OpenSHMEM Nonblocking Collectives

OpenSHMEM Spec Meeting
December 2nd, 2020

Nonblocking Collective Operation

Nonblocking collective operation is

- Same as blocking collective operation
  - Group communication operation semantics
  - Executed on all PEs of the team
- Varies in invocation, progress, and completion
  - Invocation: Upon invocation of a collective routine interface,the operation is posted and returned immediately
  - Status and completion: Learn status and completion using a different call

Need for Nonblocking Collectives

- Provide ability to overlap computation and collectives
- Invoke multiple collective operations on the same team
- Leverage offload hardware more effectively
- Support traditional use cases and emerging use cases (AI/DL, DOE workloads)

Design Goals

- Design compatible with OpenSHMEM teams and infrastructure
- Enable RDMA-based collective implementations
- Flexible ordering model
  - Support both ordered and unordered collectives
- Flexible synchronous model
  - Support synchronous and non-synchronous collectives
- Support OpenSHMEM thread model

API Design Choices

Design choice 1: A single interface for all collectives

Strawman API

shmem_collective_init(shmem_team_t team, shmem_coll_args_t collective_args, shmem_req_t req);

shmem_collective_post(shmem_req_t req);

shmem_collective_wait(shmem_req_t req);

shmem_collective_finalize(shmem_req_t req);

Collective Arguments

```
struct shmem_coll_op_args_t {
    shmem_coll_type_t           coll_type;
    shmem_coll_buffer_info_t    buffer_info;
    shmem_reduction_info_t      reduction_info;
    ucc_coll_id_t               tag;
    uint32_t                    root_pe;
};
```

API Design Choices

Design choice 2: Separate interface for each of the collective

Strawman API

*int shmem_TYPENAME_alltoall(shmem_team_t team, TYPE *dest, const TYPE *source, size_t nelems, shmem_req_t req);*

int shmem_collective_wait(shmem_req_t req);

int shmem_collective_test(shmem_req_t req);

WG Preference : Design #2 PR is available

9.10.1 SHMEM_BROADCAST_NB

Broadcasts a block of data from one PE to one or more destination PEs.

SYNOPSIS

C11:

```
int shmem_broadcast_nb(shmem_team_t team, TYPE *dest, const TYPE
*source, size_t nelems, int PE_root, uint32_t tag, shmem_req_h *request);
```

where *TYPE* is one of the standard RMA types specified by Table 5.

C/C++:

```
int shmem_TYPENAME_broadcast_nb(shmem_team_t team, TYPE
*dest, const TYPE *source, size_t nelems, int PE_root, uint32_t tag,
shmem_req_h *request);
```

where *TYPE* is one of the standard RMA types and has a corresponding *TYPENAME* specified by Table 5.

```
int shmem_broadcastmem_nb(shmem_team_t team, void *dest, const void
*source, size_t nelems, int PE_root, uint32_t tag, shmem_req_h *request);
```

https://github.com/openshmem-org/specification/pull/456

Design Goals : Open Questions

- What collectives to support ?
  - Allreduce, Alltoall, and Broadcast ? or
  - Should we have non-blocking variants for all blocking variants ?
  - Should we support nonblocking barrier?
- Persistent collectives ?
- Should we support in-place?
- Synchronous and asynchronous collectives

Asynchronous and Synchronous Collectives

- SYNC_ON_BOTH: Synchronization on both entry and exit
  - On entry, the processes/threads cannot read/write to other processes without ensuring all have entered the collective
  - On exit, the processes/threads may exit after all processes/threads have completed the reading/writing.
- NO_SYNC: No synchronization on entry or exit

Thanks!