Advancing the Search for Dark Energy with Parsl and HPC

Tom Glanzman - SLAC National Accelerator Laboratory
glanzman@stanford.edu

in close collaboration with Ben Clifford who adapted an existing DESC workflow to Parsl and continues to partner in this endeavor
The Rubin and DESC Projects

● Vera C. Rubin Observatory [formerly LSST] (DOE+NSF)
  ○ Sited on a mountain top (Cerro Pachon) in Chile
  ○ 8.3 meter diameter primary mirror
  ○ WIDE field of view (10x10 degrees)
  ○ Worlds largest digital camera (3.2 Gpixels)
  ○ Begin operation ~2022-3 with 10-year whole-sky survey program

● What is Dark Energy?
  ○ “Dark energy is the name given to the mysterious force that’s causing the rate of expansion of our universe to accelerate over time, rather than to slow down.” [ref]

● Dark Energy Science Collaboration (DOE)
  ○ >1000 scientist collaboration started in 2012
  ○ Exploit Rubin data to study clues to dark energy
Mountain top observatory (Chile)

Telescope mount (Spain)

Grinding the 8.3m lens
(Steward Observatory, Tucson, AZ)

Photos courtesy of Rubin Observatory, LSST Project/NSF/AURA
- Camera focal plane
- 189 science sensors (4k x 4k pixels)
- 12 special purpose sensors (focus, pointing)

Photos courtesy of Rubin Camera Team
The DESC Data Challenges

- No data yet! (Not until ~2022-3)
- Must hit the ground running. Therefore,
  - simulate (part of) the sky,
  - exercise the LSST project (DM) software to convert raw images into catalogs,
  - develop and test DESC-specific algorithms on the result.

- Data Challenge 2 (DC2)
  - ~300 sq. degrees of the sky (about 0.7% of entire sky)
  - 5 years of observation (one-half the Rubin survey program)

- Computational steps involved (simplified):
  - Natural parallelization: images(exposures), sensors, patches of sky, etc.
  - DC2 generates >1PB of data and consumes 10’s of millions of CPU hours
  - DOE has provided cycles at NERSC and ALCF to support this work
  - Image simulation step managed by Parsl at NERSC & ALCF (and presented at last year’s Parslfest)
Parsl @ NERSC

- Cori-KNL (primary HPC machine at NERSC)
  - 9,688 nodes each with 68 cores x 4 hardware threads
  - Modest clock speed 1.4 GHz
  - 96 GB memory per node
- Storage = GPFS ($HOME) + Lustre ($SCRATCH)
- Batch access via SLURM
- Challenges:
  - Relatively little memory/core (or hyperthread), ~1.5 GB/core
  - Disk I/O can be problematic, slow, erratic
  - SLURM queue often experiences very large dispatch latencies (hours to days), even for small jobs which can be a problem for development and production throughput
  - Rubin/LSST codes are single-threaded
Data Release Pipeline (DRP) à la Parsl

DRP consists of Rubin DM project algorithms (python/C++) representing all of the processing from raw camera images to catalogs of sky objects.

The primary Parsl apps used in this workflow.

Using *multiple HTEX executors* to match tasks to needed resources.

<table>
<thead>
<tr>
<th>Task name (parsl app)</th>
<th>Executor</th>
<th>Instances (est.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>make_tract_list</td>
<td>batch-2</td>
<td>1</td>
</tr>
<tr>
<td>make_patch_list_for_tract</td>
<td>batch-2</td>
<td>173</td>
</tr>
<tr>
<td>visits_for_tract_patch_filter</td>
<td>batch-2</td>
<td>43,506</td>
</tr>
<tr>
<td>coadd_parsl_driver</td>
<td>local</td>
<td>43,506</td>
</tr>
<tr>
<td>make_coadd_temp_exp</td>
<td>batch-3</td>
<td>360,595</td>
</tr>
<tr>
<td>assemble_coadd</td>
<td>batch-4</td>
<td>43,008</td>
</tr>
<tr>
<td>detect_coadd_sources</td>
<td>batch-4</td>
<td>43,008</td>
</tr>
<tr>
<td>multiband_parsl_driver</td>
<td>local</td>
<td>50,568</td>
</tr>
<tr>
<td>merge_coadd_detections</td>
<td>batch-4</td>
<td>8,428</td>
</tr>
<tr>
<td>deblend_coadd_sources</td>
<td>batch-4</td>
<td>50,568</td>
</tr>
<tr>
<td>measure_coadd_sources</td>
<td>batch-5</td>
<td>50,568</td>
</tr>
<tr>
<td>merge_coadd_measurements</td>
<td>batch-4</td>
<td>8,428</td>
</tr>
<tr>
<td>forced_phot_coadd</td>
<td>batch-5</td>
<td>50,568</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Executor</th>
<th># nodes/block</th>
<th># workers/node</th>
<th>Clock limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch-2</td>
<td>1</td>
<td>200</td>
<td>9:00:00</td>
</tr>
<tr>
<td>batch-3</td>
<td>400</td>
<td>22</td>
<td>10:00:00</td>
</tr>
<tr>
<td>batch-4</td>
<td>50</td>
<td>20</td>
<td>10:00:00</td>
</tr>
<tr>
<td>batch-5</td>
<td>100</td>
<td>50</td>
<td>24:00:00</td>
</tr>
</tbody>
</table>
wstat - workflow status reporting tool

- A python script to read and interpret Parsl’s `monitoring.db`
- Produce various (text-based) reports and plots.
- General tool -- *not tied to any specific workflow*
- Tabular reports including all runs, all tasks, full task history, etc.
- Full references to log files
- Example reports in the *Backup Slides*
- *Very much a work in progress* - if there is interest, contact me for github info

**Example execution timing histogram**

**Workflow summary at 2020-10-02 07:59:01.909485**

```
+--------------------------+------------------------------------------+
| workflow name           | DRPtest                                  |
| run                     | 001          <<-most current run->>       |
| run start               | 2020-09-27 10:22:05                         |
| run end                 | *pending*                                |
| run duration            | *pending*                                |
| tasks completed         | 3                                        |
| tasks completed: success| 3                                        |
| tasks completed: failed  | 0                                        |
+--------------------------+------------------------------------------+
| workflow user           | descdm@cori20                             |
| workflow rundir         | /global/cscratch1/sd/descdm/ParslRun/dr2 |
| MonitorDB               | ./monitoring.db                           |
```
Parsl Wish List

● Ability to “roll back” selected task(s) within workflow
  ○ To expedite development of both the workflow and its component tasks
  ○ In production to surgically redo selected task - and it’s downstream dependencies

● Improved executor with better control over task assignment to batch nodes
  ○ Do not start task requiring 3 hours on a batch node with only 1 hour left
  ○ Do not start task requiring 4 GB of memory on node with only 1 GB remaining
  ○ Flexibility to request #nodes/job according to task backlog
  ○ (User must specify these limits!)

● Monitoring
  ○ Extend “monitoring” to all executors
    ■ Very difficult to collect performance statistics without monitoring data
  ○ Make monitoring data reliable
    ■ Data are lost! For example, runtime (task_time_running)
  ○ Better accounting of batch jobs
    ■ For calculating efficiency, need record of #idle workers vs time, data for tasks that fail
due to batch job running out of time (ref github issue #1658)
  ○ Record task failure codes (return codes or time-out or crash or …) (ref issue #1453)

● Command/Control communication with running workflow
  ○ E.g., refresh executor parameters or other config without usual {^c, edit, restart} cycle

● Support for application-level checkpointing, e.g., dmtcp
  ○ Long-running, or batch time-outs can be restarted for better efficiency
Backup Slides
(intended to be viewed full screen)
wstat - **Workflow STATus**

wstat is a very basic text-oriented report generator using data from the Parsl monitoring.db. These reports are intended to provide a quick overall status of a running (or completed) workflow.

Obviously, monitoring must be *enabled* for this to work. Currently, only the HTEX (high-throughput executor) supports monitoring 😞.

You are welcome to take wstat out for a spin.

Github repo:  [https://github.com/TomGlanzman/Perp](https://github.com/TomGlanzman/Perp)

Caveats:

- Work in progress - some features may not quite work right: consider this a *prototype tool*
- Plot function is at a very early stage of development
- You may encounter extraneous debug statements
- Monitoring.db schema can change - and foul up the SQL in wstat
wstat “help” - listing reports and options

```
(Sat 13:35) descdm@cori20 $ python wstat -h
3.7.5 (default, Oct 25 2019, 15:51:11)
[GCC 7.3.0]
        [-l TASKLIMIT] [-d DEBUG] [-v]
        [reportType]

A simple Parsl status reporter. Available reports include:
'shortSummary', 'taskSummary', 'taskHistory', 'runNums', 'runHistory', 'plot'

positional arguments:
  reportType
      Type of report to display (default=shortSummary)

optional arguments:
  -h, --help
      show this help message and exit
  -f FILE, --file FILE
      name of Parsl monitoring database file
      (default=./monitoring.db)
  -r RUNNUM, --runnum RUNNUM
      Specific run number of interest (default = latest)
  -s, --schemas
      only print out monitoring db schema for all tables
  -t TASKID, --taskID TASKID
      specify task_id (taskHistory only)
  -S TASKSTATUS, --taskStatus TASKSTATUS
      specify task_status_name
  -l TASKLIMIT, --taskLimit TASKLIMIT
      limit output to N tasks (default is no limit)
  -d DEBUG, --debug DEBUG
      Set debug level (default = 0)
  -v, --version
      show program's version number and exit
```
wstat -- “shortSummary” example

Header

List of nodes currently running - and the number of tasks running on each

Statistics for all tasks for each Parsl “state”
wstat - “taskSummary” example

<table>
<thead>
<tr>
<th>task_id</th>
<th>task_name</th>
<th>run_num</th>
<th>status</th>
<th>hostname</th>
<th>try</th>
<th>#fails</th>
<th>submittime</th>
<th>starttime</th>
<th>endtime</th>
<th>runtime</th>
<th>stdout</th>
</tr>
</thead>
<tbody>
<tr>
<td>3842</td>
<td>multiband_parsl_driver</td>
<td>000</td>
<td>failed</td>
<td></td>
<td>0</td>
<td>1</td>
<td>2020-09-26 10:58:28</td>
<td>2020-09-26 10:58:28</td>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>3868</td>
<td>multiband_parsl_driver</td>
<td>000</td>
<td>failed</td>
<td></td>
<td>0</td>
<td>1</td>
<td>2020-09-26 10:29:06</td>
<td>2020-09-26 10:29:06</td>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>3115</td>
<td>multiband_parsl_driver</td>
<td>000</td>
<td>failed</td>
<td></td>
<td>0</td>
<td>1</td>
<td>2020-09-26 20:07:57</td>
<td>2020-09-26 20:07:57</td>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>107934</td>
<td>forced_phot_coadd</td>
<td>000</td>
<td>failed</td>
<td>nid080520</td>
<td>2</td>
<td>3</td>
<td>2020-09-26 11:19:01</td>
<td>2020-09-26 11:19:02</td>
<td>2028-09-26 11:22:24</td>
<td>0:03:22</td>
<td>335702</td>
</tr>
<tr>
<td>107935</td>
<td>forced_phot_coadd</td>
<td>000</td>
<td>failed</td>
<td>nid080520</td>
<td>2</td>
<td>3</td>
<td>2020-09-26 11:19:01</td>
<td>2020-09-26 11:19:02</td>
<td>2028-09-26 11:22:24</td>
<td>0:03:22</td>
<td>335742</td>
</tr>
<tr>
<td>107937</td>
<td>forced_phot_coadd</td>
<td>000</td>
<td>failed</td>
<td>nid080520</td>
<td>2</td>
<td>3</td>
<td>2020-09-26 11:19:01</td>
<td>2020-09-26 11:19:02</td>
<td>2028-09-26 11:22:24</td>
<td>0:03:22</td>
<td>339592</td>
</tr>
<tr>
<td>107955</td>
<td>forced_phot_coadd</td>
<td>000</td>
<td>failed</td>
<td>nid080520</td>
<td>2</td>
<td>3</td>
<td>2020-09-26 11:19:01</td>
<td>2020-09-26 11:19:02</td>
<td>2028-09-26 11:22:24</td>
<td>0:03:22</td>
<td>347536</td>
</tr>
<tr>
<td>107957</td>
<td>forced_phot_coadd</td>
<td>000</td>
<td>failed</td>
<td>nid080520</td>
<td>2</td>
<td>3</td>
<td>2020-09-26 11:19:01</td>
<td>2020-09-26 11:19:02</td>
<td>2028-09-26 11:22:24</td>
<td>0:03:22</td>
<td>346321</td>
</tr>
<tr>
<td>107962</td>
<td>forced_phot_coadd</td>
<td>000</td>
<td>failed</td>
<td>nid080520</td>
<td>2</td>
<td>3</td>
<td>2020-09-26 11:19:01</td>
<td>2020-09-26 11:19:02</td>
<td>2028-09-26 11:22:24</td>
<td>0:03:22</td>
<td>346321</td>
</tr>
</tbody>
</table>

Report consisting of every failed task

Header

Useful information for investigating problems

Including location of logs
wstat - “tasksHistory” example

Report consisting of full history for a specific task

Header

Full history of this task’s attempt to run through Parsl
wstat - “runHistory” example

(Sun 11:26) descdm@cori20 $ python wstat runHistory

3.7.5 (default, Oct 25 2019, 15:51:11)
[GCC 7.3.0]
wstat - Parsl workflow status (version 1.0.0, written for Parsl version 1.0.0:lsst-dm-282005)

+----------------------------------------------------------+
| RunNum | workflow_name | user | host | time_began | time_completed | RunDuration | #tasks_good | #tasks_bad | rundir       |
+----------------------------------------------------------+
+----------------------------------------------------------+

wstat elapsed time = 0:00:00.725202