Grid Tools

General Larsoft Meeting
Oct. 24, 2012

H. Greenlee

Overview

- This talk covers some grid tools that I have been developing with the idea of using them for MicroBooNE (in particular fall MC challenge). But these tools are mostly not MicroBooNE-specific.
- Outline
  - Job submission tools.
  - Project management tools.
  - Accessing uboone grid tools.

Grid Job Submission

- Currently we (larsoft, IF) use Dennis Box's ups product jobsub_tools as a front end to the Fermilab condor batch system.
  - Instructions on larsoft wiki.
  - Example worker scripts in larsoft svn package setup/batch (e.g. condor_lBdetMC.sh).
- My aspirational goal was to make a truly general-purpose worker script for arbitrary job running lar executable, that would “never” have to be edited.

General Purpose Worker Script: condor_lar.sh

- Input from single file or file list (in future sam dataset).
- Output to any nfs-accessible filesystem (in future gridftp?).
- Built-in support for parallelism, makeup and recovery jobs.
- Hooks for running arbitrary user-provided scripts.

Command Line Options: condor_lar.sh

```
Lar options:

-c, --config <arg>      - Configuration (fcl) file (required).
-s, --source <arg>      - Input file (full path).
-S, --source-list <arg> - Input file list (full path, one per line).
-o, --output <arg>      - Output file name.
-n, --nevts <arg>       - Number of events to process.
--nskip <arg>           - Number of events to skip.
--nskip_job <arg>       - Number of events to skip per job.
--nfile <arg>           - Number of files to process (use with option -S).
--nfile_skip <arg>      - Number of files to skip (use with option -S).
--nfile_skip_job <arg>  - Number of files to skip per job (use with option -S).
--args <args...>        - Arguments for lar command line (place at end).

Larsoft options.

-r, --release <arg>     - Release tag (default "development").
--localdir <arg>        - Larsoft local test release directory (default none).

Other options.

-h, --help              - Print help.
-i, --interactive       - For interactive use.
--group <arg>           - Group or experiment (required).
--workdir <arg>         - Work directory (required).
--outdir <arg>          - Output directory (required).
--scratch <arg>         - Scratch directory (only for interactive).
--cluster <arg>         - Job cluster.
--process <arg>         - Process within cluster.
--init-script <arg>     - User initialization script execute.
--init-source <arg>     - User initialization script to source (bash).
--end-script <arg>      - User end-of-job script to execute.
```

General Purpose Submission Script: submit_lar.sh

- Front end for jobsub.
- Provides access to features of condor_lar.sh in the most simple way possible.

Command Line Options: submit_lar.sh

```
Lar options:

-c, --config <arg>      - Configuration (fcl) file (required).
-s, --source <arg>      - Input file (full path).
-S, --source-list <arg> - Input file list (full path, one per line).
-o, --output <arg>      - Output file name.
-n, --nevts <arg>       - Number of events to process.
--nskip <arg>           - Number of events to skip.
--nskip_job <arg>       - Number of events to skip per job.
--nfile <arg>           - Number of files to process (use with option -S).
--nfile_skip <arg>      - Number of files to skip (use with option -S).
--nfile_skip_job <arg>  - Number of files to skip per job (use with option -S).
--args <args...>        - Arguments for lar command line (place at end).

Larsoft options.

-r, --release <arg>     - Release tag (default "development").
--localdir <arg>        - Larsoft local test release directory (default none).

Jobsub options.

-N, --numjobs           - Number of jobs to submit (default 1).
-g, --grid              - Use grid (default non-grid).
--opportunistic         - Use opportunistic grid (implies --grid).

Other options.

-h, --help              - Print help.
--group <arg>           - Group or experiment (default $GROUP).
--workdir <arg>         - Work directory (default current directory).
--outdir <arg>          - Output directory.
--cluster <arg>         - Job cluster.
--process <arg>         - Process within cluster.
--init-script <arg>     - User initialization script execute.
--init-source <arg>     - User initialization script to source (bash).
--end-script <arg>      - User end-of-job script to execute.
```

Project Management Tools

- Need additional features beyond simple job submission.
  - High level project description language.
    - Reduce amount of information that needs to be specified on command line.
    - Archive project description (database or svn).
    - View project descriptions on web page.
  - Testing and verification.
    - Automatically generate makeup/recovery jobs.
  - Multistage projects (e.g. generation + reconstruction).

Using XML as Project Description Language

- XML provides a widely supported and highly flexible and extensible way to store structured data.
  - Python includes several XML parsers (at least three that I know of) that make it easy to write python scripts to parse and manipulate xml documents and objects.
  - Postgres database natively supports XML datatype.
    - Includes XML parser and allows XML predicates in SQL queries.
  - Web browsers can render XML documents in human-readable form.
    - Rendering can be beautified using XML stylesheets.

Example XML Project Description

```
<?xml version="1.0"?>

<!-- Production Project -->

<!DOCTYPE project [
<!ENTITY release "S2012.09.18">
<!ENTITY name "genie_uboone">
]>

<project name="&name;">

  <!-- Group -->
  <group>uboone</group>

  <!-- Project size -->
  <numevents>5000</numevents>
  <numjobs>50</numjobs>

  <!-- Larsoft information -->
  <larsoft>
    <tag>&release;</tag>
  </larsoft>

  <!-- Project stages -->

  <stage name="gen">
    <fcl>prod&name;.fcl</fcl>
    <outdir>/uboone/data/users/greenlee/gen-uboone/&release;/&name;</outdir>
    <workdir>/uboone/app/users/greenlee/gen-uboone/&release;/&name;</workdir>
  </stage>

  <stage name="reco">
    <fcl>standard_reco_uboone.fcl</fcl>
    <outdir>/uboone/data/users/greenlee/reco-uboone/&release;/&name;</outdir>
    <workdir>/uboone/app/users/greenlee/reco-uboone/&release;/&name;</workdir>
  </stage>

  <!-- Fcl directories -->
  <fcldir>/uboone/app/users/greenlee/fcl-uboone</fcldir>

</project>
```

Project Management Script: project.py

- Job submission based on project description (XML).
- Verification of successful project stage completion.
- Submit makeup/recovery jobs.
- Daisy-chain output from previous stage to current stage.

Command Line Options: project.py

```
Project options:

--xml <-|file|url>  - Xml file containing project description.
--project <project> - Project name (required if xml file contains
                      more than one project description.
--stage <stage>     - Project stage (required if project contains
                      more than one stage).

Actions (specify one):

[-h|--help] - Print help.
--submit    - Submit all jobs for specified stage.
--check     - Check results for specified stage and print message.
--makeup    - Submit makeup jobs for specified stage.
--clean     - Delete output from specified project and stage.
```

Life Cycle of a Project (Example)

```
project.py --xml myproject.xml --submit --stage gen
```

(wait)

```
project.py --xml myproject.xml --check --stage gen
```

(continue if no errors)

```
project.py --xml myproject.xml --submit --stage reco
```

(wait)

```
project.py --xml myproject.xml --check --stage reco
```

(finished if no errors)

If errors are detected at any stage, submit and check makeup jobs.

```
project.py --xml myproject.xml --makeup --stage reco
project.py --xml myproject.xml --check --stage reco
```

Project Verification

- What gets checked by project.py --check.
  - Existence of root file for every partition.
  - Root file is openable (not corrupted, not zombie).
  - Existence of Events tree.
  - Correct number of events in Events tree.

Accessing Grid Tools (MicroBooNE Specific)

- The three scripts described in this talk (condor_lar.sh, submit_lar.sh, project.py) are stored in ubooneoffline svn repository.
  - Development versions are in trunk/products.
- Frozen versions can be accessed from MicroBooNE products area in /uboone/data/products.
  - export PRODUCTS=/uboone/data/products:$PRODUCTS (setenv PRODUCTS /uboone/data/products:$PRODUCTS)
  - setup ubtools

Tasks

- Develop postgres database tools.
  - Inserting and extracting xml project descriptions from database.
  - Generating web pages for browsing projects.
- Develop sam interface.