GASPI Proposal: Queue creation and deletion

Rui Machado
CC-HPC
Fraunhofer ITWM
Kaiserslautern, Germany

Mirko Rahn
CC-HPC
Fraunhofer ITWM
Kaiserslautern, Germany

Daniel Grünewald
CC-HPC
Fraunhofer ITWM
Kaiserslautern, Germany

Valeria Bartsch
CC-HPC
Fraunhofer ITWM
Kaiserslautern, Germany

July 1, 2015

1 Motivation and Use-case(s)

The main motivation for this proposal is to improve the ability to create GASPI-based libraries.

Currently, a GASPI-based library can use one of the available queues but has to (potentially) share it with an application or other libraries. Moreover, a clear separation of concerns is desirable from a library point of view. A library is only interested in waiting for data or notification requests that are relevant to its own internal operation.

The following code snippet illustrates how an application can use two different libraries using the current standard. Each library accepts, as a parameter, a communication queue. This communication queue will be used internally by the library.

The application initializates each library and does its own application work using the same queue.

Listing 1: GASPI application using libraries.

```
#include <GASPI.h>
#include <mylib.h>
#include <myotherlib.h>

int
main(int argc, char *argv[])
{
  ASSERT( gaspi_proc_init(GASPI_BLOCK) );

  /* Initialize my library and use Queue 0 for communication */
  ASSERT( my_lib_init(libargs, (gaspi_queue_id_t) 0));

  /* Initialize other library using Queue 0 for communication */
  ASSERT( my_other_lib_init(libargs, (gaspi_queue_id_t) 0));

  /* Do what I have to do */
  my_own_work( (gaspi_queue_id_t) 0);

  /* Done! */
  ASSERT ( gaspi_proc_term(GASPI_BLOCK) );

  return 0;
}
```

All three instances (application and libraries) will have to share the queue, leading to a non-optimal use from each instance's point of view.

To fill this gap, we propose the creation of communication queues during runtime which would be more dynamic.

This proposal includes the introduction of two new functions: **gaspi_queue_create** and **gaspi_queue_delete**. With these functions, a library can create its own communication queue(s), independent from the application (or other libraries).

2 Proposed interface

The proposed interface is as follows. We propose to add it to section 8.5 (Communication utilities).

2.0.1 `gaspi_queue_create`

The `gaspi_queue_create` procedure is a *synchronous non-local time-based blocking* procedure which creates a new queue for communication.

```
GASPI_QUEUE_CREATE ( queue
                   , timeout
                   )
```

Parameter:
(out) queue: the created queue

(in) timeout: the timeout

```
gaspi_return_t
gaspi_queue_create ( gaspi_queue_id_t queue
                   , gaspi_timeout_t timeout
                   )
```

```
function gaspi_queue_create (queue, timeout) &
& result(res) bind (C, name="gaspi_queue_create" )
integer(gaspi_queue_id_t) :: queue
integer(gaspi_timeout_t), value :: timeout
integer(gaspi_return_t) :: res
end function gaspi_queue_create
```

Execution phase:
Working

Return values:
`GASPI_SUCCESS`: operation has returned successfully
`GASPI_TIMEOUT`: operation has run into timeout
`GASPI_ERROR`: operation has finished with an error ⌟

After successful procedure completion, i. e. return value `GASPI_SUCCESS`, the communication *queue* is created and available for communication requests on it.

If the procedure returns with `GASPI_TIMEOUT`, the creation request could not be completed during the given timeout. A subsequent call to `gaspi_queue_create` has to be performed in order to complete the queue creation request.

If the procedure returns with `GASPI_ERROR`, the queue creation failed. Attempts to post requests in the queue result in undefined behaviour.

> *User advice:* The lifetime of a created queue should be kept as long as possible, avoiding repetead cycles of creation/deletion of a queue. ⌟
>
> *Implementor advice:* The maximum number of allowed queues may be limited in order to keep resources requirements low. ⌟
>
> *Implementor advice:* The communication infrastructure must be respected i. e. previously established connections (e. g. invoking `gaspi_connect`) must be able to use the newly created queue. ⌟

2.0.2 `gaspi_queue_delete`

The `gaspi_queue_delete` procedure is a *synchronous non-local time-based blocking* procedure which deletes a given queue.

```
GASPI_QUEUE_DELETE ( queue )
```

Parameter:
(in) queue: the queue to delete

Execution phase:
Working

Return values:
`GASPI_SUCCESS`: operation has returned successfully
`GASPI_ERROR`: operation has finished with an error ⌟

```
gaspi_return_t
gaspi_queue_delete ( gaspi_queue_id_t queue )
```

```
function gaspi_queue_delete ( queue ) &
& result(res) bind (C, name="gaspi_queue_delete" )
integer(gaspi_queue_id_t), value :: queue
integer(gaspi_return_t) :: res
end function gaspi_queue_delete
```

Parameter:
(in) queue: the queue to delete

Execution phase:
Working

Return values:
`GASPI_SUCCESS`: operation has returned successfully
`GASPI_TIMEOUT`: operation has run into timeout
`GASPI_ERROR`: operation has finished with an error ⌟

After successful procedure completion, i. e. return value `GASPI_SUCCESS`, the communication *queue* is deleted and no longer available for communication. It is an application error to use the queue after `gaspi_queue_delete` has been invoked.

If the procedure returns with `GASPI_ERROR`, the delete request failed.

> *User advice:* The procedure `gaspi_wait` should be invoked before deleting a queue in order to ensure that all posted requests (if any) are completed. ⌟

3 Influence on Implementations

Existing implementations must obviously implement the two new procedures.

The possibility to dynamically create a queue will lead to a larger consumption of system resources (e. g. memory).

4 Influence on Applications

4.1 Influence on Existing Applications

This proposal specifies two new functions an hence does not directly affect existing applications.

4.2 Influence on Future Applications

Future applications and, more importantly, applications that may benefit from the proposed functionality need to be updated to use the proposed procedures, compiled and linked anew with an implementation supporting the proposed functionality.

5 Influence on Performance

The creation of a queue may require information exchange between processes. An abusive use of the functionality may lead to high overhead and poor communication performance.

6 Influence on Current Specification

The introduction of proposed functionality influences the result of the procedure `gaspi_queue_num` (Section 12.3.1).

The meaning of the parameter **queue_num** of process configuration structure (Section 5.2) is also altered.

Paragraph 5 of Section 8.1 has to be updated:

From:

The number of communication queues and their size can be configured at initialization time, otherwise default values will be used. The default values are implementation dep endent. Maximum values are also dene.

To (proposed):

The number of communication queues and their size can be configured at initialization time - via process configuration - or created during run time - via `gaspi_queue_create`*. Otherwise default values will be used. The default values are implementation dep endent. Maximum values are also defined.*