2022

Persistent Memory File System

PMFS(Persistent Memory File System)

LIU CHANGPENG && YI DONG

INTEL CORPORATION | 3/3/2022

PMFS(Persistent Memory File System):

Single node userspace file system based on DAOS VOS and SPDK and relies on PMem.

PMFS Application Scenario:

It's targeted scenarios including:

- ***Integrated with upper level application like Database as the optimal File Store against kernel based POSIX FS***
- ***Support different kinds of backend block stores by SPDK like NVMe-oF target, Ceph backends and self developed backends***

PMFS DETAILS

Code files Introduction:

Changes to be committed:

modified: src/SConscript
- add pmfs to build.

new file: src/pmfs/README.md
- file for introduction for pmfs mechanism.

new file: src/pmfs/SConscript
- build file for compiling all files in pmfs.

new file: src/pmfs/include/pmfs.h
- pmfs functions and some defines that can be used for external users.

new file: src/pmfs/pmfs.c
- The realizations of pmfs APIs.

new file: src/pmfs/pmfs_internal.h
- Internal APIs that be called by pmfs APIs.

new file: src/pmfs/vos_client.c
- The realizations for client side fetch/update/punch/get_num_dkeys/list_dkeys.

new file: src/pmfs/vos_target_engine.c
- The realization of pools and containers and objects initialization and binding mechanism.

new file: src/pmfs/vos_target_engine.h
- Target engine header file for redefine the linked list persistent memory pool.

new file: src/pmfs/vos_target_fs.c
- Target file for processing tasks in spdk ring queue.
- Drains the task queues and executes the tasks in callbacks.
- It provides APIs that test APPs or external users can call:
  - ✓ vos_task_process_init
  - ✓ vos_task_process
  - ✓ vos_task_process_fini
  - ✓

new file: src/pmfs/vos_target_fs.h

- Header file for construct vos_fs_cmd_args and APIs output.

new file: src/pmfs/vos_tasks.c

- Task using spdk_ring_create and vos_target_free_tasks.

new file: src/pmfs/vos_tasks.h

- Task related structs defines :
  - ✓ VOS_OBJ_UPDATE
  - ✓ VOS_OBJ_FETCH,
  - ✓ VOS_OBJ_PUNCH,
  - ✓ VOS_OBJ_GET_NUM_DKEYS,
  - ✓ VOS_OBJ_LIST_DKEYS,

modified: src/tests/SConscript

- Add tests APPs to build.

new file: src/tests/pmfs_demo.c

- Demo app for demonstrating mkfs/mount/createdir/…

new file: src/tests/pmfs_unittest.c

- Unittest that can be added to CI.

new file: src/include/pmfs/pmfs.h

- PMFS functions and some defines that can be used for external users.

new file: src/include/pmfs/vos_target_engine.h

- Copy header file for test apps.

new file: src/include/pmfs/vos_target_fs.h

- Copy header file for test apps.

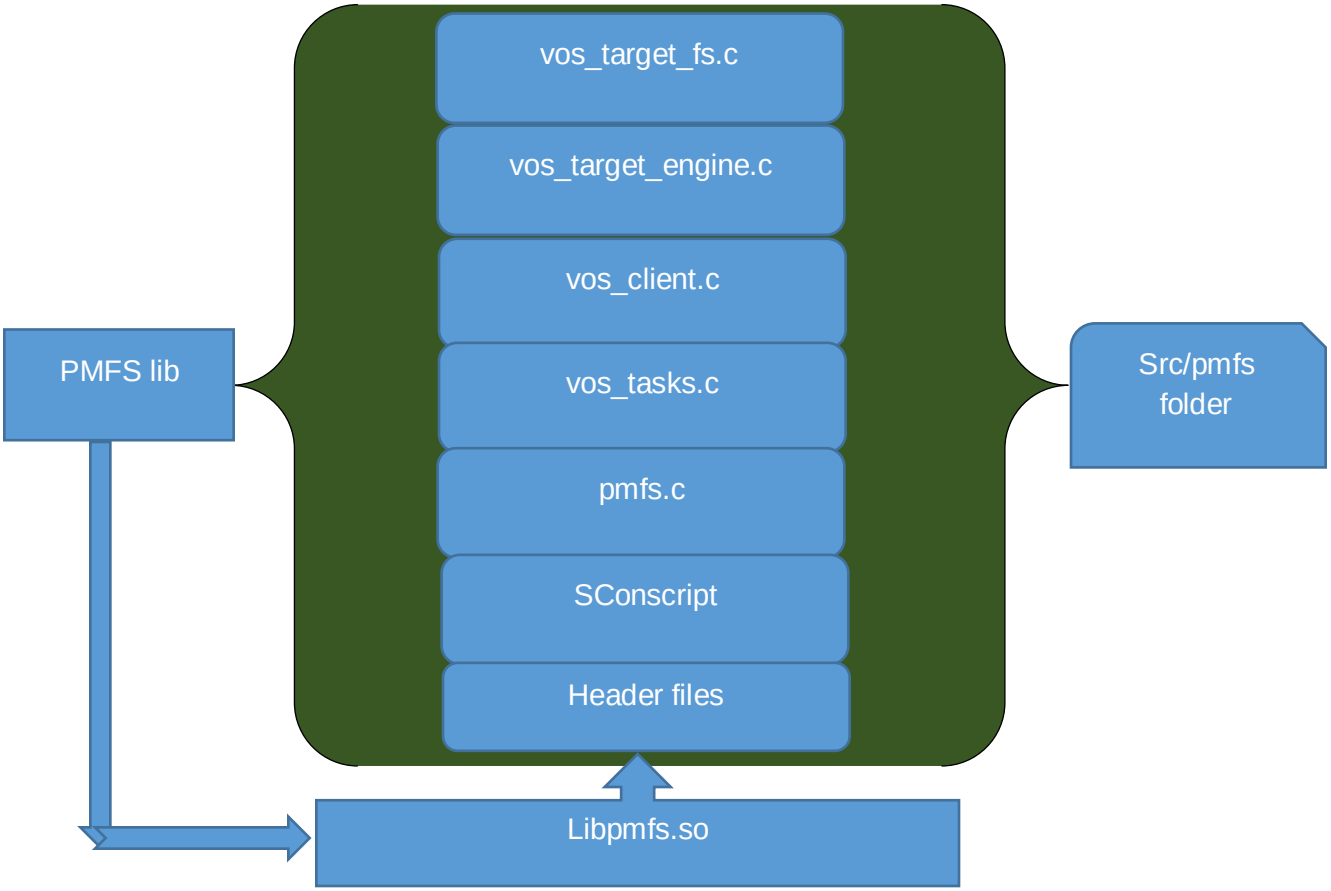
new file: src/include/pmfs/vos_tasks.h

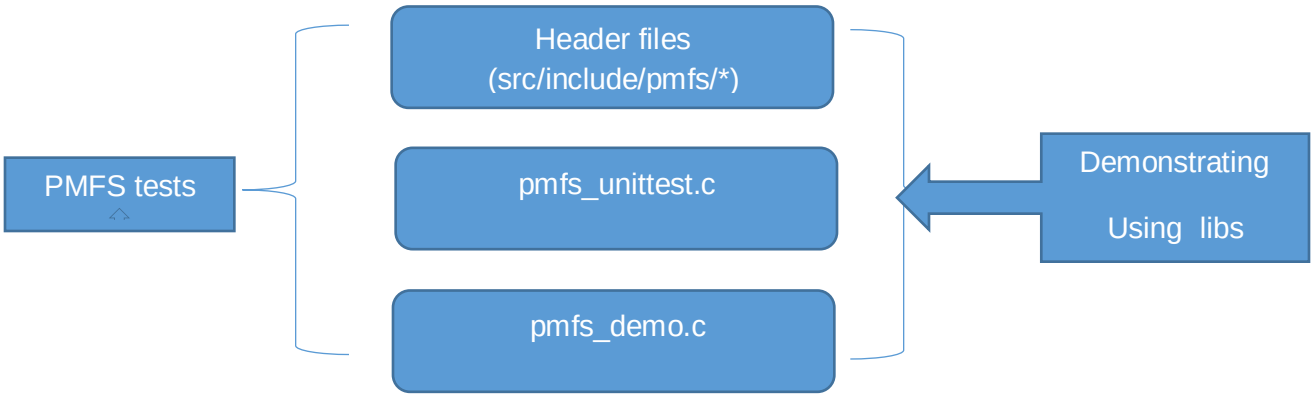
- Copy header file for test apps.

modified: utils/run_test.sh

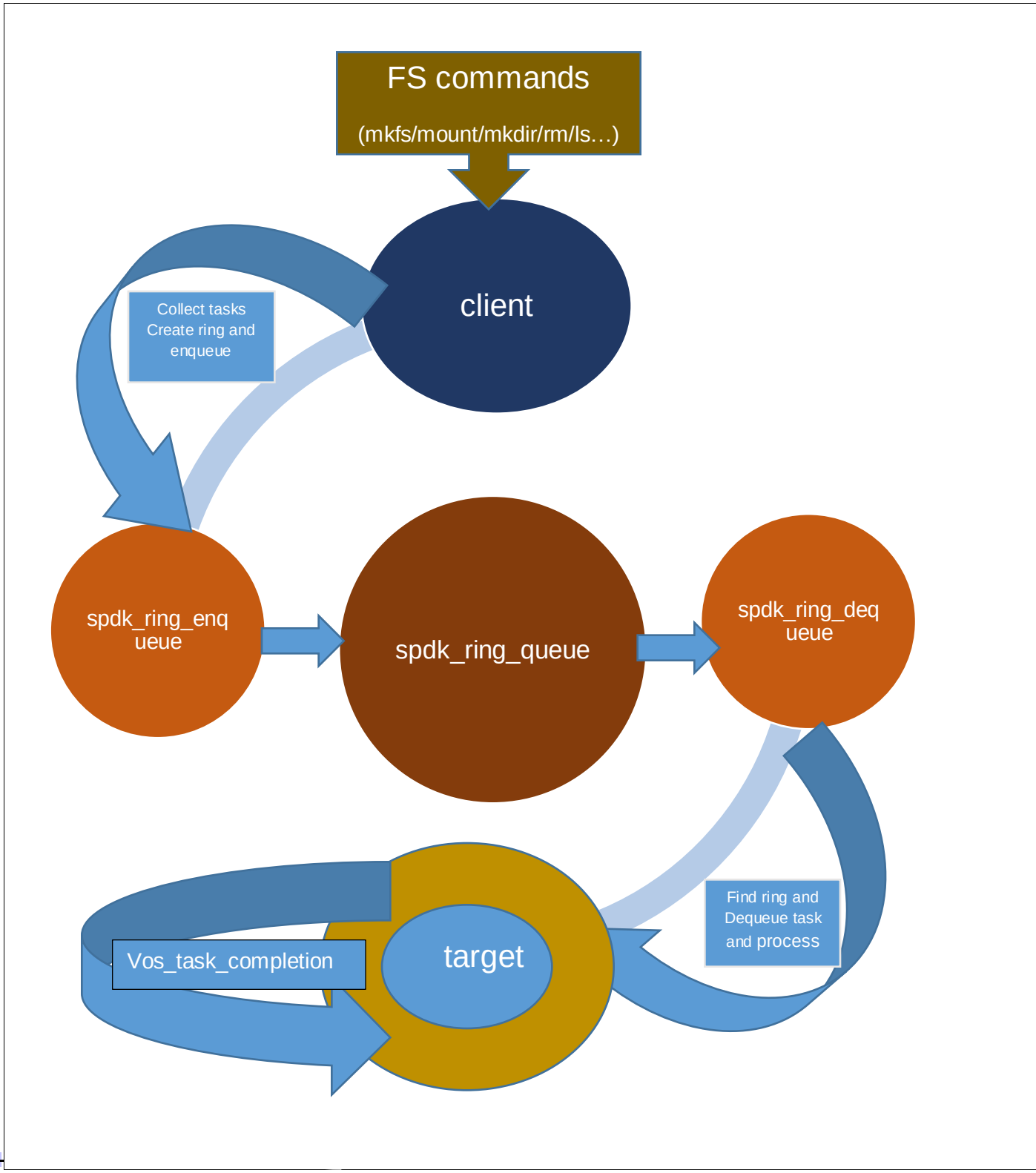
- Add pmfs_unittest to CI script.

Files relationships



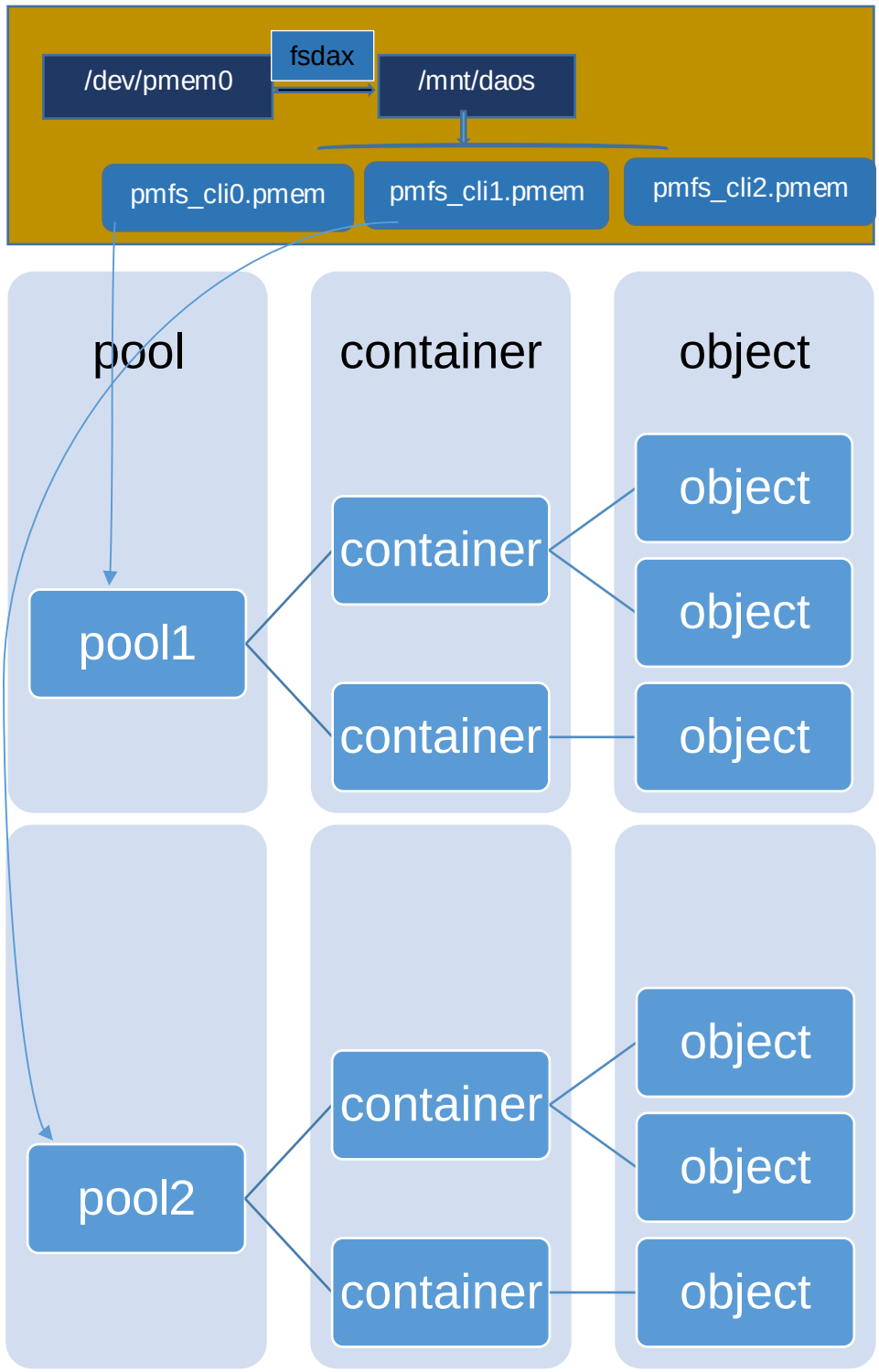


PMFS threading model



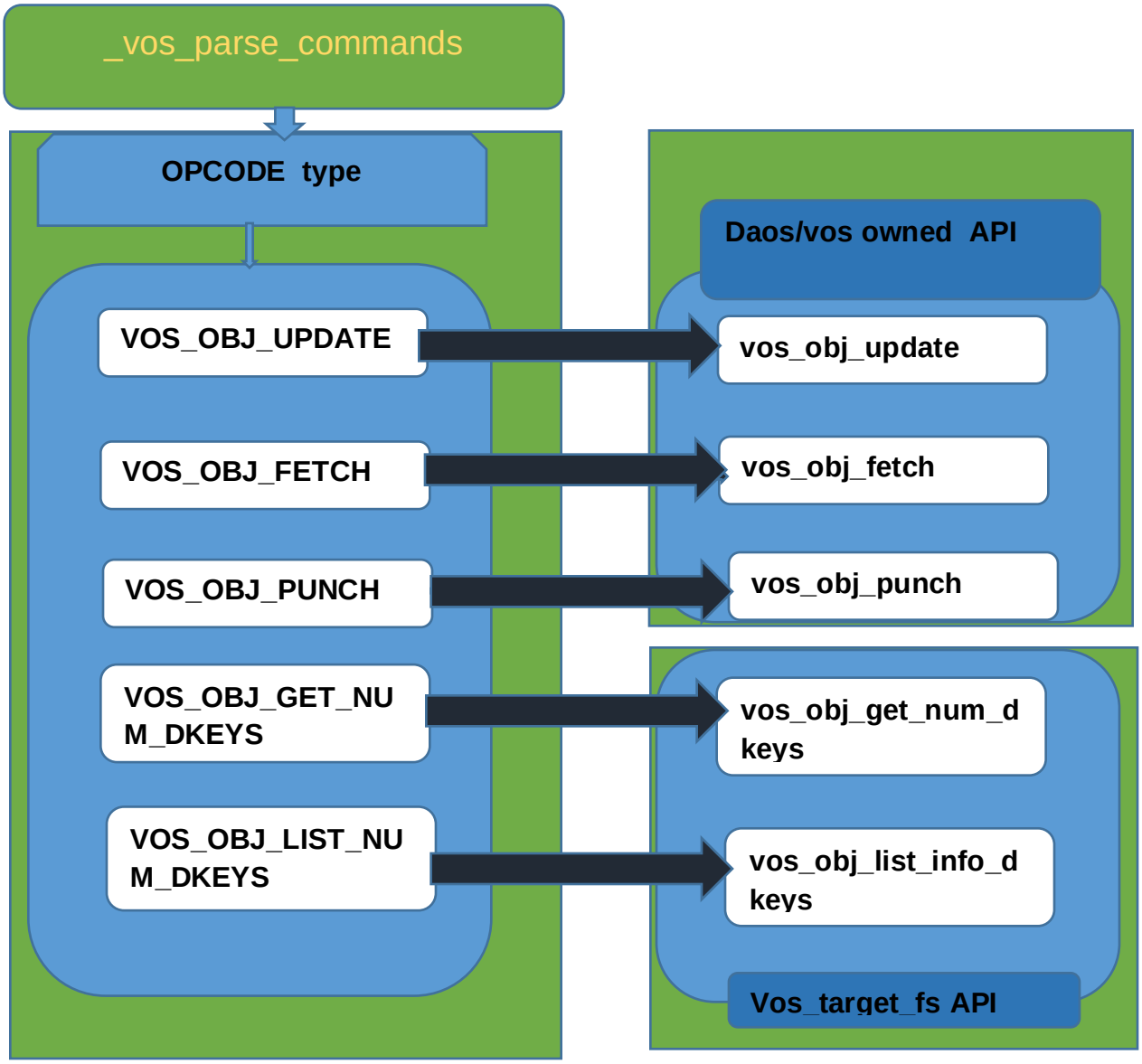
Storage hierarchy

- **Persistent memory pool**
- **Container**
- **Objects**



VOS target processes
FS command calls

- vos_task_process_init
- vos_task_process
  - ✓ while(1){ vos_task_completion}
    - vos_task_dequeue
      - vos_task_ult
        - ❖ _vos_parse_commands
- vos_task_process_fini



PMFS vos APIs

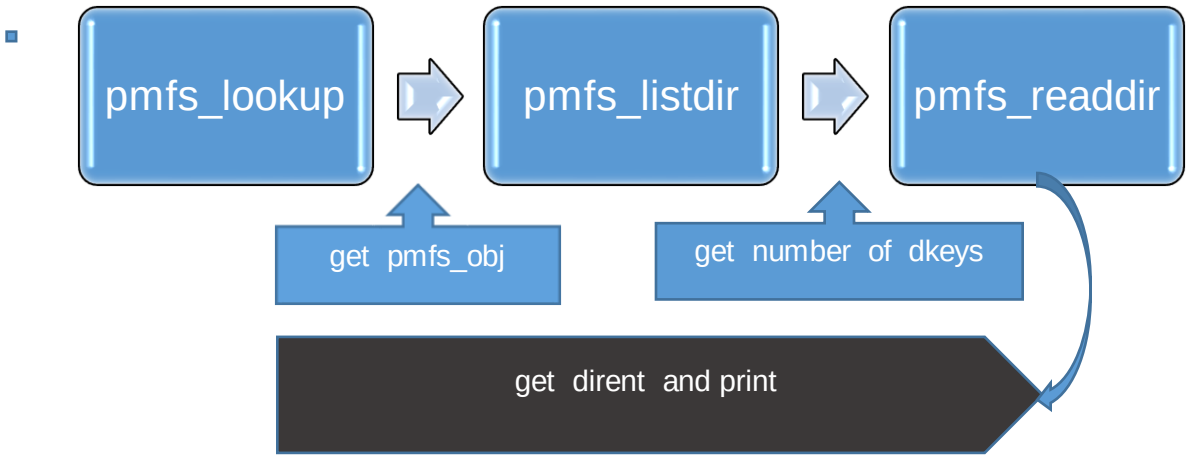
- *int pmfs_mkfs(daos_handle_t poh, uuid_t uuid);*
- *int pmfs_mount(daos_handle_t poh, daos_handle_t coh, int flags, pmfs_t **pmfs);*
- *int pmfs_umount(pmfs_t *pmfs);*
- *int pmfs_mkdir(pmfs_t *pmfs, pmfs_obj_t *parent, const char *name,*
  mode_t mode);
- *int pmfs_listdir(pmfs_t *pmfs, pmfs_obj_t *obj, uint32_t *nr);*
- *int pmfs_remove(pmfs_t *pmfs, pmfs_obj_t *parent, const char *name, bool force,*
- *daos_obj_id_t *oid);*
- *int pmfs_open(pmfs_t *pmfs, pmfs_obj_t *parent, const char *name, mode_t mode,*
- *int flags, daos_size_t chunk_size, const char *value,*
- *pmfs_obj_t **_obj);*
- *int pmfs_readdir(pmfs_t *pmfs, pmfs_obj_t *obj, uint32_t *nr, struct dirent *dirs);*
- *int pmfs_release(pmfs_obj_t *obj);*
- *int pmfs_lookup(pmfs_t *pmfs, const char *path, int flags, pmfs_obj_t **_obj,*
- *mode_t *mode, struct stat *stbuf);*
- *int pmfs_punch(pmfs_t *pmfs, pmfs_obj_t *obj, daos_off_t offset, daos_size_t len);*
- *int pmfs_write_sync(pmfs_t *pmfs, pmfs_obj_t *obj, d_sg_list_t *usr_sgl, daos_off_t off);*
- *int pmfs_read_sync(pmfs_t *pmfs, pmfs_obj_t *obj, d_sg_list_t *usr_sgl, daos_off_t off,*
  *daos_size_t *read_size);*
- *int pmfs_stat(pmfs_t *pmfs, pmfs_obj_t *parent, const char *name, struct stat *stbuf);*

VOS client APIs

- *vos_client_obj_update*
- *vos_client_obj_update_sync*
- *vos_client_obj_fetch*
- *vos_client_obj_fetch_sync*
- *vos_client_obj_punch*
- *vos_client_obj_punch_sync*
- *vos_client_obj_get_num_dkys*
- *vos_client_obj_get_num_dkys_sync*
- *vos_client_obj_list_dkys*
- *vos_client_obj_list_dkys_sync*

Example

"ls" command:



- Code flow path:
  - Client : *pmfs_lookup->fetch_entry->vos_client_obj_fetch_sync->client_task_enqueue ->spdk_ring_enqueue*
  - Target : *vos_task_completion->vos_task_dequeue->vos_task_get_ring->spdk_ring_d equeue->vos_task_ult->_vos_fs_execute_command->_vos_parse_command s->vos_obj_fetch*

Demo APP

- Setup env script:

```
#!/usr/bin/env bash
umount -l /dev/pmem0
ndctl destroy-namespace namespace0.0 --force
ndctl create-namespace --size 120G --mode fsdax --map mem -
e namespace0.0 -f
ndctl list
mkfs.xfs -f -m reflink=0 /dev/pmem0
mkdir /mnt/daos
mount -o dax /dev/pmem0 /mnt/daos
```

- config file /etc/daos_nvme.conf:

```
{
 "daos_data": {
    "config": []
 },
 "subsystems": [
    {
      "subsystem": "bdev",
      "config": [
        {
          "method": "bdev_nvme_set_options",
          "params": {
            "action_on_timeout": "none",
            "timeout_us": 0,
            "retry_count": 4,
            "arbitration_burst": 0,
            "low_priority_weight": 0,
            "medium_priority_weight": 0,
            "high_priority_weight": 0,
            "nvme_adminq_poll_period_us": 10000,
            "keep_alive_timeout_ms": 10000,
```

```
            "nvme_ioq_poll_period_us": 0,
            "io_queue_requests": 0,
            "delay_cmd_submit": true
          }
        },
        {
          "params": {
            "name": "Nvme0",
            "trtype": "PCIe",
            "traddr": "0000:68:00.0"
          },
          "method": "bdev_nvme_attach_controller"
        }
      ]
    }
  ]
}
```

./install/bin/pmfs_demo