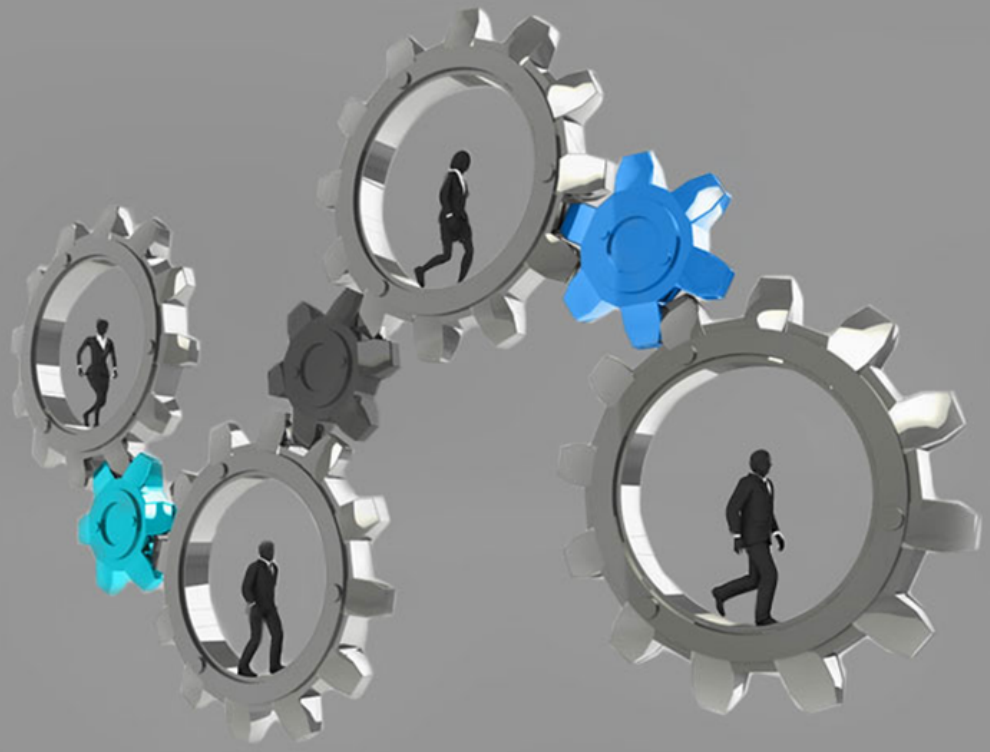
ENABLER OF CO-DESIGN



UCC Internal Abstractions

Schedules and Tasks

Manjunath Gorentla Venkata, UCF Collectives WG,

June 24th, 2020

UCC WG (June 24th)

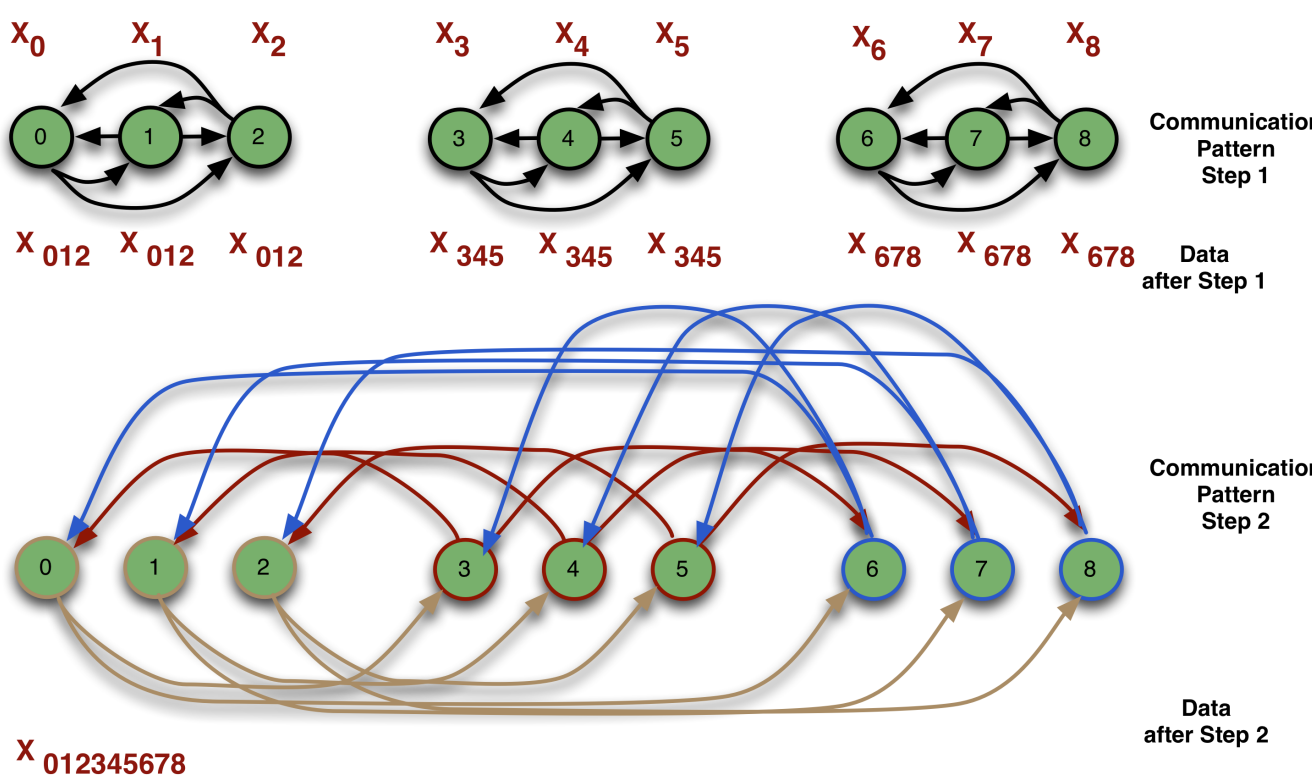
- Goals:
  - Common abstractions to express various collective implementation approaches
    - Examples : Hierarchical, Reactive or hybrid (hierarchical + reactive)
  - Flesh out the details of the abstraction

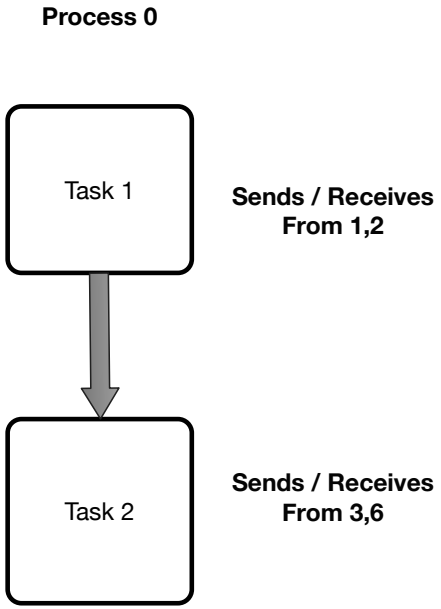
How to express the collective ?

- Abstractions
  - Schedule:
    - Expresses complete collective operation (Allreduce, Reduce, Alltoall) for a single process/rank/thread
    - Includes tasks that needs to be executed for a collective operation for a single process/rank/thread
  - Tasks:
    - A set of communication operations abstracting a single step in the algorithm
    - A task includes
      - A set of send and receive operations to complete a step (in single hierarchy / flat hierarchy implementation)
      - A collective operation (in hierarchical based implementation)
    - Let's discuss with an example, what is a schedule and tasks
  - Agree on the main abstractions required
  - Names can be changed

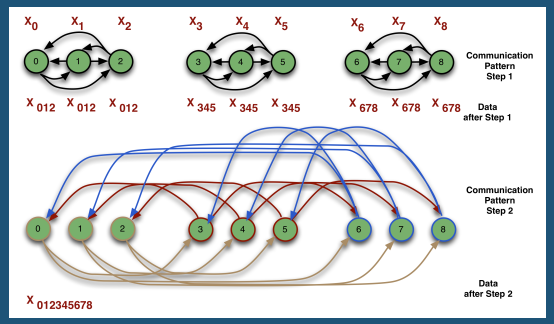
Example: Allreduce using Recursing K-ing pattern (No hierarchy)

- Intentionally choosing a complex pattern

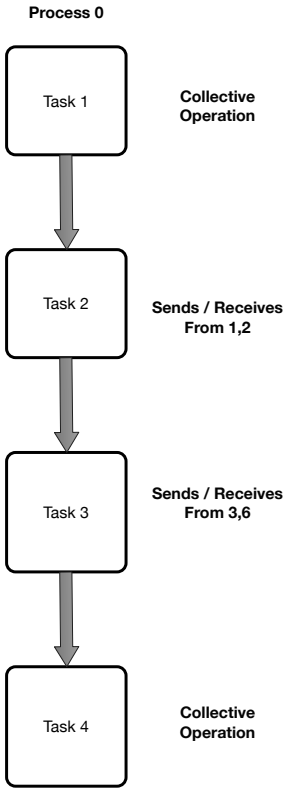




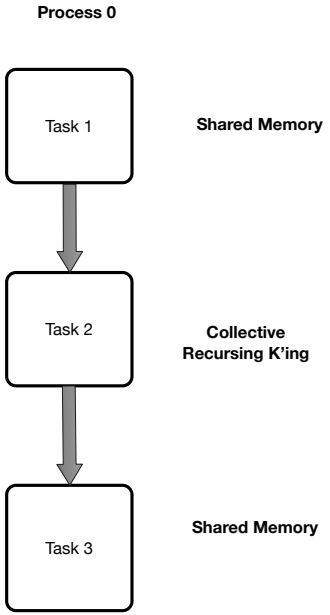
Extend Schedules and Tasks to Hierarchies



Collective operation (SHARP / Shared Memory)

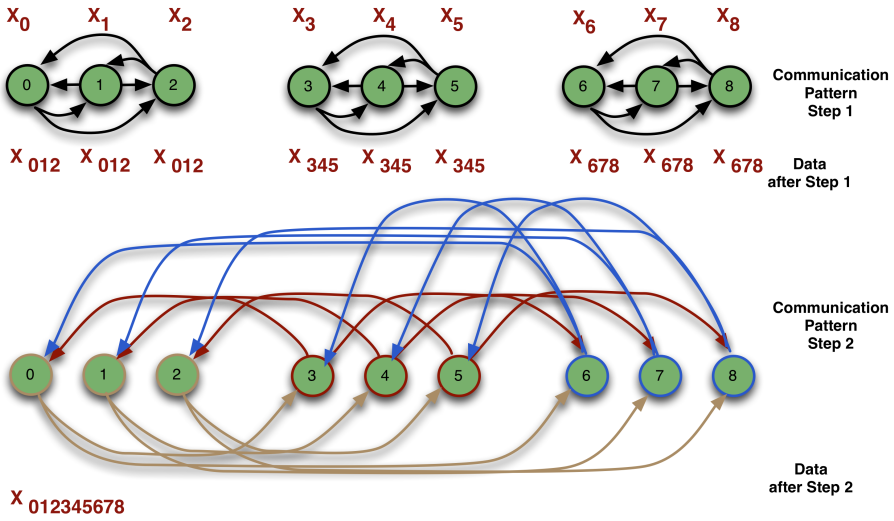


Approach 1



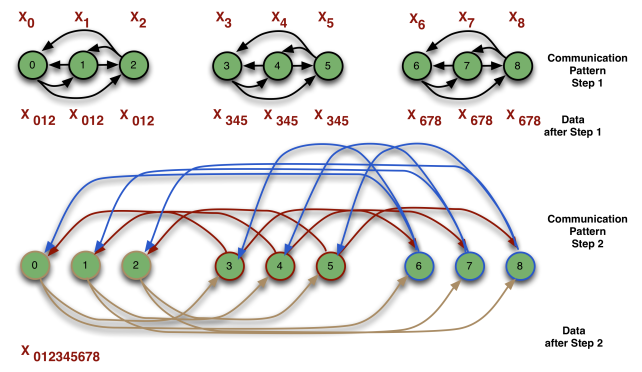
Approach 2

Example: Allreduce using Recursing K-ing pattern (No hierarchy)



- Schedule
  - Number of steps (1 and 2) in the algorithm
  - User provided info
    - Team information
    - Collective Input / Output buffers
    - Operations
  - Fragmentation information
  - What is missing ?
- Task
  - Number of sends and receives at each step (2 sends and 2 receives)
  - Communication info
    - Buffer information for send and receive
  - Network transport information
  - Preprocessing function
  - Postprocessing function
  - How to find the next task to trigger and how to trigger?
- Other optimizations
  - How to differentiate between static/dynamic information ?
- Book keeping information not captured

Example: Allreduce using Recursing K-ing pattern (With hierarchy)



Collective operation (SHARP / Shared Memory)

- Schedule
  - Number of steps (1 and 2) in the algorithm
  - User provided info
    - Team information
    - Collective Input / Output buffers
    - Operations
  - Fragmentation information
  - Relationship between the steps
  - What is missing ?
- Task
  - Number of sends and receives at each step (2 sends and 2 receives)
  - Communication info
    - Buffer information for send and receive
    - Collective primitive information
  - Network transport information
  - Preprocessing function
  - Postprocessing function
  - How to find the next task to trigger and how to trigger?
- Other optimizations
- Book keeping information not captured

Summarize : Differences between two approaches

- Schedule
  - Relationship between the steps
    - Hierarchical
      - May need to express different ordering (sequential, no-ordering)
    - Reactive
      - Might not need this information as the ordering is captured by the callbacks

- Tasks
  - How to trigger the next task ?
    - Hierarchical
      - Progress – Progresses the sends and receives
      - Trigger next task – by the progress function
    - Reactive
      - Progress – Progresses the sends and receives
      - Trigger next task – by a callback
  - Need to express collective, p2p, or both
    - Hierarchical – p2p, collective
    - Reactive – p2p

Data Structures and APIs

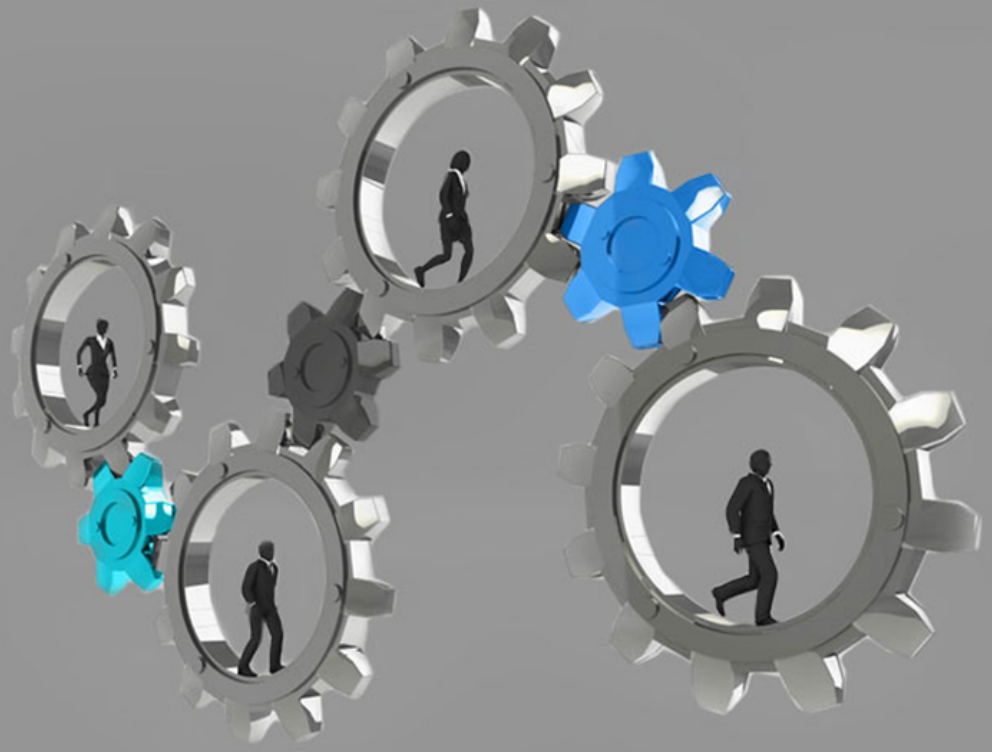
- Schedule
  - Number of steps
  - User info (Teams, collective operations)
  - Tasks (Array, list)
- Task
  - Triggering info
    - Completion callback
    - Threshold info (Number of receives, completion of sends, and next task)
    - ???
  - Communication info
    - P2p info (Buffer info)
    - Collective info
      - Collective function, Progress function (?)
  - Fragmentation information
  - Network information
  - Preprocessing function
  - Postprocessing function

APIs

- Create and Destroy (Schedule, Task)
- Progress Tasks

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.