volatile memory devices on a Linux® system. Refer to the ndctl man page for a more thorough explanation.

- Raw—presents as /dev/pmemN a block device
  - Default mode configuration when first installing NVDIMMs into a system
  - Supports filesystems with or without DAX—(ext4, xfs)
  - A raw pmem namespace does not support sector atomicity by default, see "sector" mode instead
- Sector—presents as /dev/pmemNs a block device with sector atomicity
  - Implemented using the Block Translation Table (BTT) driver that guarantees power-fail write atomicity
  - Only supports filesystems without DAX
- Memory—presents as /dev/pmemN a block device supporting device DMA
  - Supports filesystem DAX (ext4, xfs)—recommended over raw mode
  - Requires storing extra "struct page" entries on regular system memory or persistent memory
    - map=mem—regular system memory. Intended for small persistent memory capacities, 128 MiB for 8 GiB persistent memory
    - map=dev—persistent memory. Intended for large persistent memory capacities, 7.45 GiB for 1 TB persistent memory
- DAX—presents as /dev/daxN.M a character device supporting DAX

Note
DAX is not supported in RHEL 7.3 or SLES 12 SP2

  - Allows memory ranges to be allocated and mapped without the need of a filesystem
  - No interactions with the kernel page cache
  - As a character device the namespace does not support filesystems
  - Requires storing extra "struct page" entries in persistent memory

Configuring NVDIMMs using ndctl

The basic syntax to configure an NVDIMM is:

```
ndctl create-namespace -f -e namespaceN.0 -m <mode>
```

Where "N" represents the pmem namespace that will be changed. The namespace needs to be specified, e.g., namespace1.0, and the mode for the NVDIMM, raw, sector, memory, or DAX needs be given.

```
ndctl create-namespace -f -e namespace1.0 -m memory
{
  "dev":"namespace1.0",
  "mode":"memory",
  "size":16909336576,
  "uuid":"053d1cfe-3aea-4f9e-9752-68a16fdcce3c",
  "blockdev":"pmem1"
}
ndctl create-namespace -f -e namespace1.0 -m sector
{
  "dev":"namespace1.0",
  "mode":"sector",
  "size":17162027008,
  "uuid":"db94f65e-9bec-4b58-a11b-9f065b036802",
  "sector_size":4096,
  "blockdev":"pmem1s"
}
ndctl create-namespace -f -e namespace1.0 -m raw
{
  "dev":"namespace1.0",
  "mode":"raw",
  "size":17179869184,
  "blockdev":"pmem1"
}
ndctl create-namespace -f -e namespace1.0 -m dax
{
  "dev":"namespace1.0",
  "mode":"dax",
```

```
  “size”:8453619712,
  “uuid”:“aef1a696-55cc-4aca-9dcb-f234b45fd00f”,
  “daxregion”:{
    “id”:1,
    “size”:8453619712,
    “align”:4096,
    “devices”:[
      {
        “chardev”:“dax1.0”,
        “size”:8453619712
      }
    ]
  }
}
```

File System DAX Access

Mounting a filesystem (xfs, or ext4) with direct access (DAX) removes the extra copy to the page cache by performing reads and writes directly to the storage device without changing an application. Referring to the section above, NVDIMMs supported modes, mounting a filesystem in DAX mode is only supported for NVDIMMs in raw or memory mode, but not sector mode. Memory mode is the recommend setting for an NVDIMM to use filesystem DAX.

1. Configure an NVDIMM into memory mode refer to the section above Configuring NVDIMMs
2. Create ext4 or xfs file system on an NVDIMM, e.g., ext4 on /dev/pmem0
   ```
   $ sudo mkfs -t ext4 /dev/pmem0
   ```
3. Mount the file system with DAX option
   ```
   $ sudo mkdir /mnt/pmem0
   $ sudo mount -o dax /dev/pmem0 /mnt/pmem0
   ```

Note
As of this writing mounting a filesystem with DAX enabled is still an experimental feature. The following message will appear in the logs
```
$ DAX enabled. Warning: EXPERIMENTAL, use at your own risk
```

After mounting the file system with the –o DAX option, any I/O to the files under /mnt/pmem0 will be performed directly to the NVDIMM without going through the page caches. That is:

- Read/Write—Data is copied directly from/to NVDIMM i.e., a single copy
- mmap—Data is loaded/stored directly from/to NVDIMM, i.e., zero copy

DAX also bypasses the block I/O stack to minimize the software overhead.

Optionally the NVM Libraries (ex. libpmemobj, libpmem) can be used to create/access files on a pmem device. NVML adds convenient library interfaces for memory-mapped (mmap) persistence, this library is optimized for persistent memory. The recommended library for use is libpmemobj, this library provides nice wrapper functions for low-level memory manipulation. If you need to roll your own persistent memory algorithms libpmem is the recommend library to use with official documentation on the NVM Libraries found here.

File System Access

1. Create any filesystem on an NVDIMM—ext4 on /dev/pmem0

```
$ sudo mkfs -t ext4 /dev/pmem0
```

2. Mount the device

```
$ sudo mkdir /mnt/pmem0
$ sudo mount /dev/pmem0 /mnt/pmem0
```

All the I/O to the files under the "/mnt/pmem0" will be performed through the page caches, unless an application specifies O_DIRECT in open() for the read()/write() operations. The mmap() operation always accesses to the page caches.

Note
When using an NVDIMM as a traditional block device, block or sector atomicity for I/O operations is not guaranteed.

Raw Block Device Access

Opening "/dev/pmem0" allows access to an NVDIMM through the page caches without a filesystem being present. Adding the O_DIRECT option in open() allows bypassing the page caches for the read()/write() interfaces, but not for the mmap() interface.

Note
When using an NVDIMM as a traditional block device, block or sector atomicity for I/O operations is not guaranteed.

File System Access with Sector Atomicity

The Block Translation Table (BTT) assures sector atomicity when updating data on an NVDIMM. The tool "ndctl" is used to manage the BTT and other NVDIMM usage modes. Root access is needed to bind the BTT driver to a specific NVDIMM device.

Note
This step overwrites the data on pmem0 and creates a "/dev/pmem0s" device. This BTT binding persists across reboots.

1. Bind the BTT driver to the specific NVDIMM device, refer to the section above Configuring NVDIMMs
2. Create a filesystem on the new block device. E.g., Ext4 on /dev/pmem0s:

```
# mkfs -t ext4 /dev/pmem0s
```

3. Mount it:

```
# mkdir /mnt/pmem0
# /dev/pmem0s /mnt/pmem0
```

Raw Block Access with Sector Atomicity

The steps required to bind/unbind BTT is the same as the above section, File System Access with Sector Atomicity. Opening "/dev/pmem0s" allows access to NVDIMM through the page caches without a File System. Open with O_DIRECT option allows bypassing the page caches for the read()/write() interfaces, but not for the mmap() interfaces.

Known issues

1. RHEL 7.3 does not report health information using “ndctl list –health” option. This is fixed in RHEL 7.4.
2. For RHEL 7.3 and SLES 12 SP2 the command “ndctl list --dimms –health” is not supported to retrieve NVDIMMs health status. On Gen-9 HW you need to perform “modprobe acpi_ipmi” before running “ndctl list --dimms –health” to retrieve NVDIMMs health status.
3. iostat(1) does not report I/O statistics information on NVDIMM devices when DAX is used.
4. SLES 12 SP3 requires a maintenance update to properly use ndctl to configure NVDIMM. Without this maintenance update ndctl will become hung when configuring the NVDIMMs into different modes.

Troubleshooting

General

Verify that you are using up-to-date ROM, CPLD, and iLO firmware.

/dev/pmemX is missing

1. Verify that your NVDIMMs are installed correctly and that you have followed the population recommendations.
2. Verify that NVDIMM functionality is enabled in RBSU.
3. Verify that you see some NVDIMM-related messages output to the console towards the end of POST.
4. Verify that iLO reports the NVDIMMs as healthy.
5. Verify that you are running a kernel with NVDIMM enablement. If you don’t see dmesg output indicating that the ACPI NFIT table is being processed, then your kernel isn’t NVDIMM enabled.

/dev/pmemX is read-only

When (or If?) Linux complains (when mounting, for example), that a /dev/pmemX is read-only. You can also see this in the RO column of lsblk output. Check the NVDIMM status flags set in the ACPI NFIT:

```
$ grep -H . /sys/bus/nd/devices/*/nfit/flags
```

It is normal for no flags to be set and for the grep command to display no output. If you see a flag of “not_armed”, then this means that the NVDIMM is not armed for backup and so Linux marks it read-only. NVDIMMs will be marked not_armed if ROM is not able to arm them for some reason, such as an erase error. Check the IML or iLO log for errors and contact HPE support.

/sys/bus/nd or /dev/nmemX is missing

Often this means that NVDIMM functionality is not enabled in RBSU; verify that NVDIMM is enabled in RBSU. Sometimes upgrading your ROM will cause NVDIMM to become disabled.

Learn more at

HPE NVDIMM

NVM Library

ndctl github

a00036172ENW, November 2017