OpenSHMEM NonBlocking Collectives

OpenSHMEM WG

July 23rd, 2020

Need for Nonblocking Collectives

- Provide ability to overlap computation and collectives
- 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 implementations
- Support for nonblocking and split phase collectives
- Flexible ordering model
  - Support both ordered and unordered collectives
- Flexible synchronous model
  - Support synchronous and non-synchronous collectives

Design Goals : Open Questions

- Thread model
  - Should we support concurrent collectives on a same team (in SHMEM_THREAD_MULTIPLE)?
- Buffer ownership
  - Should we support in-place ?
  - When should we transfer the ownership from User / Library and vice-versa
- What collectives to support ?
  - Allreduce, Alltoall, Barrier and Broadcast ? or
  - Should we have non-blocking variants for all blocking variants ?

Interface to Customize Teams

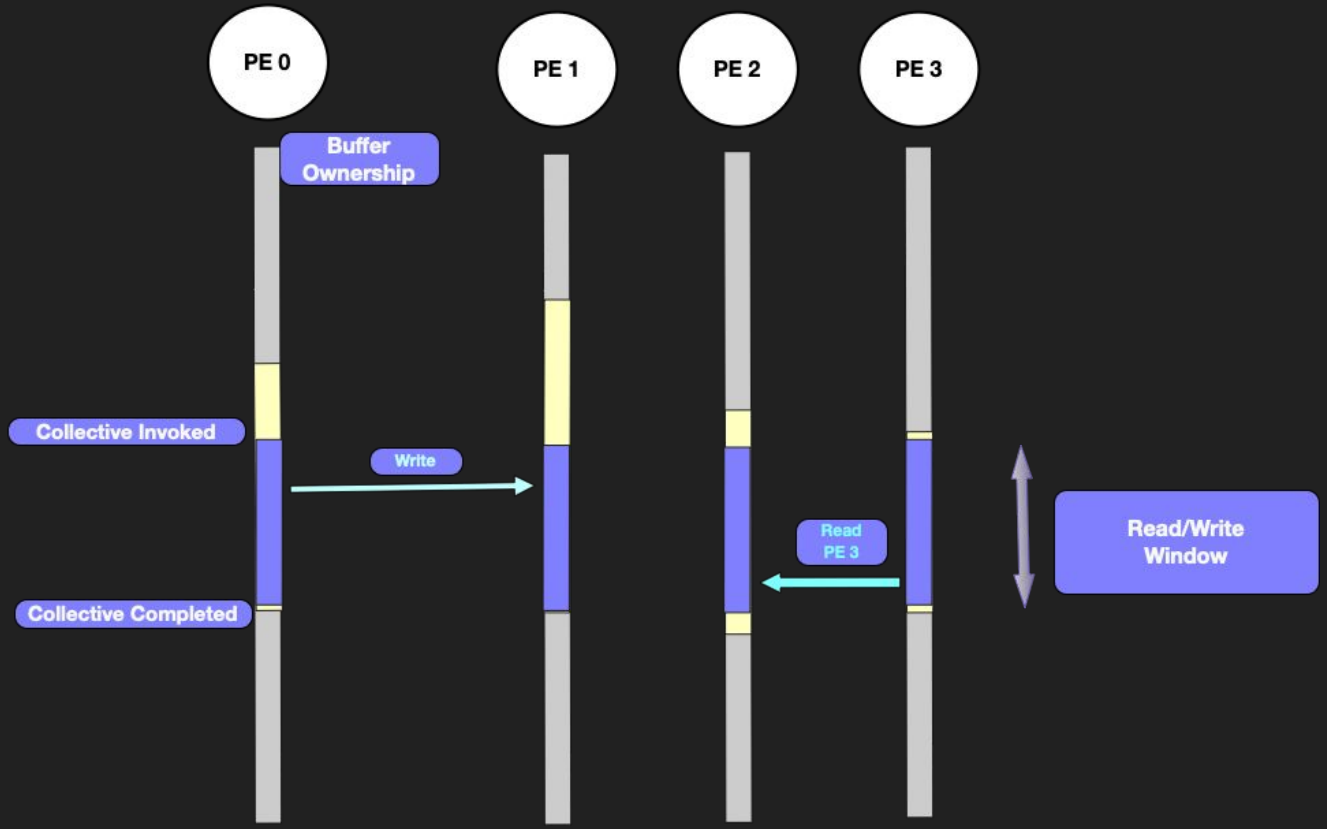
Customize team to capture the design options

```
struct shmem_team_config_t {
   int                  num_contexts;
   shmem_sync_type_t    sync_type; /* SYNC / NO-SYNC */
   bool                 ordered_type; /* ORDERED/UNORDERED */
   uint32_t             num_outstanding; /* Hint */
};
```

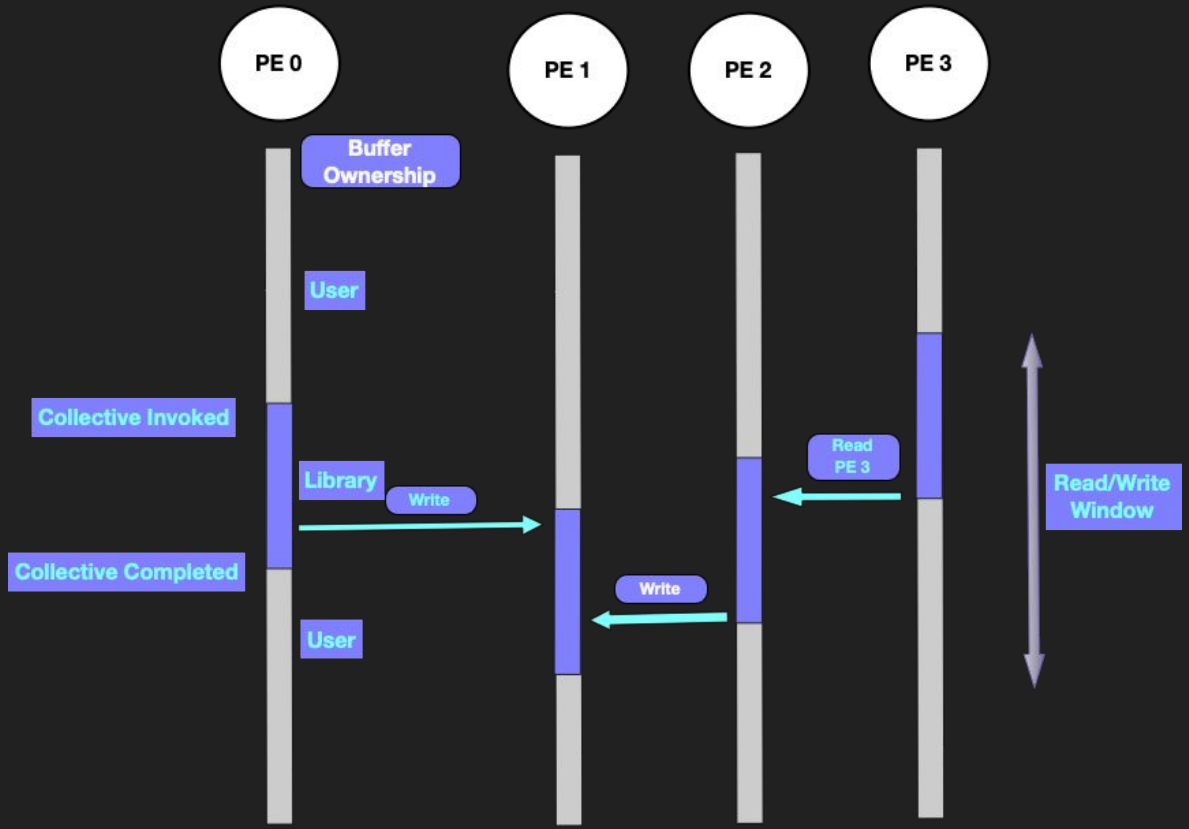
Asynchronous and Synchronous Collectives

- 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

Synchronized Collective



No-Sync Collective



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);