Project Seminar:
Parallel and Distributed Systems

Assignment 4 (Submission deadline: Dec 16th 2015, 13:30 CET)

General Rules

The assignment solutions have to be submitted at:

https://www.dcl.hpi.uni-potsdam.de/submit/

Our automated submission system is intended to give you feedback about the validity of your file upload. A submission is considered as accepted if the following rules are fulfilled:

- You did not miss the deadline.
- Your file upload can be decompressed with a zip / tar decompression tool.
- Your submitted solution contains only the source code files and a Makefile. Please leave out any Git / Mercurial repository clones or SVN / CVS meta-information.
- Your solution can be compiled using the "make" command, without entering a separate sub-directory after decompression.
- You program runs without expecting any kind of keyboard input or GUI interaction.
- **There will be no validation script for this assignment.**

If something is wrong with your submission, you will be informed via email (console output, error code). Re-uploads of corrected solutions are possible until the deadline.

All tasks must be submitted accordingly in order to pass the assignment.

Assignment 4

The fourth assignment covers programming for shared nothing systems with the Message Passing Interface (MPI). You will study the basics of MPI and implement distributed reduction and your sorting algorithms from assignment 2. You have to solve the given programming exercises in C/C++ with OpenMPI[1].

In order to properly develop, test and run your MPI application, we provide you with user accounts on our small 4-way Odroid XU-4 cluster:

```
firstname.lastname@172.16.64.151 (odroid01)
firstname.lastname@172.16.64.152 (odroid02)
firstname.lastname@172.16.64.153 (odroid03)
firstname.lastname@172.16.64.154 (odroid04)
```

Your initial password is "pvprog", please make sure to change it immediately. The machines are equipped with a mere 32GB of flash memory, so use disk space economically and don't put any large files into your home directories or elsewhere. Be fair and don't hog the cluster to yourself ☺

[1] http://www.open-mpi.de

Task 4.1 Study of MPI

Make yourself familiar with the concepts of MPI. As a starting point you can have a look at http://mpitutorial.com and http://www.open-mpi.de . Please answer the following questions (2 – 3 sentences each):

1. What are ranks?
2. What are communicators?
3. How does point-to-point communication work?
4. What are collective operations?
5. How does one-sided communication work?

Please submit a single textfile.

Task 4.2: Distributed aggregation

Implement a basic framework for distributed aggregation.

Input

Your application has to be named “dist_aggr” and accept three parameters: the operation to perform, the seed and the amount of integers to be randomly generated. For each MPI rank, make sure to add the rank id to the seed in order to have comparable results for all participants. The operation parameter is a string and can be one the following: `none`, `min`, `max`, `avg`. For the `none` operation, please perform the entire setup. This enables us to measure the initialization costs. You do not have to check the input parameters for validity.

```
Example: mpirun -np 32 dist_aggr avg seed work_size
```

Output

After the given run time is exceeded, the program must terminate with exit code 0 and produce an output file with the name of “output.bin” in the same directory. This file has to contain the result of the aggregation operation as binary (please use the “int” data type in C/C++).

Task 4.3 Distributed parallel sorting - implementation

Implement a distributed sorting algorithm using OpenMPI. You can reuse and adapt your approach of task 2.3 as well as most of your efforts of task 4.2.

Input

Your application has to be named “dist_aggr” and accept three parameters: the operation to perform, the seed and the amount of integers to be randomly generated. For each MPI rank, make sure to add the rank id to the seed in order to have comparable results for all participants. The operation parameter is a string and can be one the following: `none`, `min`, `max`, `avg`, `sort`. For the `none` operation, please perform the entire setup. This enables us to measure the initialization costs.
You do not have to check the input parameters for validity.

```
Example: mpirun -np 32 dist_aggr sort seed work_size
```

Output

After the given run time is exceeded, the program must terminate with exit code 0 and produce an output file with the name of “output.bin” in the same directory. This file has to contain the result of the aggregation operation as binary (please use the “int” data type in C/C++).