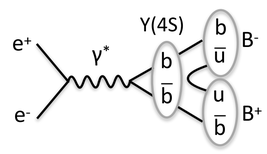
Improvement of analysis job efficiency at Belle II distributed computing system

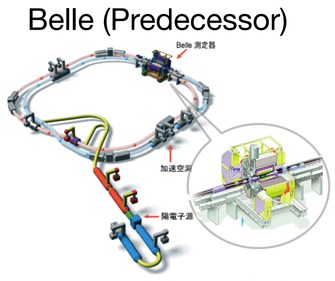
Hikari Hirata, Nagoya University

Belle II experiment

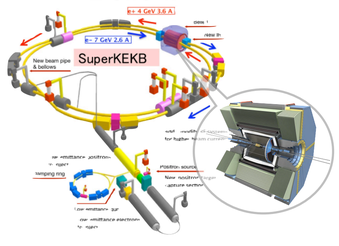
- Next-generation B factory experiment with an electron-positron collider at Tsukuba
  - -- Search CP violation from B decay, Lepton Flavor Violation, Hadron spectroscopy etc… (Competitive topics with LHCb)
    → Efficient physics analysis is important



Belle (Predecessor)



Belle II

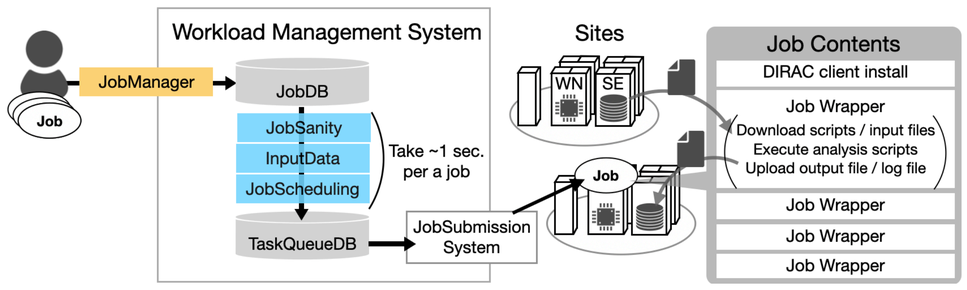


Peak luminosity	$2.1 \times 10^{34}\ \mathrm{cm^{-2}s^{-1}}$	$\times 30$	$6.0 \times 10^{35}\ \mathrm{cm^{-2}s^{-1}}$ (designed)
Integrated lumi.	$1\ \mathrm{ab^{-1}}$	$\times 50$	$50\ \mathrm{ab^{-1}}$

- -- Need huge computing resources to process and save data, produce massive simulation samples, execute analysis jobs and so on.
  → Computing requirement:
  O(10^5) CPU cores, O(100 PB) storage, O(1000) collaborators

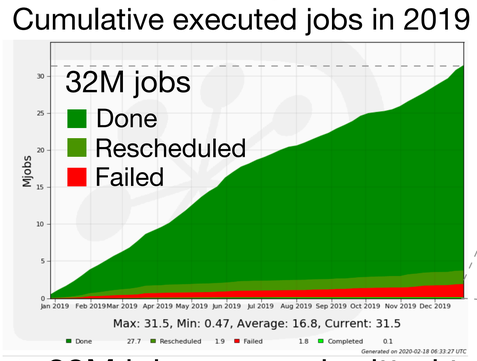
Belle II distributed computing (DC) system

- A system to distribute calculating/saving data to world-wide computing resource (cf. The predecessor used only 1 big computing site at KEK)
- Operate with following software
  - -- Processes are executed by Belle II Analysis Software Framework (basf2)
    → Distribute it to computing resources by Cern VM File System (CVMFS)
  - -- Software to interconnect among users and heterogeneous computing resources (DIRAC)
    → Extend it according to our requirement (called by BelleDIRAC)
    - Manage jobs and files on DC system
    - Submit jobs using basf2 to DC system
- Job execution workflow

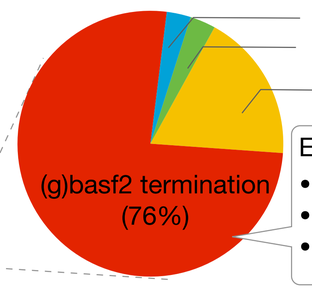


Job efficiency in 2019

Cumulative executed jobs in 2019



Failure reasons



Examples)
- Syntax error
- basf2 error
- Execution time is too long

- 32M jobs were submitted to our DC system, and 6% were failed
  → Main cause was problematic analysis script
  → Ratio of analysis jobs was the highest

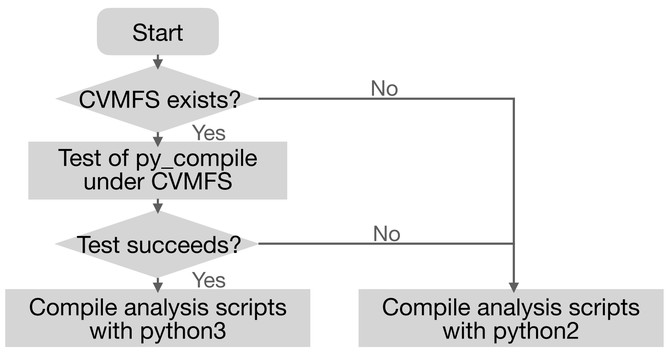
- If huge problematic jobs are submitted to DC system simultaneously, this makes job efficiency worse
  - -- It occupies worker nodes for a few minutes per 1 job to authenticate and load input data etc…
  - -- It triggers system trouble, and reduce available time of computing resources

Goal of my study

- Reduce failed jobs coming from problematic scripts
  → Firstly, implement following 2 features for analysis jobs

Python Syntax Check
at local environment

- Method: Compile scripts at local
  - -- Use a existing python module (py_compile)
    → Can detect simple syntax error without file execution. (e.g. open (), “”)
- Advantage of user: Realized a careless mistake quickly
- Prescription for different Python version:
  - basf2: python version 3.6
  - gbasf2: python version 2.7
    → Possibly kick out python3 features incompatible to python2 (e.g. fstring)
- Use python3 under CVMFS
  If not, use python3 under local environment



- Status: Complete deploying at production env.

testSteering.py

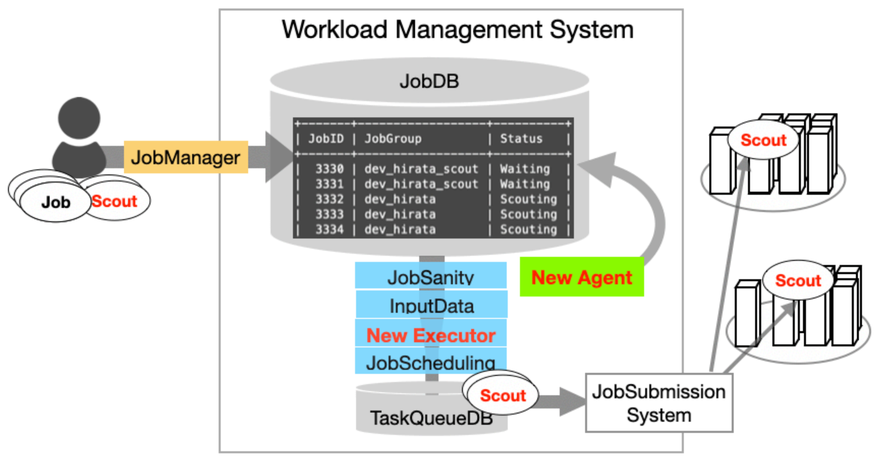
```
import basf2 as b2
test = 'Hello World'
print(f'string value: {test}')
# new feature incompatible python2
print('test) # should detect by this feature
```

Output

```
[hikari@cw02 gbasf2_dev]$ gbasf2 testSteering.py
************************************************
  File "testSteering.py", line 9
    print("test)
                ^
SyntaxError: EOL while scanning string literal
************************************************
Please fix the SyntaxError.
Are you sure to submit the project?
Please enter Y or N:
```

Execution of Scout Jobs
on Belle II DC system

- Method: Automatically submit test jobs (scout jobs), main jobs are submitted if the test succeeds.
  - Scout jobs: copied from main jobs
    - -- Has the same analysis script and the same input data → Can test under more realistic condition
    - -- Reduce processing events → Can reduce execution time for test
    - -- Information of scout jobs are registered as job parameters for main jobs
- New DIRAC module to develop
  - **New Executor** (Task for Job scheduling):
    - -- Stop to register main jobs to TaskQueueDB
  - **New Agent**
    (Component to perform actions periodically):
    - -- Monitor status of scout jobs
    - -- Change status of main jobs according to final status of scout jobs



- Workflow:
  1. Copy a part of main jobs as scout jobs, and Register all the jobs into JobDB
  2. The new executor filters out main jobs, while only scout jobs go through into TaskQueueDB
  3. Submit scout jobs to computing sites
  4. The new agent monitors status of scout jobs
  5. If scout is failed, change status of all main jobs by “Failed”, and do nothing any more
     If scout succeeds, submit main jobs to computing sites
- Status:
  - ☑ Develop all the new component at development server with single computing sites
  - ☑ Implement basic structure of the framework at certification server with multiple computing sites
  - ☑ Expose problems under certification server with multiple computing sites and solve them
  - ☐ Implement this framework to production environment
    - ✓ Step1: This framework is used only when user specify an option “--scout”
    - Step2: This framework is used by default

Test at production environment

171693548	Failed	Failed in scouting	Prod_test_scoutFail_1026	User
171693547	Failed	Failed in scouting	Prod_test_scoutFail_1026	User
171693546	Failed	Failed in scouting	Prod_test_scoutFail_1026	User
171693545	Failed	Failed in scouting	Prod_test_scoutFail_1026	User
171693544	Failed	Failed in scouting	Prod_test_scoutFail_1026	User
171693543	Failed	Application Finished With Errors	Prod_test_scoutFail_1026	UserScout
171693542	Failed	Application Finished With Errors	Prod_test_scoutFail_1026	UserScout

171693263	Done	Execution Complete	Prod_test_scout_1026	User
171693262	Done	Execution Complete	Prod_test_scout_1026	User
171693261	Done	Execution Complete	Prod_test_scout_1026	User
171693260	Done	Execution Complete	Prod_test_scout_1026	User
171693259	Done	Execution Complete	Prod_test_scout_1026	User
171693258	Done	Execution Complete	Prod_test_scout_1026	UserScout
171693257	Done	Execution Complete	Prod_test_scout_1026	UserScout

Summary

The Belle II experiment introduce a distributed computing system, but job execution efficiency was 94% in 2019. The main cause was problematic analysis scripts. We implemented a python syntax checker and execution of scout jobs so as to reduce the failed jobs. We introduced them experimentally. They will provide us efficient analysis on the system.