



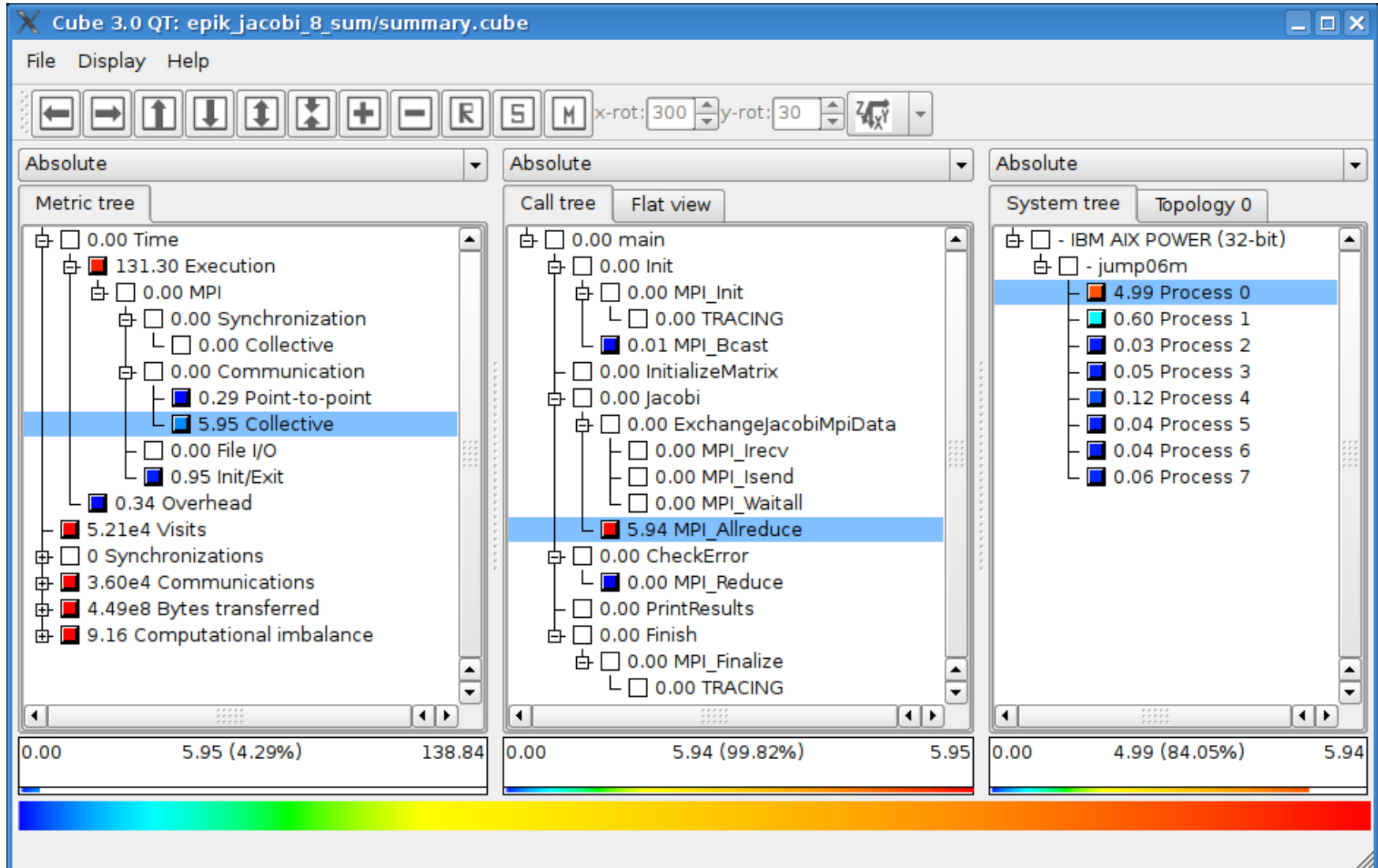
# scalasca

## Performance properties

### “The metrics tour”

Markus Geimer & Brian Wylie  
Jülich Supercomputing Centre  
[scalasca@fz-juelich.de](mailto:scalasca@fz-juelich.de)

# Scalasca analysis result

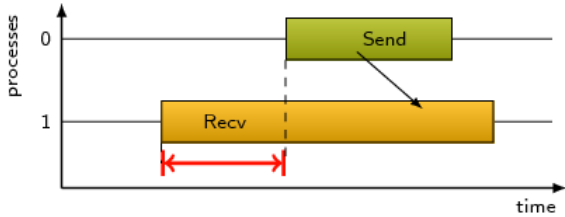


- Analysis report explorer GUI provides hyperlinked descriptions of performance properties
- Diagnosis hints suggest how to refine diagnosis of performance problems and possible remediation

**Performance properties**

### Late Sender Time

**Description:**  
Refers to the time lost waiting caused by a blocking receive operation (e.g., `MPI_Recv` or `MPI_Wait`) that is posted earlier than the corresponding send operation.



If the receiving process is waiting for multiple messages to arrive (e.g., in an call to `MPI_Waitall`), the maximum waiting time is accounted, i.e., the waiting time due to the latest sender.

**Unit:**  
Seconds

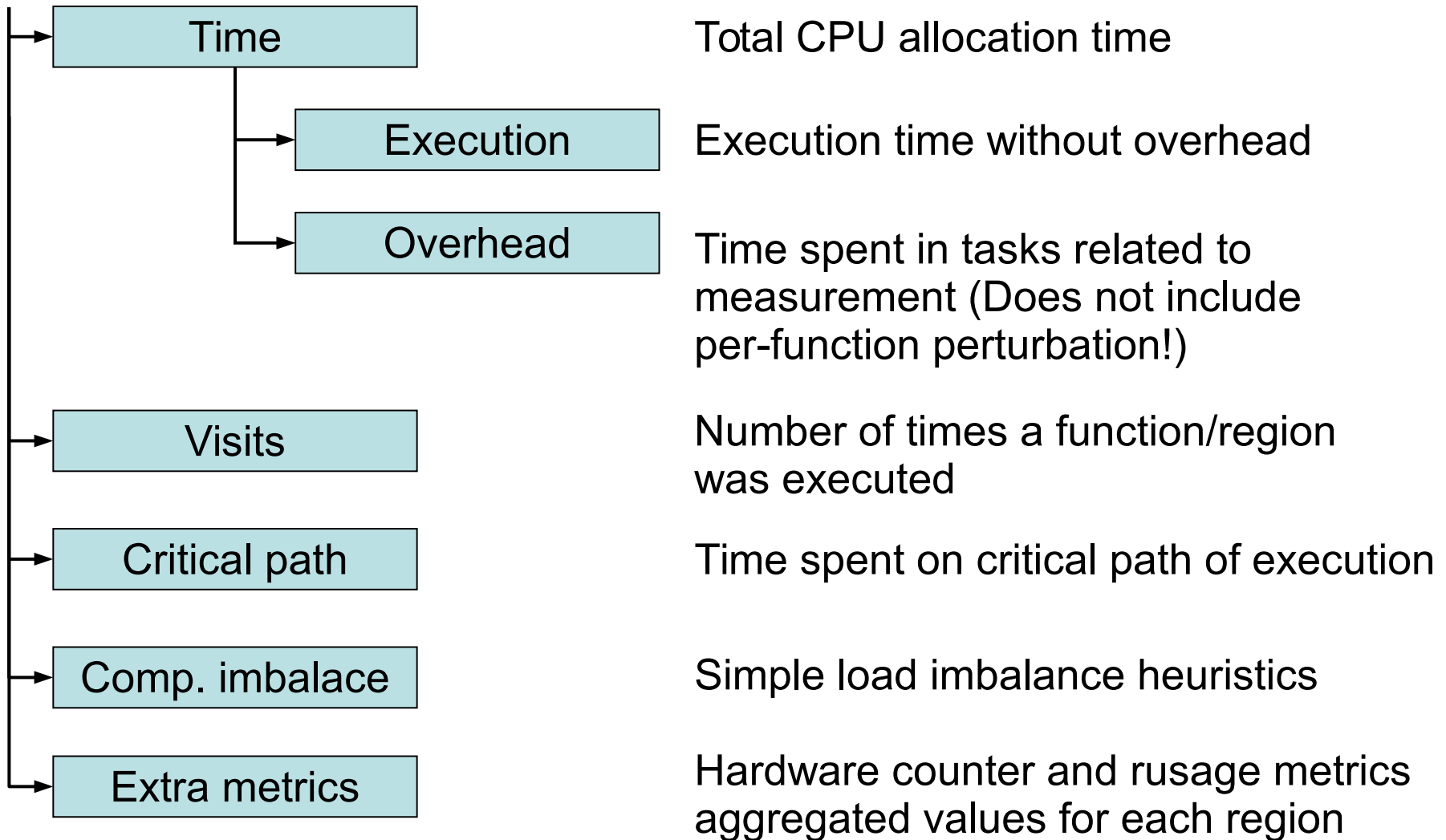
**Diagnosis:**  
Try to replace `MPI_Recv` with a non-blocking receive `MPI_Irecv` that can be posted earlier, proceed concurrently with computation, and complete with a wait operation after the message is expected to have been sent. Try to post sends earlier, such that they are available when receivers need them. Note that outstanding messages (i.e., sent before the receiver is ready) will occupy internal message buffers, and that large numbers of posted receive buffers will also introduce message management overhead, therefore moderation is advisable.



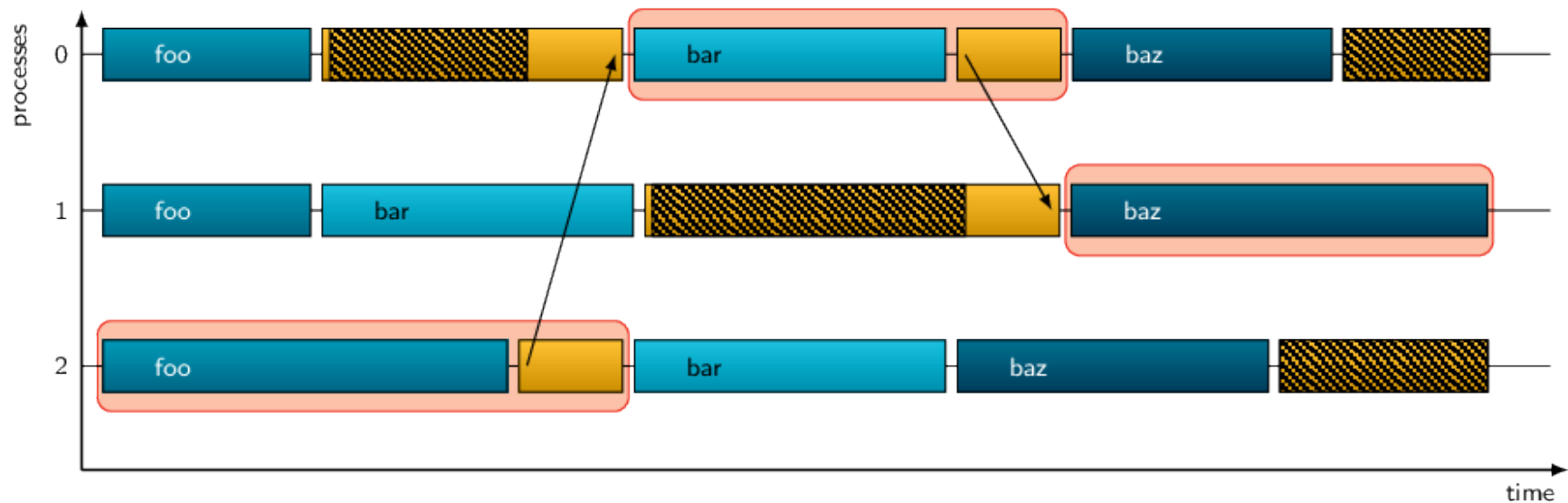
# VI-HPS

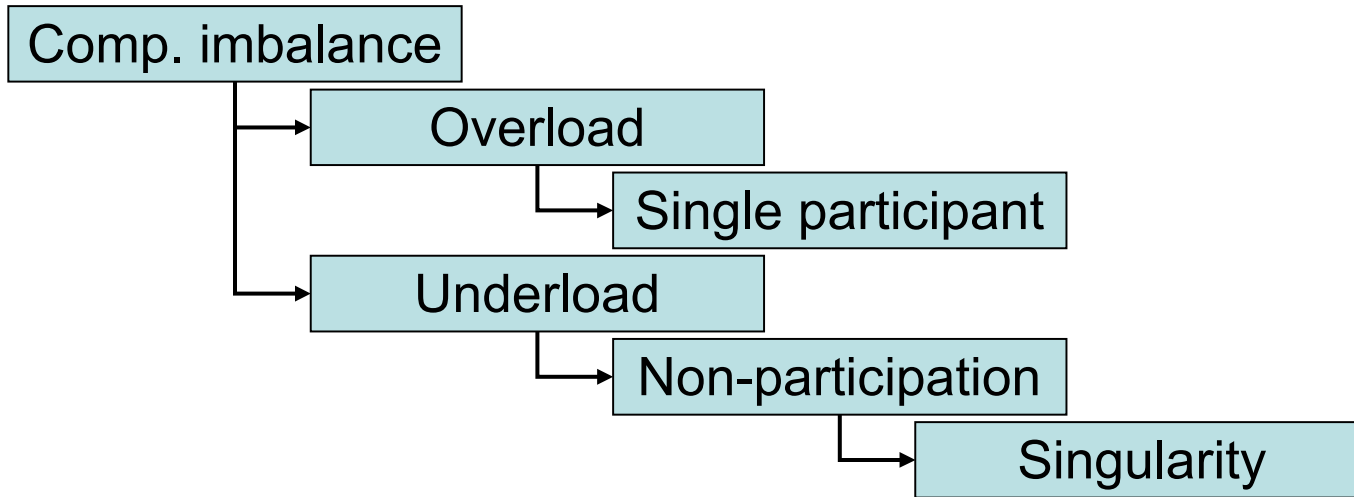


## Generic metrics



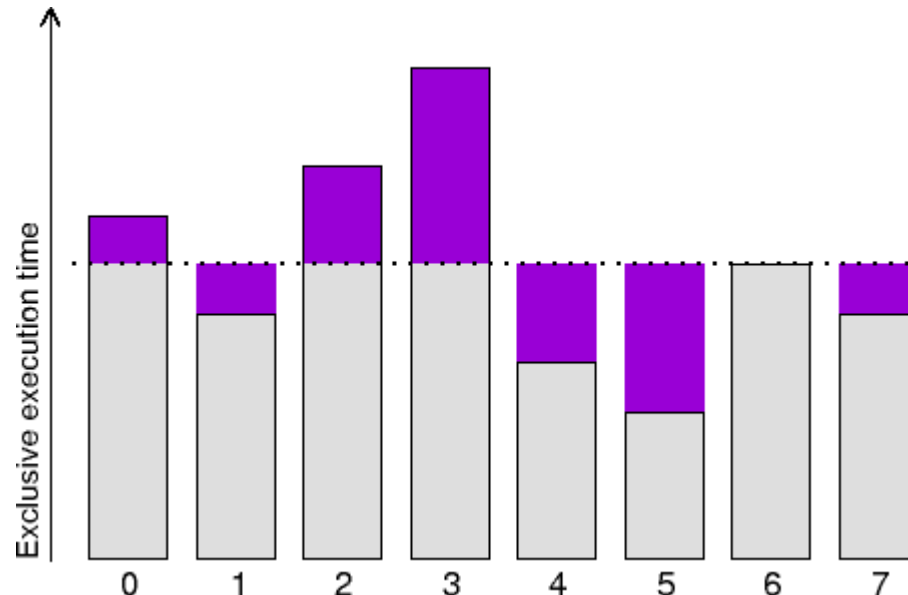
- Time on critical path of execution
  - follows causality chain from last event back to the first
- Call paths with a lot of time on the critical path are good candidates for optimisation
  - shortening events not on the critical path will only result in more waiting time at the next communication/synchronization



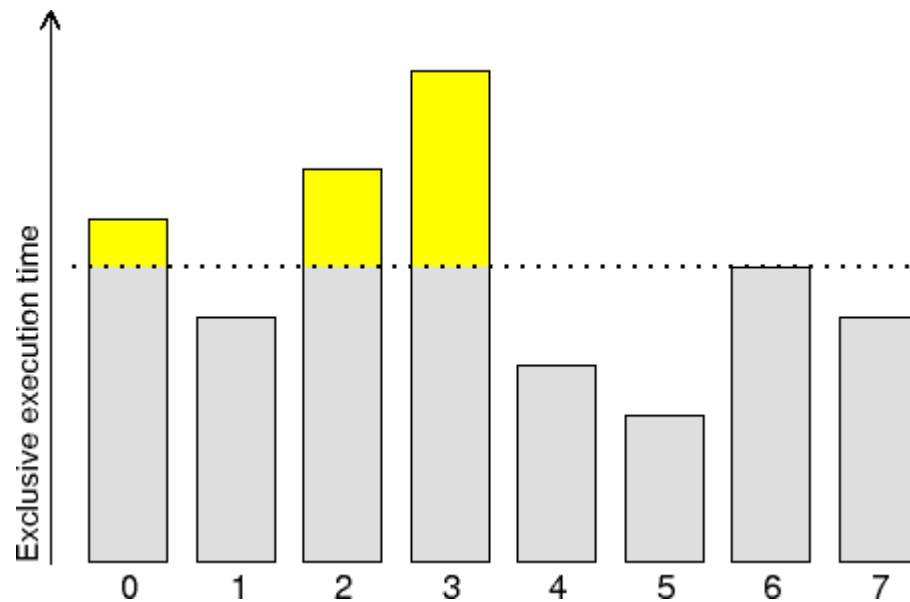




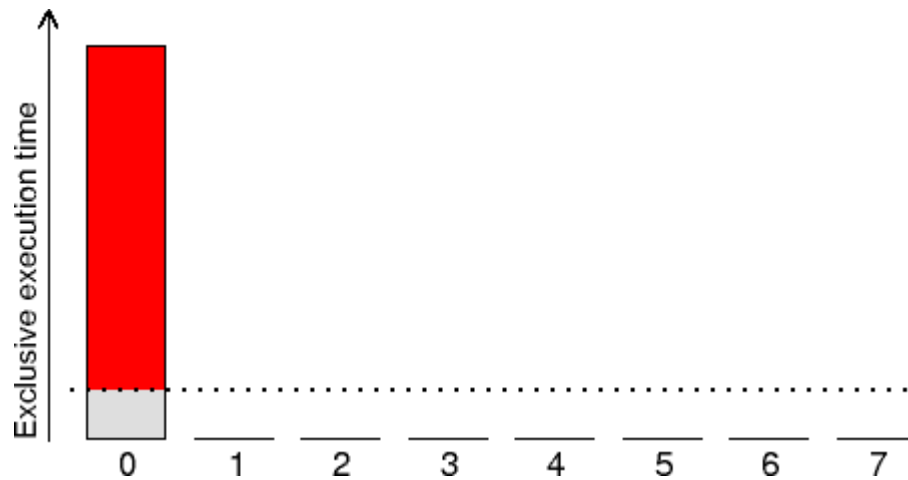
- Absolute difference to average exclusive execution time
  - Focusses only on computational parts
- Captures global imbalances
  - Based on entire measurement
  - Does not compare individual instances of function calls
- High value = Imbalance in the sub-calltree underneath
  - Expand the subtree to find the real location of the imbalance



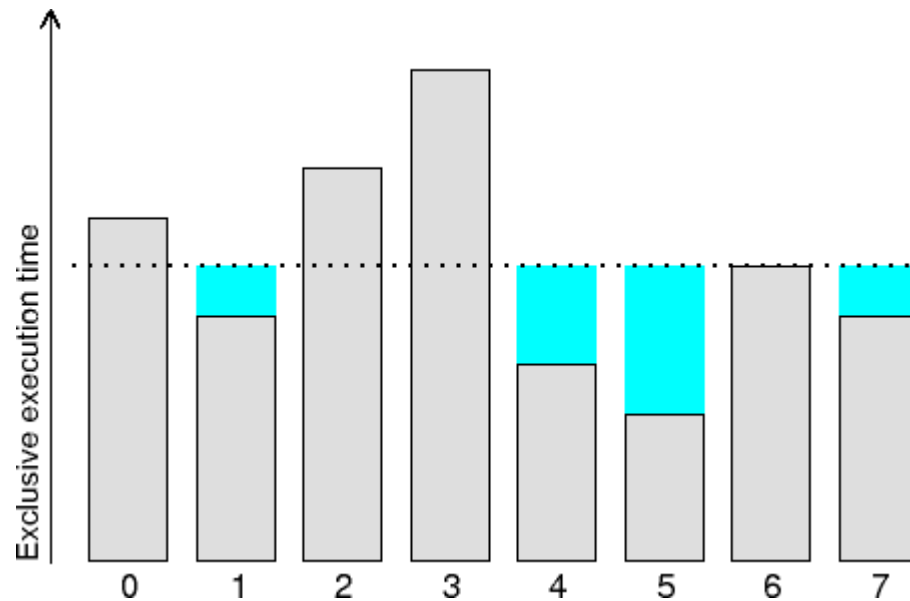
- Identifies processes/threads where exclusive execution time for the call-path was above average



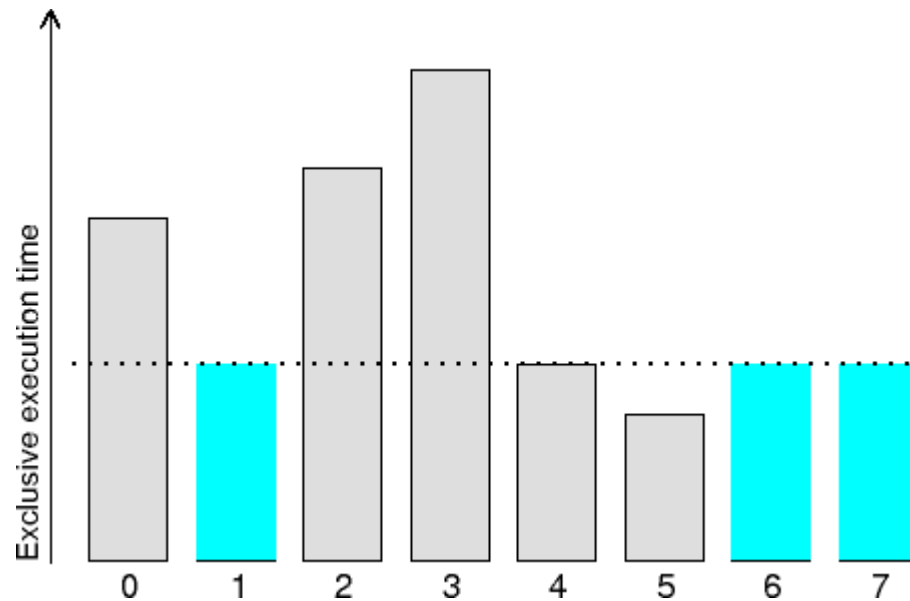
- Identifies call-paths executed by single process/thread



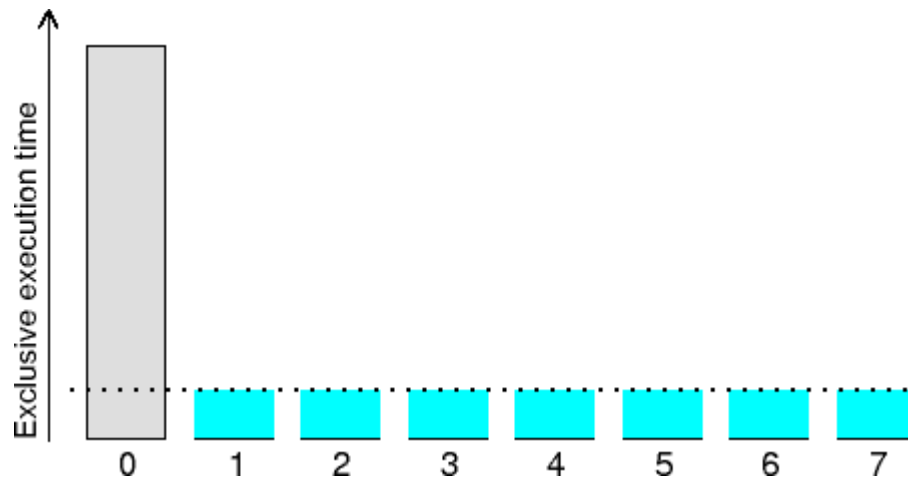
- Identifies processes/threads were exclusive execution time for the call-path was below average



- Identifies call-paths not executed by a subset of processes/threads



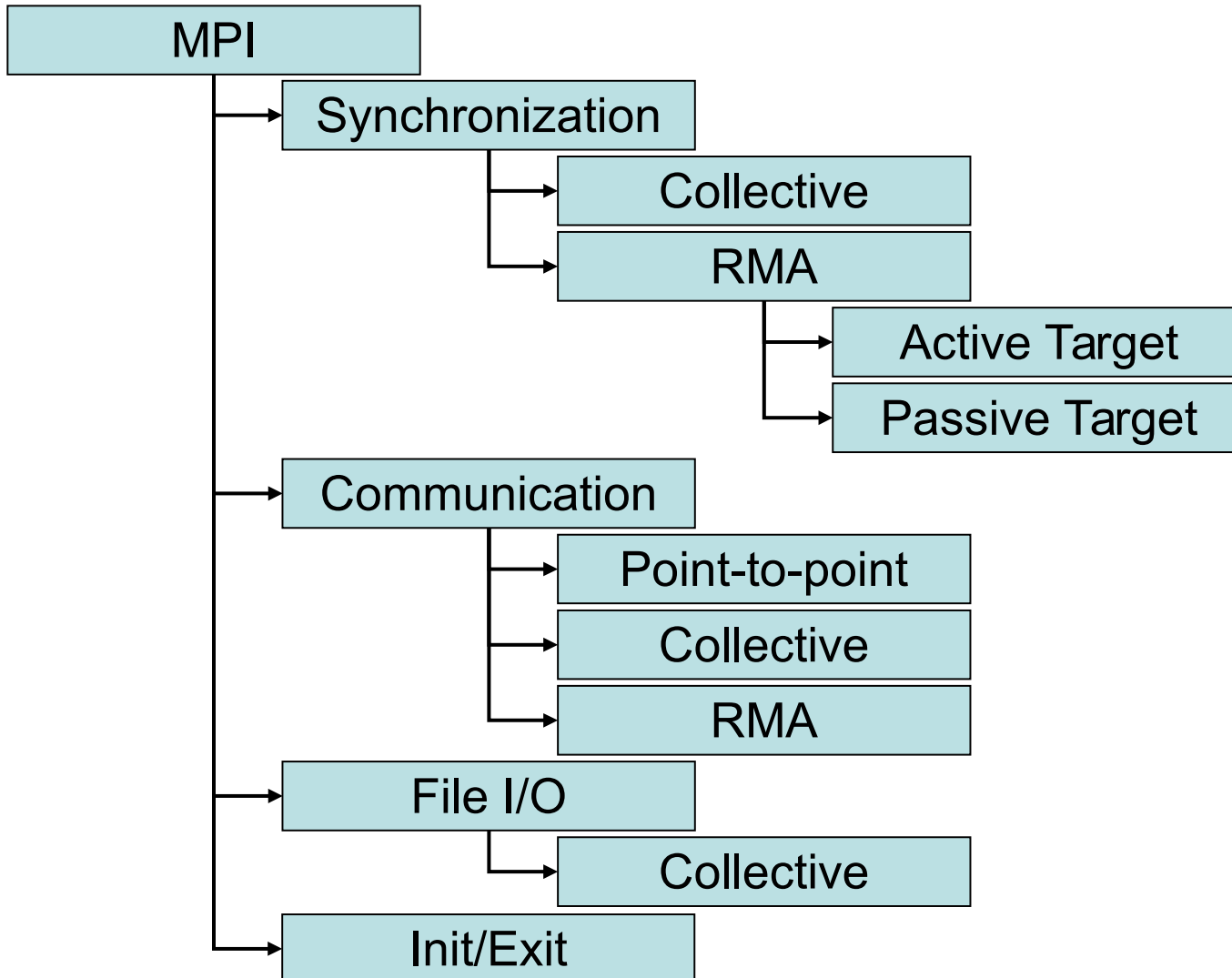
- Identifies call-paths not executed by all but a single process/thread



# VI-HPS



## MPI-related metrics





MPI

Time spent in pre-instrumented MPI functions

Synchronization

Time spent in calls to `MPI_Barrier` or Remote Memory Access synchronization calls

Communication

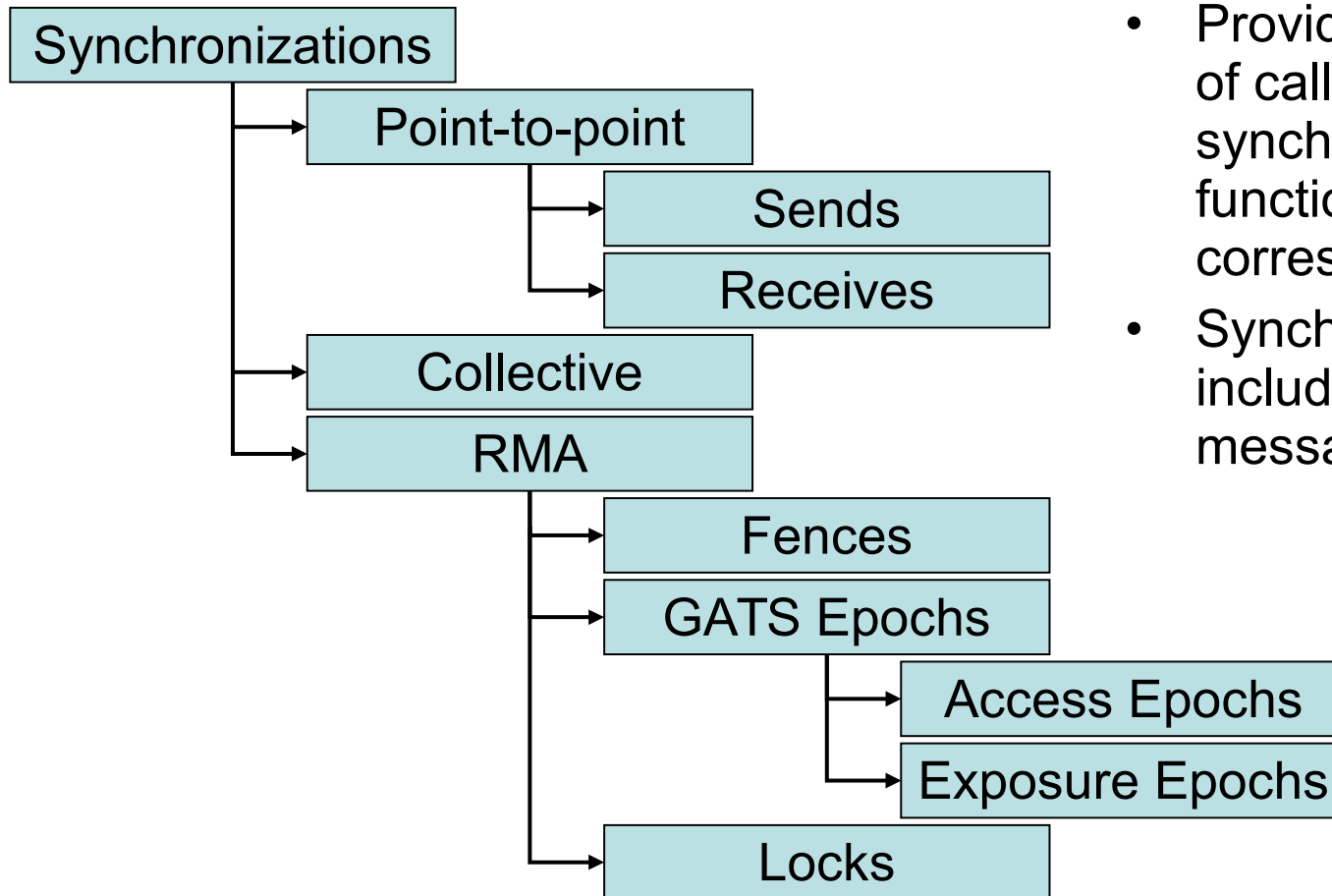
Time spent in MPI communication calls, subdivided into collective, point-to-point and RMA

File I/O

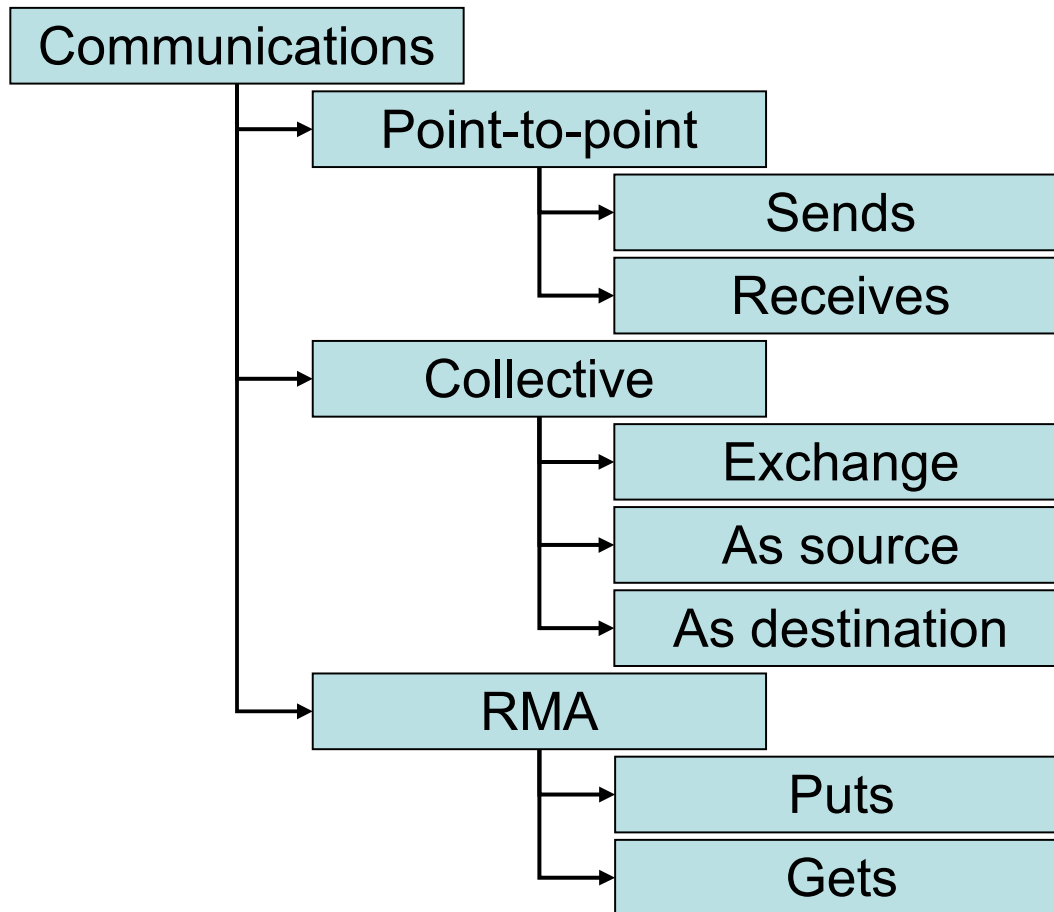
Time spent in MPI file I/O functions, with specialization for collective I/O calls

Init/Exit

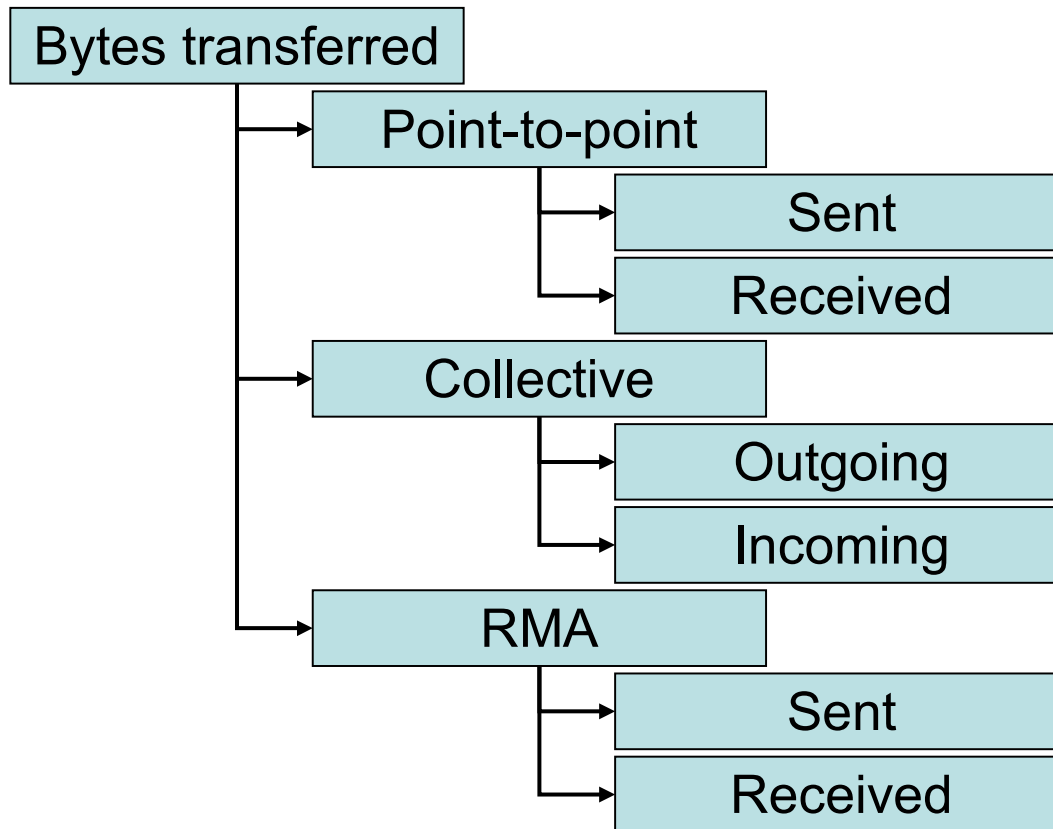
Time spent in `MPI_Init` and `MPI_Finalize`



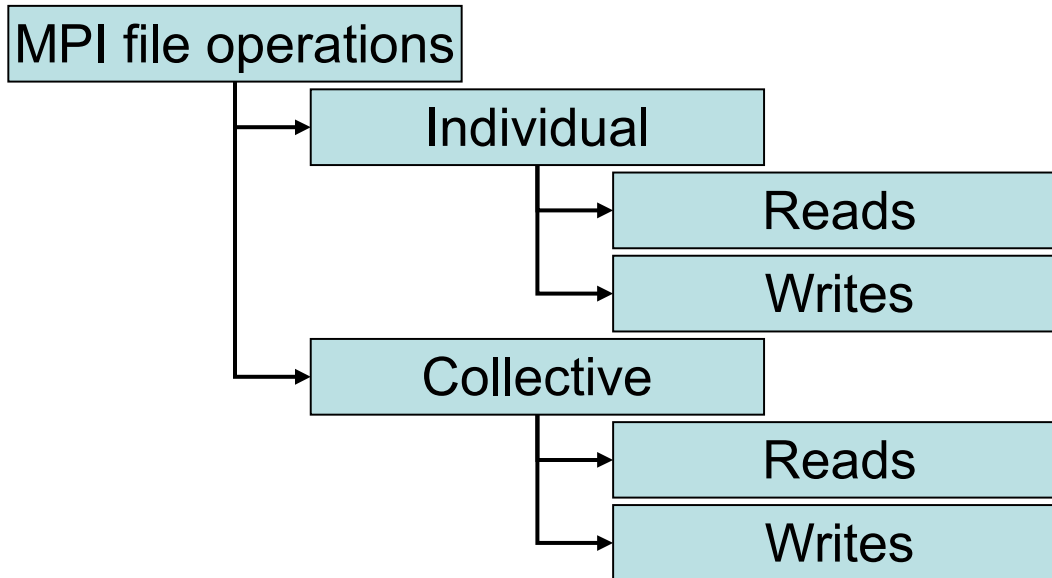
- Provides the number of calls to an MPI synchronization function of the corresponding class
- Synchronizations include zero-sized message transfers!



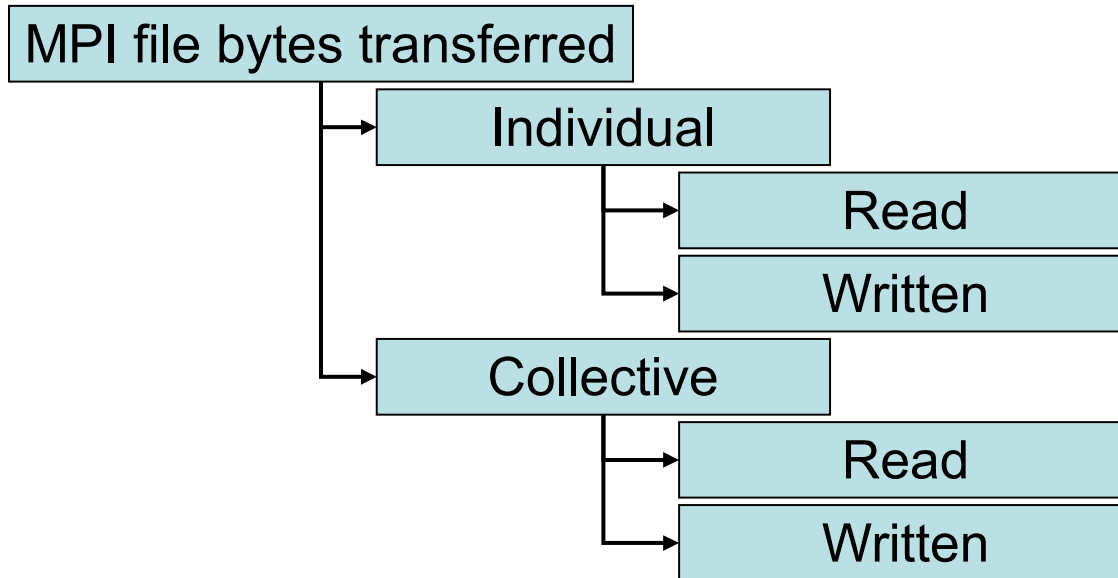
- Provides the number of calls to an MPI communication function of the corresponding class
- Zero-sized message transfers are considered synchronization!



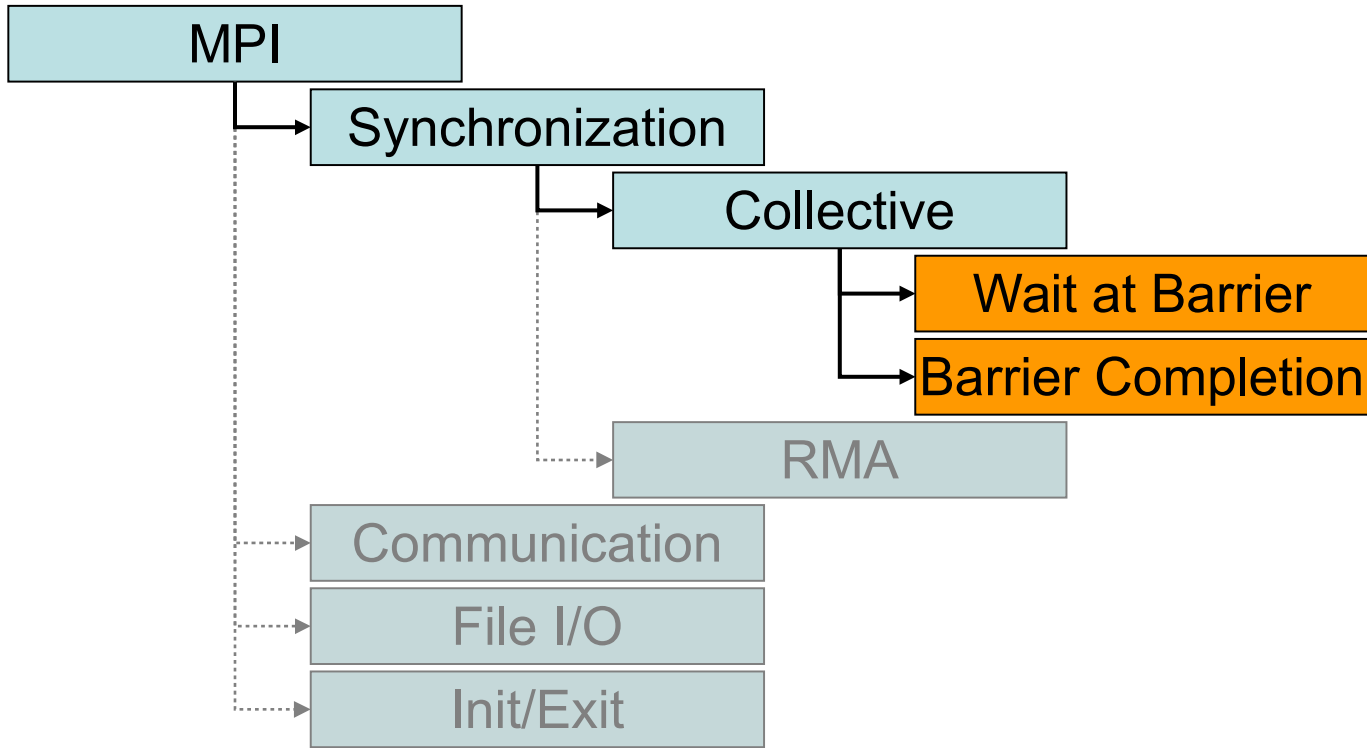
- Provides the number of bytes transferred by an MPI communication function of the corresponding class

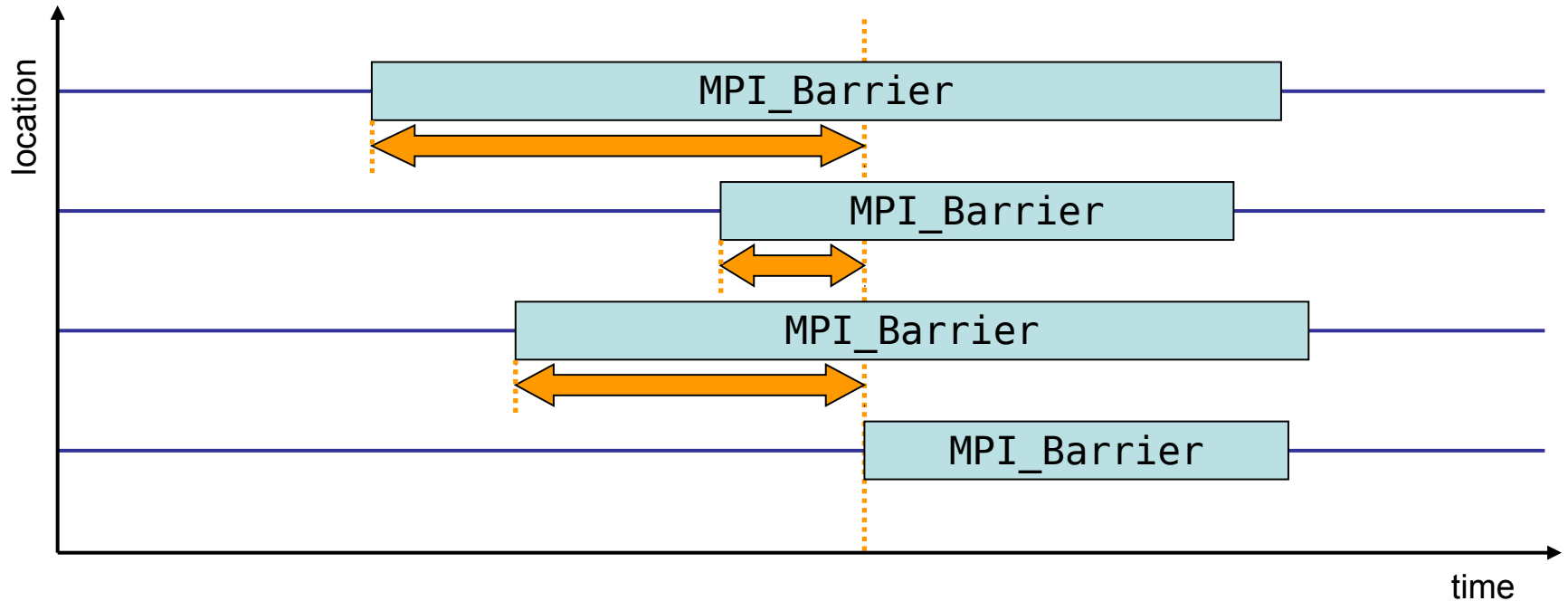


- Provides the number of calls to MPI file I/O functions of the corresponding class



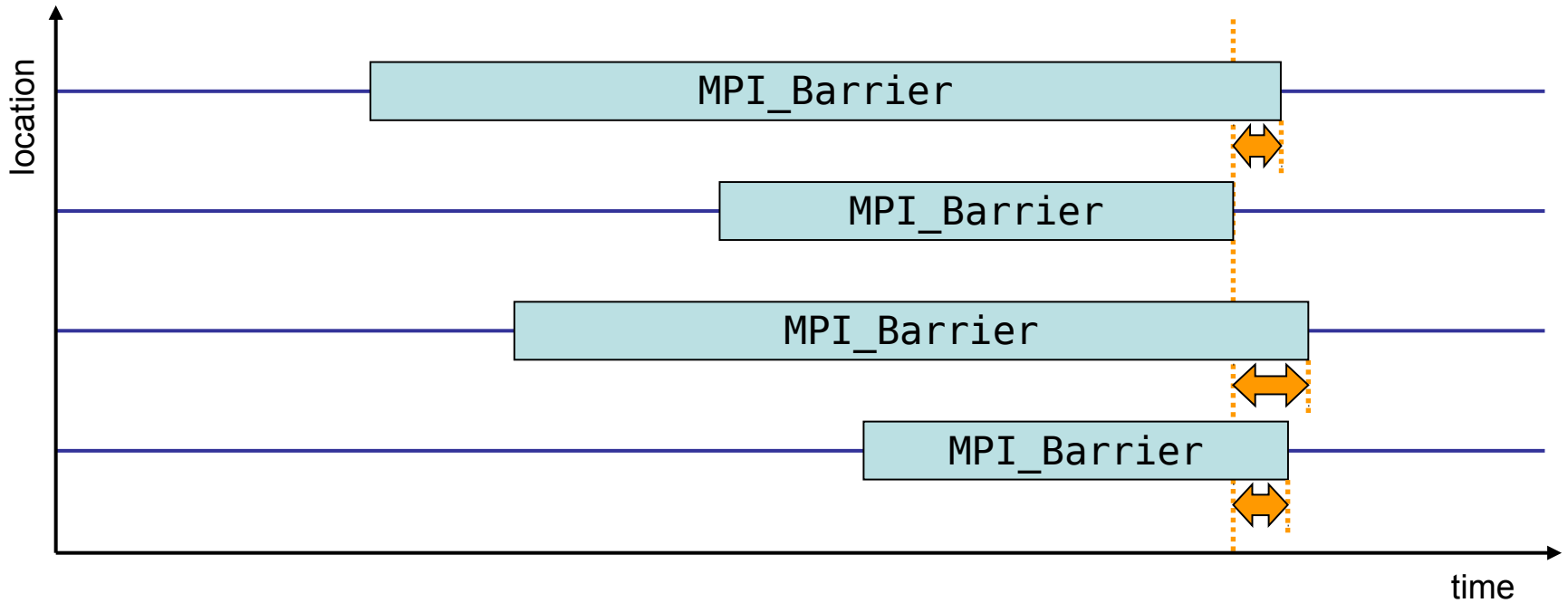
- Provides the number of bytes for MPI file I/O functions of the corresponding class



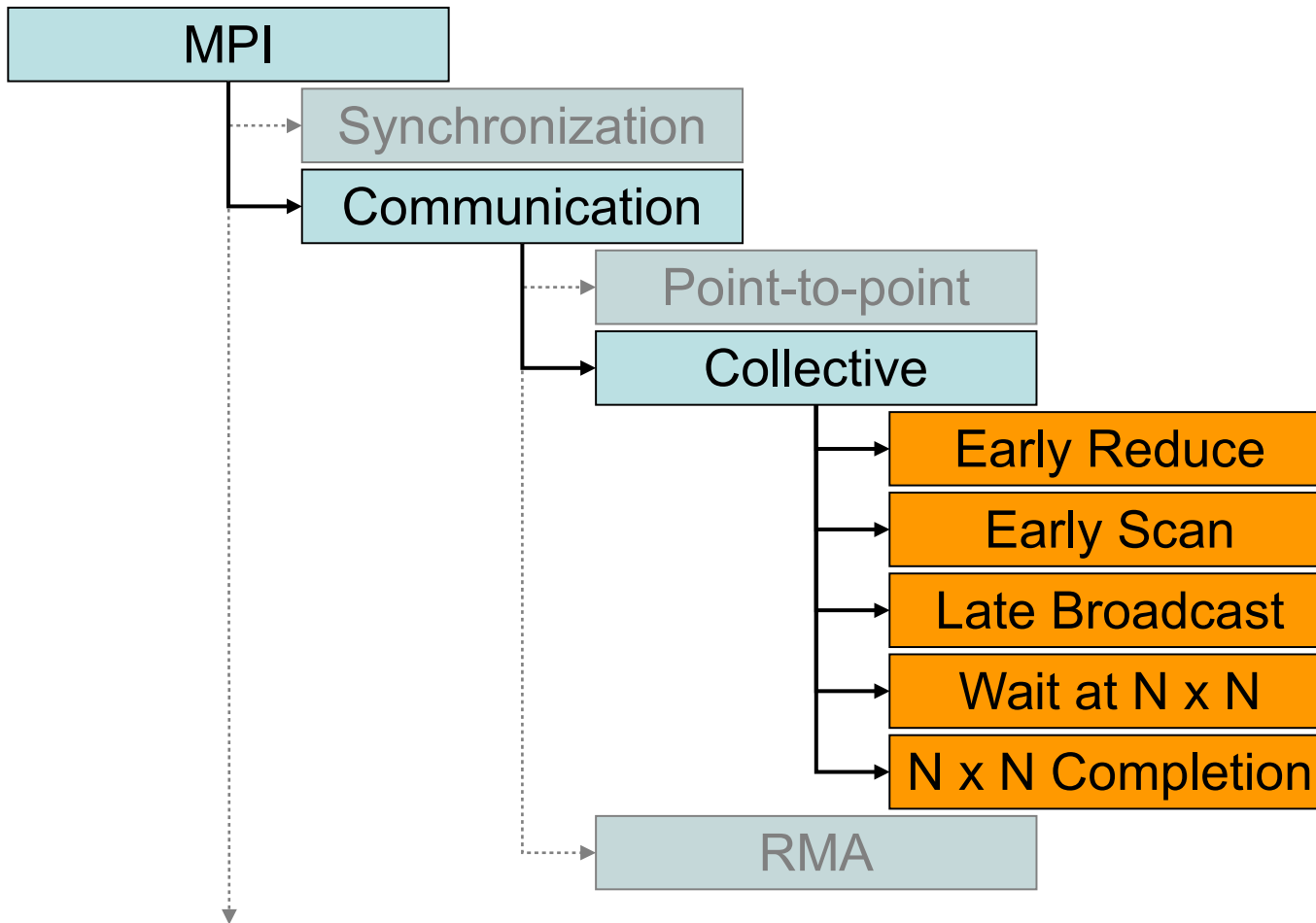


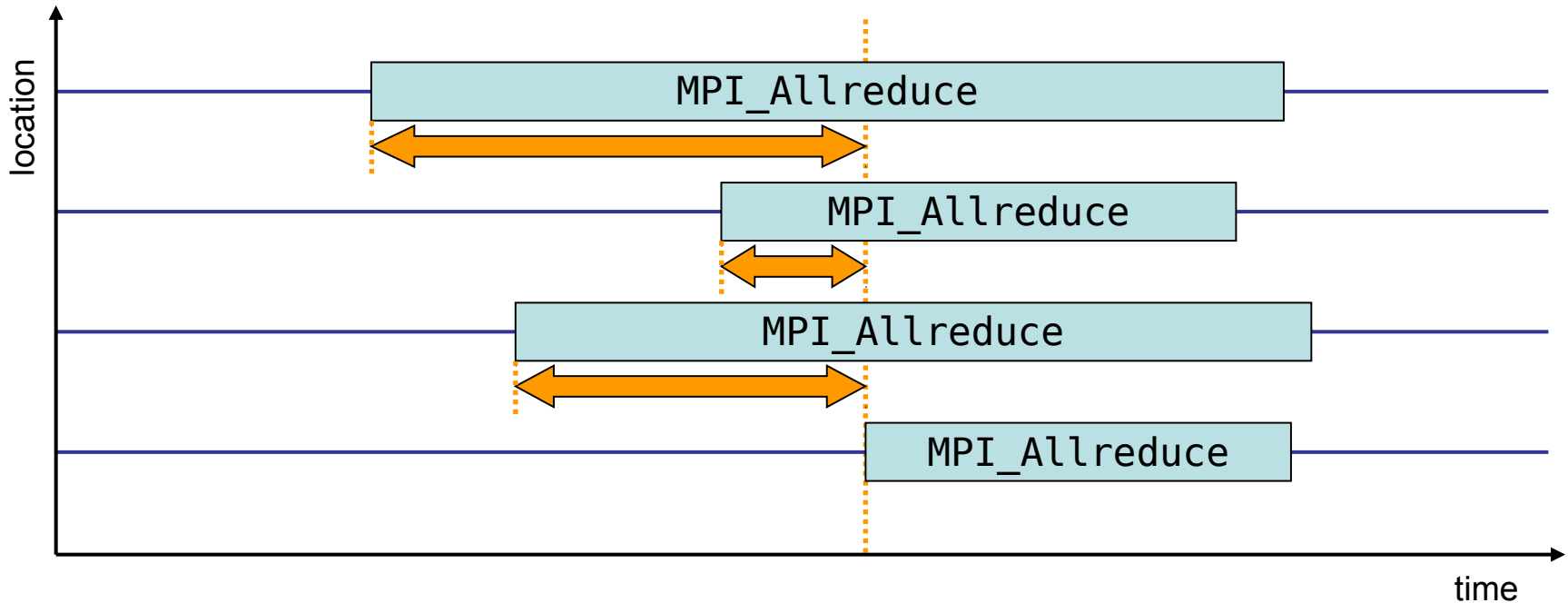
- Time spent waiting in front of a barrier call until the last process reaches the barrier operation
- Applies to: MPI\_Barrier



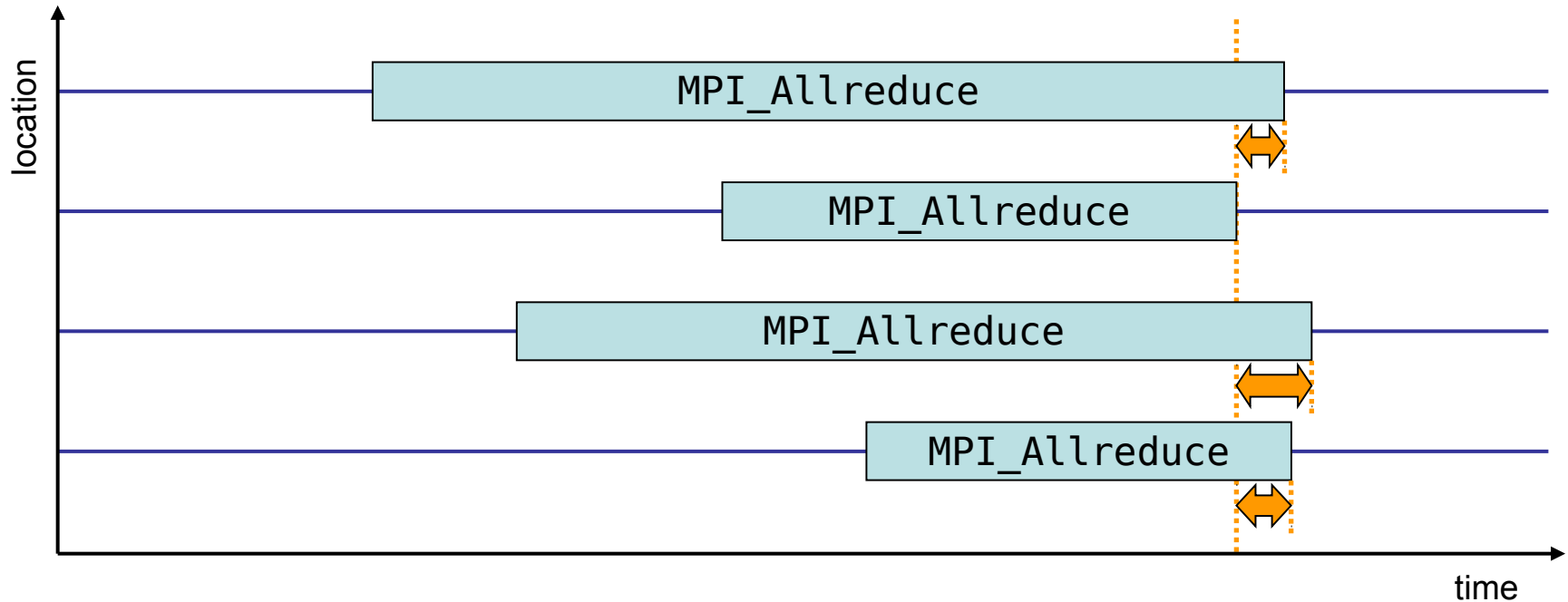


- Time spent in barrier after the first process has left the operation
- Applies to: MPI\_Barrier

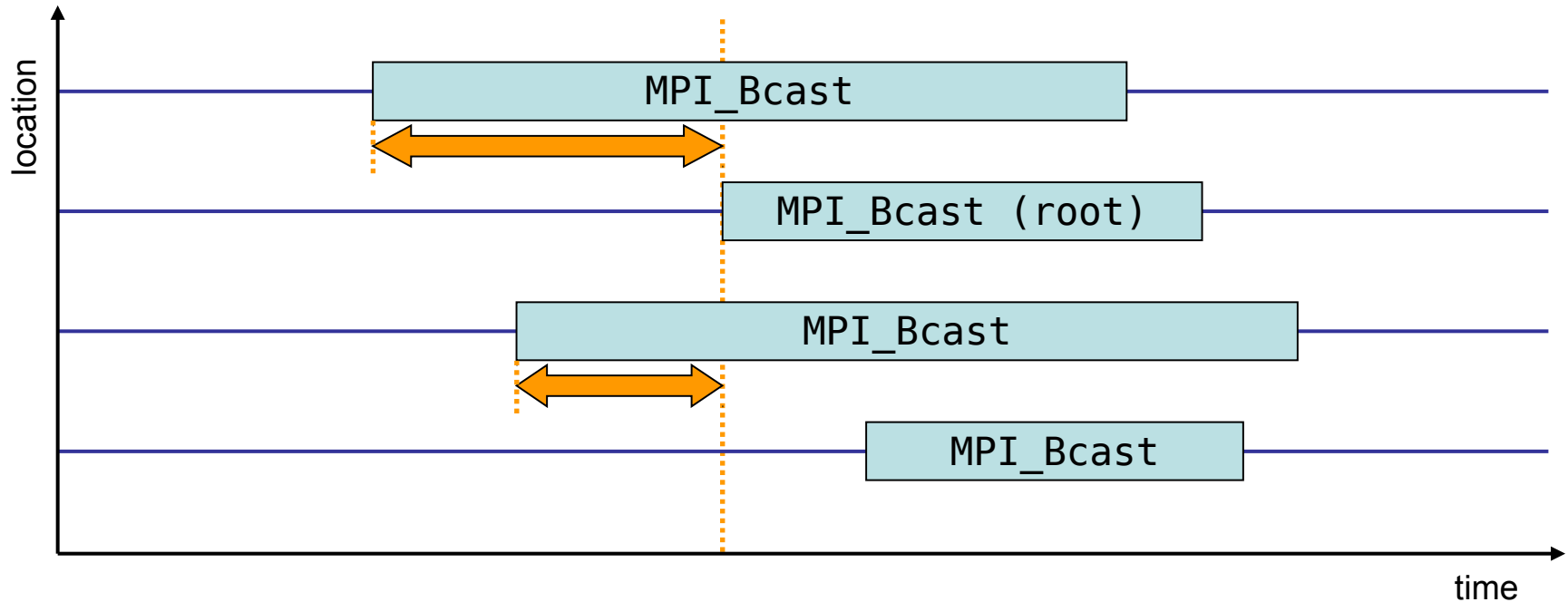




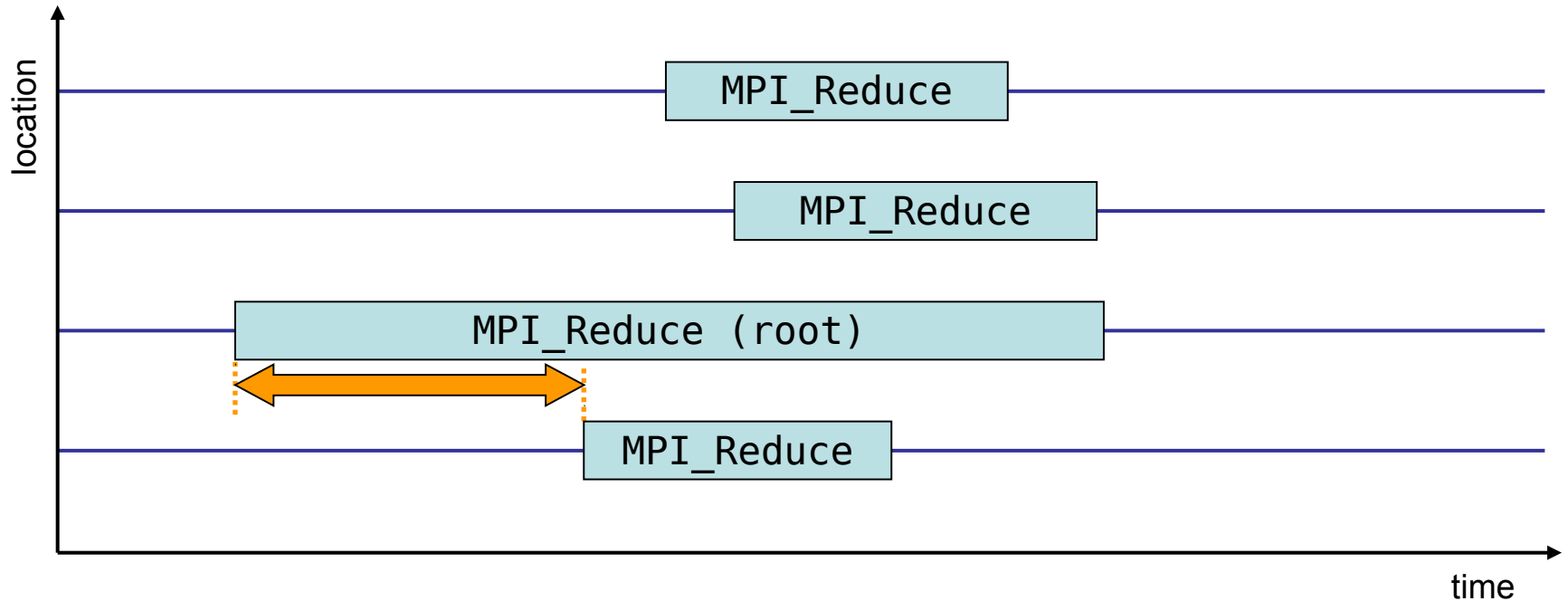
- Time spent waiting in front of a synchronizing collective operation call until the last process reaches the operation
- Applies to:  
`MPI_Allgather`, `MPI_Allgatherv`,  
`MPI_Allreduce`, `MPI_Alltoall`,  
`MPI_Reduce_scatter`, `MPI_Reduce_scatter_block`



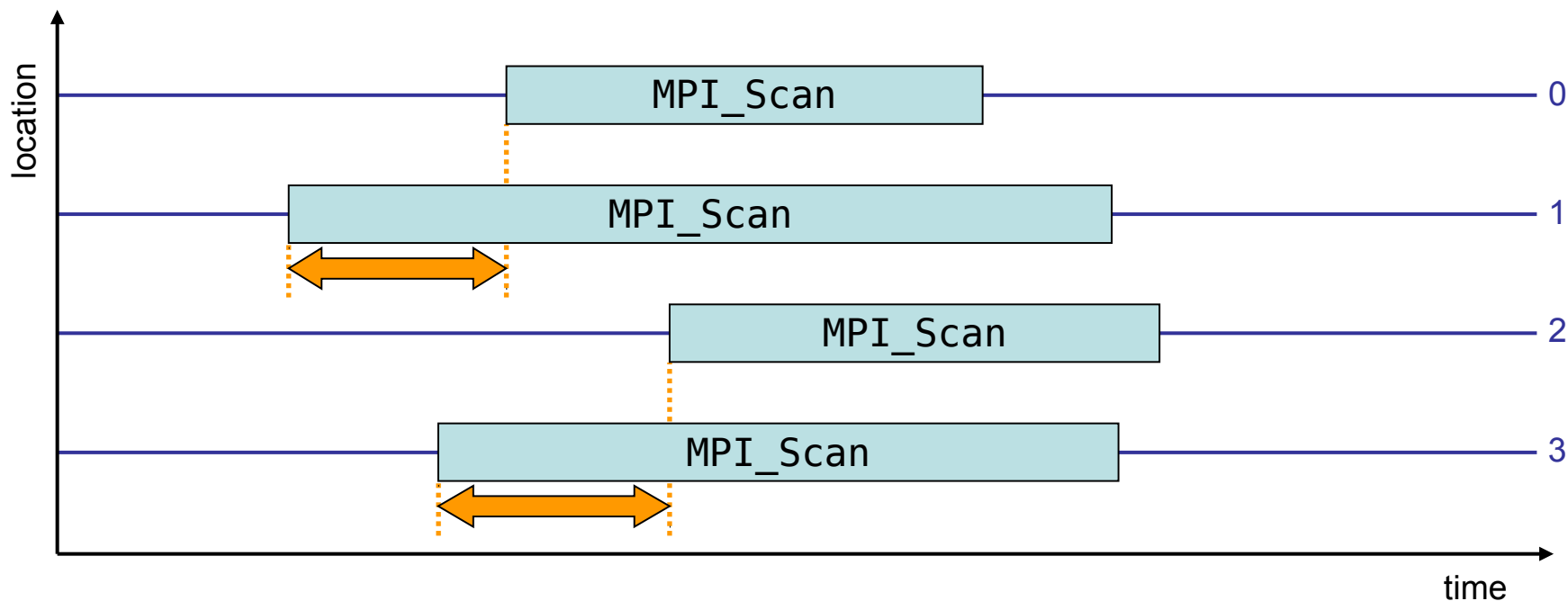
- Time spent in synchronizing collective operations after the first process has left the operation
- Applies to:  
`MPI_Allgather`, `MPI_Allgatherv`,  
`MPI_Allreduce`, `MPI_Alltoall`,  
`MPI_Reduce_scatter`, `MPI_Reduce_scatter_block`



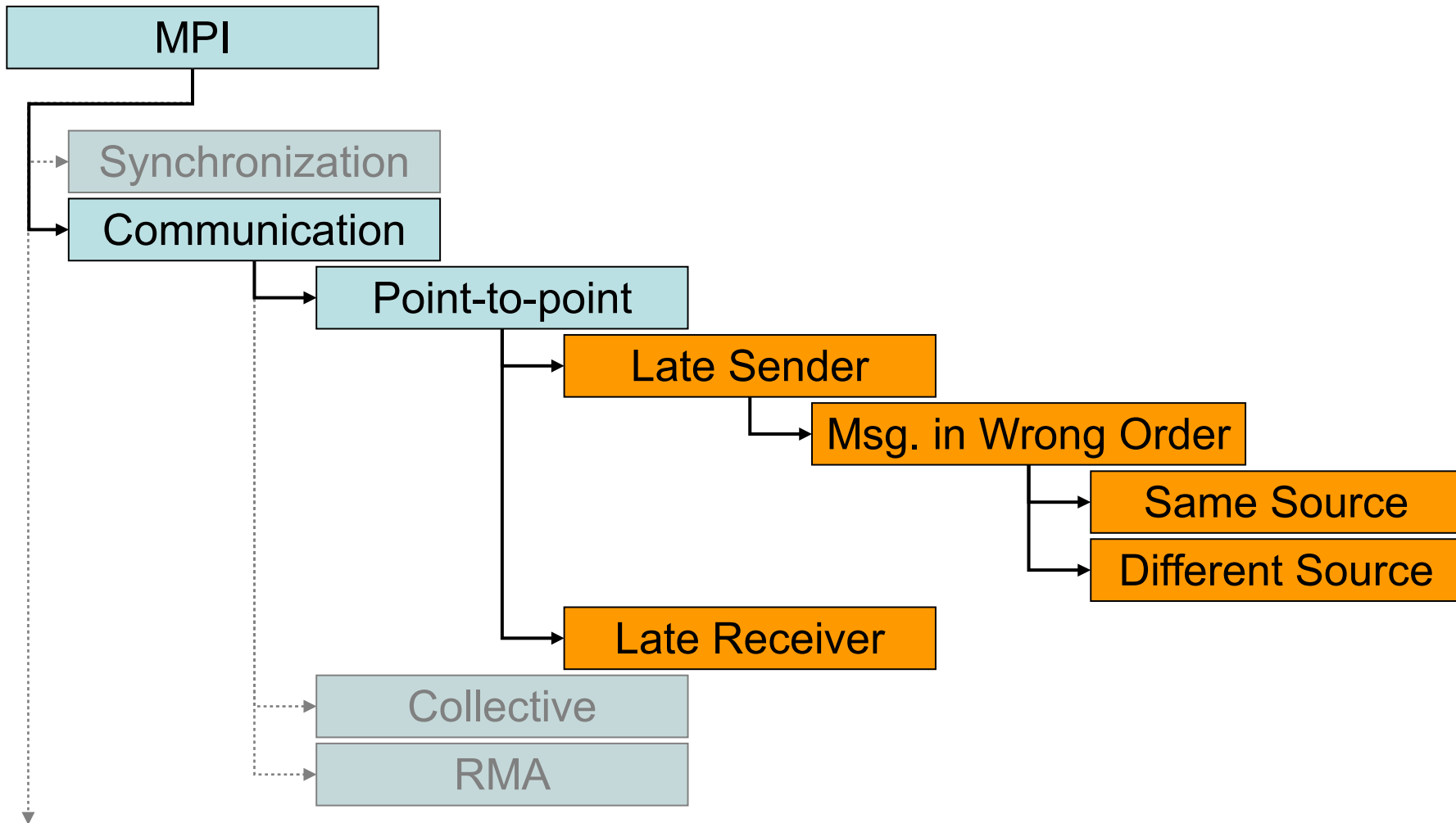
- Waiting times if the destination processes of a collective 1-to-N communication operation enter the operation earlier than the source process (root)
- Applies to: MPI\_Bcast, MPI\_Scatter, MPI\_Scatterv



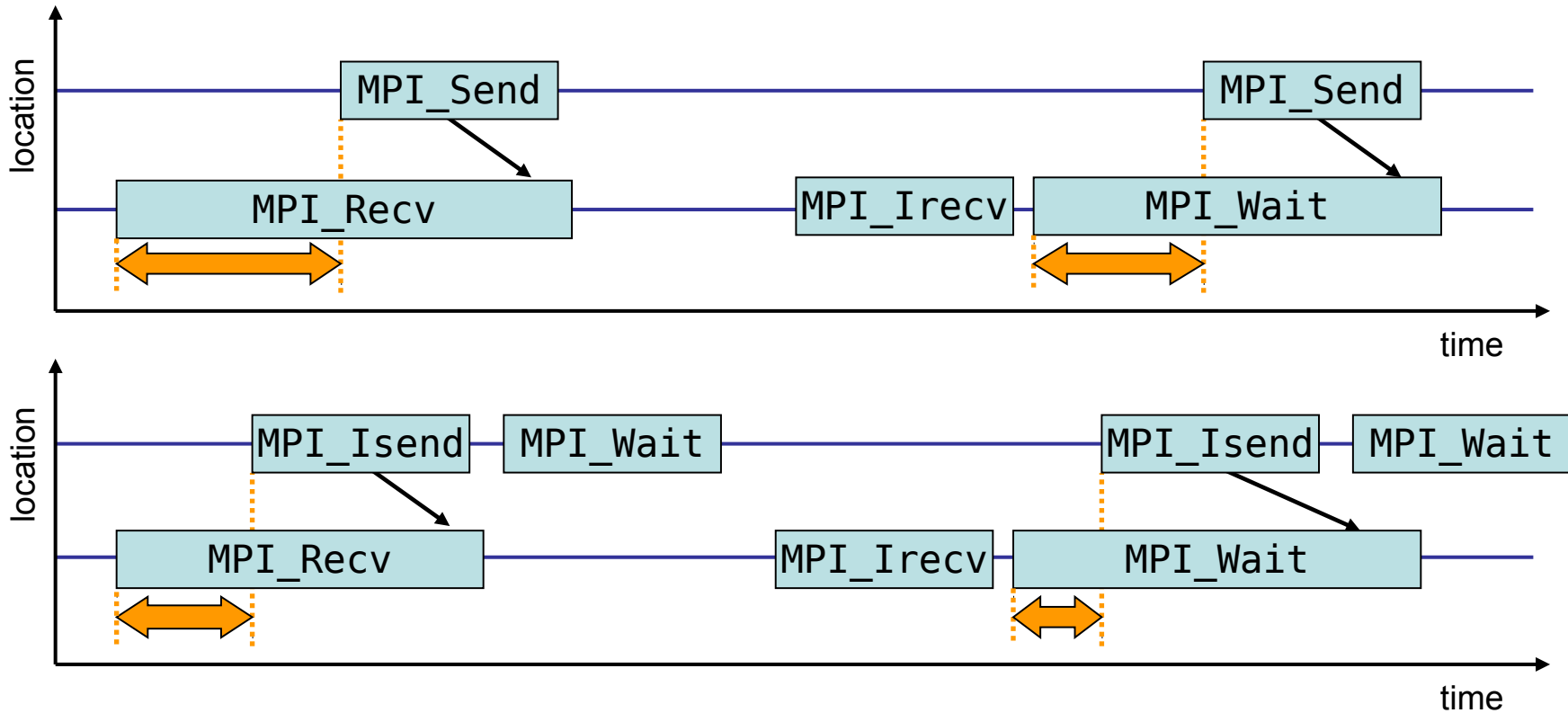
- Waiting time if the destination process (root) of a collective N-to-1 communication operation enters the operation earlier than its sending counterparts
- Applies to: MPI\_Reduce, MPI\_Gather, MPI\_Gatherv



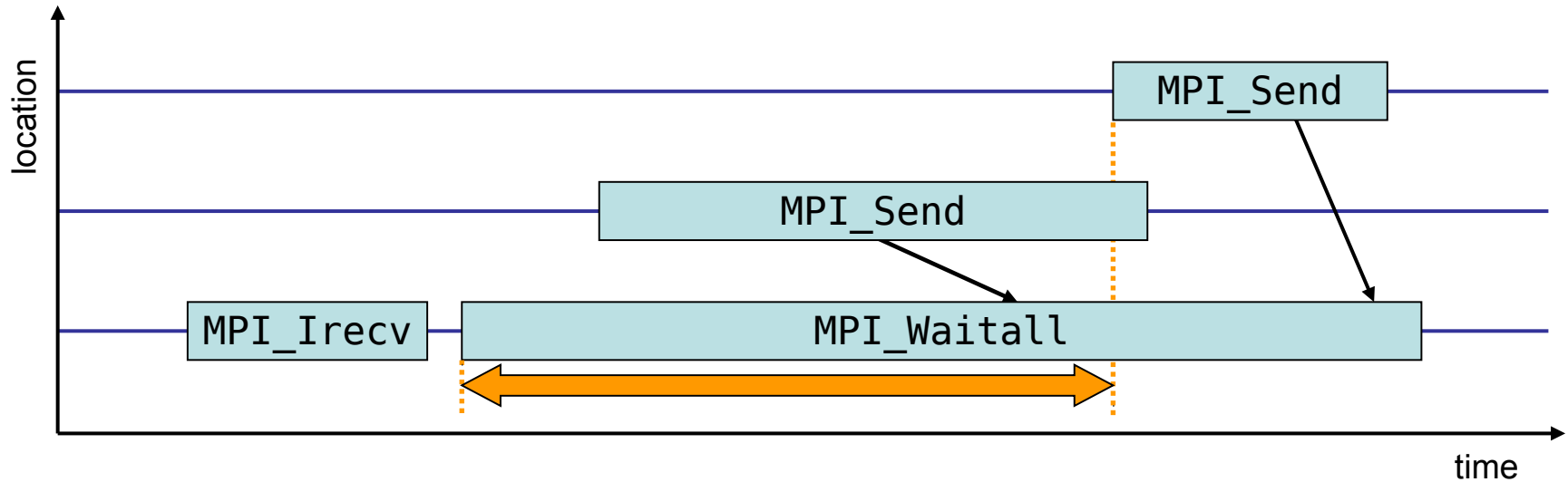
- Waiting time if process  $n$  enters a prefix reduction operation earlier than its sending counterparts (i.e., ranks  $0..n-1$ )
- Applies to: MPI\_Scan, MPI\_Exscan



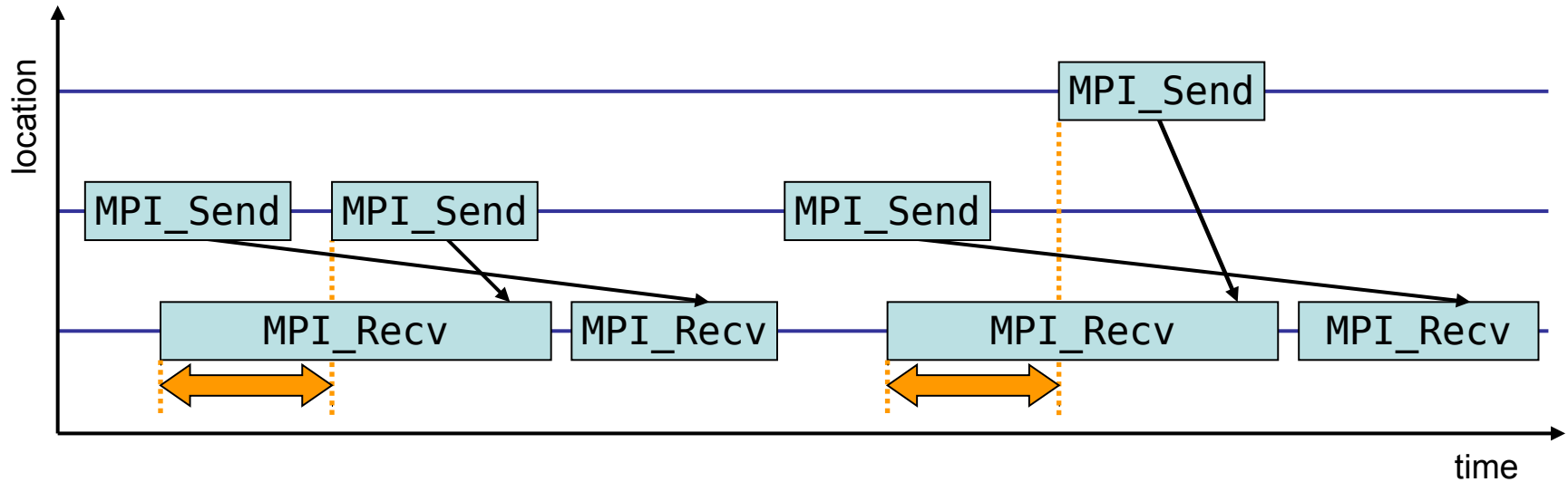




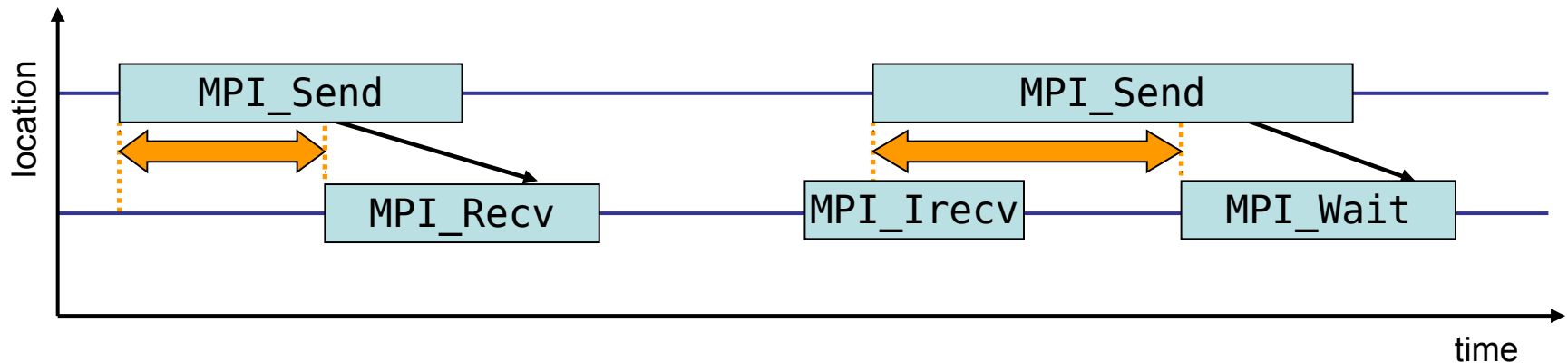
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send operation
- Applies to blocking as well as non-blocking communication



- While waiting for several messages, the maximum waiting time is accounted
- Applies to: `MPI_Waitall`, `MPI_Waitsome`

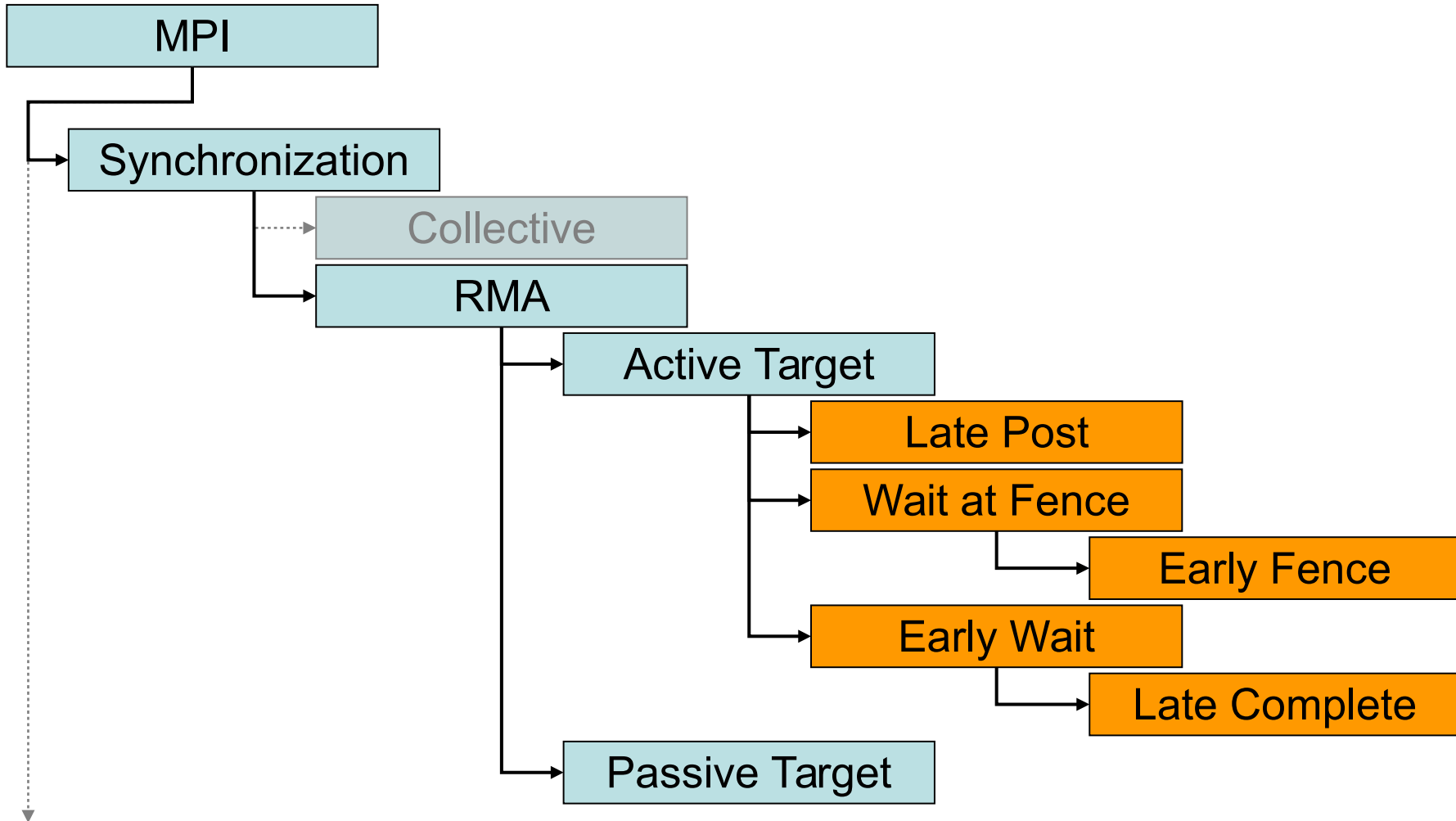


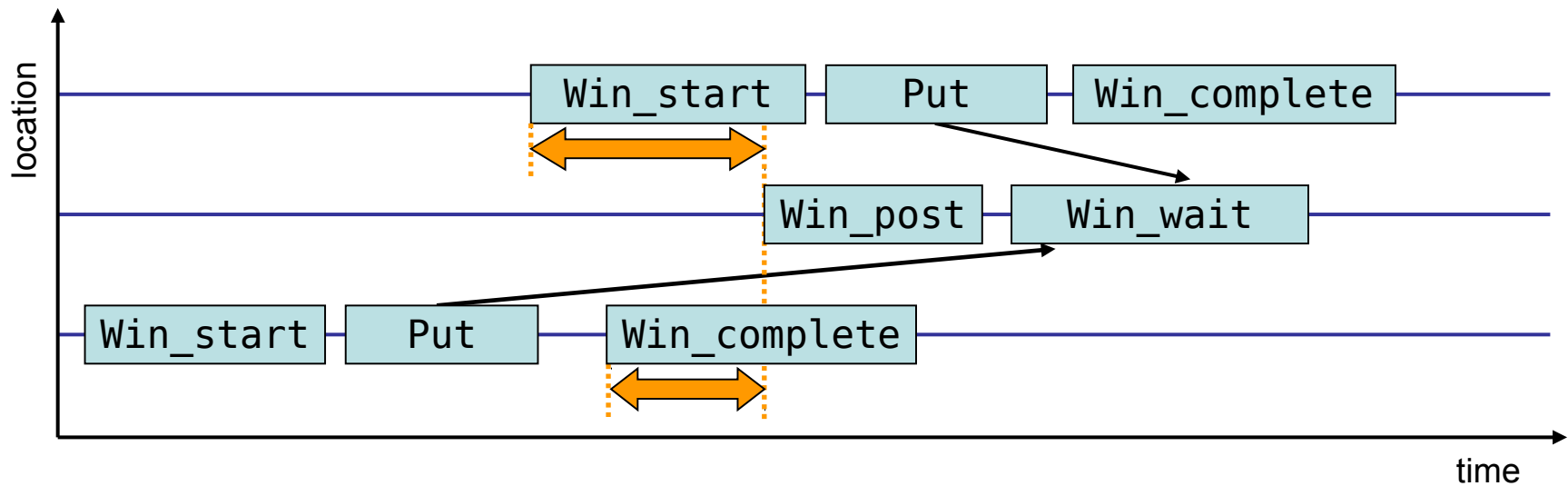
- Refers to Late Sender situations which are caused by messages received in wrong order
- Comes in two flavours:
  - Messages sent from same source location
  - Messages sent from different source locations



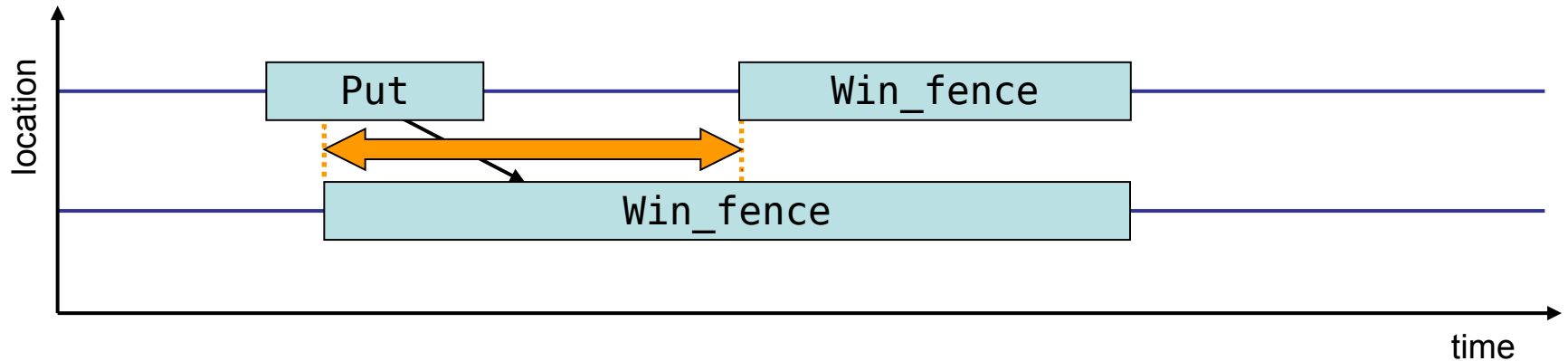
- Waiting time caused by a blocking send operation posted earlier than the corresponding receive operation
- Calculated by receiver but waiting time attributed to sender
- Does currently not apply to non-blocking sends

- The number of Late Sender / Late Receiver instances are also available
- They are divided into communications & synchronizations and shown in the corresponding hierarchies



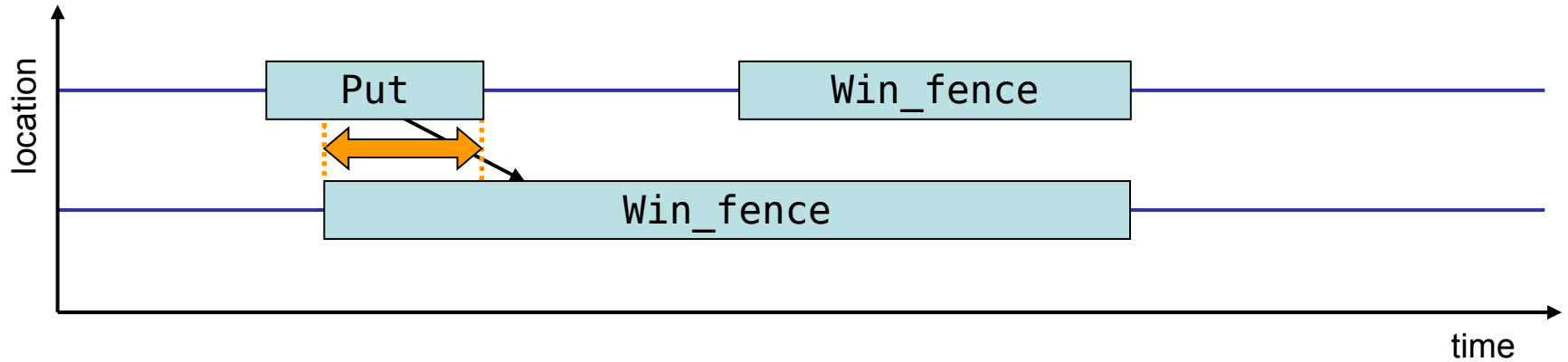


- `MPI_Win_start` (top) or `MPI_Win_complete` (bottom) wait until exposure epoch is opened by `MPI_Win_post`
- Which of the two calls blocks is implementation dependent

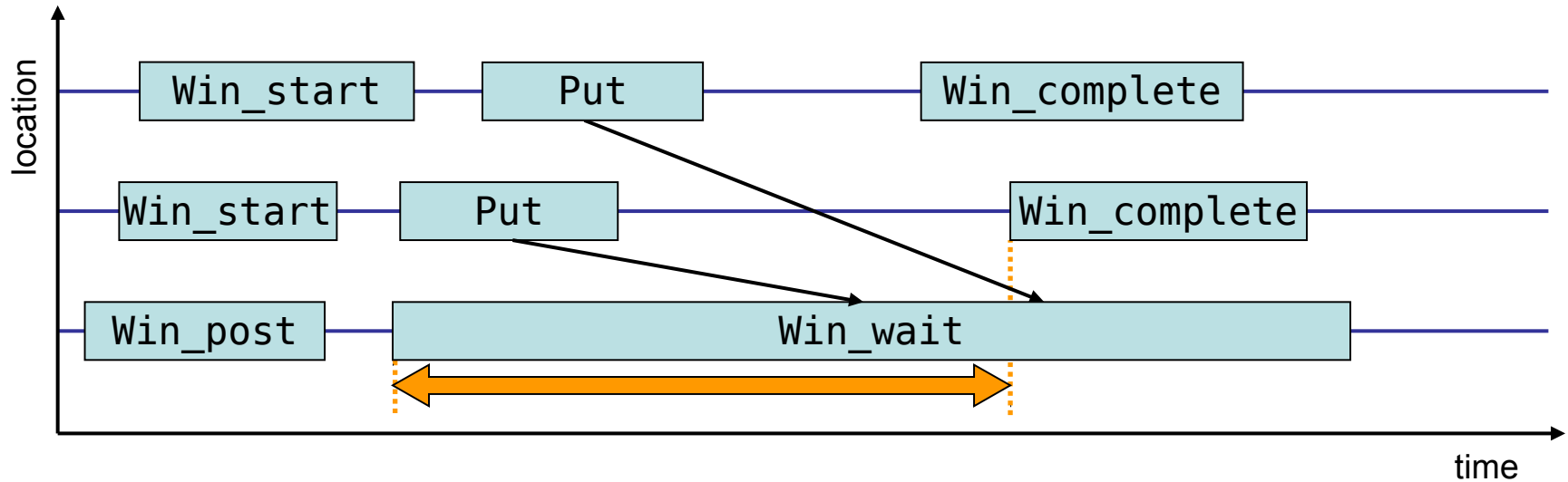


- Time spent waiting in front of a synchronizing MPI\_Win\_fence call until the last process reaches the fence operation
- Only triggered if at least one of the following conditions applies
  - Given assertion is 0
  - All fence calls overlap (heuristic)

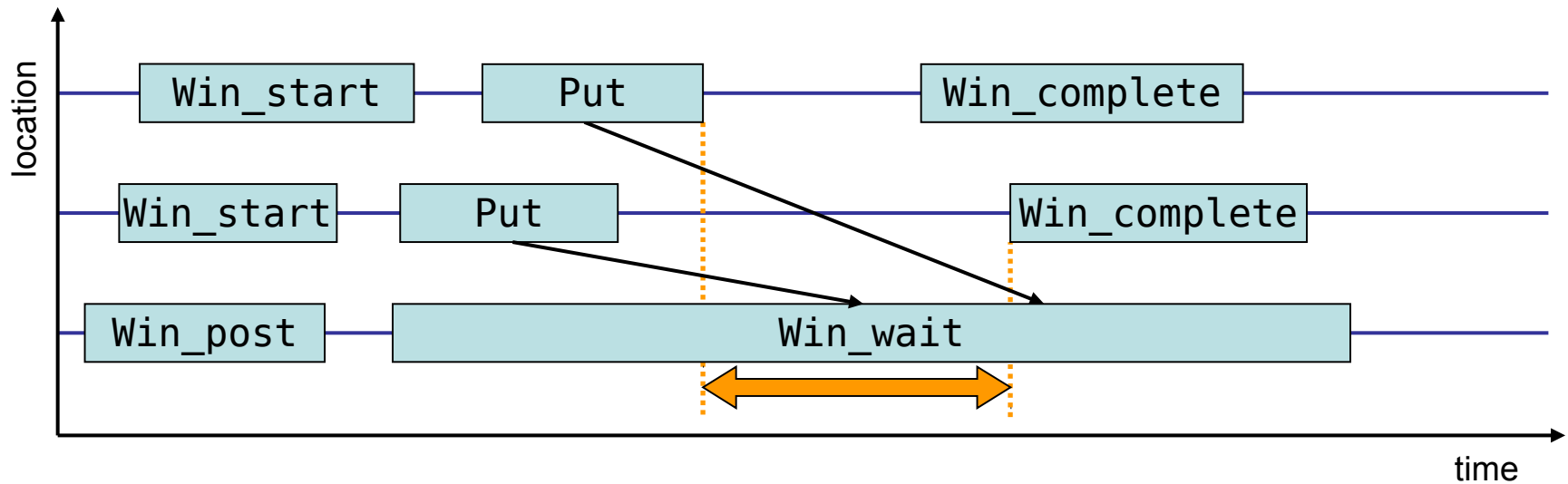




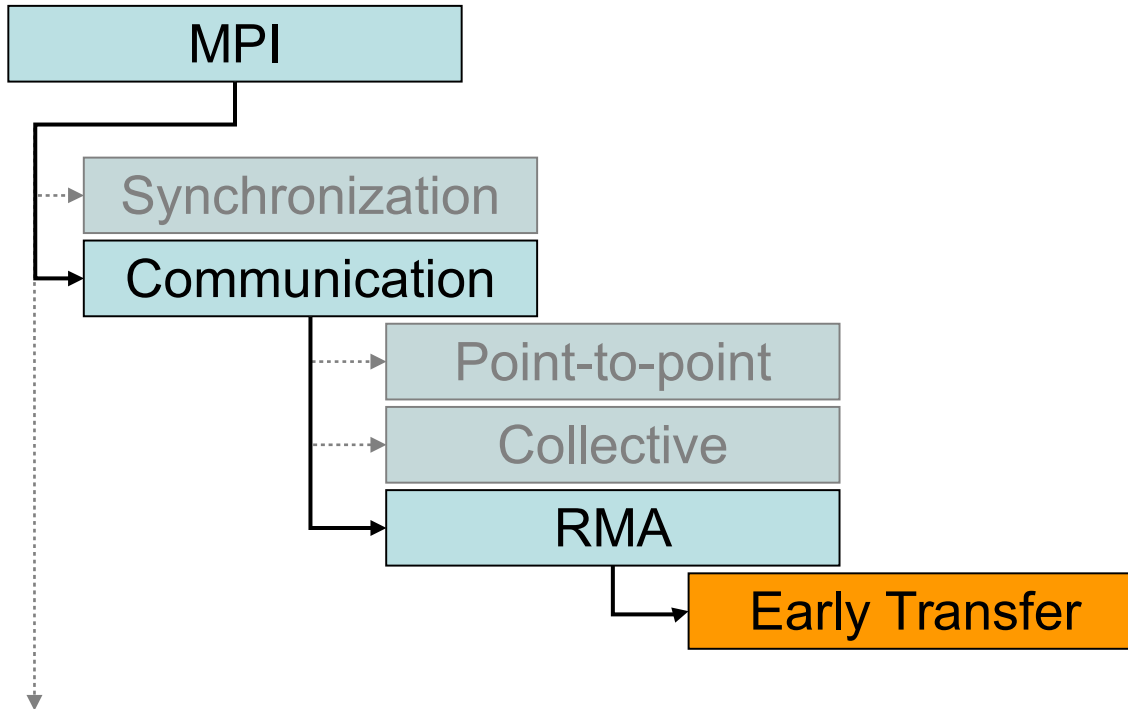
- Time spent waiting for exit of last RMA operation to target location
- Sub-pattern of Wait at Fence

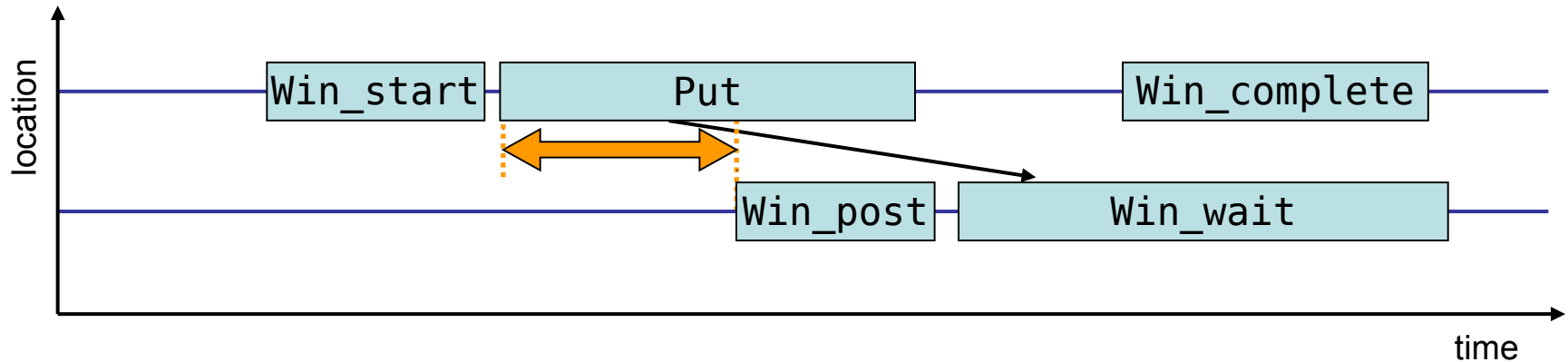


- Time spent in `MPI_Win_wait` until access epoch is closed by last `MPI_Win_complete`



- Waiting time due to unnecessary pause between last RMA operation to target and closing the access epoch by last MPI\_Win\_complete
- Sub-pattern of Early Wait



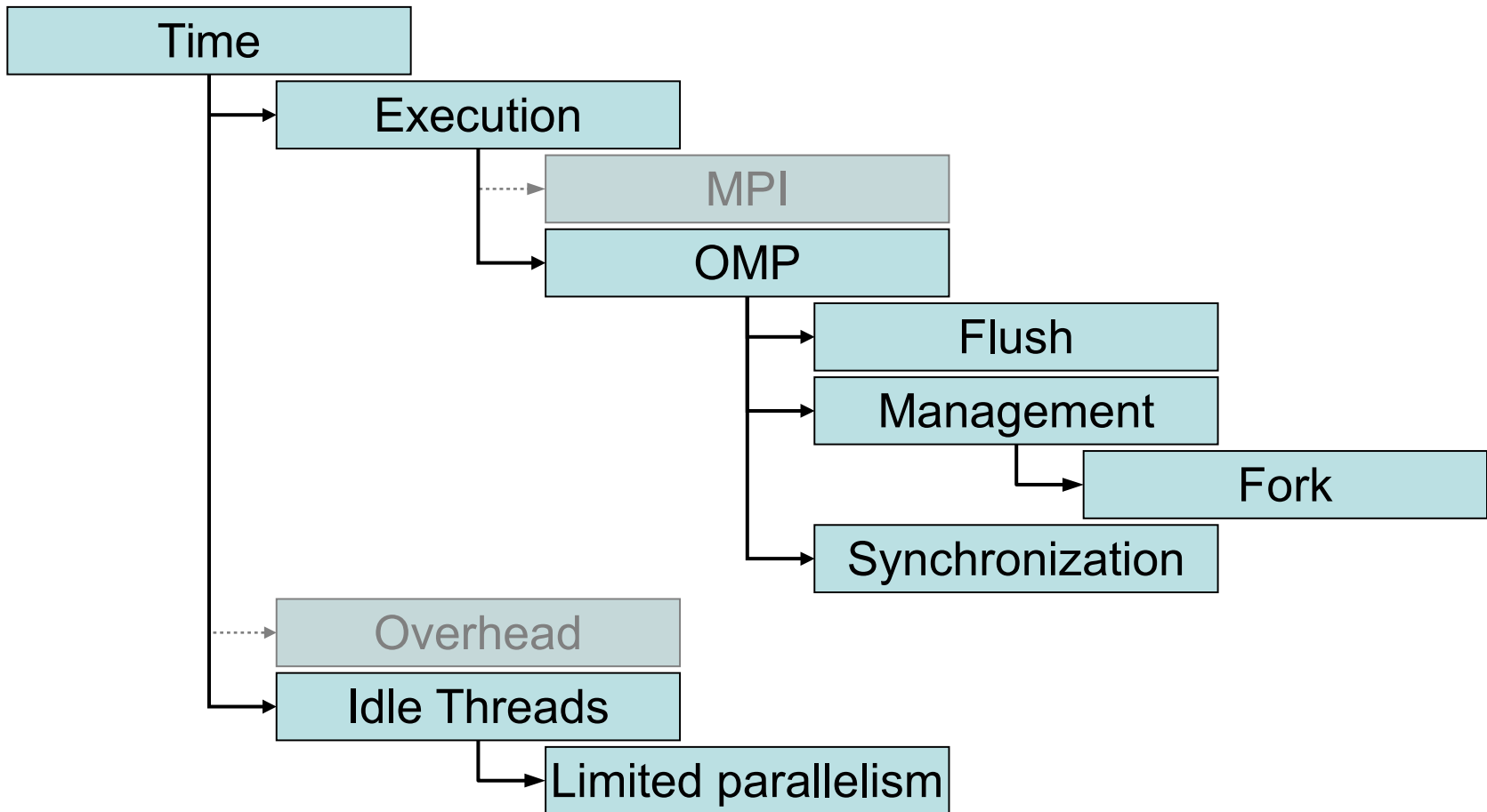


- Time spent waiting in RMA operation on origin(s) started before exposure epoch was opened on target

# VI-HPS



## OpenMP-related metrics



OMP

Time spent for OpenMP-related tasks

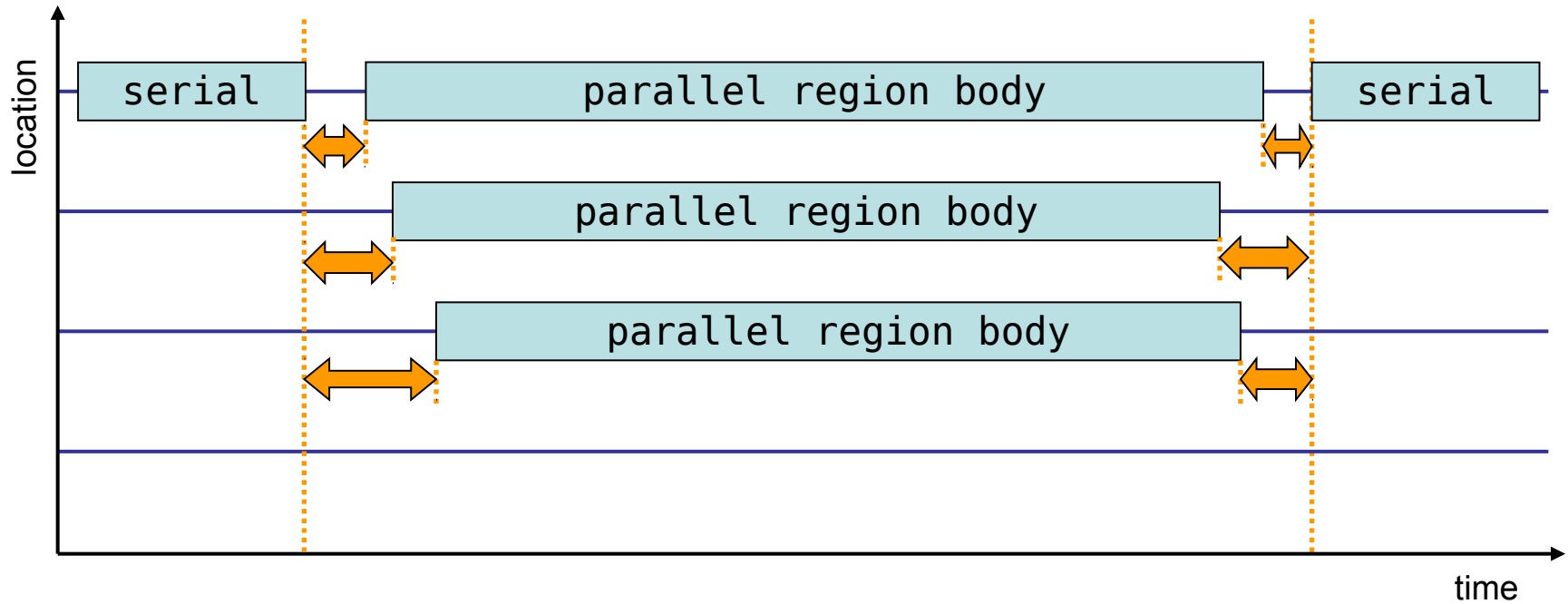
Flush

Time spent in OpenMP flush directives

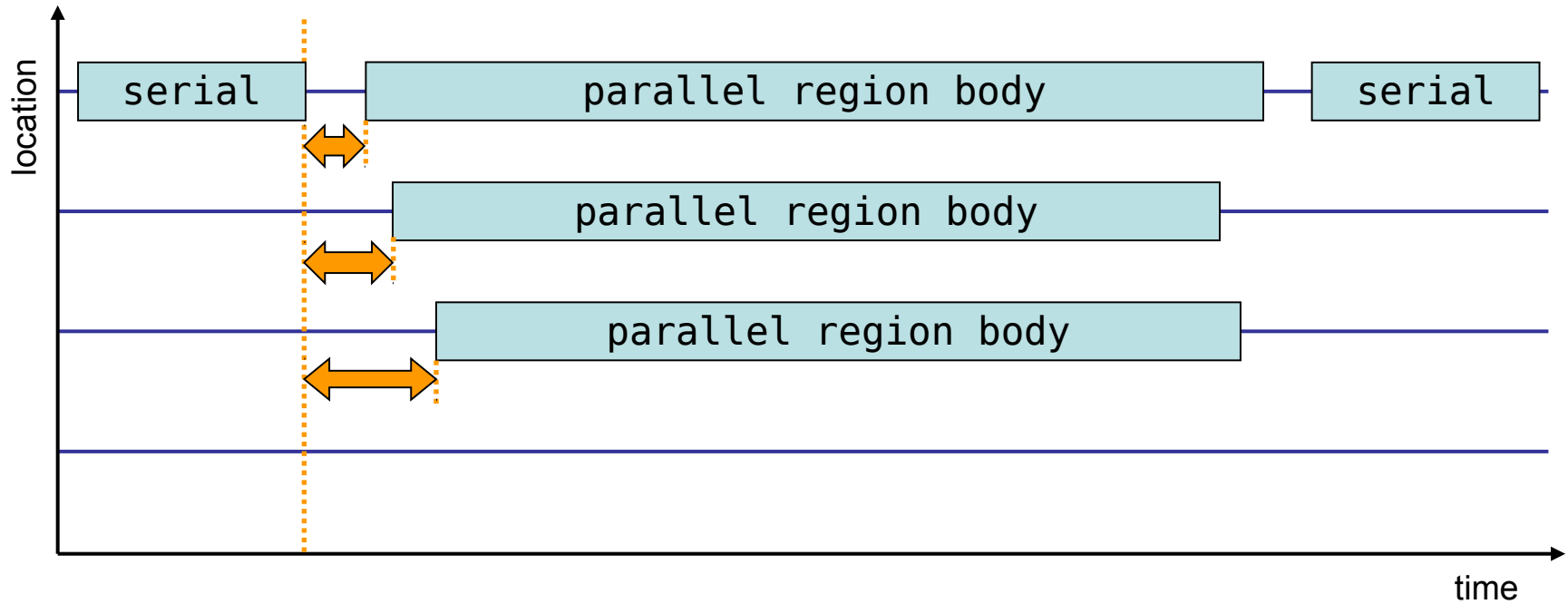
Synchronization

Time spent to synchronize OpenMP threads

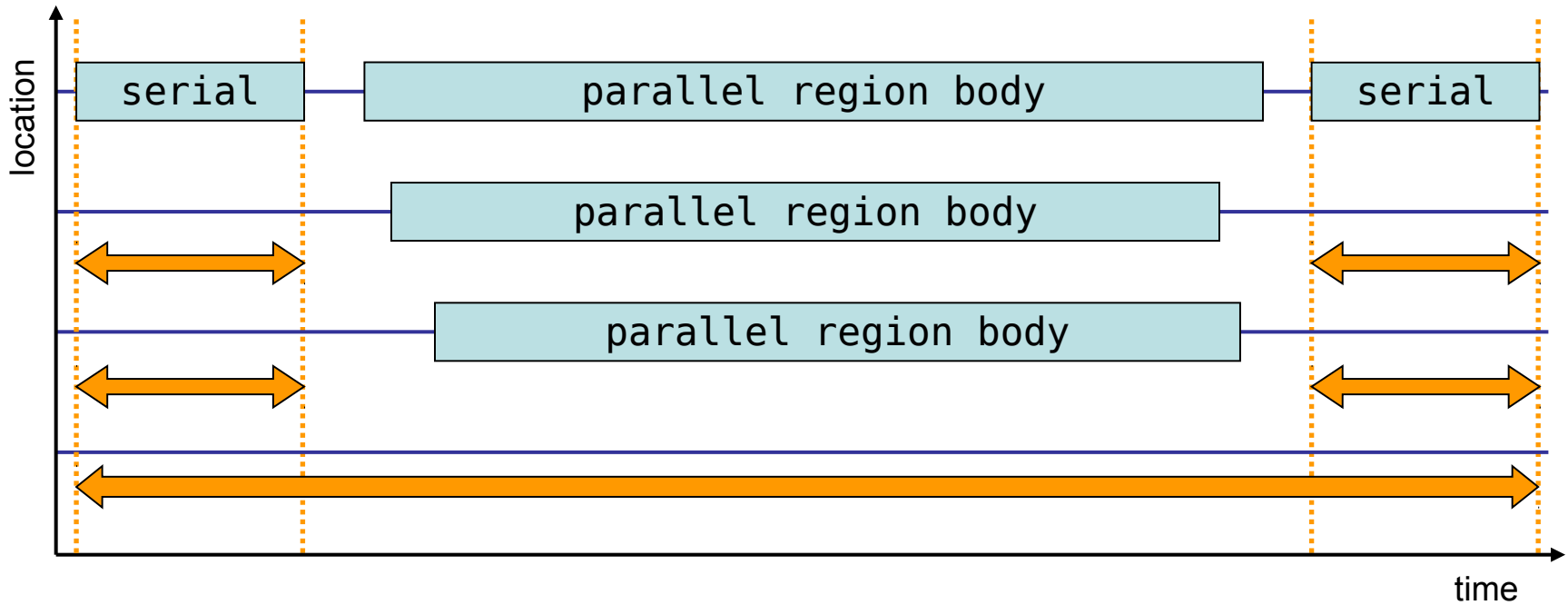




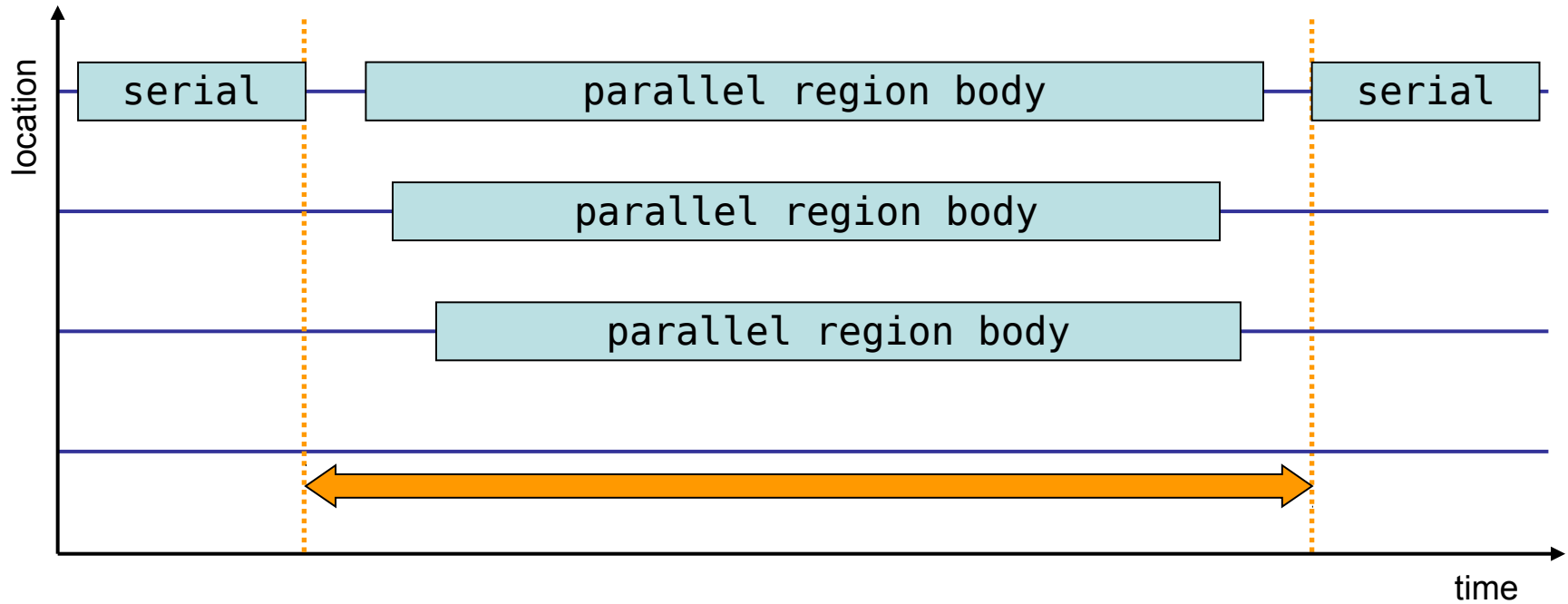
- Time spent on master thread for creating/destroying OpenMP thread teams



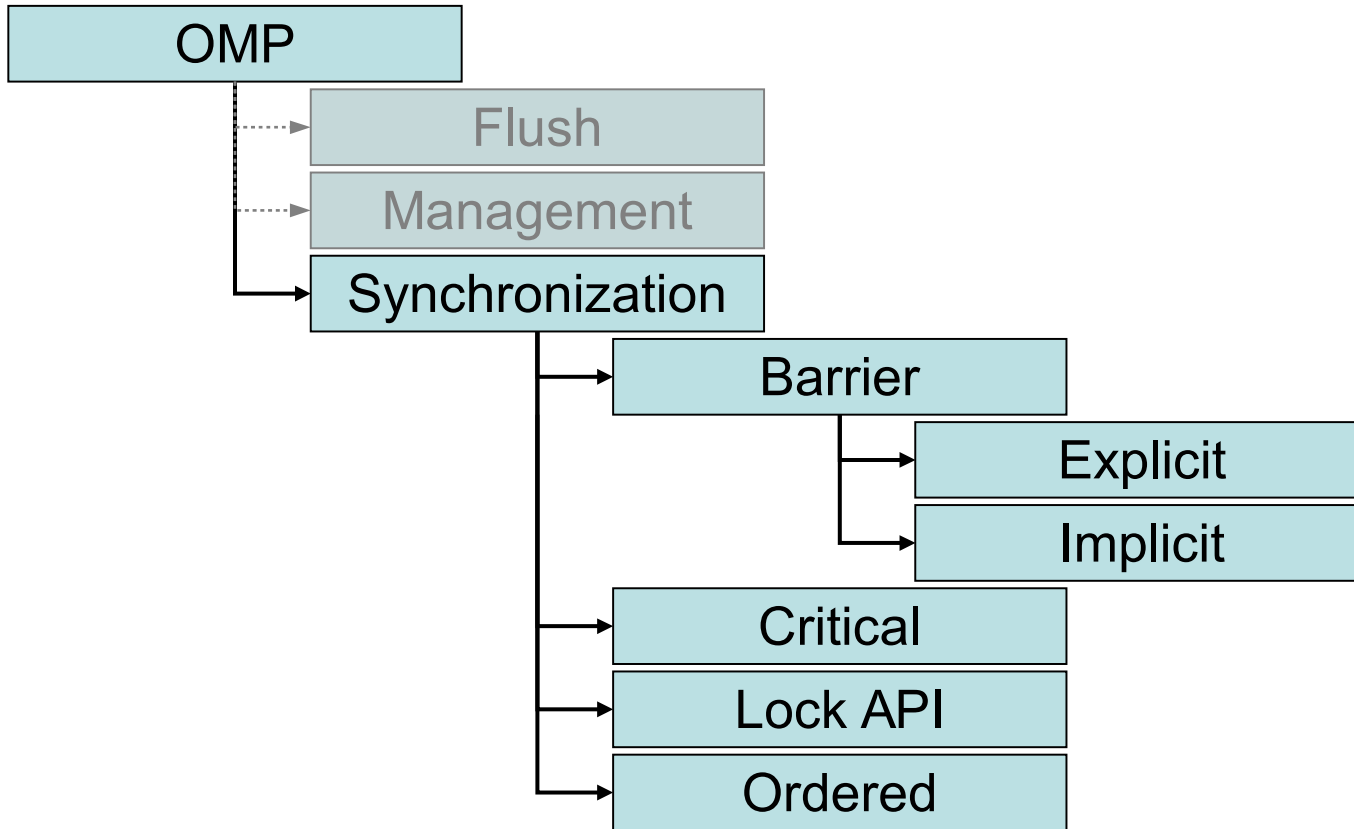
- Time spent on master threads for creating OpenMP thread teams



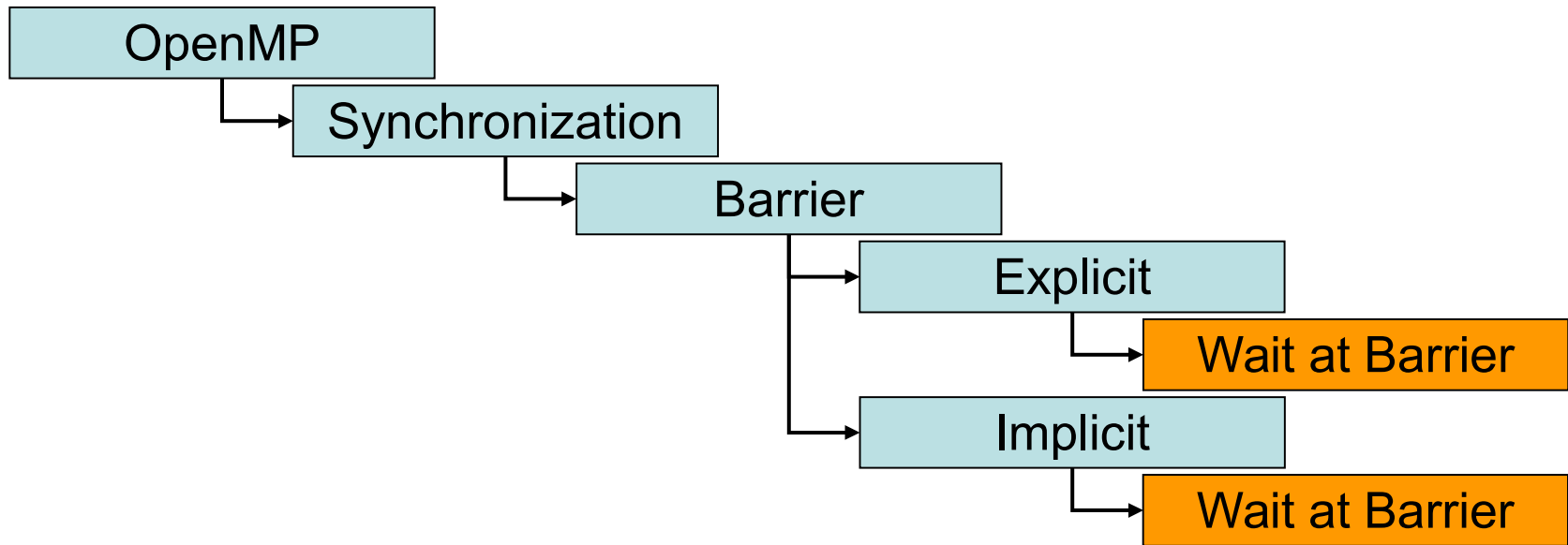
- Time spent idle on CPUs reserved for worker threads

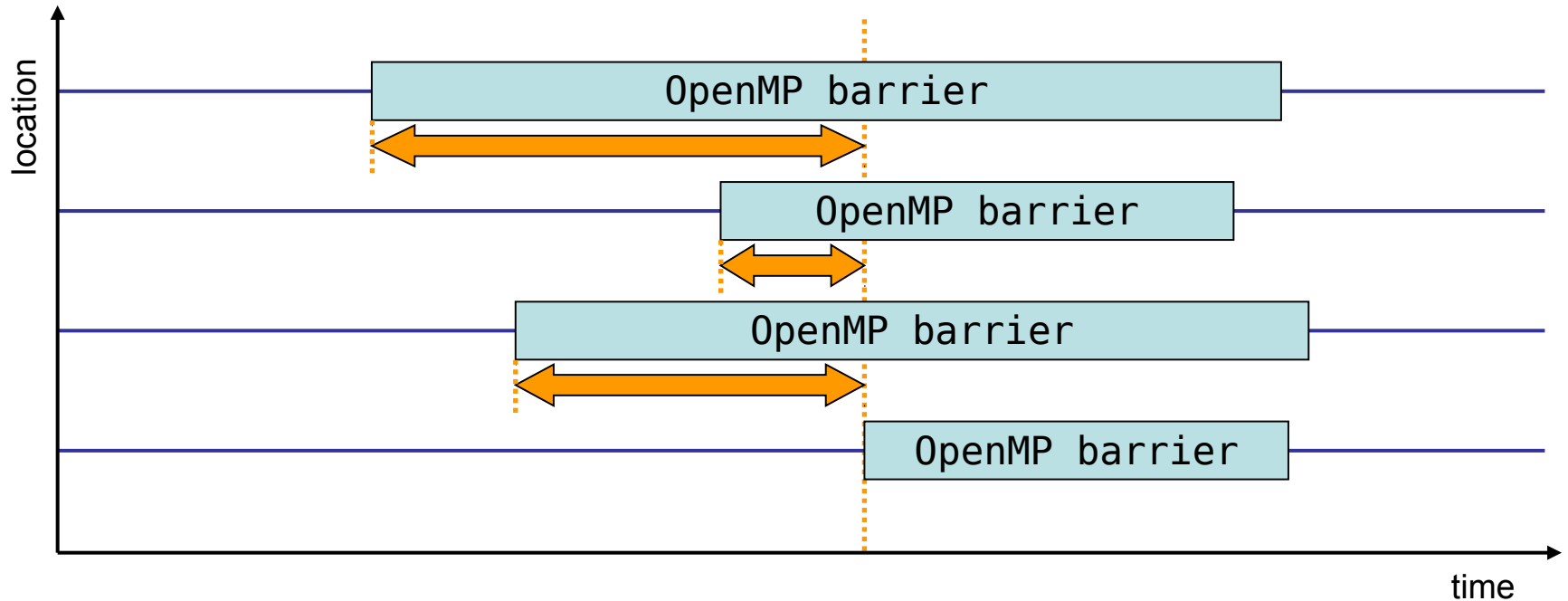


- Time spent idle on worker threads within parallel regions



- Time spent in OpenMP atomic constructs is attributed to the “Critical” metric





- Time spent waiting in front of a barrier call until the last process reaches the barrier operation
- Applies to: Implicit/explicit barriers

Happy end...

VI-HPS

