Implementing a statistical counter in GASPI

This tutorial shows you how statistical counters can be incorporated in a GASPI application using the `pgaspi` interface. It is based on the file `tutorial_profiler.c`, which comes along with this document. There are a lot of comments in `tutorial_profiler.c` explaining the actual implementation of the profiler step by step.

The `pgaspi` interface enables you to implement hooks for gaspi functions by exploiting the dynamic linking capabilites of the operating system. Thus in a first step you have to ensure that your GASPI application is dynamically linked against your GASPI library. If you use GPI-2, you can build a shared object of the non-debug version by using the following all-clause in GPI-2/src/Makefile:

```
all:
  gcc -O2 -fPIC -I$(OFED_PATH)/include -D_GNU_SOURCE -c GPI2.c
  gcc -shared -Wl,-soname,libGPI2.so GPI2.o -o libGPI2.so
  cp libGPI2.so ../lib64
  cp GASPI.h ../include
```

You can check whether your GASPI application uses a shared library using `ldd`. The actual profiler wrapper is also build as a shared library.

```
gcc tutorial_profiler.c -fPIC -shared -Idir/to/GASPI/include
    -o tutorial_profiler.so
```

Before you can run and profile your application, you have to tell your operating system, that `tutorial_profiler.so` wraps your native GASPI library. In Linux you do this by setting the `LD_PRELOAD` environment variable.

```
export LD_PRELOAD=/dir/to/profiler/tutorial_profiler.so
```

You have to ensure that this environment variable is visible on every node. One simple way to ensure this is to add this line to your `.bashrc`.

Finally you can run your GASPI application. If you use the profiler code of this tutorial, it should tell you the total numbers of bytes read and written:

```
Number of bytes written by this rank (rank 0): 0
Number of bytes read by this rank (rank 0): 13088
```