TUTORIALS

Tutorial on the DAOS API - Exercises

Mohamad Chaarawi, HPE (mohamad.chaarawi@hpe.com)

SC25 ST. LOUIS hpc ignites.

Running on NEXTGenIO system

- DAOS practicals are in the github
- DAOS is installed on the NEXTGenIO compute nodes but not the login node
  - This means compilation and running needs to be done on the compute nodes
- We provide a batch script to compile all the examples in this practical
  - `sbatch run_compile.sh`
- We provide a batch script to run all the compiled examples in this practical
  - `sbatch run_examples.sh`
- Results will be provided in the `slurm-XXXXXX.out` file

Creating a Container

- kv1.c
  - Program to insert some keys in KV object
  - Run with ./a.out pool_name
- Will fail with:

```
container open failed: DER_NONEXIST (-1005) aborting
```

- Modify to create the container “cont1”
- Answer in kv1_answer.c

- `int daos_cont_create_with_label (daos_handle_t poh, const char *label, daos_prop_t *cont_prop, uuid_t *uuid, daos_event_t *ev);`
- `int daos_cont_destroy(daos_handle_t poh, const char *cont, int force, daos_event_t *ev);`

Recap Program Flow

```
[illegible]
```

Reading from KV

- kv2.c
  - Program to insert and read some keys with variable size buffer values.
  - Run with ./a.out pool_name
- Will only insert the keys.
- Modify to query the size of the key values and get each value.
- Answer in kv2_answer.c

KV put/get example

```
/** init, connect, cont_open */

oid.hi = 0;
oid.lo = 1;
daos_obj_generate_oid(coh, &oid, DAOS_OF_KV_FLAT, 0, 0, 0);
daos_kv_open(coh, oid, DAOS_OO_RW, &kv, NULL);


/** set val buffer and size */
daos_kv_put(kv, DAOS_TX_NONE, 0, "key1", val_len1, val_buf1, NULL);
daos_kv_put(kv, DAOS_TX_NONE, 0, "key2", val_len2, val_buf2, NULL);

/** to fetch, can query the size first if not known */
daos_kv_get(kv, DAOS_TX_NONE, 0, "key1", &size, NULL, NULL);
get_buf = malloc (size);
daos_kv_get(kv, DAOS_TX_NONE, 0, "key2", &size, get_buf, NULL);
daos_kv_close(kv, NULL);

/** free buffer, cont_close, disconnect, finalize */
```

Enumerating Keys from KV

- kv3.c
  - Program to insert and enumerate 20 keys.
  - Run with ./a.out pool_name
- Will only insert the keys
- Modify to enumerate all keys in list_key()
- Answer in kv3_answer.c

KV list example

```
/** enumerate keys in the KV */
daos_anchor_t   anchor = {0};
d_sg_list_t     sgl;
d_iov_t         sg_iov;

/** size of buffer to hold as many keys in memory */
buf = malloc(ENUM_DESC_BUF);
d_iov_set(&sg_iov, buf, ENUM_DESC_BUF);
sgl.sg_nr                = 1;
sgl.sg_nr_out            = 0;
sgl.sg_iovs              = &sg_iov;
```

```
daos_key_desc_t kds[ENUM_DESC_NR];

while (!daos_anchor_is_eof(&anchor)) {
  /** how many keys to attempt to fetch in one call */
  uint32_t       nr = ENUM_DESC_NR;


  memset(buf, 0, ENUM_DESC_BUF);
  daos_kv_list(kv, DAOS_TX_NONE, &nr, kds, &sgl,
               &anchor, NULL);

  if (nr == 0)
    continue;
  /** buf now container nr keys */
  /* kds arrays has length of each key */
}
```

Accessing Array of Integers

- array1.c
  - Program to create an integer array and write/read using the Array API.
  - Run with ./a.out pool_name
- Will fail as array is not created
- Modify and add the TODO items
  - Create the array
  - Set the Array IOD
  - Write the populated buffer of 100 integers
  - Read those integers
- Answer in array1_answer.c

DAOS Array example

```
/** create array - if array exists just open it */
daos_array_create(coh, oid, DAOS_TX_NONE, 1, 1048576, &array, NULL);


daos_array_iod_t iod;
d_sg_list_t     sgl;
daos_range_t    rg;
d_iov_t         iov;


/** set array location */
iod.arr_nr = 1; /** number of ranges / array iovec */
rg.rg_len = BUFLEN; /** length */
rg.rg_idx = rank * BUFLEN; /** offset */
iod.arr_rgs = &rg;


/** set memory location, each rank writing BUFLEN */
sgl.sg_nr = 1;
d_iov_set(&iov, buf, BUFLEN);
sgl.sg_iovs = &iov;


daos_array_write(array, DAOS_TX_NONE, &iod, &sgl, NULL);
daos_array_read(array, DAOS_TX_NONE, &iod, &sgl, NULL);
daos_array_close(array, NULL);
```

Multi-Level KV with Array and Single Value

- mkv1.c
  - Program to create a multi-level KV with 1 dkey that has 2 akeys:
    - 1 SV akey
    - 1 Array value of 100 integers
  - Run with ./a.out pool_name
- Modify and add the TODO items
  - Update 1 dkey with 2 akeys: set IODs for the update operation
  - Fetch each akey individually: set IOD for each update operation
- Answer in mkv1_answer.c

DAOS Object Update Example

```
daos_obj_open(coh, oid, DAOS_OO_RW, &oh, NULL);
d_iov_set(&dkey, "dkey1", strlen("dkey1"));

d_iov_set(&sg_iov, buf, BUFLEN);
sgl[0].sg_nr = 1;
sgl[0].sg_iovs = &sg_iov;
sgl[1].sg_nr = 1;
sgl[1].sg_iovs = &sg_iov;

d_iov_set(&iod[0].iod_name, "akey1", strlen("akey1"));
d_iov_set(&iod[1].iod_name, "akey2", strlen("akey2"));

iod[0].iod_nr = 1;
iod[0].iod_size = BUFLEN;
iod[0].iod_recxs = NULL;
iod[0].iod_type = DAOS_IOD_SINGLE;

iod[1].iod_nr = 1;
iod[1].iod_size = 1;
recx.rx_nr = BUFLEN;
recx.rx_idx = 0;
iod[1].iod_recxs = &recx;
iod[1].iod_type = DAOS_IOD_ARRAY;

daos_obj_update(oh, DAOS_TX_NONE, 0, &dkey, 2, &iod, &sgl, NULL);
```

DAOS Object Fetch Example

```c
daos_obj_open(coh, oid, DAOS_OO_RW, &oh, NULL);
d_iov_set(&dkey, "dkey1", strlen("dkey1"));

d_iov_set(&sg_iov, buf, BUFLEN);
sgl[0].sg_nr = 1;
sgl[0].sg_iovs = &sg_iov;
sgl[1].sg_nr = 1;
sgl[1].sg_iovs = &sg_iov;

d_iov_set(&iod[0].iod_name, "akey1", strlen("akey1"));
d_iov_set(&iod[1].iod_name, "akey2", strlen("akey2"));

iod[0].iod_nr = 1;
iod[0].iod_size = BUFLEN; /** if size is not known, use DAOS_REC_ANY and NULL sgl */
iod[0].iod_recxs = NULL;
iod[0].iod_type = DAOS_IOD_SINGLE;

iod[1].iod_nr = 1;
iod[1].iod_size = 1; /** if size is not known, use DAOS_REC_ANY and NULL sgl */
recx.rx_nr = BUFLEN;
recx.rx_idx = 0;
iod[1].iod_recxs = &recx;
iod[1].iod_type = DAOS_IOD_ARRAY;

daos_obj_fetch(oh, DAOS_TX_NONE, 0, &dkey, 2, &iod, &sgl, NULL, NULL);
```

```
git clone https://github.com/adrianjhpc/ObjectStoreTutorial.git
srun -N 1 --nvram-options=1LM:1000 --pty /bin/bash
cd Exercises/DAOS/
./build.sh
./kv1_answer pool01
./kv2_answer pool01
./kv3_answer pool01
./array1_answer pool01
./mkv1_answer pool01
```