GASPI Proposal: Various Changes

V. End and C. Simmendinger

22.06.2016

Typos and Minor Clarifications

Contrary Definition of Successful Return of `gaspi_wait`

Clarification on Asynchronous and Synchronous Implementation of Collectives

Different definitions of GASPI_TEST

Clarification on Collective Queue

Change of Allreduce Listing

Typos and Minor Clarifications

- p. 8: specifiaction $\rightarrow$ specification
- p. 8: IN, OUT $\rightarrow$ *in*, *out*
  To keep consistent to the funcion signatures, where in and out are written in lower case letters and italisized
- p. 38: ascendinging $\rightarrow$ ascending
- p. 87: The blocking `gaspi_passive_send` is the routine ...
  $\rightarrow$ `gaspi_passive_send` is the routine...
  The function is a time-based blocking procedure, as stated in the next sentence.

Contrary Definition of Successful Return of `gaspi_wait`

[p.64] After successful procedure completion, i. e. return value GASPI_SUCCESS, the hitherto posted communication requests have been processed by the network infrastructure and the queue is cleaned up. After that, any communication request which has been posted to the given queue can be considered as completed on the local and remote side. For completed requests the transmitted data is available for the application.

[p.65] User advice: Return value GASPI_SUCCESS means, that the data of all posted write requests has been transferred to the remote side. It does not mean, that the data has arrived at the remote side. However, write accesses to the local source location will not affect the data that is placed in the remote target location.

Proposed Change

[p.64] After successful procedure completion, i. e. return value GASPI_SUCCESS, the hitherto posted communication requests have been processed by the network infrastructure and the queue is cleaned up. After that, any communication request which has been posted to the given queue can be considered as completed on the local ~~and remote~~ side. ~~For completed requests the transmitted data is available for the application.~~

[p.65] User advice: Return value GASPI_SUCCESS means, that the data of all posted write requests in this queue ~~has been transferred~~is in transfer to the remote side. It does not mean, that the data has arrived at the remote side. However, write accesses to the local source location will not affect the data that is placed in the remote target location.

Clarification on Asynchronous and Synchronous Implementation of Collectives

[p.98] Collective operations support both synchronous and asynchronous implementa- tions as well as time-based blocking. That means, progress towards successful procedure completion can be achieved either inside the call (for a synchronous implementation) or outside of the call (for an asynchronous implementation) before the procedure exits.

[p.99] Progress towards successful `gaspi_barrier` *completion is achieved even if the procedure exits due to timeout. The barrier is then continued in the next call of the procedure.*

[p.102] Progress towards successful `gaspi_allreduce` *completion is achieved even if the procedure exits due to timeout. The reduction operation is then continued in the next call of the procedure.*

Proposed Change

Change procedure specifications to:

> *Progress towards successful* `gaspi_<collective_routine>` *completion may be achieved even if the procedure exits due to timeout. The reduction operation is ~~then~~ continued in the next call of the procedure.*

Different definitions of GASPI_TEST

[p.9] [...] The special value 0 (defined as GASPI_TEST) indicates that the procedure will complete all local operations. The procedure subsequently returns the current status without waiting for data from other processes (non-blocking).

[p.13] GASPI_TEST is another predefined timeout value which blocks the procedure for the shortest time possible, i. e. the time in which the procedure call processes an atomic portion of its work.

[p.98] Timeout = 0 makes an atomic portion of progress in the operation if possible. If progress is possible, the procedure returns as soon as the atomic portion of progress is achieved. Otherwise, the procedure returns immediately. Here, an atomic portion of progress is defined as the smallest set of non-dividable instructions in the current state of the collective operation.

Proposed Changes

[p.9] A timeout is a mechanism required by procedures that might block (see blocking above). Timeout here is defined as the maximum time (in milliseconds) a called procedure will wait for outstanding communication from other processes. The special value 0 (defined as GASPI_TEST) indicates that the procedure ~~will complete all local operations~~ will process an atomic portion of its work, if possible. The procedure subsequently returns the current status without waiting for data from other processes (non-blocking).

[p.13] GASPI_TEST is another predefined timeout value which blocks the procedure for the shortest time possible, i. e. the time in which the procedure call processes an atomic portion of its work, if possible.

In addition, on p. 98 "Timeout = 0" should be changed to "GASPI_TEST".

Clarification on Collective Queue

> *[p.14] Collective operations have their own queue and hence typically will be synchronised independently from the operations on other queues (separation of concerns).*

It may not be clear to the reader, that GASPI will handle this queue internally and the application programmer does not have to take care of flushing the queue. This should be stated unmistakenely.

Proposed Change

We propose to explicitely state that the handling of the queue is done by the implementation:

> *Collective operations have their own queue and hence typically will be synchronised independently from the operations on other queues (separation of concerns). This queue is handled internally by the* GASPI *library, i.e., the user does not have to wait on this queue at any point in the application.*

Change of Allreduce Listing

On p. 98 of the specification, an example for the usage of `gaspi_allreduce_user` with GASPI_TEST is given:

```
1  while (gaspi_allreduce_user( buffer_send
2                             , buffer_receive
3                             , num
4                             , size_element
5                             , reduce_operation
6                             , reduce_state
7                             , group
8                             , GASPI_TEST
9                             ) != GASPI_SUCCESS
10       )
11 {
12       work();
13 }
```

Proposed Change

```
1  while (ret = (gaspi_allreduce_user( buffer_send
2                                    , buffer_receive
3                                    , num
4                                    , size_element
5                                    , reduce_operation
6                                    , reduce_state
7                                    , group
8                                    , GASPI_TEST
9                                    )) != GASPI_SUCCESS
10       )
11 {
12       if ( ret == GASPI_ERROR)
13       {
14            error_handle(&ret);
15       }
16       work();
17 }
```