# DAOS System Deployment

## Preflight Checklist

This section covers the preliminary setup required on the compute and storage nodes before deploying DAOS.

### Time Synchronization

The DAOS transaction model relies on timestamps and requires time to be synchronized across all the storage and client nodes. This can be done via NTP or any other equivalent protocol.

### User & Group

DAOS requires users and groups to be synchronized on both storage and client nodes.

### Runtime Directory Setup

DAOS uses a series of Unix Domain Sockets to communicate between its various components. On modern Linux systems Unix Domain Sockets are typically stored under /run or /var/run (usually a symlink to /run) and are a mounted tmpfs file system. There are several methods for ensuring the necessary directories are setup.

A sign that this step may have been missed is when starting daos\_server or daos\_agent you may see the message:

mkdir /var/run/daos\_server: permission denied

Unable to create socket directory: /var/run/daos\_server

#### Non-default Directory

By default daos\_server and daos\_agent will use the directories /var/run/daos\_server and /var/run/daos\_agent respectively. To change the default location that daos\_server uses for its runtime directory either uncomment and set the socket\_dir configuration value in install/etc/daos\_server.yaml or pass the location to daos\_server on the command line using the -d flag. For the daos\_agent an alternate location can be passed on the command line using the -runtime\_dir flag.

#### Default Directory (non-persistent)

Files and directories created in /run and /var/run only survive until the next reboot. However if reboots are infrequent an easy solution while still utilizing the default locations is to manually create the required directories. To do this execute the following commands.

daos\_server:

* mkdir /var/run/daos\_server
* chmod 0755 /var/run/daos\_server
* chown user:user /var/run/daos\_server (where user is the user you will run daos\_server as)

daos\_agent:

* mkdir /var/run/daos\_agent
* chmod 0755 /var/run/daos\_agent
* chown user:user /var/run/daos\_agent (where user is the user you will run daos\_agent as)

#### Default Directory (persistent)

If the server hosting daos\_server or daos\_agent will be rebooted often systemd provides a persistent mechanism for creating the required directories called tmpfiles.d. This mechanism will be required every time the system is provisioned and requires a reboot to take effect.

To tell systemd to create the necessary directories for DAOS:

* Copy the file utils/systemd/daosfiles.conf to /etc/tmpfiles.d   
  cp utils/systemd/daosfiles.conf /etc/tmpfiles.d
* Modify the copied file to change the user and group fields (currently daos) to the user daos will be run as
* Reboot the system and the directories will be created automatically on all subsequent reboots.

### Root Privilege Access

Several tasks (e.g. storage access, hugepages configuration) performed by the DAOS server requires elevated permissions on the storage nodes.

NVMe access through SPDK as an unprivileged user can be enabled by first running sudo daos\_server prep-nvme -p 4096 -u bob. This will perform the required setup in order for daos\_server to be run by user "bob" who will own the hugepage mountpoint directory and vfio groups as needed in SPDK operations. If the target-user is unspecified (-u short option), the target user will be the issuer of the sudo command (or root if not using sudo). The specification of hugepages (-p short option) defines the number of huge pages to allocate for use by SPDK.

The configuration commands that require elevated permissions are in src/control/mgmt/init/setup\_spdk.sh (script is installed as install/share/setup\_spdk.sh).

The sudoers file can be accessed with command visudo and permissions can be granted to a user to execute a specific command pattern (requires prior knowledge of daos\_server binary location):

linuxuser ALL=/home/linuxuser/projects/daos\_m/install/bin/daos\_server prep-nvme\*

See daos\_server prep-nvme --help for usage.

### Storage Detection & Selection

While the DAOS server will eventually auto-detect all the usable storage, the administrator will still be provided the ability through the configuration file (see next section) to whitelist or blacklist the storage devices to be (or not) used. This section covers how to manually detect the storage devices potentially usable by DAOS in order to populate the configuration file when the administrator wants to have finer control over the storage selection.

#### SCM

This section addresses how to verify that Optane DC Persistent memory (DCPM) is correctly installed on the storage nodes and how to configure it in interleaved mode to be used by DAOS in AppDirect mode. Instructions for other type of SCM may be covered in the future.

DCPM can be configured and managed through the IpmCtl[[1]](#footnote-2) library and associated tool. The ipmctl command just be run as root and has pretty detailed man pages and help output (use “ipmctl help” to display it).

The list of NVDIMMs can be displayed as follows:

$ ipmctl show -dimm

DimmID | Capacity | HealthState | ActionRequired | LockState | FWVersion

==============================================================================

0x0001 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

0x0101 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

0x1001 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

0x1101 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

As for affinity to CPU, use the following command:

# ipmctl show -dimm

DimmID | Capacity | HealthState | ActionRequired | LockState | FWVersion

==============================================================================

0x0001 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

0x0101 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

0x1001 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

0x1101 | 502.5 GiB | Healthy | 0 | Disabled | 01.00.00.5127

Moreover, DAOS requires DCPM to be configured in interleaved mode. A command mode option (--set-interleaved) can be used as a "one-shot" invocation of daos\_server and must be run as root. SCM modules will be configured into interleaved regions with memory mode set to "app-direct" with one set per socket (each module is assigned to socket and reports this via its NUMA rating). This configuration may require a reboot and daos\_server will exit on completion of the task.

This can be done manually via the following commands:

# to verify there is non-volatile memory type

$ ipmctl show -a -topology | egrep 'Capac|MemoryType'

  MemoryType=DCPM

  Capacity=502.5 GiB

  MemoryType=DCPM

  Capacity=502.5 GiB

  MemoryType=DCPM

  Capacity=502.5 GiB

  […]

$ ipmctl create -goal PersistentMemoryType=AppDirect

A reboot is required after those changes.

#### SSDs

DAOS supports only NVMe-capable SSDs that are accessed directly from userspace through the SPDK library.

In order to make the SSDs available to SPDK, the setup script is run to unbind the devices from original kernel drivers and then bind the devices to a generic driver through which SPDK can communicate. The devices are then bound back to the original drivers when management activities have been completed in the control plane.

A command mode option (--show-storage) can be used as a "one-shot" invocation of daos\_server and must be run as root to display all the locally attached SSDs (and also SCM modules) usable by DAOS.

$ daos\_server show-storage

[…]

Listing attached storage...

NVMe:

- id: 0

model: 'INTEL SSDPED1K375GA '

serial: 'PHKS7335006W375AGN '

pciaddr: 0000:81:00.0

fwrev: E2010420

namespace:

- id: 1

capacity: 375

The pciaddr field above is what should be used in the server configuration file to identified NVMe SSDs.

### Network Interface Detection & Selection

To display the supported OFI provider, use the following command:

# /scratch/standan/daos\_m/opt/ofi/bin/fi\_info -l

psm2:

version: 1.7

ofi\_rxm:

version: 1.0

ofi\_rxd:

version: 1.0

verbs:

version: 1.0

UDP:

version: 1.1

sockets:

version: 2.0

tcp:

version: 0.1

ofi\_perf\_hook:

version: 1.0

ofi\_noop\_hook:

version: 1.0

shm:

version: 1.0

ofi\_mrail:

version: 1.0

The fi\_pingpong test (delivered as part of OFI/libfabric) can be used to verify that the targeted OFI provider works fine:

node1$ fi\_pingpong -p psm2

node2$ fi\_pingpong -p psm2 ${IP\_ADDRESS\_NODE1}

bytes #sent #ack total time MB/sec usec/xfer Mxfers/sec

64 10 =10 1.2k 0.00s 21.69 2.95 0.34

256 10 =10 5k 0.00s 116.36 2.20 0.45

1k 10 =10 20k 0.00s 379.26 2.70 0.37

4k 10 =10 80k 0.00s 1077.89 3.80 0.26

64k 10 =10 1.2m 0.00s 2145.20 30.55 0.03

1m 10 =10 20m 0.00s 8867.45 118.25 0.01

Further details will be added to this section in a future revision.

## Server Configuration

This section addresses how to configure the DAOS servers on the storage nodes before starting it.

### Certificate Generation

The DAOS security framework relies on certificates to authenticate administrators. The security infrastructure is currently under development and will be delivered in DAOS v1.0.

### Server Configuration File

The daos\_server configuration file is parsed when starting the daos\_server process. The configuration file location can be specified on the command line (daos\_server -h for usage) or default location (install/etc/daos\_server.yml).

Parameters will be parsed and populated with defaults (located in config\_types.go), if not present in the configuration.

Command line parameters take precedence over configuration file values. If not specified on the command line, configuration file values will be applied (or parsed defaults).

For convenience, either active parsed config values are written to the directory where the configuration file was read from or /tmp/ if that fails.

An example of the configuration file with descriptions is provided in the GitHub source[[2]](#footnote-3).

If the user shell executing the daos\_server has environment variable CRT\_PHY\_ADDR\_STR set, the user os environment will be used instead of the configuration file. In this situation a "Warning: using os env vars..." message will be printed to the console and no environment variables will be added as specified in the env\_vars list within the per-server section of the server config file. This behavior provides backward compatibility with historic mechanism of specifying all parameters through environment variables.

The following section lists the format, options, defaults and descriptions available in the configuration file.

#### Configuration File Options

This section lists the default empty configuration listing all the options (living documentation of the config file). Live examples are available at <https://github.com/daos-stack/daos/tree/master/utils/config>

The Location of this configuration file is determined by first checking for the path specified through the -f option of the daos\_server command line. Otherwise, /etc/daos\_server.conf is used.

**Name associated with the DAOS system**

Immutable after reformat.

name: daos

**Access points**

To operate, DAOS will need a quorum of access point nodes to be available.

Immutable after reformat.

Hosts can be specified with or without port, default port below assumed if not specified.

default: hostname of this node at port 10000 for local testing

access\_points: ['hostname1:10001','hostname2:10001','hostname3:10001']

access\_points: [hostname1,hostname2,hostname3]

**Force default port**

Force different port number to bind daos\_server to, this will also be used when connecting to access points if no port is specified.

default: 10000

port: 10001

**Path to CA certificate**

If not specified, DAOS will start in insecure way which means that anybody can administrate the DAOS installation and access data.

ca\_cert: ./.daos/ca.crt

**Path to server certificate and key file**

Discarded if no CA certificate is passed.

default: ./.daos/daos\_server.{crt,key}

cert: ./.daosa/daos\_server.crt

key: ./.daosa/daos\_server.key

**Fault domain path**

Immutable after reformat.

default: /hostname for a local configuration w/o fault domain

fault\_path: /vcdu0/rack1/hostname

**Fault domain callback**

Path to executable which will return fault domain string.

Immutable after reformat.

fault\_cb: ./.daos/fd\_callback

**Use specific OFI interfaces**

Specify either a single fabric interface that will be used by all spawned servers or a comma-separated list of fabric interfaces to be assigned individually.

By default, the DAOS server will auto-detect and use all fabric interfaces if any and fall back to socket on the first eth card, otherwise.

fabric\_ifaces: [qib0,qib1]

**Use specific OFI provider**

Force a specific provider to be used by all the servers.

The default provider depends on the interfaces that will be auto-detected:

ofi+psm2 for Omni-Path, ofi+verbs;ofi\_rxm for Infiniband/RoCE and finally

ofi+socket for non-RDMA-capable Ethernet.

provider: ofi+verbs;ofi\_rxm

**Storage mount directory**

TODO: If no pre-configured mountpoins are specified, DAOS will auto-detect NVDIMMs, configure them in interleave mode, format with ext4 and mount with the DAX extension creating a subdirectory within scm\_mount\_path.

This option allows to specify a preferred path where the mountpoints will be created. Either the specified directory or its parent must be a mount point.

default: /mnt/daos

scm\_mount\_path: /mnt/daosa

**NVMe SSD whitelist**

Only use NVMe controllers with specific PCI addresses.

Immutable after reformat, colons replaced by dots in PCI identifiers.

By default, DAOS will use all the NVMe-capable SSDs that don't have active mount points.

bdev\_include: ["0000:81:00.1","0000:81:00.2","0000:81:00.3"]

**NVMe SSD blacklist**

Only use NVMe controllers with specific PCI addresses. Overrides drives listed in nvme\_include and forces auto-detection to skip those drives.

Immutable after reformat, colons replaced by dots in PCI identifiers.

bdev\_exclude: ["0000:81:00.1"]

**Use Hyperthreads**

When Hyperthreading is enabled and supported on the system, this parameter defines whether the DAOS service thread should only be bound to different physical cores (value 0) or hyperthreads (value 1).

default: false

hyperthreads: true

**Use the given directory for creating unix domain sockets**

DAOS Agent and DAOS Server both use unix domain sockets for communication with other system components. This setting is the base location to place the sockets in.

default: /var/run/daos\_server

socket\_dir: ./.daos/daos\_server

**Number of hugepages to allocate for use by NVMe SSDs**

Specifies the number (not size) of hugepages to allocate for use by NVMe through SPDK. This indicates the total number to be used by any spawned servers. Default system hugepage size will be used and hugepages will be evenly distributed between CPU nodes.

default: 1024

nr\_hugepages: 4096

**Force specific debug mask for daos\_server (control plane).**

By default, just use the default debug mask used by daos\_server.

Mask specifies minimum level of message significance to pass to logger.

Currently supported values are DEBUG and ERROR.

default: DEBUG

control\_log\_mask: ERROR

**Force specific path for daos\_server (control plane) logs.**

default: print to stderr

control\_log\_file: /tmp/daos\_control.log

When per-server definitions exist, auto-allocation of resources is not performed. Without per-server definitions, node resources will automatically be assigned to servers based on NUMA ratings, there will be a one-to-one relationship between servers and sockets.

servers:

Rank to be assigned as identifier for server.

Immutable after reformat.

Optional parameter, will be auto generated if not supplied.

rank: 0

Logical CPU assignments as identified in /proc/cpuinfo (e.g. [0-24] for CPU 0 to 24).

Immutable after reformat.

cpus: [0-20]

**Use specific OFI interfaces.**

Specify the fabric network interface that will be used by this server.

Optionally specify the fabric network interface port that will be used by this server but please only if you have a specific need, this will normally be chosen automatically.

fabric\_iface: qib0

fabric\_iface\_port: 20000

**Force specific debug mask (D\_LOG\_MASK) at start up time.**

By default, just use the default debug mask used by DAOS.

Mask specifies minimum level of message significance to pass to logger.

default: ERR

log\_mask: WARN

**Force specific path for DAOS debug logs.**

default: /tmp/daos.log

log\_file: /tmp/daos\_server1.log

**Pass specific environment variables to the DAOS server**

Empty by default.

env\_vars:

ABT\_MAX\_NUM\_XSTREAMS=100

CRT\_TIMEOUT=30

**Define a pre-configured mountpoint for storage class memory to be used by this server.**

Path should be unique to server instance (can use different subdirs).

Either the specified directory or its parent must be a mount point.

scm\_mount: /mnt/daos/1

**Backend block device type. Force a SPDK driver to be used by this server instance.**

Options are:

"nvme" for NVMe SSDs (preferred option)

"malloc" to emulate a NVMe SSD with memory

"file" to emulate a NVMe SSD with a regular file

"kdev" to use a kernel block device

Immutable after reformat.

default: nvme

bdev\_class: nvme

**Backend block device configuration to be used by this server instance.**

Immutable after reformat.

When bdev\_class is set to nvme, bdev\_list is the list of unique NVMe IDs that should be different across different server instance.

Colons replaced by dots in PCI identifiers.

bdev\_list: ["0000:81:00.0"] # generate regular nvme.conf

**Rank to be assigned as identifier for server.**

Immutable after reformat.

Optional parameter, will be auto generated if not supplied.

rank: 1

Logical CPU assignments as identified in /proc/cpuinfo (e.g. [0-24] for CPU 0 to 24).

Immutable after reformat.

cpus: [21-40]

**Use specific OFI interfaces.**

Specify the fabric network interface that will be used by this server. Optionally specify the fabric network interface port that will be used by this server but only if you have a specific need, this will normally be chosen automatically.

fabric\_iface: qib0

fabric\_iface\_port: 20000

**Force specific debug mask (D\_LOG\_MASK) at start up time.**

By default, just use the default debug mask used by DAOS. Mask specifies minimum level of message significance to pass to logger.

default: ERR

log\_mask: WARN

**Force specific path for DAOS debug logs.**

default: /tmp/daos.log

log\_file: /tmp/daos\_server2.log

**Pass specific environment variables to the DAOS server**

Empty by default.

env\_vars:

ABT\_MAX\_NUM\_XSTREAMS=200

CRT\_TIMEOUT=100

**Define a pre-configured mountpoint for storage class memory to be used by this server.**

Path should be unique to server instance (can use different subdirs).

scm\_mount: /mnt/daos/2

**Backend block device type. Force a SPDK driver to be used by this server instance.**

Options are:

"nvme" for NVMe SSDs (preferred option)

"malloc" to emulate a NVMe SSD with memory

"file" to emulate a NVMe SSD with a regular file

"kdev" to use a kernel block device

Immutable after reformat.

When bdev\_class is set to malloc, bdev\_number is the number of devices to allocate and bdev\_size is the size in GB of each LUN/device.

# bdev\_class: malloc

# bdev\_number: 1

# bdev\_size: 4

When bdev\_class is set to file, bdev\_list is the list of file paths that will be used to emulate NVMe SSDs. The size of each file is specified by

bdev\_size in GB unit.

bdev\_class: file

bdev\_list: [/tmp/daos-bdev1,/tmp/daos-bdev2]

bdev\_size: 16

When bdev\_class is set to kdev, bdev\_list is the list of unique kernel block devices that should be different across different server instance.

bdev\_class: kdev

bdev\_list: [/dev/sdc,/dev/sdd]

## Server Startup

DAOS currently relies on PMIx for server wire-up and application to server connection. As a result, the DAOS servers can only be started via orterun (part of OpenMPI). A new bootstrap procedure is under implementation and will be available for DAOS v1.0. This will remove the dependency on PMIx and will allow the DAOS servers to be started individually (e.g. independently on each storage node via systemd) or collectively (e.g. pdsh, mpirun or as a Kubernetes Pod).

### Parallel Launcher

As stated above, only orterun(1) is currently supported.

The list of storage nodes can be specified in a host file (referred to as ${hostfile}). The DAOS server and the application can be started separately but must share a URI file (referred as ${urifile}) to connect. The ${urifile} is generated by orterun using (--report-uri filename) at the server and used at the application with (--ompi-server file:filename). Also, the DAOS server must be started with the --enable-recovery option to support server failure. See the orterun(1) man page for additional options.

To start the DAOS server, run:

orterun -np <num\_servers> --hostfile ${hostfile} --enable-recovery --report-uri ${urifile} daos\_server

The --enable-recovery is required for fault tolerance to guarantee that the fault of one server does not cause the others to be stopped.

Hostfile used here is the same as the ones used by Open MPI. See the mpirun documentation[[3]](#footnote-4) for additional details.

The --allow-run-as-root option must be added to the command line to allow the daos\_server to run with root priviledged on each storage nodes.

### Systemd Integration

A preliminary systemd script to manage the DAOS server is available under utils/system. That being said, this startup method is not supported yet since the current DAOS version still relies on PMIx for DAOS server wireup.

### Kubernetes Pod

DAOS service integration with Kubernetes is planned and will be supported in a future DAOS version.

### Service Monitoring

On start-up, the daos\_server will create and initialize the following components:

* gRPC server to handle requests over client API
* dRPC server to handle requests from IO servers over the UNIX domain socket
* storage subsystems for handling interactions with NVM devices
* SPDK environment using a shared memory segment identifier causing the process to act as a primary in multi-process mode. From there, the main process can respond to requests over the client API for information through the SPDK interface.

The daos\_shell is a transitory tool used to exercise the management api and can be used to verify that the DAOS servers are up and running. It is to be run as a standard, unprivileged user as follows:

$ daos\_shell –l storagenode1:10001,storagenode2:10001 storage list

“storagenode” should be replaced with the actual hostname of each storage node. This command will show whether the DAOS server is properly running and initialized on each storage node. A more comprehensive and user-friendly tool built over the management API is under development. A first version will be available for DAOS v1.0.

## Firmware Upgrade

Firmware on an NVMe controller can be updated from an image on local storage (initially installing from a local path on the host that is running daos\_server but to be extended to downloading remotely from central storage location).

When the controller is selected and an update firmware task runs, controller data is accessed through an existing linked list through the binding fwupdate call and a raw command specifying firmware update with local image (specified by filepath) and slot identifier. The firmware update is followed by a hard reset on the controller.

## Storage Burn-in

Burn-in testing can be performed on discovered NVMe controllers. By default this involves a 15-minute slow burn-in test with a mixed read/write workload issued by fio but test duration and load strength should be user configurable. Burn-in should run in the background to allow administrators to use the control-plane for other tasks in the meantime.

The fio repo is to be built and needs to be referenced when building the SPDK fio\_plugin. The plug-in can then be run by fio to exercise the NVMe device through SPDK. Currently the output of the burn-in is displayed in the shell and control is returned to the user after completion. Future iterations may perform this as a background task.

## DAOS Formatting

Distributed formatting through the DAOS servers for both SCM and SSDs is under development and will be documented there once available.

Meanwhile, SCM should be formatted manually as an ext4 filesystem and mounted with the dax option prior to start the DAOS servers on each storage node:

# Create a /dev/pmem\* device for one NUMA node

$ ndctl create-namespace

{

  "dev":"namespace1.0",

  "mode":"fsdax",

  "map":"dev",

  "size":"2964.94 GiB (3183.58 GB)",

  "uuid":"842fc847-28e0-4bb6-8dfc-d24afdba1528",

  "raw\_uuid":"dedb4b28-dc4b-4ccd-b7d1-9bd475c91264",

  "sector\_size":512,

  "blockdev":"pmem1",

  "numa\_node":1

}

# Show regions with 0 free space after namespace created

$ ipmctl show -a -region

SocketID | ISetID | PersistentMemoryType | Capacity | FreeCapacity | HealthState

================================================================================================

0x0000 | 0xb1887f48651e2ccc | AppDirect | 3012.0 GiB | 3012.0 GiB | Healthy

0x0001 | 0xf77a7f481e352ccc | AppDirect | 3012.0 GiB | 0.0 GiB | Healthy

$ mkfs.ext4 /dev/pmem1

[…]

$ mount -o dax /dev/pmem1 /mnt/daos

If SCM is emulated with DRAM, then a tmpfs filesystem should be mounted:

$ mount –t tmpfs –o size=16G tmpfs /mnt/daos

Replace 16G with the desired tmpfs size.

## Agent Configuration

The DAOS Agent is not required in DAOS v0.4 since authentication support is not fully landed yet. Instructions on how to setup and start the agent will be provided in the next revision of this document.

## System Validation

To validate that the DAOS system is properly installed, the daos\_test suite can be executed:

orterun -np <num\_clients> --hostfile ${hostfile} --ompi-server file:${urifile} ./daos\_test

daos\_test requires at least 8GB of SCM (or DRAM with tmpfs) storage on each storage node.

1. https://github.com/intel/ipmctl [↑](#footnote-ref-2)
2. https://github.com/daos-stack/daos/tree/master/utils/config [↑](#footnote-ref-3)
3. <https://www.open-mpi.org/faq/?category=running#mpirun-hostfile> [↑](#footnote-ref-4)