High Performance Computing

Informatik **Sheet 10**

Prof. Dr. B. Schmidt & Dr. Ch. Hundt

1 Introduction to GASPI (0P)

GASPI (Global Adress Space Programming Interface) is a set of specifications (`www.gaspi.de`) which is implemented in GPI-2 (`http://gpi-site.com.www488.your-server.de/docs/`). You can download the source code at `https://github.com/cc-hpc-itwm/GPI-2` and install GPI-2 by executing
`./install.sh -p /path/to/install -with-mpi<=path/to/mpi/installation>`.
To run your application, type
`gaspi_run -m <machinefile> [OPTIONS] <gaspi program>`
where `machinefile` is a file with the hostnames of nodes.

Download the source code from the reader and unzip it.

(i) Take a look at `aux/success_or_die.h`. Explain what happens in the macro defined there.

(ii) Complete `basic/helloworld.c` such that every process prints "Hello from rank x of y".

(iii) GPI-2 uses segments to manage memory access. Explain what segments are and how those are used. Create a segment in `basic/segments.c`.

(iv) Explain what queues are used for in GPI-2. Name at least one pitfall in using those queues. **Hint:** Check `write_notify_and_wait` at `aux/queue.h` and explain what happens there.

(v) Use `basic/onesided.c` to send `VLEN` many doubles to the next process. What happens if you do not use the notification mechanism? What happens if your data transfer is out of bounds?

2 GPI-2 and MPI (0P)

GPI-2 and MPI are not mutually exclusive but can be used together. You have to invoke `MPI_Init()` **before** you invoke `gaspi_proc_init()`. For such a mixed mode, it is assumed that you start the application with mpirun (or mpiexec etc.). Take a look back at solving the Poisson equation from sheet 4. We created a custom datatype and used `Scatterv` to scatter the initial data:

```
// create tile data type
MPI::Datatype tile_t =
    MPI::DOUBLE.Create_vector (local_height, local_width , width)
               .Create_resized(0, sizeof(double));
tile_t.Commit();

int32_t counts[num_ranks], displs[num_ranks];
for (uint64_t proc = 0; proc < num_ranks; ++proc) {
    const uint64_t i = proc / radix, j = proc % radix;
    counts[proc] = 1;
    displs[proc] = i*(local_height-2)*width+j*(local_width-2);
}

// scatter the tiles
MPI::COMM_WORLD.Scatterv(image.data(), counts, displs, tile_t,
                         ying.data(), local_height*local_width,
                         MPI::DOUBLE, 0);
```

(i) Check the methods `write_list_and_wait` and `write_list_notify_and_wait` in `queue.c`. Make sure you understand what every parameter is good for. How can we use those methods to replicate MPI's `Scatterv` with custom data types?

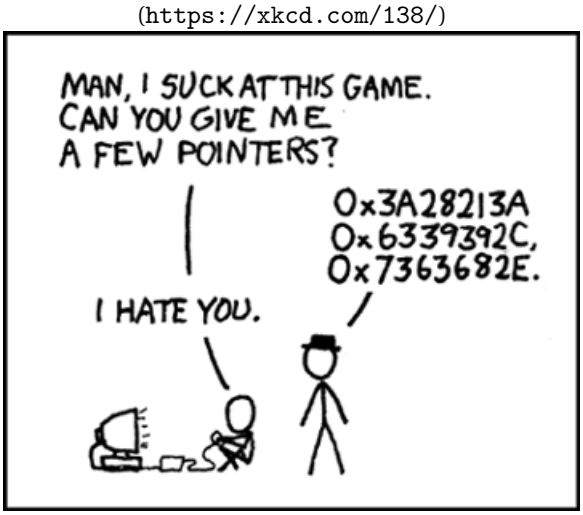
(ii) Change the scatter method to use GPI-2 instead of MPI. Take a look at the segment creation in the code and leave the rest as it is.

(iii) **Bonus task:** Replace MPI completely by changing the communication during calculation and the gathering of the data in the end.

(iv) **Bonus task:** The bandwidth degrades if too many communication operations with few data (< 1 kB) are used. Write a 1D decomposition of the matrix. What are the downsides of such a decomposition compared to a 2D decomposition?

Pointer

(`https://xkcd.com/138/`)



Every computer, at the unreachable memory adress 0x-1, stores a secret. I found it and it is that all the humans ar– SEGMENTATION FAULT.