Report: Homework 2 - Movie Rendering Application Deployment

Jan Schlenker

March 26, 2015

Instructor:	Dipl.-Ing. Dr. Simon Ostermann
Parts solved of the sheet:	Tasks 1-6
Total points:	10

1 How to run the programme

First of all extract the archive file `homework_2.tar.gz`:

```
$ tar −xzf homework_2.tar.gz
$ cd homework_2
```

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/
```

Now you can run the the remote renderer programme:

```
$ ./myRemoteRenderer.sh user@karwendel.dps.uibk.ac.at 16 clean
```

2 Programme explanation

The files of the the programme are structured as follows:

- The `myRemoteRenderer.sh` script contains the main programme and calls `render.sh` remotely
- The `render.sh` script will be copied to the remote host and basically contains homework_1
- The `bin` directory contains the binaries `povray` and `gm` which will be copied to remote host
- The `inputdata` directory contains the necessary files for the `povray` binary which will be copied to remote host

- The `jobs` directory contains the jobs which are executed by the `render.sh` script via `qsub` and which will be copied to remote host

The scripts and the jobs are written in shell script. One advantage of shell script for the tasks is that grid engine commands like `qsub` and `qacct` are directly available in shell script.

Below is the programme explanation task by task:

- **Task 1:** To copy all necessary files to the remote host, `myRemoteRenderer.sh` uses `rsync`. The reason for choosing `rsync` is that the command recognizes already transfered data on the remote host.
- **Task 2:** For rendering the files on the cluster `myRemoteRenderer.sh` calls `render.sh` on the remote host.
- **Task 3:** To extract the job execution times the `render.sh` script stores the job_ids and calls the `qacct` command for every job, when the rendering and merging is done.
- **Task 4:** The `myRemoteRenderer.sh` script just uses `rsync` again to get the gif file.
- **Task 5:** If the user passed ”clean” as the third argument, `myRemoteRenderer.sh` deletes the generated files on the remote host via `ssh`.
- **Task 6:** Figure 1 shows the execution times for an example of 64 frames rendered by 16 processors.

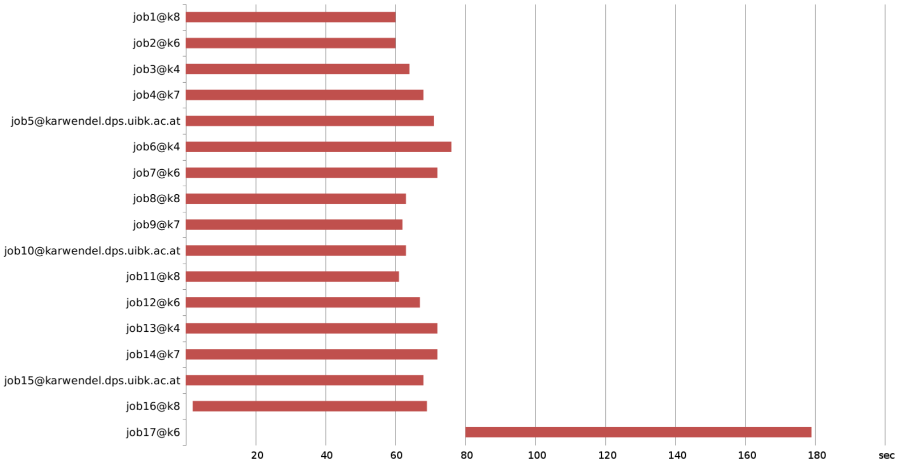

Figure 1: Gant chart for M=64 and N=16

3 Results

The test environment consisted of a grid of 40 processors in total. 16 where used for rendering the images and 1 for merging the images. The chart shows that 16 jobs started nearly at the same time and also needed nearly the same amount of time. If we had used e.g. 70 frames instead of 64, 6 processors would propably needed more time, because 64 mod 16 = 0 and 70 mod 16 = 6. After the job rendering there is a time gap between the rendering jobs and the merger job. This can be due to shell script overhead and transfer overhead between the nodes of the grid.

One measurement problematic the programme has is that only the execution times given by `qacct` are considered and not the overheads. Either way the chart shows a comprehensible result.