A lightweight task submission and management infrastructure

Xiaomei Zhang Bing SUO Xianghu Zhao TianYan
CHEP2016, San Fransico
Oct 10, 2016
Experiments at IHEP

- **Current experiments**
  - BESIII
  - DYB
  - YBJ

- **New experiments**
  - JUNO
  - LHAASO
  - HXMT
  - CEPC

BESIII (Beijing Spectrometer III at BEPCII)

DYB (Daya Bay Reactor Neutrino Experiment)

JUNO (Jiangmen Underground Neutrino Observatory)

LHAASO

Large High Altitude Air Shower Observatory

HXMT

Hard X-Ray Moderate Telescope
Multi-VO distributed computing

- More than one experiments express interests on using or evaluating distributed computing resources

- DIRAC was adopted to build distributed computing system for BESIII in 2012

- Multi-VO supports has been extended in one set-up to run JUNO and CEPC jobs

- Task Submission and Management System (TSMS) is one of the necessary steps for new experiments to have a try and use in their early stages
Goals:

- Automatically manage massive jobs
- Highly extensible for new experiments
- Ease the procedure to use heterogeneous resources

Life cycle of a task (a batch of jobs in one submission)

- split->submit->run->status monitor->output retrieval -> reschedule

Diagram:

- Task
  - Job1
  - ... 
  - jobn
- Dirac
- Batch
- worknode
- SE
- Monitor
JSUB Architecture

- **Job submission**
  - split
  - submit
  - workflow control

- **Job management**
  - status monitoring
  - reprocessing
  - logs retrieve
  - repository

- **Dataset management**
  - query
  - register

**Backend**
- Dirac
- Pbs
- Condor
- Other

**File Catalog**
User Interface

- YAML used to describe tasks
  - simple, clear, friendly with python
  - Data transformed into tree-like dictionary
- Shell commands are provided

```
-bash-3.2$ ls -l
jsub_remove
jsub_resubmit
jsub_status
jsub_submit
-bash-3.2$ jsub_submit -h
usage: jsub_submit <options> <job profile>
options:
  -h  show the help information
  -v  show JSUB version
job profile: the file where the parameters of master job are defined

-bash-3.2$ jsub_status -h
usage: jsub_status <options> <target>
options:
  -h  show the help information
  -v  show JSUB version
  -all show the status of all the jobs the user ever submitted
target: could be JobID, JobGroup
```
Job Split and Submit

- YAMLParser
- Splitter -- Split jobs with various ways
- JobFactory -- Collect Job related parameters to generate Job submission scripts
- ExpParser -- Get runtime card or options for applications execution
- Workflow – Define runtime workflow to create RT scripts
- MonService – Collect and Record job info

Diagram:

- YAMLParser
- Splitter
- JobSteps
- ExpParser
- Workflow
- JobFactory
- MonService
- WMS
- jobProfile
- core components
- jobStatus DB
Workflow control

- Adopt ideas from “DIRAC workflow”
  - Highly extensible and reusable
- Classes for organizing run-time operations
  - Workflow
  - Step
    - Group of modules to complete a process, eg. preparation, simulation, reconstruction, complete
  - Module
    - Basic units for one operation or command, eg. checkenv
TaskId uniquely identify the task, created after submitted

Tasks can be monitored and controlled both through commands and web portal (next slide)

- Jsub_list, list all the tasks and the status
- Jsub_status <taskId>, show task progress and information
- Jsub_resubmit <taskId>, reschedule all or failed jobs
- Jsub_remove <taskId>, delete the whole task

Detailed task info collected from each stage of task processing

- Physicists can easily track down specific task
Task monitor and management (2)
Dataset management

- Dirac File Catalog (DFC) used to build metadata and replica catalog
- Dataset class is built to contact with DFC
  - Query dataset with conditions or name
  - Register dataset for output of jobs
- Users can simply use Dataset as input data of task, move bulks of data by dataset
Implementation and Extensions

- Develop in python
- Object-oriented programming are used to make it reusable
  - Inheritance, polymorphism, reflection
- Main code architecture
  - Cores
  - Interface and Common services
  - Plug-in for Exps
CEPC and JUNO case (1)

- **CEPC**

  ![CEPC diagram](image1.png)

  - `.stdhep` → `sim` → `rec`
  - `sim.slcio` → `rec.slcio`

- **JUNO**

  ![JUNO diagram](image2.png)

  - `sim` → `cali` → `rec`
  - `sim.root` → `cali.root` → `rec.root` → `.gdim`

---

### CEPC/JUNO

- **Step**
- OptParser
- JobFactory
- WorkFlow modules

### JSUB Core

- **parse, split, generate, submit, getStatus...**

---

### CEPC/JUNO Modules

#### Dirac/Condor modules
- CepcSimParser
  - `simu_macro`
  - `event_macro`
  - `_prepare()`
  - `parser()`
  - `generateOpts()`

#### JobFactory
- `properties`
- `createJobSet()`
- `setSubParam()`
- `setSpecialParam()`

---

### JUNO Modules

#### JobFactory
- `setSpecialParam()`

---

### ExpParser
- `opts`
- `parser()`
- `generateOpts()`

---

### CepcJobFactory
- `setSpecialParam()`

### JunoJobFactory
- `setSpecialParam()`

---

### CepcSimParser
- `simu_macro`
- `event_macro`
- `_prepare()`
- `parser()`
- `generateOpts()`

### CepcRecParser
- `reco_xml`
- `_prepare()`
- `parser()`
- `generateOpts()`
CEPC and JUNO case (2)

CEPC_test.yaml

```
Experiment:
  Name: cepc
JobSteps:
  - Type: Sim
    Executable: Mokka
    JobOption: /home/cc/suob/jsub/workfiles/sim
    ReturnData: true
  - Type: Rec
    Executable: Marlin
    JobOption: /home/cc/suob/jsub/workfiles/rec
    ReturnData: true

Splitter:
  Type: ByFile
  EventMaxPerJob: 20
  SeedStart: auto
  InputData: ./stdhep.list

Backend:
  Name: dirac
  Site:
    - CLOUD.IHEP.cn
    - GRID.JINR.ru
  JobGroup: suob160302_7
  OutputSe: WHU-USER
  OutputDir: /test/cepc/0302_7
```
In 2016, about 130K CEPC jobs have been successfully submitted through JSUB

Cumulative Jobs by UserGroup
34 Weeks from Week 52 of 2015 to Week 35 of 2016
Summary

- JSUB is a lightweight and extensible task submission and management tool
  - Small experiments or experiments in early stage can extend it for quickly access of distributed resources for massive production
- JSUB has been implemented in Python
  - Dirac and Condor supported
- Real cases of CEPC and JUNO experiments have shown its extensibility and simplicity
- CEPC usage in 2016 has proved its stability