Report: Homework 3 - Grid Computing

Jan Schlenker

April 23, 2015

Instructor:	Dipl.-Ing. Dr. Simon Ostermann
Parts solved of the sheet:	Tasks 1-3
Total points:	10

1 How to run the programme

First of all extract the archive file `homework_4.tar.gz`:

```
$ tar -xzf homework_4.tar.gz
$ cd homework_4
```

Afterwards move/copy the binary files `gm` and `povray` to the `bin/` directory and the files `scherk.args`, `scherk.ini` and `scherk.pov` to the `inputdata/` directory:

```
$ cp <gm-file-path> <povray-file-path> bin/
$ cp <scherk-files-dir>/scherk* inputdata/
```

At last run the gridRender.sh script:

```
$ ./gridRender.sh
```

2 Programme explanation

The files of the the programme are structured as follows:

- The `gridRender.sh` script contains the main programme
- The `bin` directory contains the binaries `povray` and `gm` which will be copied to the grid instances
- The `inputdata` directory contains the necessary files for the `povray` binary which will be copied to the grid instances

Below is the programme explanation task by task:

- **Task 1:** The `gridRender.sh` script just creates a proxy and copies the files to the appropriate instancens. One interesting thing to mention is the grid resource `login.leo1.uibk.ac.at`, because for running jobs on this resource the addition `/jobmanager-sge` is compulsory.

- **Task 2:** The `gridRender.sh` script executes a `globus`-job for each resource with a specific subset of frames. The merging of the images is done on the `karwendel` machine, after all files have been copied to it.
- **Task 3:** The `gridRender.sh` script takes timestamps before and after each grid-resource execution. The results are described in the next section.

3 Results

The environment contains 3 grid resources, 2 were used for measurements:

- `karwendel.dps.uibk.ac.at`
- `login.leo1.uibk.ac.at/jobmanager-sge`

Only one processor of each machine got a grid job, so that there is a better comparison between the resources. For faster results it would be much better to make use of all processors.

Measurement were made for 8, 16, 32, 64 and 128 frames. Table 1 shows the measurement results, where T_k is the execution time of the `karwendel` machine and T_L the execution time of the `leo1` machine (both without copying the results to the image merge machine).

Frames	$\mathbf{T}_K$ in s	$\mathbf{T}_L$ in s	$\mathbf{T}_K$**/(Frames)**/2 in s	$\mathbf{T}_L$**/(Frames)**/2 in s
8	81,47	91,14	20,37	22,79
16	141,70	171,25	17,71	21,41
32	272,36	301,54	17,02	18,85
64	543,87	602,28	17,00	18,82
128	1066,39	1153,45	16,67	18,02

Table 1: Measurements

The `karwendel` machine processor seems to be faster than the `leo1` machine one. While the number of frames increases the time/frame decreases for both machines. This is probably due to the relatively lesser job submitting overhead.

For a load imbalance as little as possible the time/frame and the measurement with the largest number of frames (smallest uncertainty) are crucial. The best distribution for 128 frames for the same environment and the same condition would be:

$karwendel = \frac{18,02}{16,67+18,02} = 52\%$

$leo1 = \frac{16,67}{16,67+18,02} = 48\%$

Of course the utilization of all processors of all machines would reveal another result.