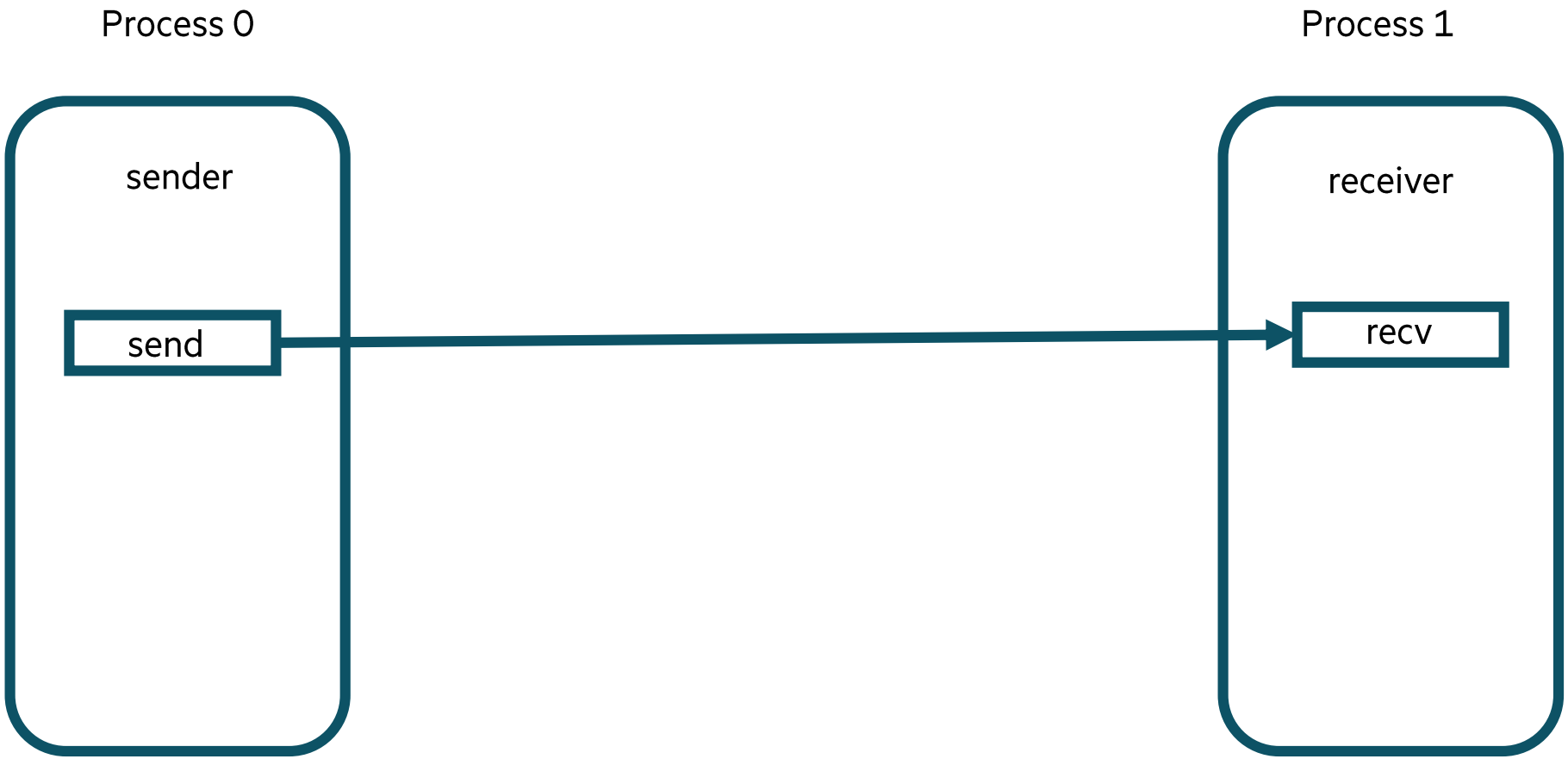
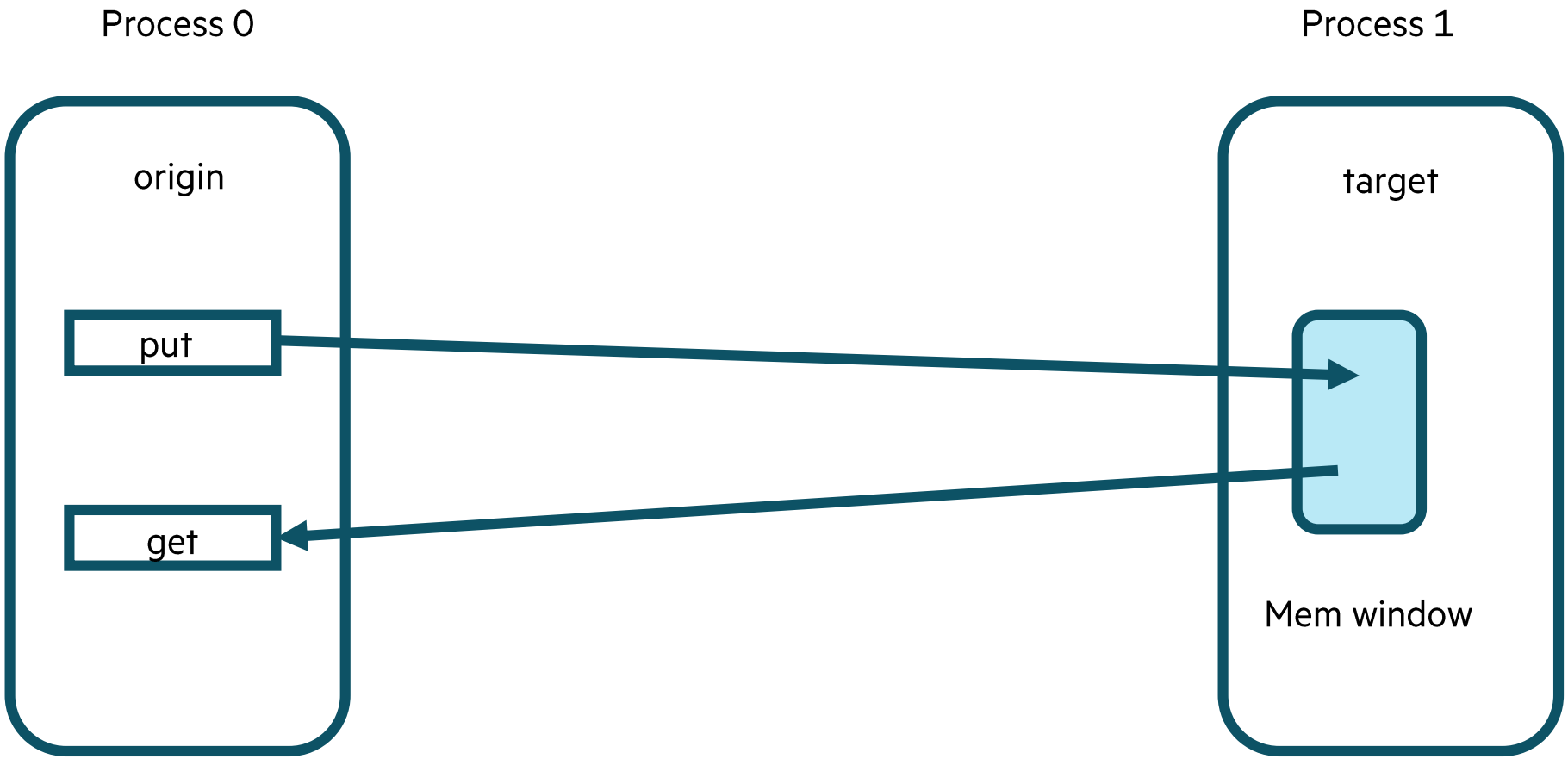
Hewlett Packard
Enterprise

NOTIFIED COMMUNICATION

Nils Imhoff

June 10, 2022


Process 0
Process 1
sender
receiver
send
recv


Process 0
Process 1
origin
target
put
get
Mem window

ONE-SIDED OPERATIONS

- Create & allocate memory window

- Access to remote window (RMA)
  - Put
  - Get
  - Accumulate

- Synchronization
  - Late sender/ late receiver problem
  - requires guarantee consistency
  - Active-target (Flush)
  - Passive-target (lock/unlock)

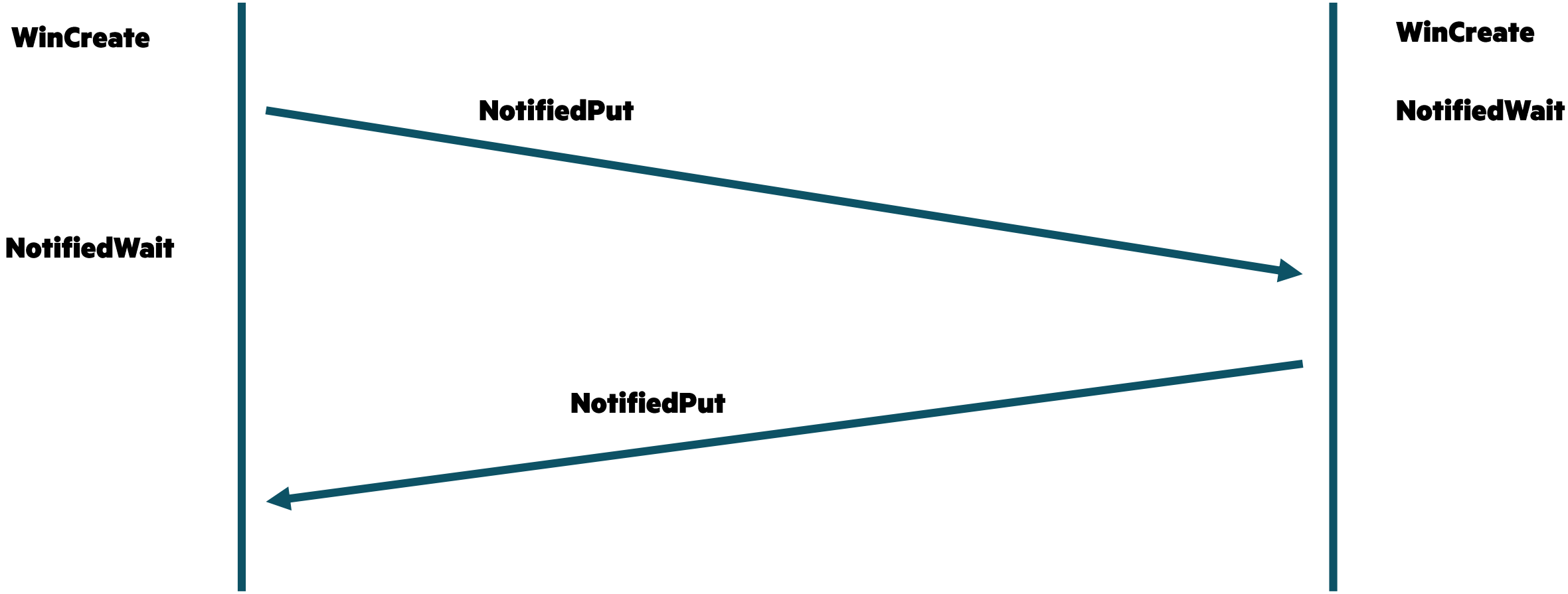
NOTIFIED COMMUNICATION

- One-sided read and writes
- remote completion with notifications.
- Asynchronous execution model

WEAK SYNCHRONIZATION

- Synchronization via Notification
  - Identified by notification ID
  - Logical association of memory location and notification

PING PONG


WinCreate
NotifiedPut
NotifiedWait
WinCreate
NotifiedWait
NotifiedPut

PUT NOTIFY

- MPI_Put_notify
- Do a Put to the target area.
- send a notification
  - a value that the target rank can wait for
  - and know that something has happened

```
MPI_Put_notify(const void *origin_addr, int origin_count, MPI_Datatype
               origin_datatype, int target_rank, MPI_Aint target_disp,
               int target_count, MPI_Datatype target_datatype, MPI_Win
               win, uint32_id)
```

GET NOTIFY

- MPI_Get_notify
- Do a Get from the target side to the origin side
- send a notification
  - a value that the target rank can wait for
  - and know that something has happened

```
MPI_Get_notify(void *origin_addr, int origin_count, MPI_Datatype
               origin_datatype, int target_rank, MPI_Aint target_disp,
               int target_count, MPI_Datatype target_datatype, MPI_Win
               win, uint32_id)
```

WAIT NOTIFY

- MPI_Win_wait_notify
- Waits until it receives notification with matching ID

```
MPI_Win_wait_notify(MPI_Win win, uint32 id)
```

TEST NOTIFY

- MPI_Test_notify
- Tests if notification with matching id was already received

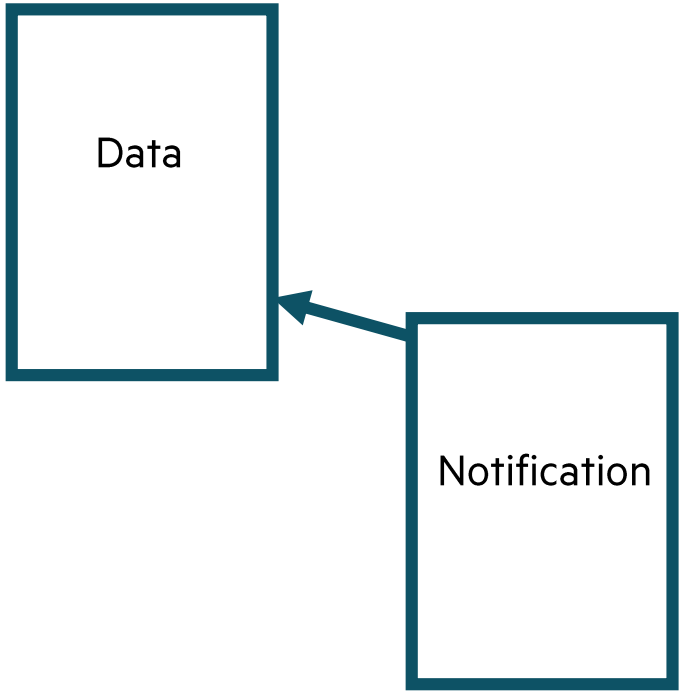
- `MPI_Win_test_notify(MPI_Win win, int *flag, uint32 id)`

PERFORM PUT NOTIFY

- `IBV_WR_RDMA_WRITE_WITH_IMM`
- Direct access to remote memory region
- immediate data
  - Available in Work Completion
  - Receive Request

PERFORM GET NOTIFY

- Chained work requests
- Last work request for notification



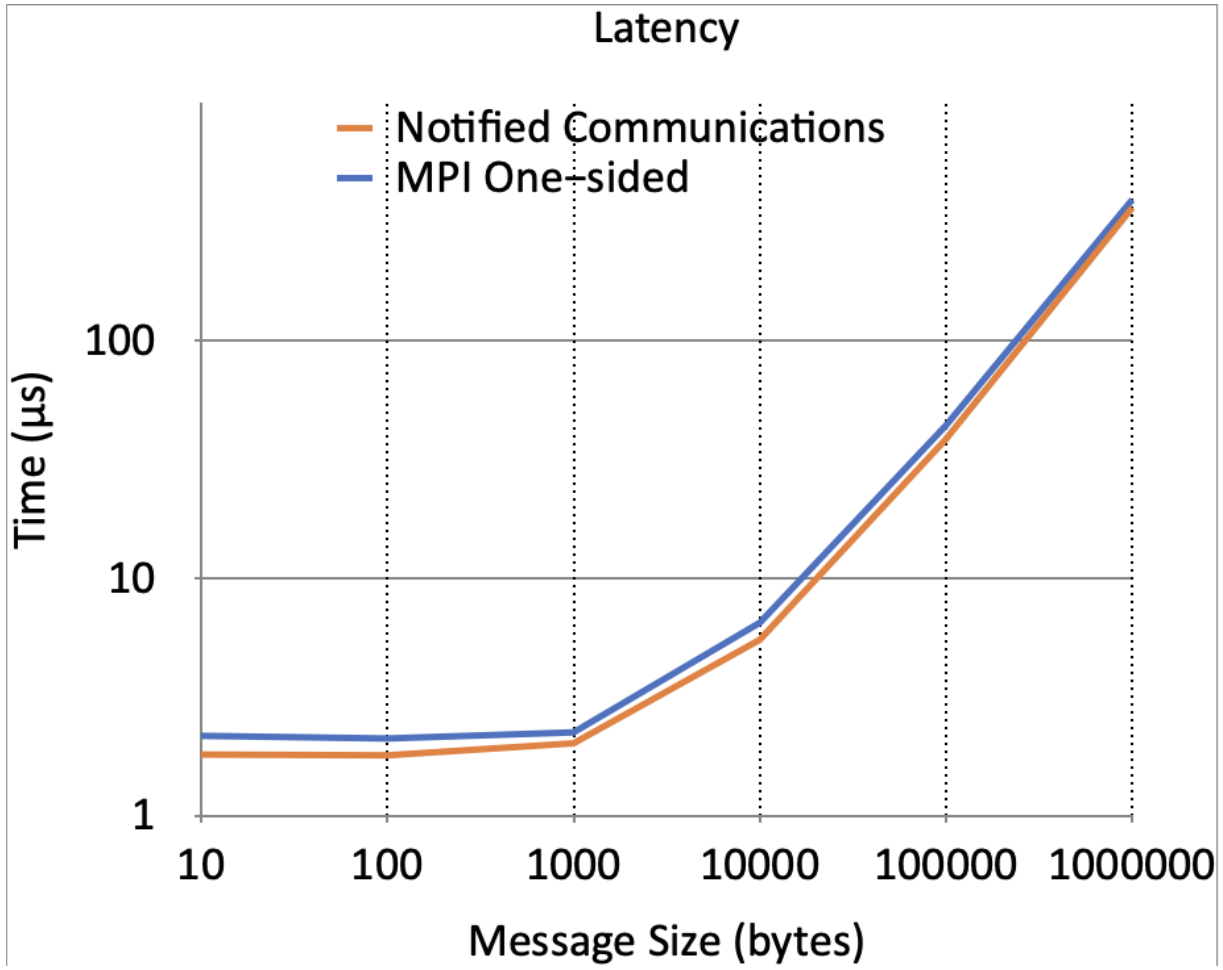
PERFORM WAIT NOTIFY / TEST NOTIFY

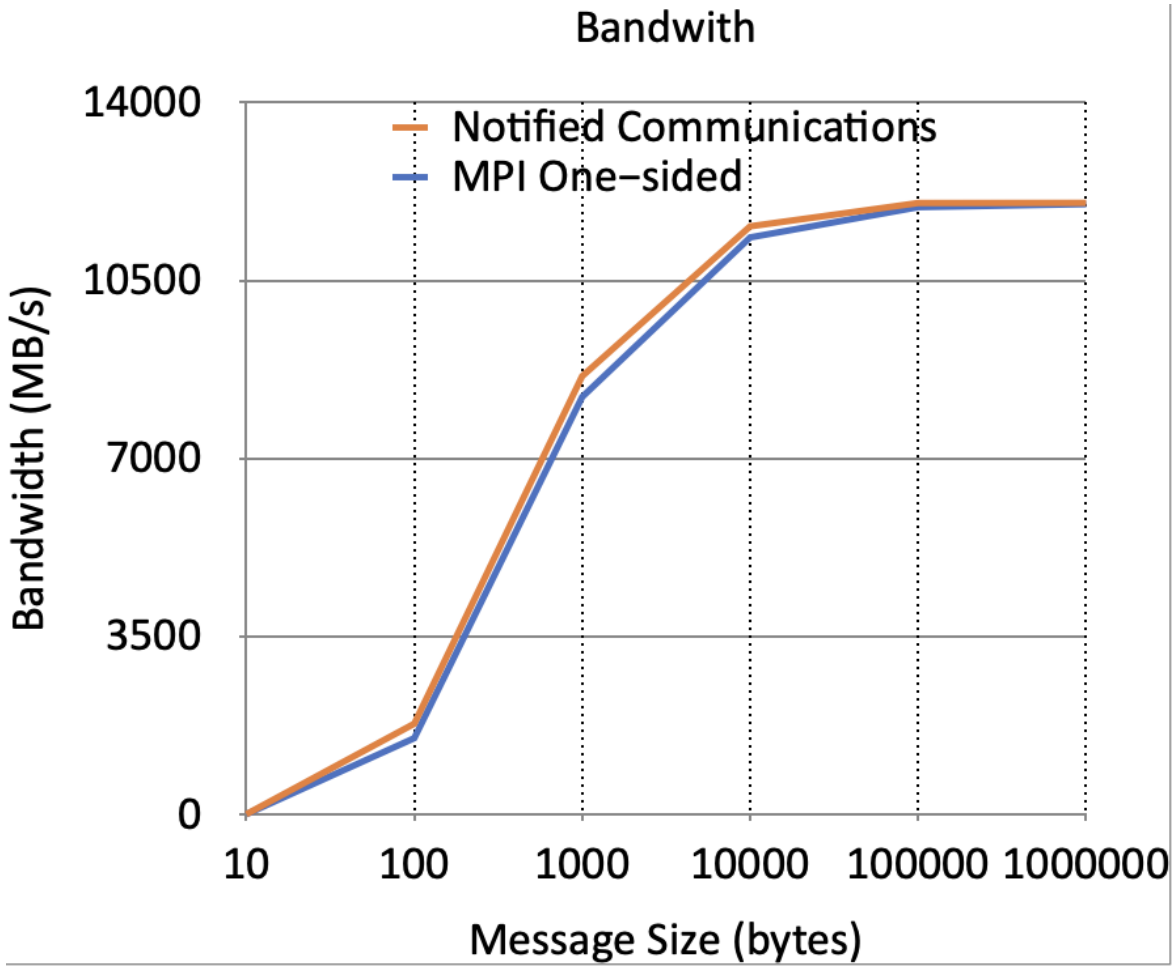
- Waiting for notification
- Get notification from the completion queue

PING PONG

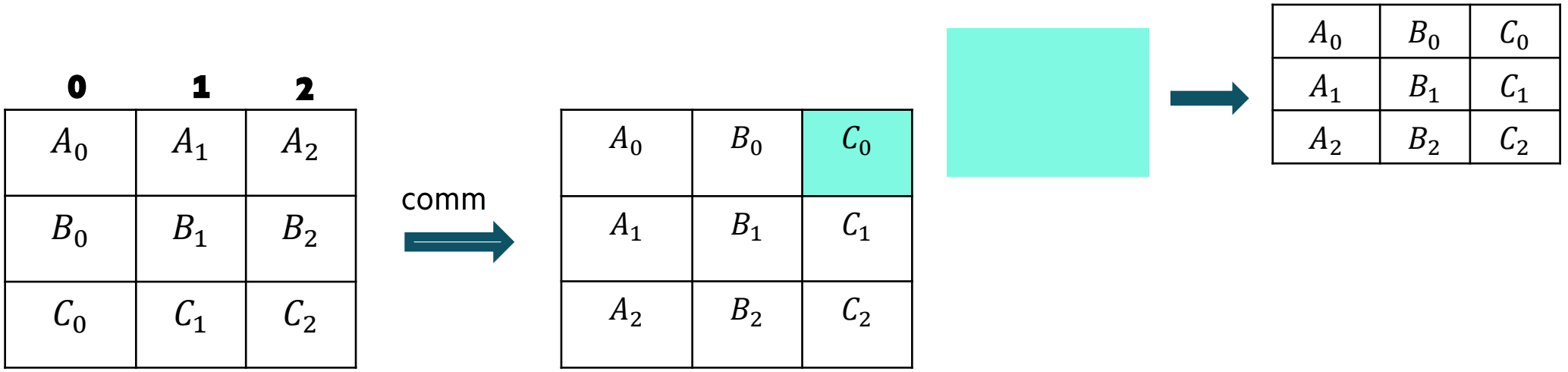
```
if (first_sender)
{
  MPI_Put_notify(&baseptr_send[0], num_item_to_send, MPI_UINT64_T, send_to, 0,
                 num_item_to_send, MPI_UINT64_T, win_send, tag);
  MPI_Win_wait_notify(win_send, id);
}
else
{
  MPI_Win_wait_notify(win_send, id);
  MPI_Put_notify(&baseptr_send[0], num_item_to_send, MPI_UINT64_T, send_to, 0,
                 num_item_to_send, MPI_UINT64_T, win_send, id);
}
```

BENCHMARK RESULTS





MATRIX TRANSPOSE



Matrix Transpose =>

Global Transpose + Local Transpose => MPI_Alltoall + Local Transpose

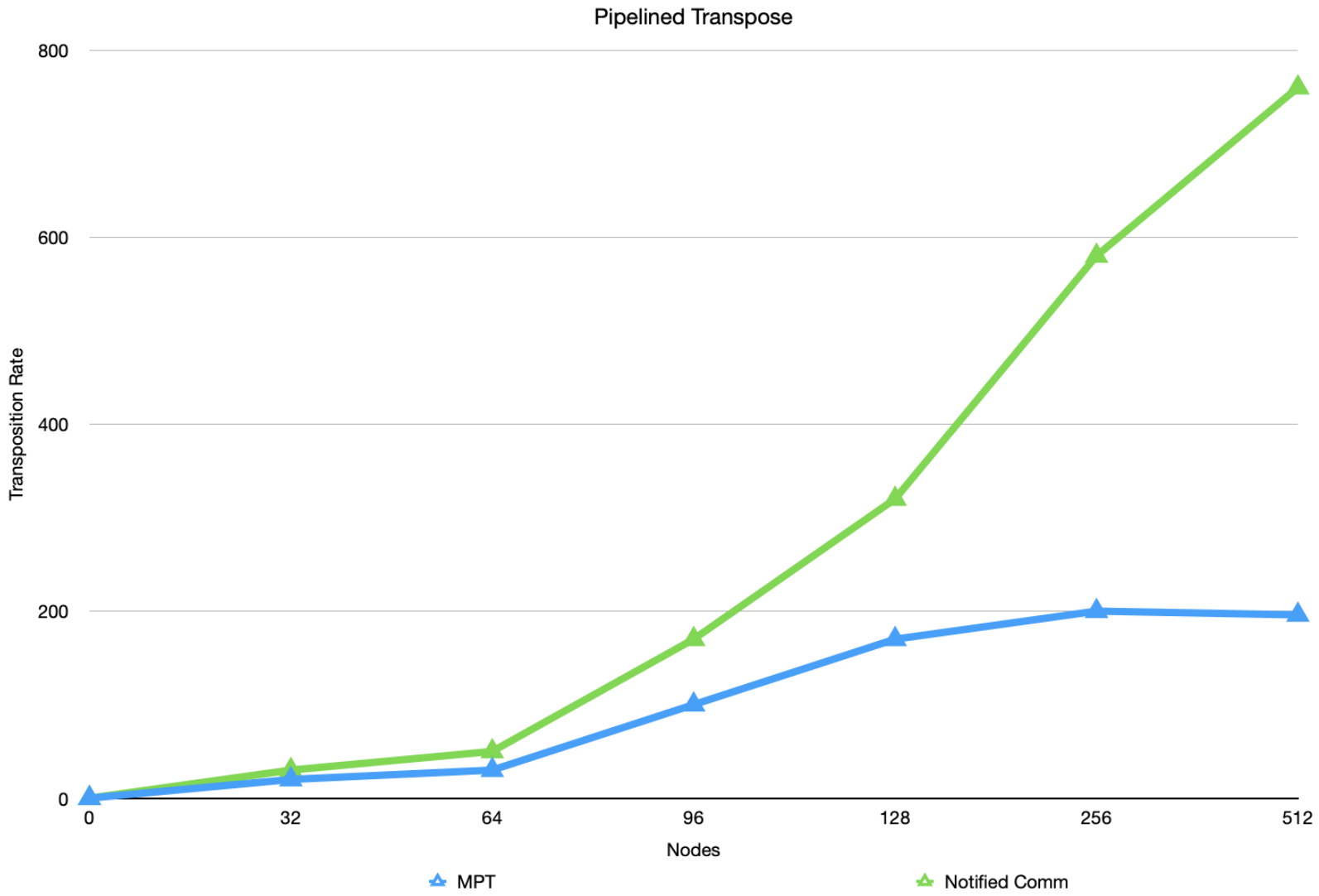
SCHEMATIC IMPLEMENTATION

```
#pragma omp parallel{
    if(this_is_first_thread()){

      for(i = 0; i < P-1; i++){
          MPI_Put_notify(&submatrix[i], i);
      }
    }
    while(!complete){
      for(j = 0; j < my_num_submatrices; j++){
        if(MPI_Win_test(win,j)== MPI_SUCCESS){
          transpose(&submatrix[j]);

        }
      }
    }
}
```

BENCHMARK RESULTS



CONCLUSION

- Our implementation is better than the current One-Sided Communication
- Shows potential
- No late send / late receive

THANK YOU

nils.imhoff@hpe.com