Hands on Parallel Programming with OpenMPI

Taha Atahan Akyıldız

Tutorials are available at =>
https://docs.truba.gov.tr/

https://docs.truba.gov.tr/education/openmpi/index.html

You can find the examples at =>
https://github.com/kamerkaya/computhon2021-1/tree/main/MPI

Flow

- What is OpenMPI?
- Truba Architecture
- Topologies and Brain Teasers
- Important MPI Concepts
- Primitive OpenMPI commands
- Collective Communication
- Derived Data Types
- Message Packing-Unpacking
- Vertex Degree Counting Example

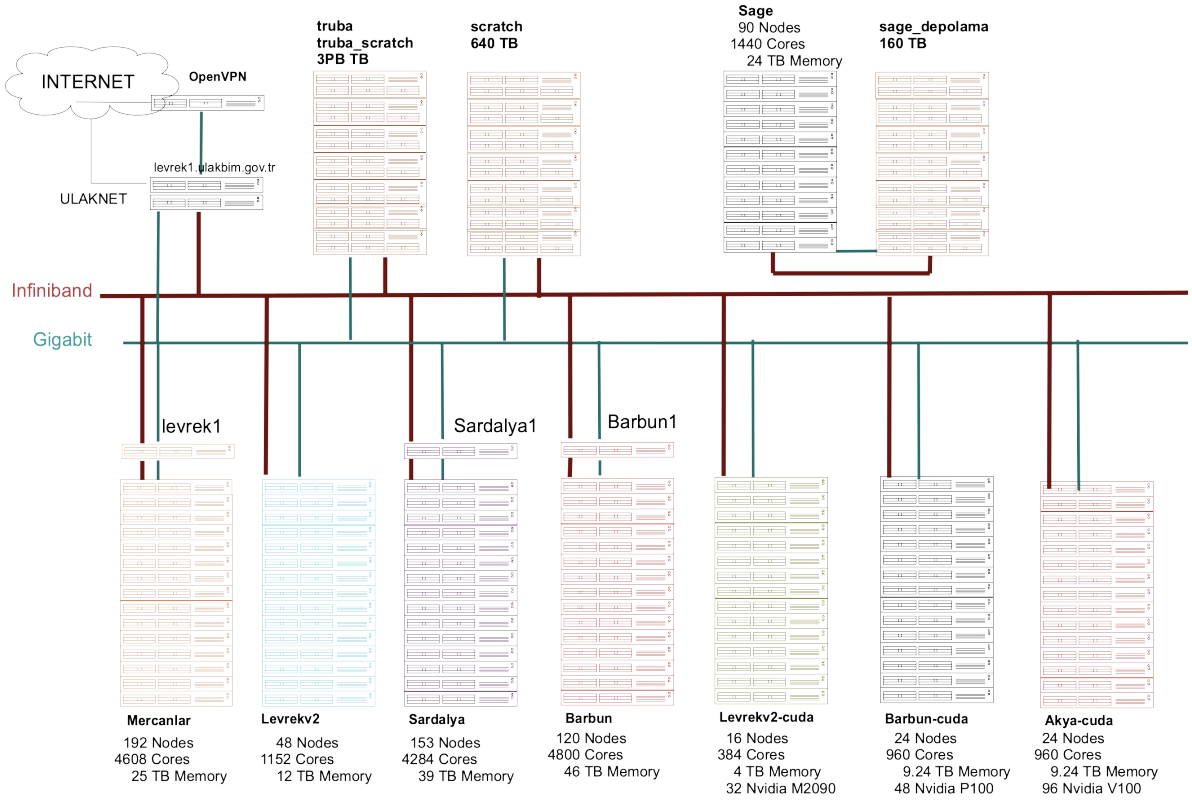
Open MPI https://www.open-mpi.org/

OpenMPI is an **open source** interface for carrying out **information transfer between processes**. It is implemented with respect to the MPI standard.

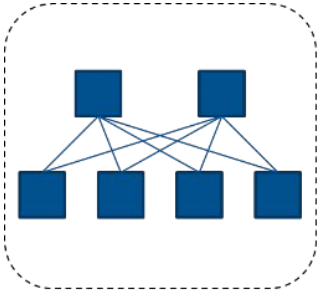
It was developed by a **consortium** of **academic**, **research** and **industry** partners.

With the OpenMPI standart a lot of low level concerns related to parallel programming is abstracted away from the programmer. Some of these are **thread safety and concurrency**, **network and process fault tolerance**, i**nfrastructure related differences**, etc.

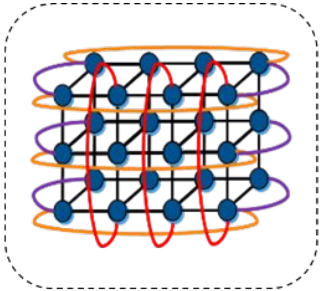
Truba Architecture



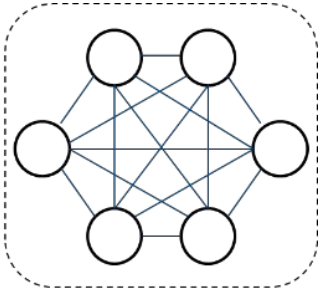
Cluster Topologies



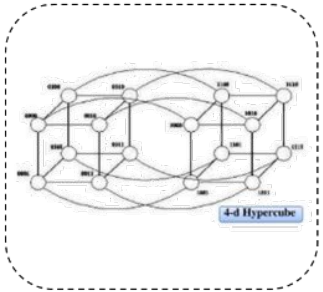
Fat Tree



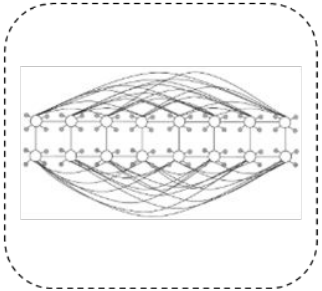
Torus



Dragonfly



Hypercube



HyperX

Important MPI Concepts

Data Types: MPI_CHAR, MPI_INT, MPI_FLOAT, MPI_DOUBLE

Groups and Communicators: Resources can be divided into different groups via this interface and MPI commands can be run only between these groups. For example, **MPI_COMM_WORLD** represents all processors in the system.

- Climate Models
- Distributed Linear Algebra
- Quicksort

Primitive OpenMPI Commands

MPI_Init(&argc, &argv): Initialize MPI environment.

MPI_Finalize(): Terminate MPI environment.

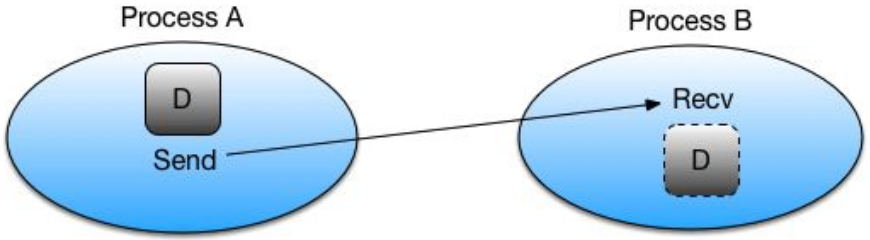
MPI_Comm_size(MPI_comm communicator, &num_procs): Provides the number of processes.

MPI_Send(message, count, data_type, destination, tag, MPI_comm communicator): Sends a message to destination.

MPI_Recv(message, count, data_type, source, tag, MPI_comm communicator, status): Receives message from source.

More on Send-Recv

Process A decides to send data to process B. It first packs the data into a buffer. Rank A then calls MPI_Send to create a message for rank B. The communication device is responsible for routing the message to the correct destination.

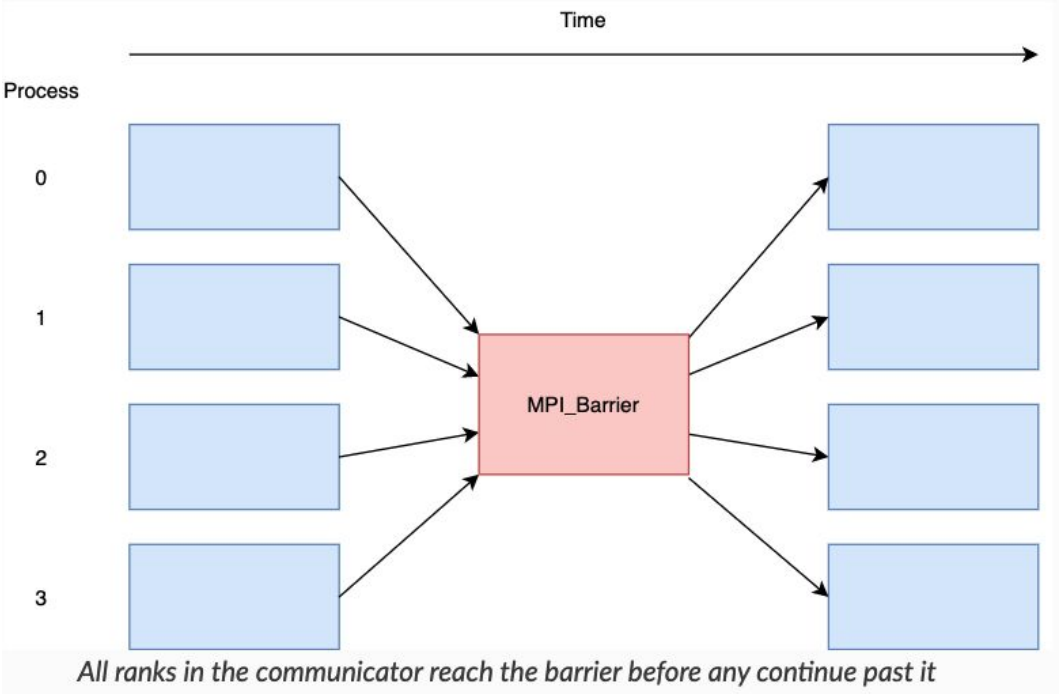


Process B must know that it is about to receive a message and acknowledge this by calling MPI_Recv. This sets up a buffer for writing the incoming data and instructs the communication device to listen for the message. The message will not actually be sent before the receiving process calls MPI_Recv, even if MPI_Send has been called.

Collective Communication (Barrier)

int MPI_Barrier(MPI_Comm comm):

- is blocking.
- enables collective synchronization.



Collective Communication

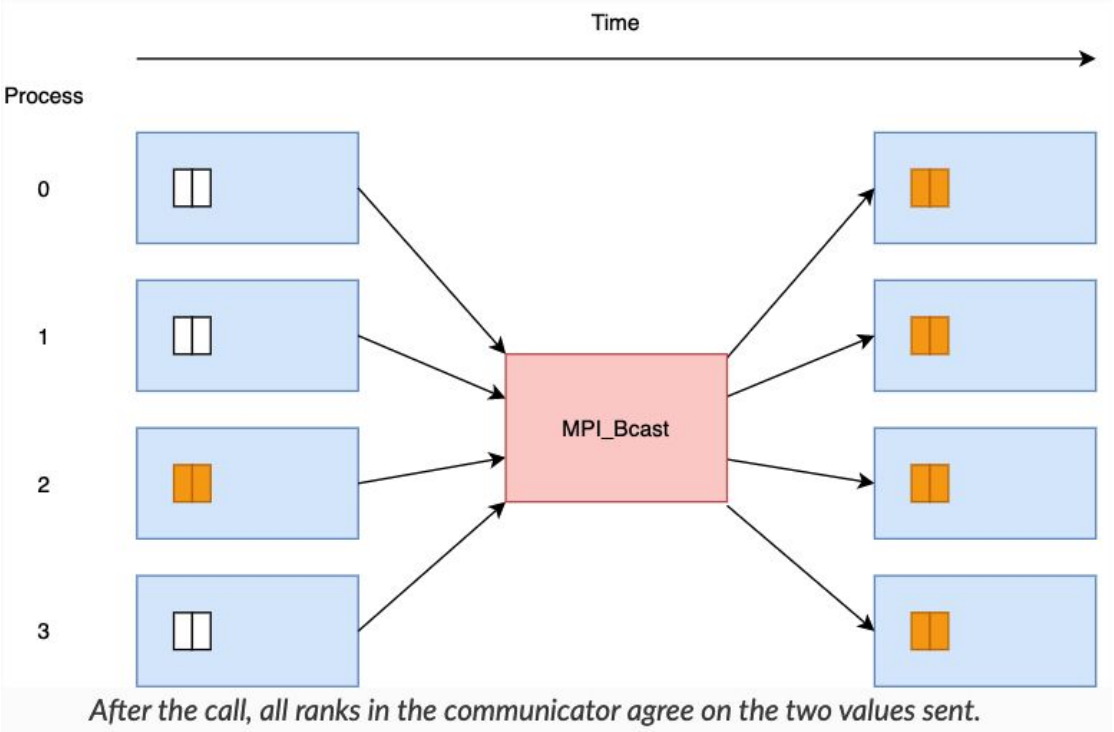
Parallel programs needs to collaborate for efficient communication and correct computation. Some common collective operations are:

- **Barrier:** Ensure all processes have reached a certain point.
- **Broadcast:** Share data with all ranks.
- **Reduce:** Compute based on data from all ranks.
- **Scatter,gather:** Rearrange data across ranks.

Collective Communication (Broadcast)

int MPI_Bcast(void * buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

- Also blocking.
- On top of synchronization it also distributes data from a rank to all ranks.

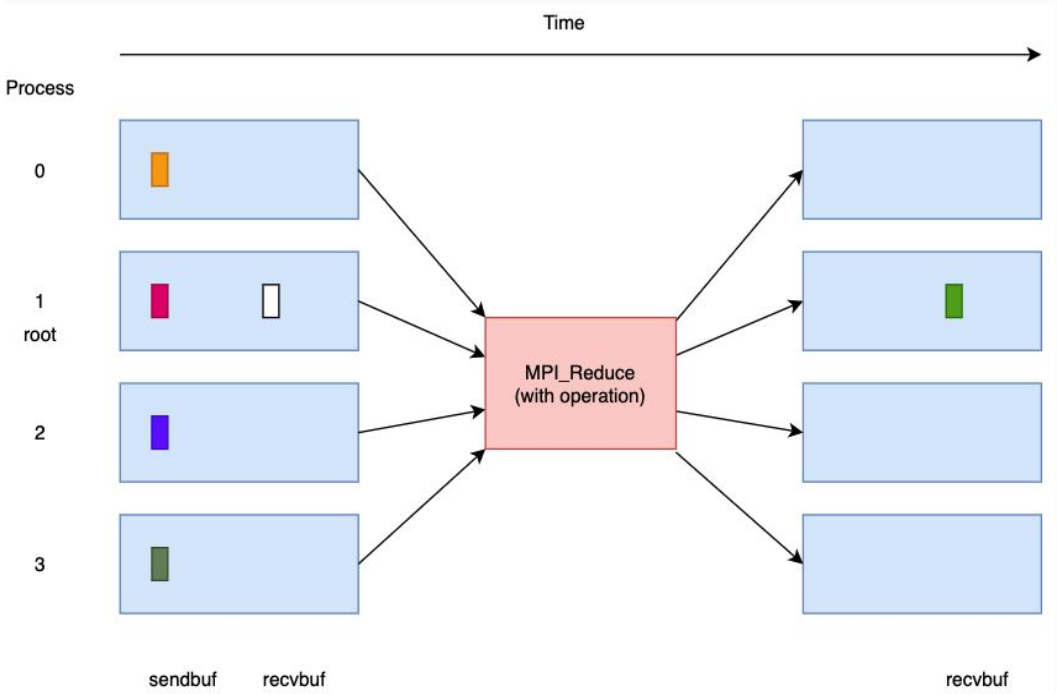


After the call, all ranks in the communicator agree on the two values sent.

Collective Communication (Reduce)

int MPI_Reduce(const void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)

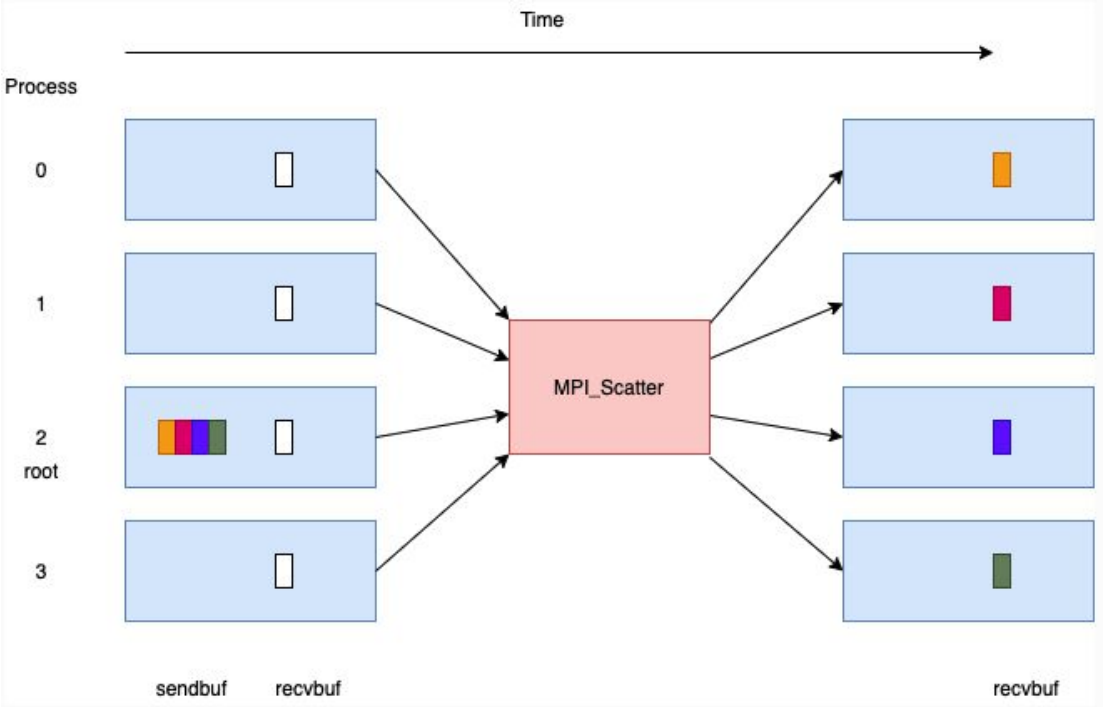
- Also blocking.
- The root rank receives a value computed by combining a value (with the provided information) from each other rank in the communicator.



Collective Communication (Scatter)

int MPI_Scatter(const void * sendbuf, int sendcount, MPI_Datatype send_dt, void * recvbuf, int recvcount, MPI_Datatype recv_dt, int root, MPI_Comm comm)

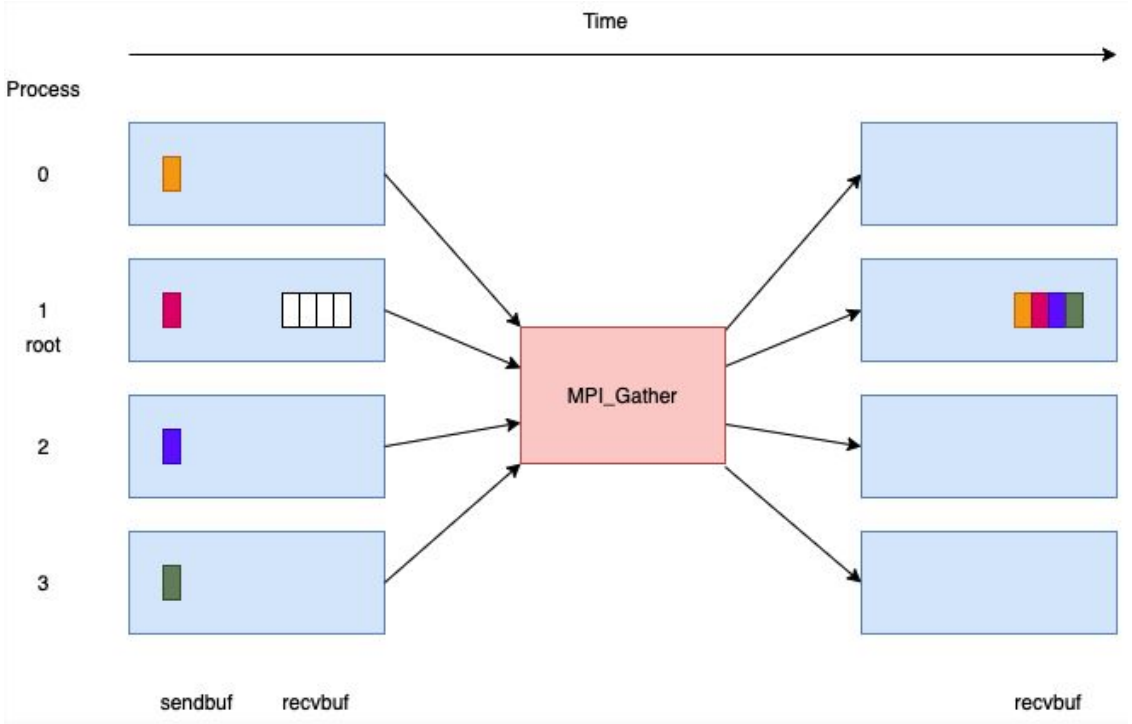
- Blocking communication.
- Distributes different pieces of information to all the other ranks.



Collective Communication (Gather)

int MPI_Gather(const void * sendbuf, int sendcount, MPI_Datatype sendtype, void * recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

- Blocking communication.
- Gathers different pieces of information from all the other ranks to the root.



Non-Blocking (Asynchronous) Communication

One problem with the communication alternatives we have seen is that they are all blocking. For collective operations synchronization is usually wanted but for point to point operations, we are usually waiting unnecessarily! This time can be utilized for more computation.

- int MPI_Isend(const void * buff, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request *request)
- int MPI_Irecv(void * buff, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Request request)

When the non-blocking versions are used the main thread is not blocked by the send and receive functions.

Derived Data Types

We can also define **custom data types** for communication.

We just need to provide the low level information of the primitive data types that forms our custom data type. This is because MPI does not know how to represent user-defined datatypes in memory by itself.

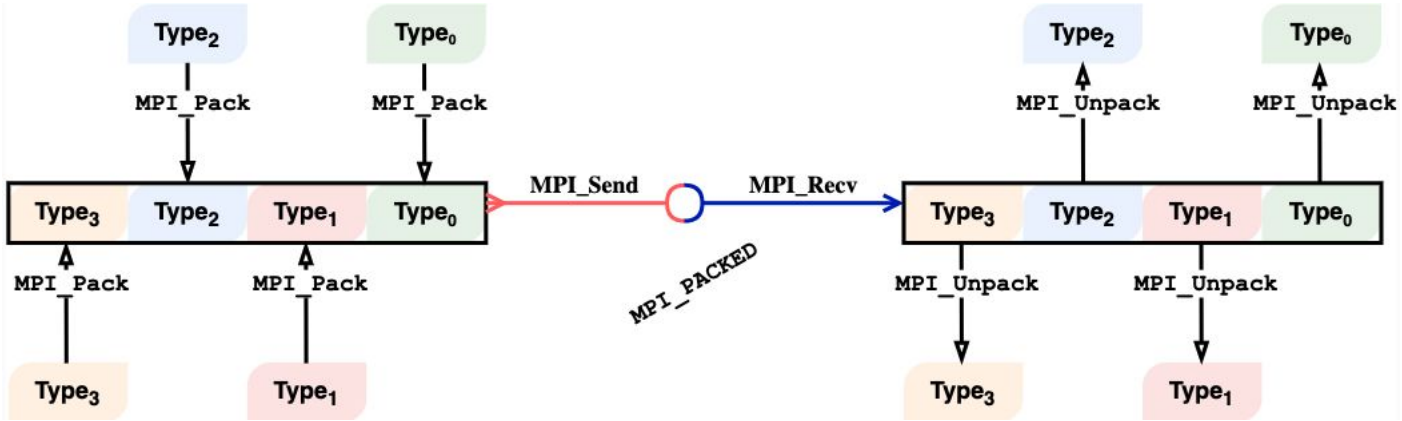
Providing **displacements**, **block_lenghts** and **type signature** takes care of this problem.

```
int MPI_Type_create_struct(int count, const int array_of_block_lengths[], const MPI_Aint array_of_displacements[], const MPI_Datatype array_of_types[], MPI_Datatype *newtype)
```

We can use the above command to create an **OpenMPI version** of the **C language struct**. Bear in mind that this version is only helpful for storing data.

Message Packing

MPI allows the programmer to **communicate heterogeneous collections** into a single message, without defining a full-fledged custom data type.



References

- https://enccs.github.io/intermediate-mpi/
- https://pdc-support.github.io/introduction-to-mpi/
- https://www.open-mpi.org/
- https://docs.truba.gov.tr/education/openmpi/index.html

Thanks for listening!
Questions?