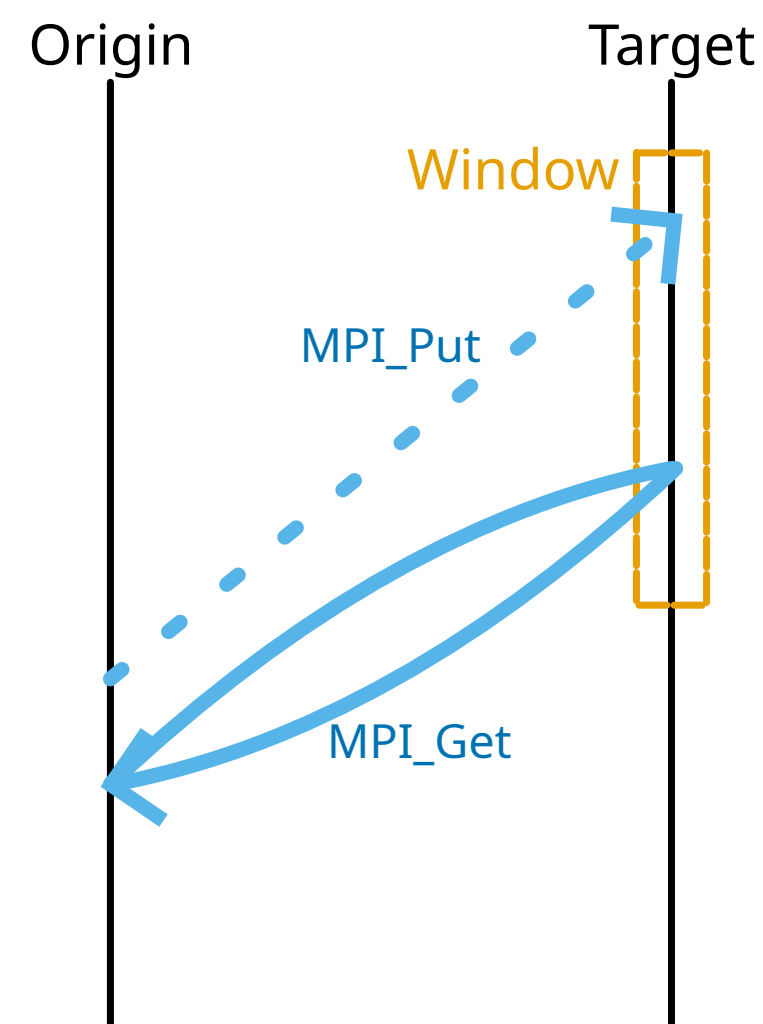


MPI One-Sided Communications (or MPI RMA)

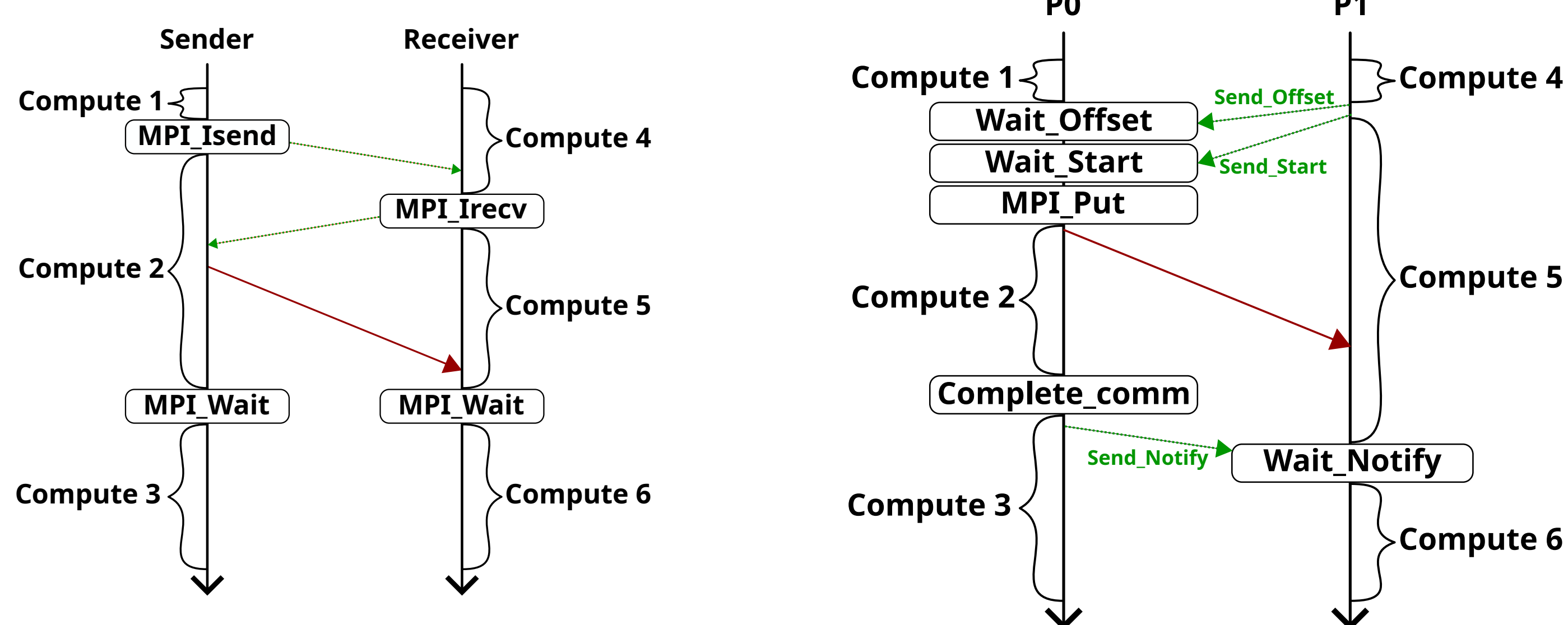
- Benefit : Decoupling communications and synchronizations
- Better communication/computation overlap
- Each MPI process exposes a part of its memory (*window*) to remote MPI processes



- **Challenge: MPI RMA programs are complex to write**
 - ▷ Need to ensure memory consistency
- **Our goal:** Help developers migrate their programs towards MPI RMA

Generalized Post-Start-Complete-Wait (GPSCW)

- A general approach to transform nonblocking two-sided communications into their one-sided counterparts while reducing and merging synchronizations



- Send/Wait_Offset: *target* tells *origin* where to put the data
- Send/Wait_Start: *target* tells *origin* that it is ready to receive data

Optimizing GPSCW

Goal: An MPI code using nonblocking communications and the transformation we propose

```

allocate(r_buf);
...
read(r_buf);
...
for(int i=0; i<num_it; i++){
  if(my_rank == sender){
    for(int j=0; j<n; j++){
      MPI_Isend(s_buf[j],...,receiver,...);

      ...
      MPI_Waitall();
      write(s_buf);
    }
  }
  if (my_rank == receiver){
    for(int j=0; j<n; j++){
      MPI_Irecv(r_buf[j],...,sender,...);

      ...
      MPI_Waitall();
      read(r_buf);
    }
  }
}

```

```

allocate(r_buf);
if(my_rank == receiver)
  for(int j=0; j<n; j++){
    Send_Offset(r_buf[j]);
    ...
    read(r_buf);
    ...
    if(my_rank == receiver)
      Send_Start();
    ...
    for(int i=0; i<num_it; i++){
      if(my_rank == sender)
        for (int j=0; j<n; j++){
          Wait_Offset();
          Wait_Start();
          MPI_Put(s_buf[j], ...);
        }
      ...
      Complete Comms();
      Send_Notify();
      write(s_buf);
    }
  }
  if(my_rank == receiver){
    Complete Comms();
    for(int j=0; j<n; j++){
      Wait_Notify();
      read(r_buf);
      Send_Start();
    }
  }
}

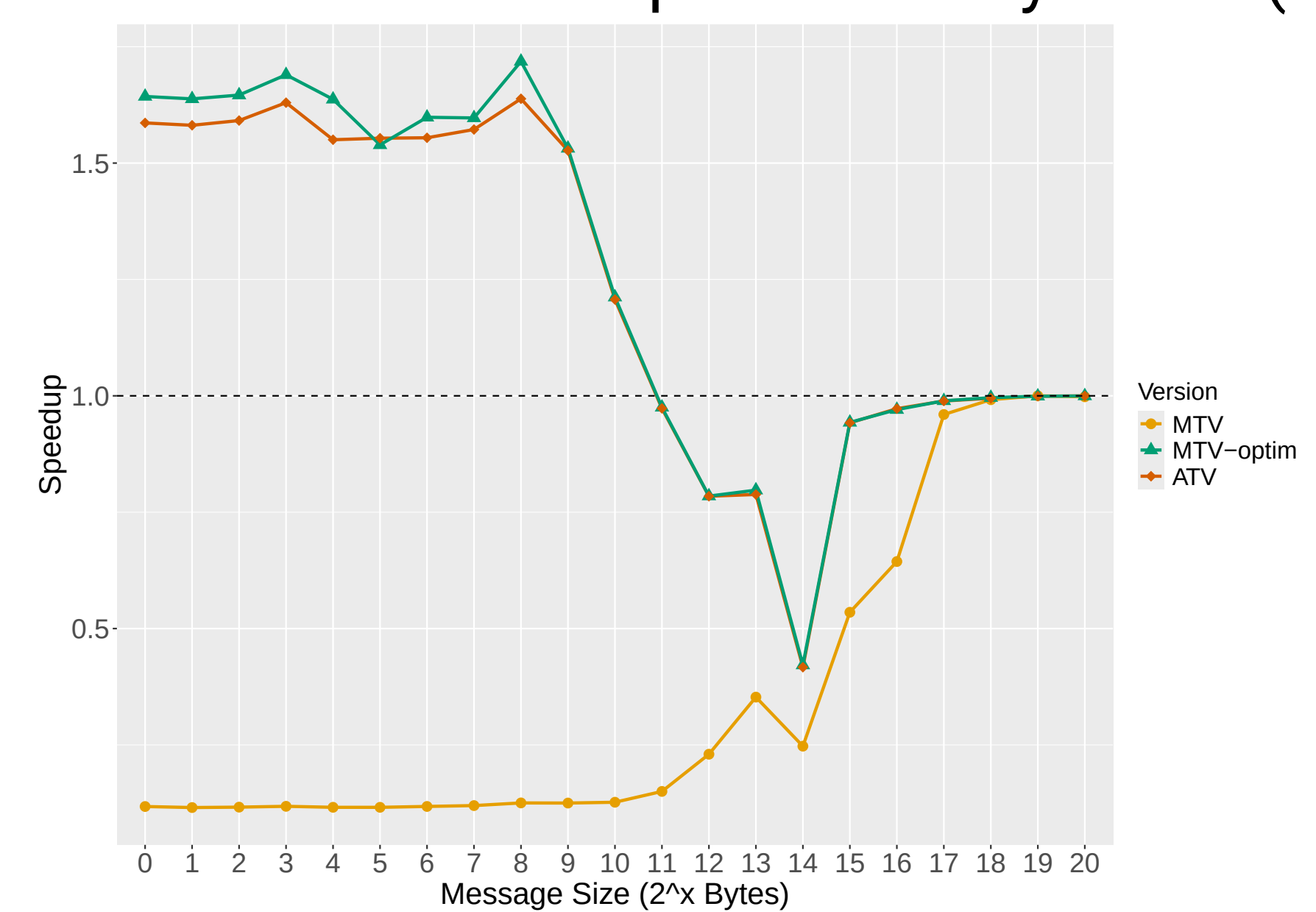
```

OSCAR

- Automatize the transformation to GPSCW at compile time
- Three parts using LLVM passes:
 - ▷ Inline Pass: inlines the relevant functions
 - ▷ Analysis Pass: determines where to put the synchronizations
 - ▷ Transformation Pass: performs the transformations
- Several transformations from a single code depending on:
 - ▷ Synchronization model: **Lockall, Fence, PSCW**
 - ▷ Communication: **Put, Get**

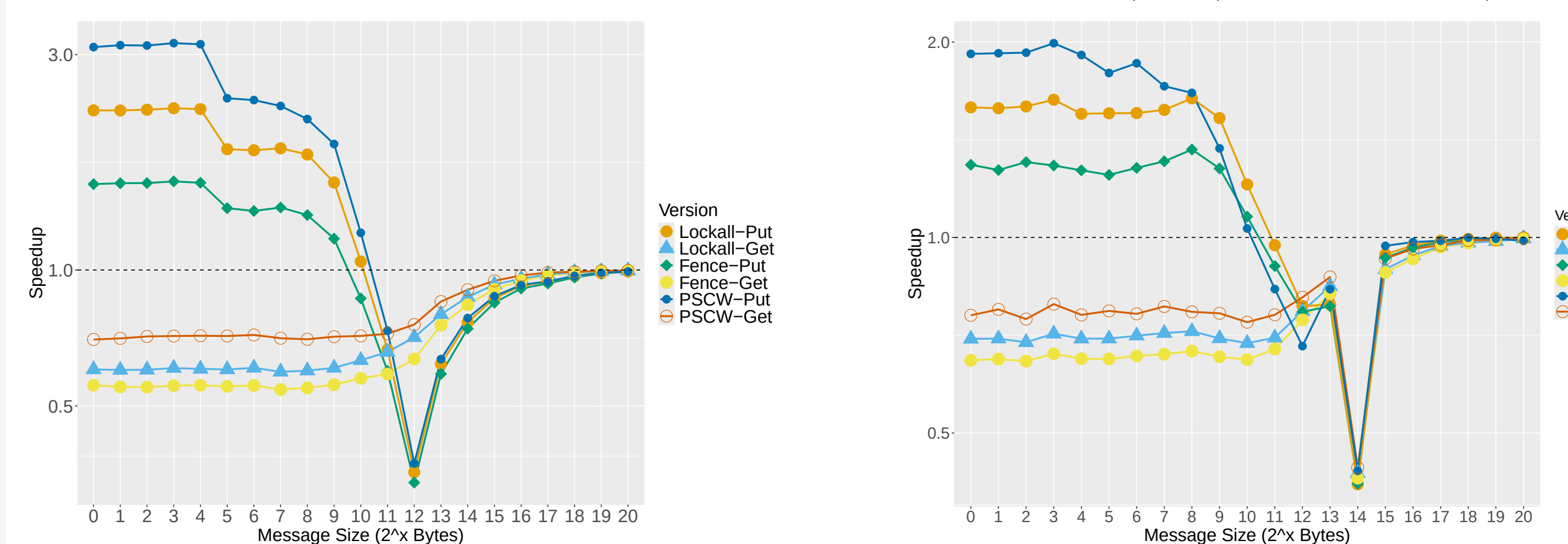
Transformation on the OSU benchmarks

Speedup compared to the Original Version (OV) using nonblocking pt2pt communications; **MTV**=naive manual transformation of OV; **MTV-opt**=manual transformation with optimizations; **ATV**= automatic transformation performed by Oscar (**Lockall-Put**).



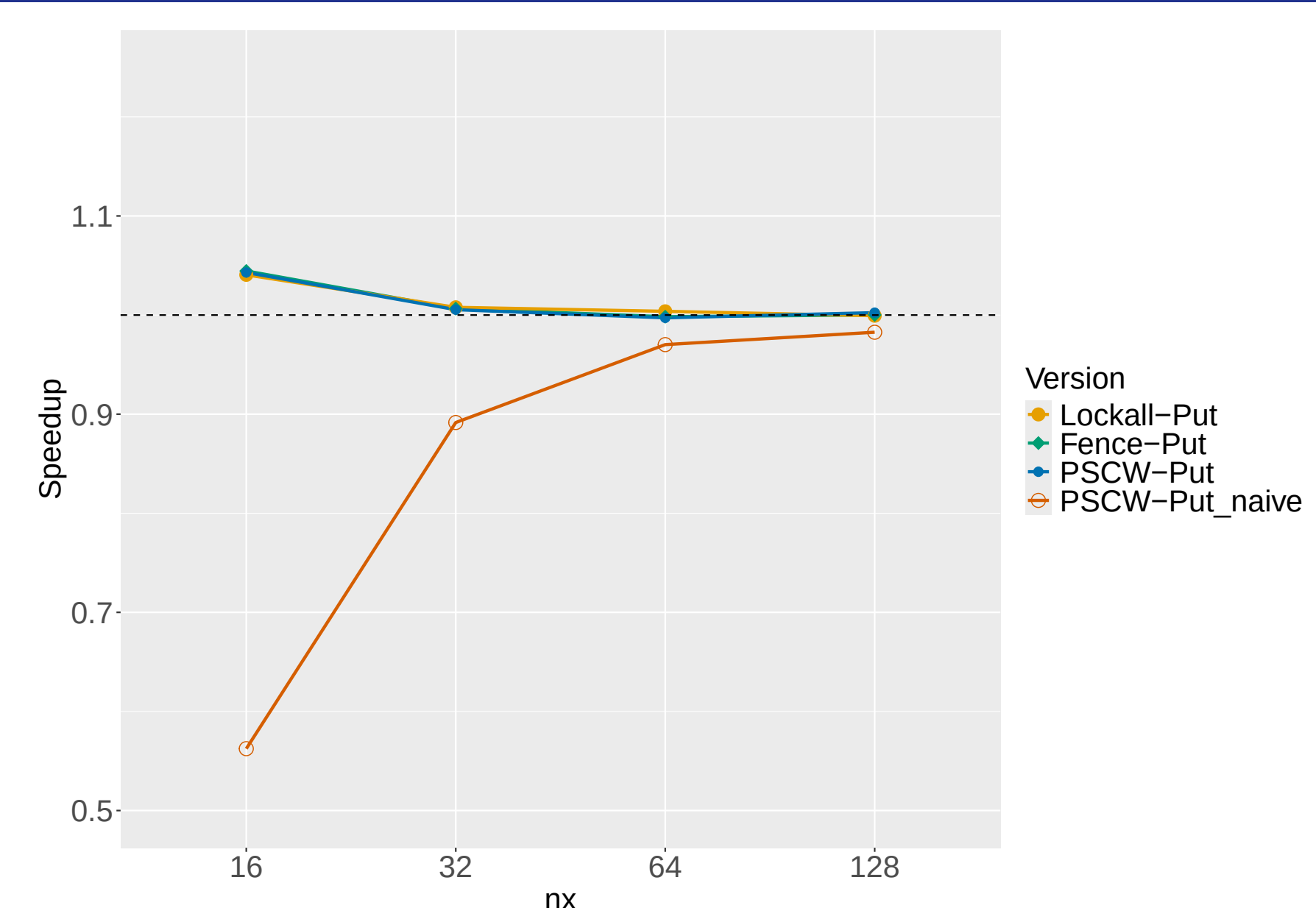
- MTV slower than OV: same number of synchronizations but uses higher-level, less efficient application level functions.
- On small messages, optimized versions send the buffer

Comparison between all the automatic transformations proposed by OSCAR on two different clusters: Amour (left) and Pise (right).



- OSCAR determines which transformation suits the best the setup.

Transformation on HPCG



- Transformation not efficient because original code does not take advantage of the nonblocking semantics.

Conclusion and Perspective

- New tool to transform codes that use nonblocking pt2pt communications into codes optimized for MPI RMA
- Find regions in the code where MPI RMA may be beneficial
- Applicability to other PGAS libraries/languages