# Motivation and Scope of Changes to MPI Semantic Terms Section

Semantic Terms Working Group

# Overview

- What is missing?
- Semantic Terms in MPI-3.1 – are they still correct?
  - Blocking / nonblocking
  - Collective
  - Local / non-local
- Current status of Sect. 2.4 Semantic Terms
- Solution → Issue #96, PR #116

# What is missing?

- Persistent:
  - Since MPI-4.0, now **two** Sections on persistent operations:
    - pt-to-pt and
    - collective
  - Common definitions should be in Semantic Terms

- Operations:
  - 1049 x the word "*operation*" in MPI-3.1 (and 1202x in MPI-4.x currently)
  - But definition of MPI operations **is missing**

# Motivation - original terms of reference

- Instruction: define “persistent” in context of “persistent collectives”
- There is/was no definition of the term “persistent” at all in MPI
  - The word is used without explanation in the body of the document.

- Persistent “fits” in the sequence {blocking, nonblocking, persistent}
- However, this is talking about MPI operations, not MPI procedures
  - Insight: there is no such thing as a “persistent procedure” in MPI.

- There is/was no definition of “MPI operation” at all in MPI!
- We need at least a definition of operation that is good enough to define “persistent operation”

# Defining “MPI operation”

- Need to differentiate **persistent operations** from **blocking** and **nonblocking** ones.
- Persistent MPI operations are expressed using 4 MPI procedures:

  `MPI_<thing>_init, MPI_Start[all], MPI_{Test|Wait}[all|some], MPI_Request_free`
- These can be seen as 4 state transitions between 2 operation states:
  - Initialisation (*->inactive), starting (inactive->active),
  - Completion (active->inactive), freeing (inactive->*)
- Nonblocking operation: 2 state transitions
  - Initialisation+starting & Completion+freeing
- Blocking operation: all together in one routine

# Semantic Terms in MPI-3.1 – are they still correct?

**2.4 Semantic Terms**

When discussing MPI procedures the following semantic terms are used.

**nonblocking** A procedure is nonblocking if it may return before the associated operation completes, and before the user is allowed to reuse resources (such as buffers) specified in the call. The word complete is used with respect to operations and any associated requests and/or communications. An operation completes when the user is allowed to reuse resources, and any output buffers have been updated.

**blocking** A procedure is blocking if return from the procedure indicates the user is allowed to reuse resources specified in the call.

**local** A procedure is local if completion of the procedure depends only on the local executing process.

**non-local** A procedure is non-local if completion of the operation may require the execution of some MPI procedure on another process. Such an operation may require communication occurring with another user process.

**collective** A procedure is collective if all processes in a process group need to invoke the procedure. A collective call may or may not be synchronizing. Collective calls over the same communicator must be executed in the same order by all members of the process group.

# Semantic Terms in MPI-3.1 – are they still correct?

**nonblocking** A procedure is nonblocking if it may return before the associated operation completes, and before the user is allowed to reuse resources (such as buffers) specified in the call. The word complete is used with respect to operations and any associated requests and/or communications. An operation **completes** when the user is allowed to reuse resources, and any output buffers have been updated.

Additionally, the term "operation" is undefined → coming later

Examples:

- MPI_File_read_all_begin:
  - This procedure is **nonblocking** because you must not reuse the buffer upon return.
  - This procedure is non-local. It may ***block*** *(in the normal English sense of the word*) until all processes of the process group have called this procedure.
  - To name a procedure "nonblocking" although it may block – is this a good idea?
    In other words, MPI-3.1 allows that a routine may both **block** (English meaning) and is **nonblocking** (based on the MPI definition) .
- MPI_Bcast_init (coming with MPI-4):
  - This procedure is **nonblocking** because you must not free the buffer address upon return.
  - This procedure is non-local because it may ***block*** until all processes of the process group have called this procedure.

# Semantic Terms in MPI-3.1 – are they still correct?

**nonblocking** A procedure is nonblocking if it may return before the associated operation completes, and before the user is allowed to reuse resources (such as buffers) specified in the call. The word complete is used with respect to operations and any associated requests and/or communications. An operation **completes** when the user is allowed to reuse resources, and any output buffers have been updated.

~~Broken~~

Result:

- The MPI definition of nonblocking is broken, because in conflict with the normal use of English blocking **in a dangerous way**:
  A user may program deadlocks because he/she may overs that the procedure is non-local.
  There are 23 MPI nonblocking (MPI sense) procedures that are allowed to block (English sense)

How to resolve:

- Introduce the new semantic term "**incomplete**" based on the definition above.
- Define nonblocking as incomplete AND local
- and only for operation-related procedures

# Semantic Terms in MPI-3.1 – are they still correct?

> **blocking** A procedure is blocking if return from the procedure indicates the user is allowed to reuse resources specified in the call.

Problem:
- When we fix “nonblocking”, then we have also to fix “blocking”

Important:
- **Non**…… should be identical to logically **not** ……

How to resolve:
- Define “blocking” also only for operation-related procedures
- And just: An MPI procedure is **blocking** if it is not nonblocking ☺

# Semantic Terms in MPI-3.1 – are they still correct?

> **collective** A procedure is collective if all processes in a process group need to invoke the procedure. A collective call may or may not be synchronizing. Collective calls over the same communicator must be executed in the same order by all members of the process group.

Question: These are two different concepts:

- A procedure is collective if all processes in a process group need to invoke the procedure.
- A collective call may or may not be synchronizing.

Some routines are

- Collective and they are allowed to synchronize but need not to synchronize (MPI_Bcast, MPI_Bcast_init)
- Collective and they are **not** allowed to synchronize (MPI_Ibcast)
- Collective and must synchronize (MPI_Barrier)

Overseen when introducing the nonblocking collectives in MPI-3.0

This exception is clearly defined in the routine definition of MPI_Barrier

→ **Unclear / weakly defined definition and therefore** needs to be clarified.

# Semantic Terms in MPI-3.1 – are they still correct?

**collective** A procedure is collective if all processes in a process group need to invoke the procedure. **A collective call may or may not be synchronizing**. Collective calls over the same communicator must be executed in the same order by all members of the process group.

~ Unclear ~

Result:

- The definition fits well to blocking collective operations, but since MPI-1.1, we now have many chapters with collective operations and different types of collective procedures:
  - Always: **must be called by all processes** of the group
  - Initialization procedures of collective operations must be called in the **same sequence.**
  - Initiation procedures for nonblocking collective operations and the starting of persistent collectives are **local**,
  - whereas all others, especially the blocking collectives and the persistent collective initialization procedures are allowed to **synchronize**.

How to resolve:

- Rewriting this definition that it fits to all collective procedures.

# Semantic Terms in MPI-3.1 – are they still correct?

**local** A procedure is local if completion of the procedure depends only on the local executing process.

**non-local** A procedure is non-local if completion of the operation may require the execution of some MPI procedure on another process. Such an operation may require communication occurring with another user process.

Issue:
- Local / non-local is related to progress

# Semantic Terms in MPI-3.1 – are they still correct?

**local** A procedure is local if completion of the procedure depends only on the local executing process.

**non-local** A procedure is non-local if completion of the operation may require the execution of some MPI procedure on another process. Such an operation may require communication occurring with another user process.

Important:
- Local / non-local is related to progress
- → any change in the definition must not change the understanding of progress

Clarification:
- Let's name this definition of "local" as *"strong local"*.
- Let's check whether this meaning of "local" is really used in MPI?

# Semantic Terms in MPI-3.1 – are they still correct?

**local** A procedure is local if completion of the procedure depends only on the local executing process.

Problems:

- Let's look at MPI_Cancel and MPI_Test_cancelled of a nonblocking MPI_Issend?
  - MPI-3.1, section 8.7, Example 8.9 on pages 358+359: "The program is correct.":

```
Process 0                Process 1
_________                _________
MPI_Issend(dest=1);      MPI_Finalize();
MPI_Cancel();
MPI_Wait(…, &status);
MPI_Test_cancelled(&status, &flag);
MPI_Finalize();
```

I added this local inquiry call to show the problem

  - Both, MPI_Cancel and MPI_Wait must be local by definition of MPI_Cancel.
  - MPI_Cancel may require communication to check, whether the message is already received on the other process, i.e., whether flag == true or false must be returned:
    - As stated in MPI-1.1 to MPI-3.1, in the advice to implementors for MPI_Test_cancelled (MPI-3.1 page73 lines 21-23): "*Note that, while communication may be needed to implement MPI_CANCEL, this is still a local operation, since its completion does not depend on the code executed by other processes*."
    - That this MPI_Cancel is able to communicate, MPI-1.2 already mentioned for a similar example (MPI-2 page 24, lines 12-14 ): "*An implementation may need to delay the return from MPI FINALIZE until all potential future message cancellations have been processed*."

Result:

- Therefore, "local" was seen as weak local since MPI-1.1 → The definition of local is broken.

# Semantic Terms in MPI-3.1 – are they still correct?

**local** A procedure is local if completion of the procedure depends only on the local executing process.

– Broken –

**non-local** A procedure is non-local if completion of the operation may require the execution of some MPI procedure on another process. Such an operation may require communication occurring with another user process.

A weak-local definition would be needed for MPI_Cancel

Problems:

- Is non-local the contrary of local?
- The contrary of local would be something like
  - **non-local** A procedure is non-local if completion of the operation may require the execution of some **(MPI or non MPI)** procedure on another process. Such an operation may require communication occurring with another user process.

  → Gap between non-local and local → broken
- What means "of some MPI procedure on another process"?
  - Is it "*may require the execution of some **specific semantically-related*** MPI procedure", then non-local is identical to **contrary of weak local**.
  - Or is it "*may require the execution of* some **(specific or unspecific)** MPI procedure", then non-local is nearby to the **contrary of strong local**.
- This means, this definition is not clear enough → and therefore double broken.

This would be needed to be consistent with MPI_Cancel

# Semantic Terms in MPI-3.1 – are they still correct?

**local** A procedure is local if completion of the procedure depends only on the local executing process.

**non-local** A procedure is non-local if completion of the operation may require the execution of some MPI procedure on another process. Such an operation may require communication occurring with another user process.



Result:
- Local should be defined as weak local
- Non-local should be defined as the contrary of weak local
- MPI-1.1 to MPI-3.1 ever wanted to have local defined as “weak local” that implementations can implement all MPI procedures as flexible and efficient as possible.

How to resolve:
- We take the non-local definition and add “specific semantically-related”
- We define local as not non-local

Remark:
- The terms complete/incomplete and blocking/nonblocking are only useful for operation-related MPI procedures,
- Whereas local/non-local can be used for all MPI procedures.

# Current status of Sect. 2.4 Semantic Terms

**operation** — Missing —

**persistent** — Missing —

**nonblocking** A procedure is nonblocking if it may return before the associated operation completes, and before the user is allowed to reuse resources (such as buffers) specified in the call. The word complete is used with respect to operations and any associated requests and/or communications. An operation completes when the user is allowed to reuse resources, and any output buffers have been updated.

— Broken —

**blocking** A procedure is blocking if return from the procedure indicates the user is allowed to reuse resources specified in the call.

— Broken —

**local** A procedure is local if completion of the procedure depends only on the local executing process.

— Broken —

**non-local** A procedure is non-local if completion of the operation may require the execution of some MPI procedure on another process. Such an operation may require communication occurring with another user process.

— Broken —

**collective** A procedure is collective if all processes in a process group need to invoke the procedure. A collective call may or may not be synchronizing. Collective calls over the same communicator must be executed in the same order by all members of the process group.

— Unclear —

# Solution → Issue #96, PR #116

**Semantic Terms** has now three subsections:

- MPI operations
  - Operations consist of 4 stages: initialization, starting, completion, freeing stages
  - 3 forms: blocking, nonblocking and persistent operations
  - Collective / noncollective operations
- MPI procedures
  - Non-local / local (defined for all MPI procedures)
  - An MPI operation is implemented as a set of one ore more MPI procedures.
  - An MPI operation-related procedure implements at least part of a stage of an MPI operation.
  - Properties of operation-related MPI procedures:
    - Initialization / Starting / initiation / completing / incomplete / freeing procedure
    - Nonblocking / blocking procedure
    - Collective procedure
- MPI Datatypes
  - (existing text is unchanged)

Some additional comments on exceptions within the chapters

New Annex A.2, showing

- the set of procedures that form an operation
- The properties of the involved procedures

**Important**:

- We do not change the definition of any MPI procedure.
- The semantic terms should be consistent to the rest of the MPI standard.

# Lets look at pdf on Issue #96 and PR #116

- Issue: https://github.com/mpi-forum/mpi-issues/issues/96
- Pull request: https://github.com/mpi-forum/mpi-standard/pull/116

- pdf: https://github.com/mpi-forum/mpi-standard/files/4718105/mpi40-report-67fa10e-20200602.pdf

- Any open questions?

# Thanks for listening

Questions?