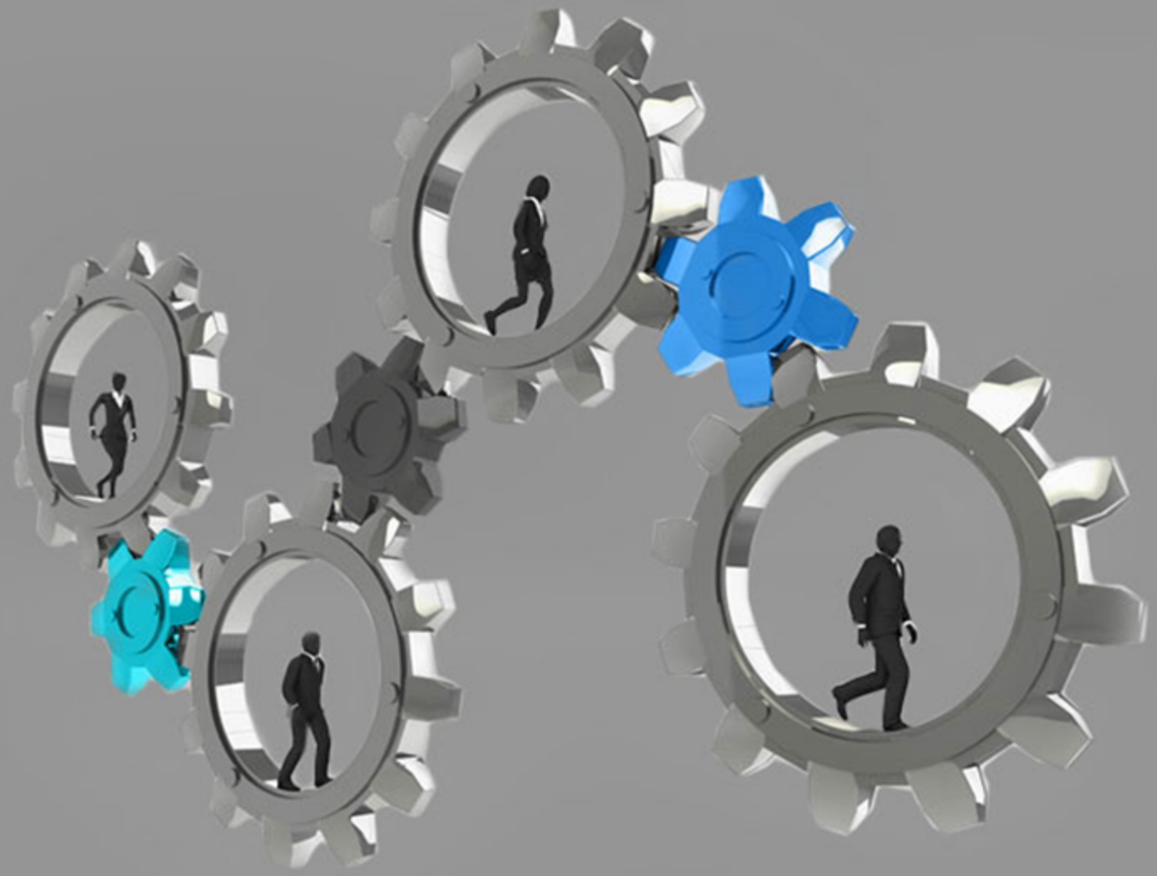
ENABLER OF CO-DESIGN

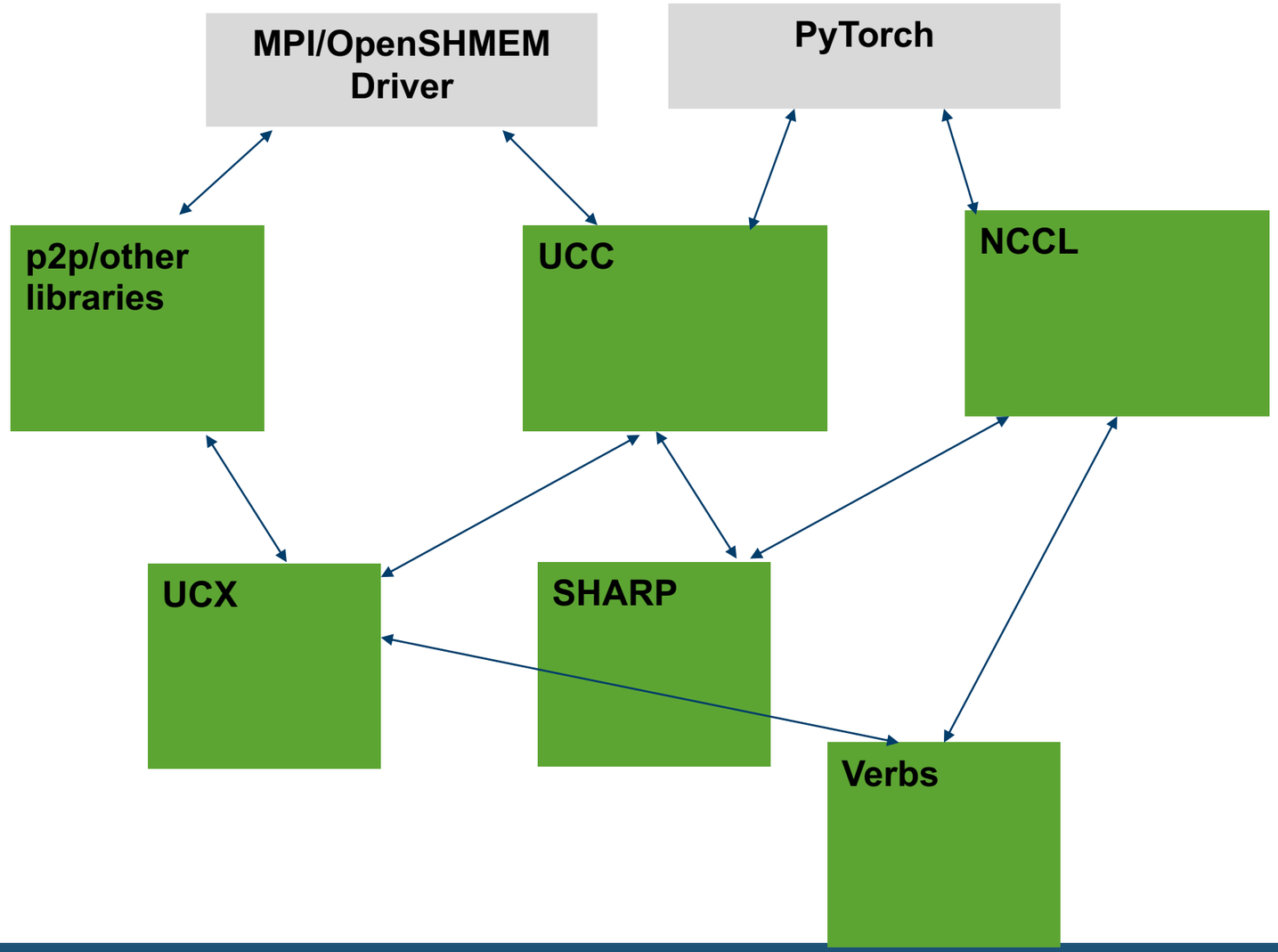
UCF



Resource Sharing between UCC and other libraries

Manjunath Gorentla Venkata

Resource sharing between UCC and other libraries (UCX, SHARP, or Shared Memory)



UCC Resources

- Library, Context, Team, Endpoints, Memory
- Stateless objects
  - Library, memory, and endpoints
- Objects with state
  - Context and team
  - Depends on the implementation but very likely all are with state

Sharing resources between libraries

- Information required to share between the libraries
  - It is easy - share the pointer !
    - Caution : should interpret the structure and semantics in the same way; tight coupling of structures
- Creating objects
  - All resources that needs to be shared are created in a shared mode; libraries are aware that the objects are shared
  - Objects should be created in a threaded mode
- Atomicity
  - Multiple threads sharing the same resources; need to lock around critical structures
    - Stateless objects – less state to lock
    - Objects with state – all state is locked before changed/updated
  - All resources should be created in a multi-threaded mode (?)
- Destroy/Fatal error should be conveyed other libraries
  - The control thread which manages the resources should inform another library
- What is the granularity of sharing ?
  - Fine grained or coarse grained
  - Depends on the use case – UCX (context, worker, eps), SHARP (trees, communicator), Shared Memory (Memory chunk/ whole memory), Verbs (QPs/PD)

Let’s try a solution: Lazy resource association

UCF Unified Communication Framework

- High-level
  - All objects that needs lazy resource association will be created in that mode
    - All upper dependencies objects are in created in this mode as well
    - For example, If UCC context is created in shared mode, then library object should be created in shared mode.
  - Library creates association between the resources and objects explicitly
    - ucc_associate_resources(library_object, object_type, params, shared_resource)
  - Library deletes/ends the association between the objects explicitly
    - ucc_disassociate_resources(shared_resource)
  - Introduce new error codes for disassociated resources

  - Atomicity
    - Responsibility of the implementations
      - Ensure that the critical structures are protected
      - Ensure the resources are shareable
  - Co-ordination protocol between the libraries
    - Responsibility of the user (MPI or PyTorch Driver)
      - Co-ordinate associate and dissociate resources

Strawman API

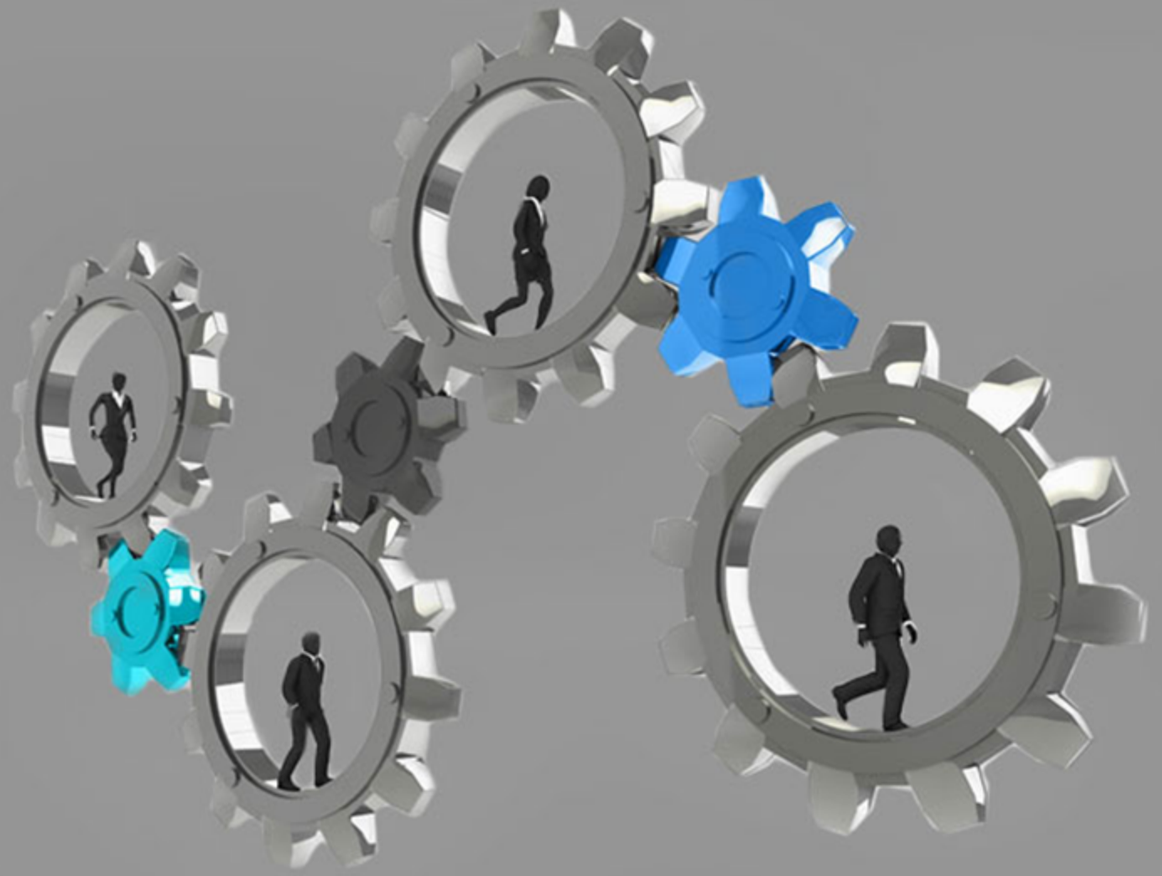
```
/**
 *  @ingroup UCC_LIB
 *
 *  @brief The @ref ucc_associate_resources provides a lazy and dynamic way
 *  to attach resources to the UCC objects.
 *
 *
 *  @param [in]       lib_p               Input library object
 *  @param [in]       params              Parameters to attach the resources
 *  @param [in,out]   ucc_resource_obj    Resource object to be modified
 *
 *  @parblock
 *
 *  @b Description
 *
 *  A local resource association operation for the UCC objects. The parameters
 *  specify the UCC objects to be associated with the resources. The resources are provided as
 *  a key/value/length triplet parameters. The values are user and implementation defined.
 *  The keys are defined has a pattern UCC_RESOURCE_OBJ_*_*. The first wild card specifies
 *  the object such as library, context, and team. The second wild card is user and implementation
 *  specific identifer.  The user is responsible for allocating the memory for keys and values.
 *
 *  The resource association operation can be called on the object, which is of correct type .i.
e.,
 *  UCC_RESOURCE_TYPE_ASSOCIATED.
 *  On success, the resources are associated with the object.
 *
 *  @endparblock
 *
 *  @return Error code as defined by ucc_status_t
 */

ucc_status_t ucc_associate_resources(ucc_lib_h lib_p, ucc_resource_params_t
                                     *params, void *ucc_resource_obj);
```

Gaps

- Sharing is limited to libraries with the same version.
- The API doesn’t support export of resources created by UCC
  - Should we support it now ? Use case ?
- Sharing is limited to a single process
  - Inter-process sharing ?
  - Need to figure out how to share the information about structures – shared memory or IPC
- Needs support from other libraries (UCX, SHARP) for safe execution
  - UCX for example could support something like this:
    - ucp_import_worker(ucp_context, ucp_worker) /* Like shared PD */
    - ucp_unimport_worker(ucp_worker)

ENABLER OF CO-DESIGN

UCF



Thank You

The UCF Consortium is a collaboration between industry, laboratories, and academia to create production grade communication frameworks and open standards for data centric and high-performance applications.