Next evaluation of POLAR software tests using resource of Centrum Informatyczne Świerk

Artur Narwid

1 Introduction

In this report an ability of parallel Polar data processing using cluster of Centrum Informatyczne Świerk (hereafter CIŚ) has been presented. Chapter two contain a description of testing environment in chapter three details of parallel calculation described. Chapter four and five presents respectively example (sort of how to) of CIŚ cluster using in context of Polar data analysis and results of conducted tests. In appendix A some useful commands has been described.

2 Testing environment

Polar data testing calculation has been carried out in CIŚ cluster. The software used in mentioned analysis downloaded from official Polar software repository. Details about an access to Polar repository and about code compilation and needed library has described in previous report.

The data used in tests came from first phase of Polar detector tests. This data can be found on Swiss server `raidpolar1` in `/RAID/polar/QM1/TestInit_20130829/data_raw` folder. The QM1 catalog contain 1075 files saved in binary format defined by CERN root framework (details about this format can be found at `http://root.cern.ch/drupal/content/root-files-1`) Those files contain 43 run of data acquisition, precisely 25 files `*.root` per each run. For all 43 runs following sequence of processing had to be done: pedestal subtraction, cross talk correction, energy calibration an data merging. The `process` program which can be found in `/software/analysis/ProcessData/` in Polar repository is dedicated for mentioned sequence of analysis.

3 Parallel data processing

In a case of Polar software there is no possibility of parallel data processing in a level of single code instruction. This software does not support multithreading. One chance of doing parallel data processing is to divided whole input data set into smaller subsets and run for each one an instance of `process` program. The number of subsets is limited by a logic of Polar data processing. Taking this into account, I prepared python's script called `splitter.py` which takes as input parameters following variables: path to input set, path to place where output should be write and the number of subsets on which initial set should be divided. The `splitter.py` script consist of three function. First one, called `readconfig()` reads input parameters from polarCIS.conf configuration file. The following format of configuration file has been adopted: `parameter_name:value` (see, Fig. 1.)

```
input:./QM1
output:./results_QM1
path_to_process:./software/analysis/ProcessData/
subset: 21
```

Figure 1: Format of configuration file read by `splitter.py`. Four parameters must be set, `input:` define path to the initial set of data, `output:` define path to place where results of data analysis should be write, `path_to_process:`

determines the path to `process` program, and `subset:` contains number of subsets.

The second function called `prepdata()` reads initial set of data and on the basis of names of `*.root` files it determines the number of acquisition queue (runs). The number of queue is a very important parameters because it determines the maximum value of subsets. On the basis of number of queue and the number of subsets third function called `splittask()` creates commands sequence which will call separate instances of `procces` program (one instance per one subset). Mentioned sequence of commands are writing into files (one file, one command) called "taskX.sh", where X is a successive number of file (or subset). The `taskX.sh` files will be a parameters for `qsub` command which requests resources of CIŚ cluster.

4 Example

Let's assume that we want to make an analysis of Polar data which are in mentioned QM1 catalog in four parallel processes. Following steps should be performed.

1. Set parameters in configuration `polarCIS.conf` file.

2. Run `splitter.py` by command

`python splitter.py`

As an output user should see information like this:

```
No of subsets: 3.0
No of runs: 43
For n-1 nodes: 14.0 runs
Last node: 15.0 runs
process all -t QM1 -r 0000 0013 -i QM1/ -o results_QM1_1 -ow
process all -t QM1 -r 0014 0027 -i QM1/ -o results_QM1_2 -ow
process all -t QM1 -r 0028 0042 -i QM1/ -o results_QM1_3 -ow
procPath: polar2/software/analysis/ProcessData/
```

Also, where "splitter.py" was run, four files (four because we decided to have four parallel process) should be created, namely, `task1.sh`, `task2.sh`, `task3.sh` and `task4.sh`. This files should contained instructions like this:

```
#!/bin/bash
process all -t QM1 -r 0000 0010 -i QM1 -o results_QM1_1 -ow
```

Use `process help` command to see details about options.

3. Start task files on CIS cluster. To run tasks defined in task1-4.sh following commands need to be used.

```
qsub -q short taks1.sh
qsub -q short task2.sh
qsub -q short task3.sh
qsub -q short task4.sh
```

If we want to avoid typing four times above commands we can place them into some shell file. Then save this file as for instance `run_tasks`, then `chmod 755 run_tasks` and run it by `./run_task` command.

5 Tests results

Method discussed in chapter two and three has been tested on CIŚ cluster. The time of processing data was measured and the results of measurements was shown in Fig.2. As can be seen from this figure the initial set of data was divided from two to fourteen subsets. For each of subsets Polar's data has processed. The real and CPU usage time was measured. If we divided an initial set to two subsets the real time of data processing was equal about 140 minutes. When we divided it to fourteen

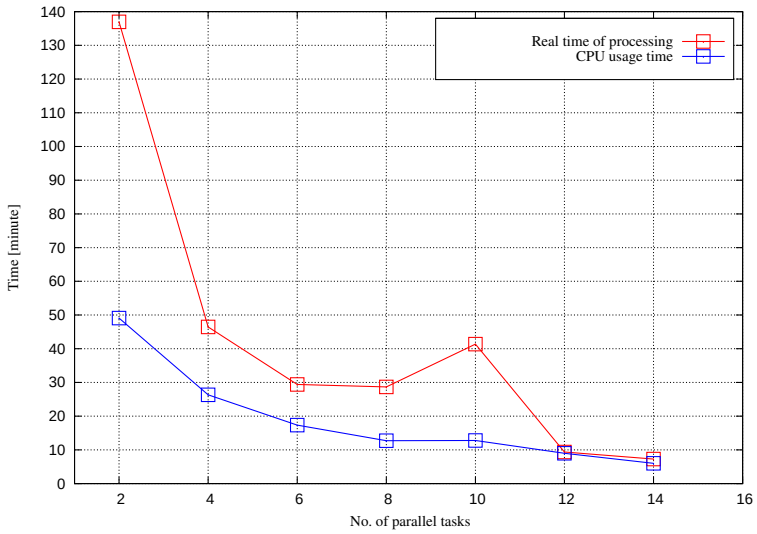

Figure 2: Real time data processing and CPU usage time vs. numbers of parallel processes. The initial set of Polar data was divided into several subsets. Initially it was divided into two subsets, then four and so on up to fourteen. The real time of processing and CPU usage time has a decreasing trend. However, when the number of subset has been set at ten, the real time of data processing increased compared to the previous value. Probably, one of the reason was high cluster load (many users used it).

subsets we was able to processed them in 8 minutes. For single CIŚ node when I did not use proposed method of parallel analysis the time of whole process took about $2h05m$. For comparison, Polar data processing was run on dedicated Swiss server and the time of whole analysis was order of 2 hours (precisely, 1h56m).

Appendix A

Below we present some useful commands which can be used during work with CIŚ cluster.

qsub

This command submit jobs on server. It has following options,

```
[-a date_time] [-A account_string] [-b secs]
[-c checkpoint_options] [-C directive_prefix] [-d path] [-D path]
[-e path] [-f] [-h] [-I ] [-j join ] [-k keep ]
[-l resource_list ] [-m mail_options] [-M user_list]
[-N name] [-o path] [-p priority] [-P user[:group]]
[-q destination] [-r c] [-S path_list]
[-t array_request] [-u user_list]
[-v variable_list] [-V ] [-W additional_attributes] [-X] [-z] [script]
```

Details about all above options can be found at

`http://www.clusterresources.com/torquedocs21/commands/qsub.shtml`

Example:

`qsub -q short task1.sh`,
This command submit a short queue for tasks defined in task1.sh file.

qstat

This command show status of jobs. It has following options,

```
qstat [-f [-1]][-W site_specific] [job_identifier...  | destination...]
[time]
qstat [-a|-i|-r|-e] [-n [-1]] [-s] [-G|-M] [-R] [-u user_list]
[job_identifier...  | destination...]
qstat -Q [-f [-1]][-W site_specific] [destination...]
qstat -q [-G|-M] [destination...]
qstat -B [-f [-1]][-W site_specific] [server_name...]
qstat -t
```

Details about `qstat` options can be found at
`http://www.clusterresources.com/torquedocs21/commands/qstat.shtml`

Example:
`qsub -u user_name`,
This command will show some useful information about jobs requested by a given user.

qdel

The qdel command deletes jobs. It has a following options,

```
qdel [-m <message>|-p|-W <delay>|-t <array_range>] <JOBID>[ <JOBID>]...
| 'all' | 'ALL'
```

Details about `qdel` options can be found at
`http://www.clusterresources.com/torquedocs21/commands/qdel.shtml`