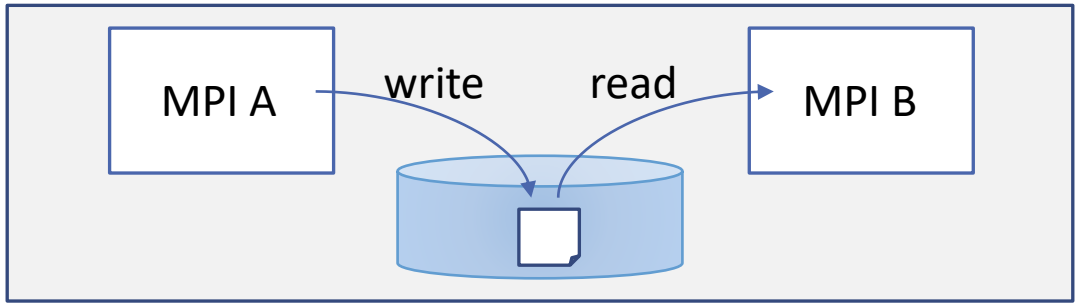
CoToCoA: A Communication-Efficient Framework for Coupling Programs

Takeshi Nanri (Kyushu U.), Yuto Katoh (Tohoku U.), Keiichiro Fukazawa (Kyoto U.), Yohei Miyake (Kobe U.) and Kazuya Nakazawa (Kobe U.)

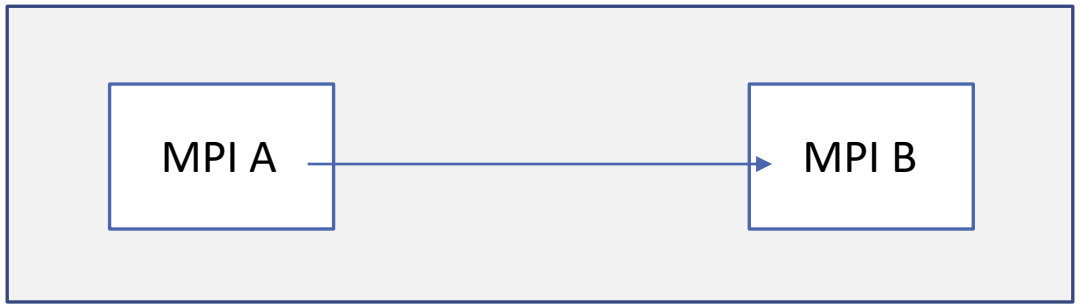
poster at HPC Asia 2022, 12-14 Jan, 2022

Motivation

Instead of "via files", connect two MPI programs by communication.



Connection via a file



Connection by communication

Design goals of CoToCoA (Code To Code Adapter)

1. Minimal modification to existing MPI programs
2. Asynchronous execution
3. Low overhead of data transfer

Overview of CoToCoA

Connection via a coupler program that controls connection.

- Goal 1: Minimize modification to existing MPI programs.

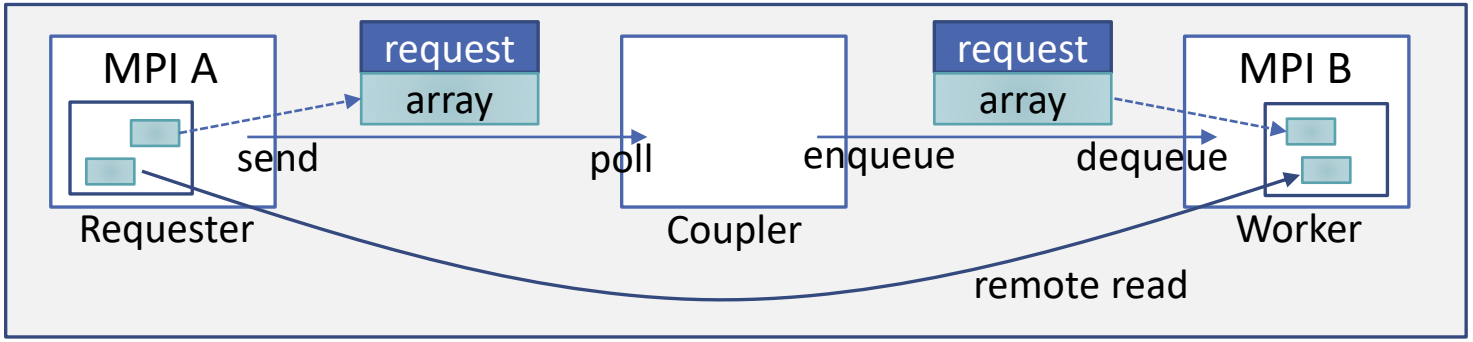
Requester-Worker model.

- Goal 2: Asynchronous execution.

Two styles of data-transfer:

1. Attach an array to a request
2. Remote read from the requester

- Goal 3: Low overhead of data transfer.



Sample code of a requester

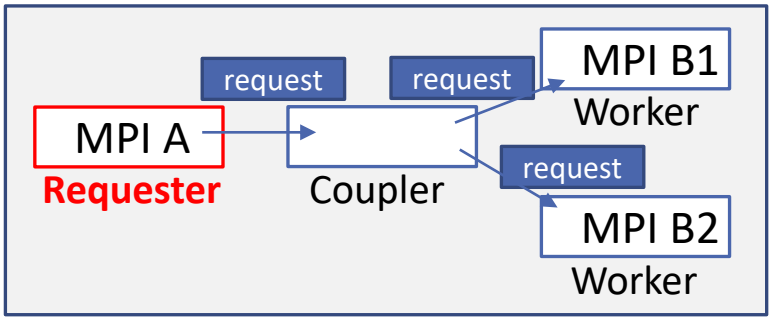
```
CTCAR_init();
MPI_Comm_size(CTCA_subcomm, &nprocs);
MPI_Comm_rank(CTCA_subcomm, &myrank);

for ( ... ) {

 calculation

  if (myrank == 0)
    CTCAR_sendreq_withreal8( ... );
}

CTCAR_finalize();
```



Modify existing MPI program:

- Replace subroutines
  - MPI_Init to CTCAR_init
  - MPI_Finalize to CTCAR_finalize
- Replace MPI_COMM_WORLD to "CTCA_subcomm"
- Add "sendreq" to send a request to the coupler
  - It will be transferred to a worker.
  - An array can be attached to a request.

Sample code of a worker

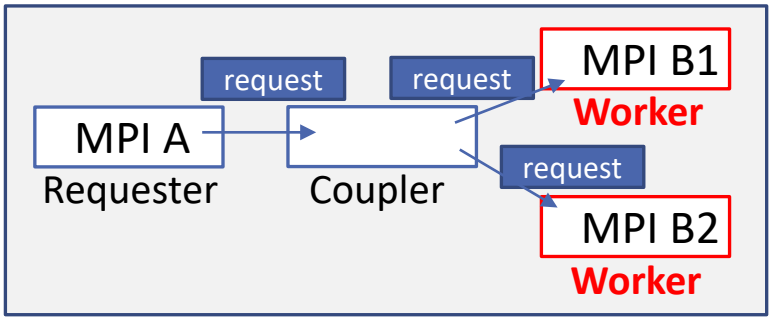
```
CTCAW_init(0, 4);
  // Worker Program #0, 4procs / group
MPI_Comm_size(CTCA_subcomm, &nprocs);
MPI_Comm_rank(CTCA_subcomm, &myrank);

while(1) {
  CTCAW_pollreq_withreal8( ... );
  if (CTCAW_isfin())
    break;

   calculation

  CTCAW_complete();
}

CTCAW_finalize();
```



Modify existing MPI program:

- Replace subroutines and MPI_COMM_WORLD like the requester.
- Repeat until the finalization signal arrives.
  - Poll for a request.
  - Handle the request.
  - Notify "completion" to the coupler.

Additional facility:

- More than one MPI programs can be workers.
  - Choices are done at the coupler.
- Requests to a worker program can be distributed to multiple process groups.

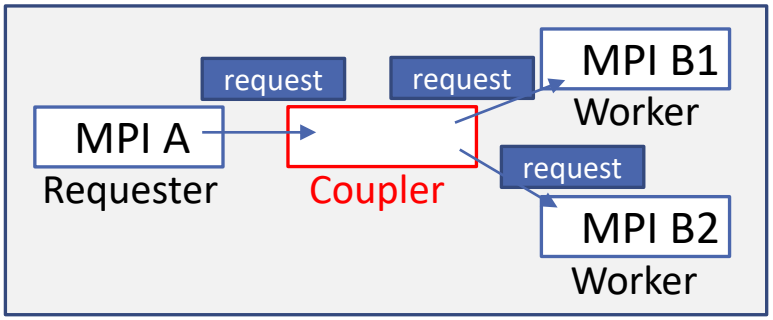
Sample code of a coupler

```
CTCAC_init();
MPI_Comm_size(CTCA_subcomm, &nprocs);
MPI_Comm_rank(CTCA_subcomm, &myrank);

while (1) {
  CTCAC_pollreq_withreal8( ... );
  if (CTCAC_isfin())
    break;


  CTCAC_enqreq_withreal8( ... );
}

CTCAC_finalize();
```



Create a new one:

- Repeat until the finalization signal arrives.
  - Poll for a request.
  - Enqueue the request to an appropriate worker.

Possible roles of the coupler:

- Convert the attached array.
- Choose the worker program according to the parameters of the request.
- Balance loads among process groups.

Remote read from the requester

```
CTCAR_init();

CTCAR_regarea_real8(data, size, &areaid);

...
```

```
CTCAW_init();

CTCAW_regarea_real8(&areaid);

...

  CTCAW_readarea_real8(areaid, rank, size,
                       offset, data);
```

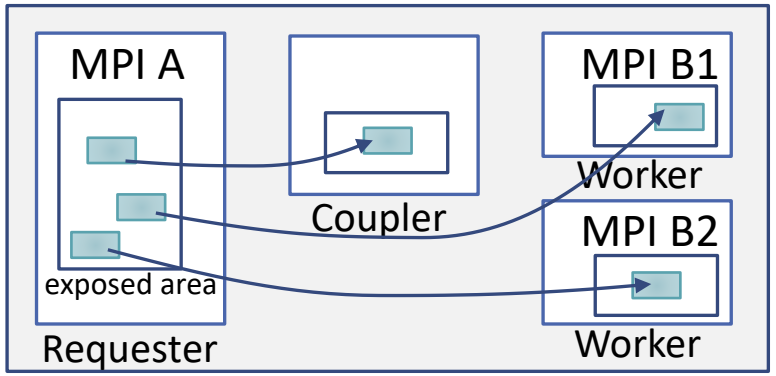
```
CTCAC_init();

CTCAC_regarea_real8(&areaid);

...
```

Requester can expose its memory area by "regarea".

- "regarea" is a collective operation.
  - All workers and the coupler must call "regarea" to receive the ID of the area.
- Workers and the coupler can perform remote read on the exposed area.
  - Offset and size should be specified.
  - Remote write to the exposed area is also possible.



Execution of CoToCoA

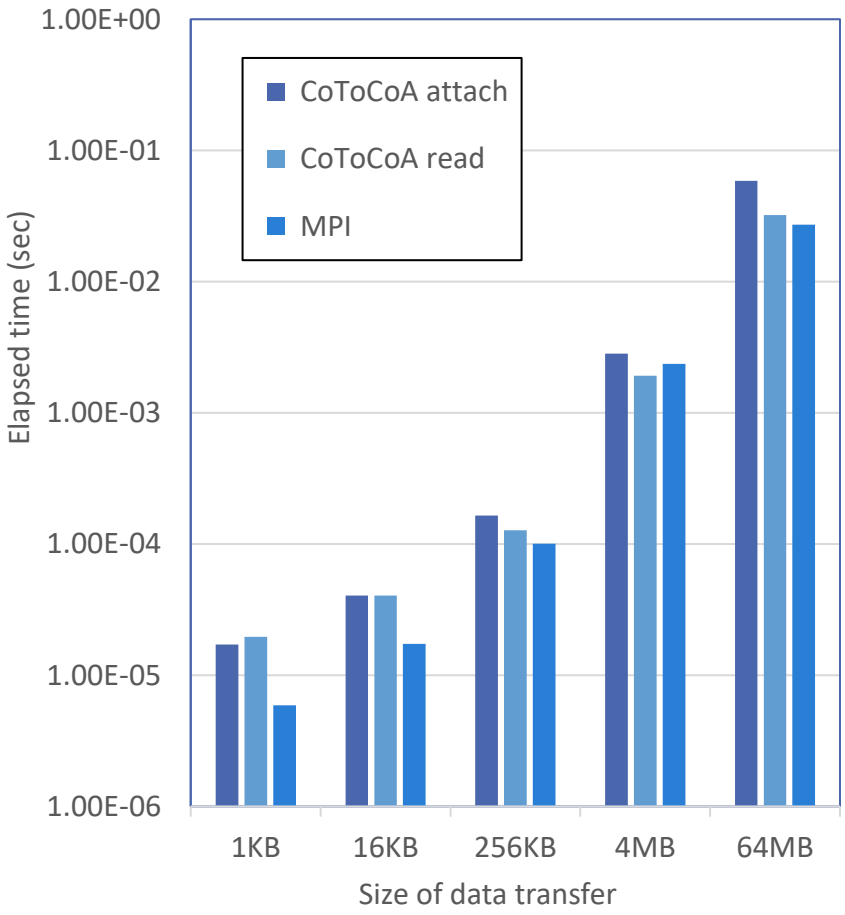
Use MPMD (Multiple Program Multiple Data) facility of MPI libraries.

Example:

```
mpiexec -n 4 ./requester : -n 1 ./coupler : -n 6 ./worker
```

- 4 processes for the requester
- 1 process for the coupler
- 6 processes for the worker

Performance evaluation



- Compare elapsed time of a request
  - From "sendreq" to "complete"
  - Difference among three data-transfer styles:
    - CoToCoA attach: Attach to a request
    - CoToCoA read: Remote read
    - MPI: Use MPI_Send and _Recv
  - Environment
    - ITO subsystem A of Kyushu Univ., Japan
      - Intel Xeon, Mellanox InfiniBand FDR
      - RedHat Enterprise 7, Intel MPI
    - 4 nodes
      - 4 procs / node
      - 4 procs for requester, one proc for coupler and 6 procs for worker.

Conclusion

Introduced a communication-efficient framework for coupling programs, CoToCoA.

- Minimal modification to existing MPI programs.
- Asynchronous execution.
- Low overhead of data transfer.

Examined that the overhead of CoToCoA is sufficiently low.

Future works

- Development of coupled simulations on CoToCoA.
  - Example:
    - "Application of Cross-Reference Framework CoToCoA to Macro- and Micro-Scale Simulations of Planetary Magnetospheres "

CoToCoA is available at
https://doi.org/10.5281/zenodo.5655841