GASPI proposal: Memory Ordering

Olaf Krzikalla
Technische Universität, Dresden, Germany,
olaf.krzikalla@tu-dresden.de

July 3, 2015

1 Introduction

Consider the following GASPI-like code snippet:

```
variable = 123;
gaspi_write (&variable, target_rank, ...);
```

The programmer clearly intends to transfer a value of 123 to the target rank. However the current GASPI standard does not give a guarantee, that that value is actually transferred to the target rank. If the CPU provides out-of-order execution, then it is possible, that the local synchronous write operation to `variable` stalls until after the RDMA operation issued by `gaspi_write` has started. In that case a wrong value would arrive at target rank.

This problem does not only arise with GASPI functions reading or writing a variable but also with synchronization functions:

```
        rank_1                  rank_2


    variable = 234;
    gaspi_barrier();        gaspi_barrier();
                            gaspi_read(&variable, rank_1, ...);
```

In that case the `gaspi_barrier` call at `rank_1` has to ensure, that the value of 234 is visible, before it sends a barrier-enter event to `rank_2`.

Currently these examples might be a corner cases, which - at least in contemporary systems and GASPI implementations - won't occur. Nevertheless an addition of the intended memory consistency model to the standard is advisable in order to be able to formally proove the correctness of GASPI applications. Also, it explicitely advises library implementers to check, whether memory barriers might be neccessary for their particular target systems.

2 Proposed Changes

In section 3.4 (GASPI segments) add the following paragraph:

> Local synchronous accesses to variables in a memory segment by a thread are sequentially consistent to calls of GASPI functions by that same thread.

3 Influence on Implementation

Library implementers have to check, whether a memory barrier has to be placed at the entry and/or exit point of various functions:

- all entry points of functions, which asynchronously read or write memory,
- all exit points of functions, which synchronize asynchronous memory accesses,
- all entry and exit points of collectives.

4 Influence on Applications

There is no influence to existing applications.

The proposal *Memory provided by applications* by ITWM enables application programmers to use memory segments shared accross process boundaries. Together with this proposal the application programmer can safely use the GASPI notification technique and GASPI barriers for the coordination of memory accesses to such shared memory segments. In the following program `variable` resides in a shared memory segment, which can be seen by Process 1 and Process 2. The assertion will only hold, if memory ordering is required for GASPI synchronization functions.

```
   Process 1               Process 2

variable = 234;
gaspi_barrier();        gaspi_barrier();
                        assert(variable == 234);
```

5 Influence on Performance

The performance of calls to GASPI functions might be slightly affected, since memory barriers might be placed at some places inside the functions. If the effect becomes measureable, than it might be advisable to introduce GASPI functions with weaker ordering requirements.