Report: Homework 1 - Parallel Movie Rendering with Pov-Ray

Jan Schlenker

March 16, 2015

Instructor:	Dipl.-Ing. Dr. Simon Ostermann
Parts solved of the sheet:	Tasks 1-8

1 How to run the programme

First of all extract the archive file homework_1.tar.gz:

```
$ tar -xzf homework_1.tar.gz
$ cd homework_1
```

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 main programme:

```
$ ./main.sh
```

2 Programme explanation

The files of the the programme are structured as follows:

- The **main.sh** script contains the main programme
- The **bin** directory contains the binaries povray and gm
- The **inputdata** directory contains the necessary files for the povray binary
- The **jobs** directory contains the jobs which are executed by the main script via qsub

The main programme main.sh and the jobs are written in shell script. One advantage of shell script for the tasks is that grid engine commands like qsub and qhost are directly available in shell script.

Below is the programme explanation task by task:

- **Task 1:** To render all png-files with one processor in the grid the programme uses the qsub command once to execute the povray job under jobs/job_task1.pbs. The povray job itself uses the INI-file under input-data/scherk.ini to determine how many png-files should be rendered. For task 2 it is important to give the job a name.
- **Task 2:** The programme waits until all png-files are rendered. To achieve this, image merge script (bin/gm) is submitted as a job as well, which waits with the help of the -hold-jid flag for job_task1. The main programme itself waits with the help of the -sync flag. When job_task1 finishes, job_task2 will start to merge all images.
- **Task 3:** For the measurement the programme subtracts a timestamp after task 2 from a timestamp before task 1.
- **Task 4 & 5:** First of all the programme extracts the number of frames (M) which should be rendered from the INI-file. Then it extracts the number of available processors (N) in the grid via the qhost command. If there are more processors than frames, only M processors will be used for rendering. Otherwise the frames are equally split on the processors following the rule from the task sheet. To achieve this, the qsub command is used multiple times to execute the povray job with the parameters Subset_Start_Frame and Subset_End_Frame.
- **Task 6:** For the measurement the programme subtracts a timestamp after task 5 from a timestamp before task 4.
- **Task 7 & 8:** The results are executed and presented with the help of the bc and awk command.

3 Results

The test environment consisted of a grid of 40 processors in total. Measurements were made for 2, 4, 8, 16, 32, 64 and 128 frames. Table 1 on page 3 shows the sequential and parallel execution times as well as the speedup and the efficency for the frame numbers.

While T_{seq} is roughly doubled with every doubling of the frames, T_{par} is increased much slower. Either way the increase of T_{par} gets bigger with the increase of the frames. This is due to the sequential merge in T_{par} (cf. Amdahl's law). The speedup and the efficency are increased quite steadily until the frame number is bigger than the number of processors. Then more and more processors have to render more than one frame, so that the total execution time gets bigger.

All in all the results show that the use of multiple processors can decrease

the rendering time of the pictures enormously and that the time difference of T_{seq} and T_{par} gets bigger with the number of frames.

Frames	**CPUs for T_{par}**	**T_{seq} in sec**	**T_{par} in sec**	**Speedup**	**Efficency**
2	2	38,15	23,86	1,60	0,04
4	4	76,18	27,89	2,73	0,07
8	8	147,47	35,90	4,11	0,10
16	16	292,41	49,91	5,85	0,14
32	32	582,99	75,93	7,68	0,19
64	40	1161,30	141,94	8,18	0,20
128	40	2319,09	277,01	8,37	0,21

Table 1: Measurements

Total points: /10