RPC transport/Mercury discussion

- NUMA affinity & scaling
- Abort in-flight RPC/bulk
- Flow control – limiting #inflight RPCs
- RPC rate/throughput
- RPC in-place packing/unpacking
- Group & collective

Xuezhao Liu @DAOS team, HPDD, Intel
July 2016

NUMA affinity & scaling

- Expectation
  - Concurrent threads share accessing the same NIC
    - Affinity kept for executing, memory accessing, reply/request; without lock contention
- Current DAOS implementation
  - Each server thread creates one dtp_context
    - One dtp_context maps to separate na_class/na_context/hg_cass/hg_context (separate listening addresses)
    - Each thread regularly progresses its context separately until being canceled
    - Registers RPC to every hg_class; reply received by the request sending thread
  - Client process creates one global dtp_context
    - Connects to different server context to send RPC
    - Progress context when user calls daos_eq_poll
- Possible enhancement in future?
  - Bridge OFI's tag matching to user (user can specify the tag to send RPC)?
  - Can use single na_class (listening address) but with multiple contexts over it?
    - Remove/minimize the contention for concurrent progress on different context
    - Avoid busy polling

Abort in-flight RPC/bulk

- User tracks RPC request, when timeout:
  - Cancel the original RPC request (hg_handle)
    - Completion callback be triggered with (cb_info->ret == HG_CANCELED)
  - If the RPC involves bulk operation, then need to destroy bulk handle (HG_Bulk_free)
- Bulk transferring can be canceled (HG_Bulk_cancel)
- A few problem/question
  - Can reuse the original hg_handle to resend the request or must create a new one?
  - Now problem occurs when server-side bulk transferring in-progress race with client-side bulk handle destroying

```
HG: Error in /root/mercury_github/src/mercury_bulk.c:546
    # hg_bulk_transfer_cb(): Error in NA callback
HG: Error in /root/mercury_github/src/mercury_bulk.c:546
    # hg_bulk_transfer_cb(): Error in NA callback
*** Error in `./dtp_echo_srv': double free or corruption (out): 0x00007f97c017d720 ***
*** Error in `./dtp_echo_srv': malloc(): memory corruption: 0x00007f97c011d930 ***
```

Flow-control – limiting #inflight RPCs

- Problem
  - CCI limits max in-flight msg sent per connection (for example VERBS_CONN_TX_CNT for verbs)
  - Mercury does not impose any limit for that
  - If DAOS issue too many async IOs, it will meet:

```
NA: Error in /scratch/DAOS_OPT/src/mercury_github/src/na/na_cci.c:959
   # na_cci_msg_send_unexpected(): cci_sendv() failed with CCI_EAGAIN
HG: Error in /scratch/DAOS_OPT/src/mercury_github/src/mercury_core.c:1332
   # hg_core_forward_na(): Could not post send for input buffer
HG: Error in /scratch/DAOS_OPT/src/mercury_github/src/mercury_core.c:2854
   # HG_Core_forward(): Could not forward buffer
```

  - Now DAOS limits test case's number of async IO as a workaround

- Solution
  - DAOS internally limits the #inflight RPCs, but it does not know low layer such as CCI's limitation
  - Could be handled by mercury and transparent to upper layer?

RPC rate/throughput

- DAOS IOR bandwidth basically can saturate network (1 server, 1 client, 8 async IO)
- DAOS KV update/fetch about 8.1K ops per second (1 server, 1 client, 8 async IO)
  - One operation is one 64B RPC + 64 B bulk transferring
- Mercury benchmark (over cci verbs)
  - RPC noop round-trip average 6.1uS
  - RPC+Bulk_read round-trip (uS) (the 2nd number in the first 3 rows numbers is with checksum disabled, others are enabled)

Length	Server-side						Client-side			
	Get input		Bulk create	Bulk transferring		Respond	Request forwarding		**RPC + Bulk read round-trip**	
	eager	RDMA		eager	RDMA		eager	RDMA	eager	RDMA
1	4.4 4.3	4.3	37.5	2.8 2.7	8.1	1.8 1.6	1.5 1.0	1.5	64.5 56.9	64.6
1k	41.1 25.8	4.2	37.6	2.8	7.9	1.8	13.6 8.8	1.5	109.5 91.8	65.1
2k	76.1 47.3	4.2	37.3	2.8	8.3	1.7	25.4 16.5	1.5	155.8 120.7	65.2
512k		4.4	69.3		140.7	1.8		1.5		231.1
1M		4.3	96.3		281.8	1.8		1.5		390.8

RPC inplace packing/unpacking

- DAOS RPC packing/unpacking
  - Similar method to use hg_proc_xxx to pack/unpack request/reply
  - Use structure description of msg fields (no boost macro used)

```
struct dtp_msg_field *dmg_pool_destroy_in_fields[] = {
        &DMF_UUID,                              /* pd_pool_uuid */
        &DMF_STRING,            /* pd_grp */
        &DMF_INT                /* pd_force */
};
struct dtp_msg_field DMF_UUID = DEFINE_DTP_MSG("dtp_uuid", 0, sizeof(uuid_t),
                                                dtp_proc_uuid_t)
```

  - DAOS RPC request/reply need to be compactly packed (DAOS takes care of the possible padding)

- A step ahead …
  - Can the memory copy be avoided for hg_proc_xxx?
  - DAOS packs its request by itself and hand over the raw buffer pointer to low layer to send out
  - At server-side, DAOS gets a raw buffer pointer and parses it by itself
  - Similar as lustre …

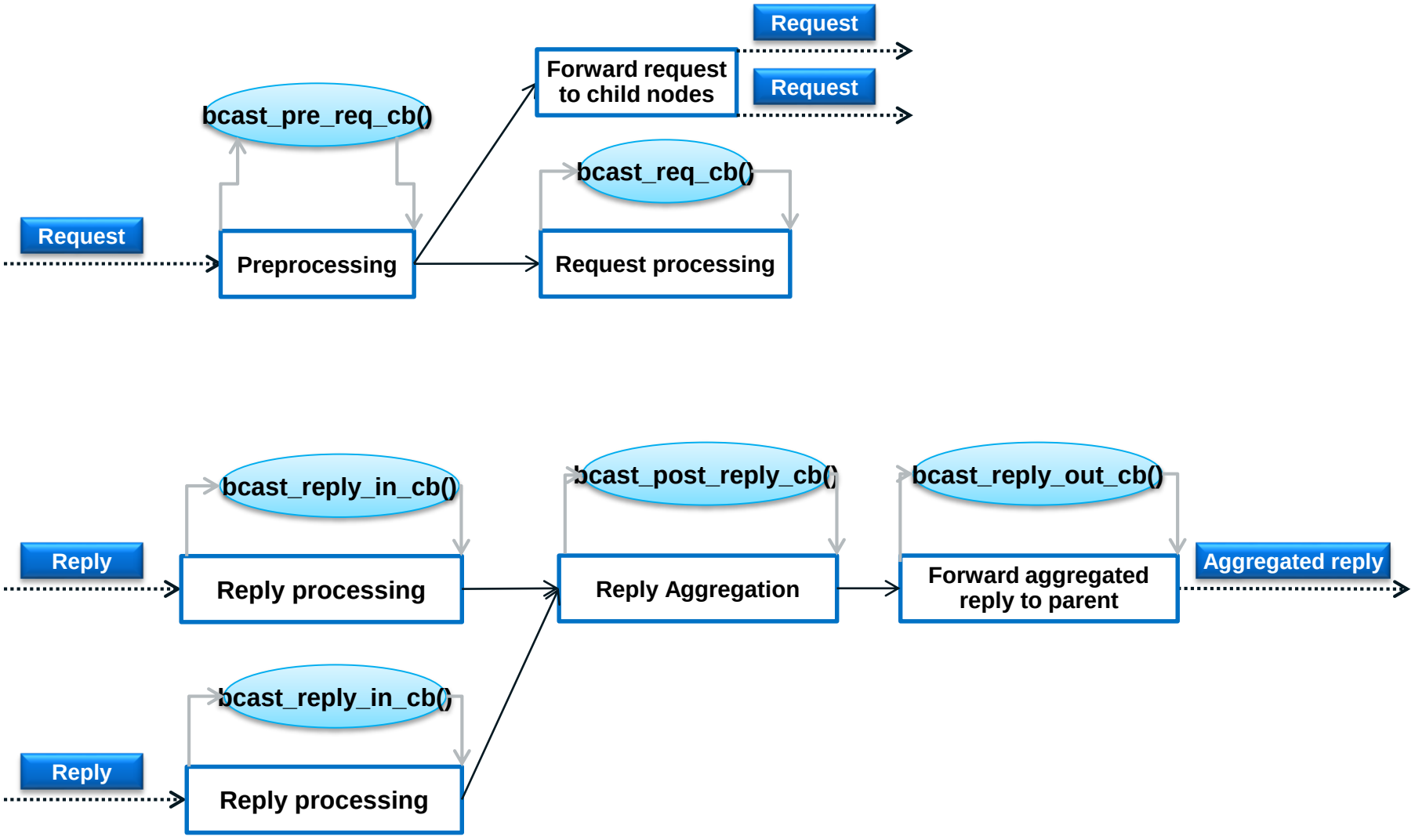
Group & collective

- Use cases
  - Collective commit, pool create, DAOS capability distribution, client eviction ...

- Collective completely implemented over mercury
  - Upper layer RPCs share a common RPC handler and request completion callback, to forward request and aggregate reply.
  - Request/reply repeat unpacked/packed during the request/reply propagating

- Collective implemented inside mercury
  - Adding group concept to mercury, a na_group layer?
    - Group membership and ranking can be maintained by upper layer
  - Mercury RPC broadcast APIs

Service callbacks for broadcast



Server collective extension to mercury