

# GPI-2

A library that implements the GASPI specification

## Introduction

Programming Next Generation Supercomputers: GPI-2 is an API library for asynchronous interprocess, cross-node communication. It provides a flexible, scalable and fault tolerant interface for parallel applications.

The GPI-2 library ([www.gpi-site.com/gpi2/](http://www.gpi-site.com/gpi2/)) implements the GASPI specification (Global Address Space Programming Interface, [www.gaspi.de](http://www.gaspi.de)). GASPI is a Partitioned Global Address Space (PGAS) API. It aims at scalable, flexible and failure tolerant computing in massively parallel environments.

## Modules

The GPI-2, version 1.0.2 is available on Anselm via module gpi2:

```
$ module load gpi2
```

The module sets up environment variables, required for linking and running GPI-2 enabled applications. This particular command loads the default module, which is gpi2/1.0.2

## Linking

Link with -lGPI2 -libverbs

Load the gpi2 module. Link using **-lGPI2** and \*\*\* **-libverbs** switches to link your code against GPI-2. The GPI-2 requires the OFED infinband communication library ibverbs.

## Compiling and linking with Intel compilers

```
$ module load intel
$ module load gpi2
$ icc myprog.c -o myprog.x -Wl,-rpath=$LIBRARY_PATH -lGPI2 -libverbs
```

## Compiling and linking with GNU compilers

```
$ module load gcc
$ module load gpi2
```

```
$ gcc myprog.c -o myprog.x -Wl,-rpath=$LIBRARY_PATH -lgpi2 -libverbs
```

## Running the GPI-2 codes

`gaspi_run`

`gaspi_run` starts the GPI-2 application

The `gaspi_run` utility is used to start and run GPI-2 applications:

```
$ gaspi_run -m machinefile ./myprog.x
```

A machine file (**machinefile**) with the hostnames of nodes where the application will run, must be provided. The\*\*\* machinefile lists all nodes on which to run, one entry per node per process. This file may be hand created or obtained from standard `$PBS_NODEFILE`:

```
$ cut -f1 -d"." $PBS_NODEFILE > machinefile
```

machinefile:

```
cn79
cn80
```

This machinefile will run 2 GPI-2 processes, one on node cn79 other on node cn80.

machinefle:

```
cn79
cn79
cn80
cn80
```

This machinefile will run 4 GPI-2 processes, 2 on node cn79 o 2 on node cn80.

Use the **mpiprocs** to control how many GPI-2 processes will run per node

Example:

```
$ qsub -A OPEN-0-0 -q qexp -l select=2:ncpus=16:mpiprocs=16 -I
```

This example will produce `$PBS_NODEFILE` with 16 entries per node.

## `gaspi_logger`

`gaspi_logger` views the output form GPI-2 application ranks

The `gaspi_logger` utility is used to view the output from all nodes except the master node (rank 0). The `gaspi_logger` is started, on another session, on the master node - the node where the `gaspi_run` is executed. The output

of the application, when called with `gaspi_printf()`, will be redirected to the `gaspi_logger`. Other I/O routines (e.g. `printf`) will not.

## Example

Following is an example GPI-2 enabled code:

```
#include <GASPI.h>
#include <stdlib.h>

void success_or_exit ( const char* file, const int line, const int ec)
{
    if (ec != GASPI_SUCCESS)
    {
        gaspi_printf ("Assertion failed in %s[%i]:%dn", file, line, ec);
        exit (1);
    }
}

#define ASSERT(ec) success_or_exit (__FILE__, __LINE__, ec);

int main(int argc, char *argv[])
{
    gaspi_rank_t rank, num;
    gaspi_return_t ret;

    /* Initialize GPI-2 */
    ASSERT( gaspi_proc_init(GASPI_BLOCK) );

    /* Get ranks information */
    ASSERT( gaspi_proc_rank(&rank) );
    ASSERT( gaspi_proc_num(&num) );

    gaspi_printf("Hello from rank %d of %dn",
                rank, num);

    /* Terminate */
    ASSERT( gaspi_proc_term(GASPI_BLOCK) );

    return 0;
}

Load modules and compile:

$ module load gcc gpi2
$ gcc helloworld_gpi.c -o helloworld_gpi.x -Wl,-rpath=$LIBRARY_PATH -lGPI2 -libverbs
```

Submit the job and run the GPI-2 application

```
$ qsub -q qexp -l select=2:ncpus=1:mpiprocs=1,place=scatter,walltime=00:05:00 -I
qsub: waiting for job 171247.dm2 to start
qsub: job 171247.dm2 ready
```

```
cn79 $ module load gpi2
cn79 $ cut -f1 -d"." $PBS_NODEFILE > machinefile
cn79 $ gaspi_run -m machinefile ./helloworld_gpi.x
Hello from rank 0 of 2
```

At the same time, in another session, you may start the gaspi logger:

```
$ ssh cn79
cn79 $ gaspi_logger
GASPI Logger (v1.1)
[cn80:0] Hello from rank 1 of 2
```

In this example, we compile the helloworld\_gpi.c code using the **gnu compiler** (gcc) and link it to the GPI-2 and ibverbs library. The library search path is compiled in. For execution, we use the qexp queue, 2 nodes 1 core each. The GPI module must be loaded on the master compute node (in this example the cn79), gaspi\_logger is used from different session to view the output of the second process.