Practice on the BESIII offline software system (BOSS)

Weidong Li (李卫东), Tao Lin (林韬)

lintao@ihep.ac.cn
IHEP

BESIII National Day Workshop, Hohhot
Oct 1, 2023

Outline

Overview

- You will learn the following from this talk:
  - Understand the basics of software environment. Especially, you will know how to setup a new BOSS version from scratch.
  - Understand the basics of the data processing chain in BOSS. You will also know the data formats and how to read them.
  - Develop a simple Gaudi algorithm and understand how to use and configure Algorithm and Service in the Gaudi framework.
  - Understand the computing service including the interactive logon services and the batch system. You will practice on how to submit batch jobs.
- Prerequisites
  - You should know how to login `lxslc7.ihep.ac.cn` using SSH and setup the X11 forward for the visualization.
  - You should know the basics of Unix/Linux commands.
  - You should know the basics of programming using C/C++ or Python.

Familiar with the software environment

- If it is your first time when you login the `lxslc7` without any setup, you can only use the system commands.
- In order to let the system know where to find the commands and libraries, you can modify **environment variables**.
- You can check following environment variables:

```
$ echo $PATH
$ echo $LD_LIBRARY_PATH
$ echo $PYTHONPATH          # Python
$ echo $CMAKE_PREFIX_PATH   # CMake
$ echo $CMTPATH             # CMT
$ echo $CMTPROJECTPATH      # CMT
```

- You may already know how to modify these variables
  - For sh/bash, use `export`.
  - For csh/tcsh, use `setenv`.
- The setup of software environment is actually modifying the environment variables.

Setup the BOSS from scratch

- The setup of the BOSS software environment includes
  - Setup the external libraries and tools, such as GCC, CMT, ROOT, Geant4.
  - Setup the official BOSS software, including generators, simulation, reconstruction and analysis.
  - Setup your own working directory.
- The tool CMT is used to help users setup the environment variables easily and automatically.
  - In principle, you can setup the environment variables manually.
  - But it is recommended to use the official way to avoid the problem.
- The official BOSS software is located under `/cvmfs`:

```
/cvmfs/bes3.ihep.ac.cn/bes3sw/Boss # Contains all the releases
/cvmfs/bes3.ihep.ac.cn/bes3sw/Boss/7.0.6 # version 7.0.6

/cvmfs/bes3.ihep.ac.cn/bes3sw/cmthome # Contains all the setup related scripts
/cvmfs/bes3.ihep.ac.cn/bes3sw/cmthome/cmthome-7.0.6-Slc6Centos7Compat
```

Practice: the first time you setup BOSS basic envs (1)

- In this example, `/besfs5/users/$USER/boss-practices` is used.
  - If you don't have such directory, you can choose other directories.
- The version `7.0.6` is used.

step 1: create the working directory:

```
$ mkdir /besfs5/users/${USER}/boss-practices
$ cd /besfs5/users/${USER}/boss-practices
$ mkdir cmthome
$ mkdir 7.0.6 # your working directory for 7.0.6
```

step 2: copy the official cmthome to your local cmthome

```
$ cd cmthome
$ cp -r /cvmfs/bes3.ihep.ac.cn/bes3sw/cmthome/cmthome-7.0.6-Slc6Centos7Compat .
```

step 3: go to your cmthome

```
$ cd cmthome-7.0.6-Slc6Centos7Compat
```

Practice: the first time you setup BOSS basic envs (2)

step 4.1: edit `requirements` and use your work area

Find the following part in the file:

```
#Add your worarea to CMTPATH
#macro WorkArea "/home/bes/maqm/cvmfs/705"
```

Replace the previous part with follwoing part:

```
macro WorkArea "/besfs5/users/${USER}/boss-practices/${BES_RELEASE}"
path_remove CMTPATH  "${WorkArea}"
path_prepend CMTPATH "${WorkArea}"
```

step 4.2: edit `setupCVS.sh/.csh` and replace the CVS account

```
export CVSROOT=':pserver:bes3@koala.ihep.ac.cn:/bes/bes'
```

step 5: Setup the basic environment with GCC, CMT, CVS

```
$ source setupCMT.sh # if you are csh/tcsh users, please use the .csh
$ source setupCVS.sh
$ cvs login # You can use the public account if you don't have csv account yet.
$ cmt config
$ source setup.sh
```

Practice: the first time you setup BOSS basic envs (3)

step 6: Check the basic environments

```
$ cmt show macro WorkArea
# Package cmt_standalone v0 defines macro WorkArea as '/besfs5/users/${USER}/boss-pra
#
# Selection :
WorkArea='/besfs5/users/${USER}/boss-practices/${BES_RELEASE}'
$ cmt show macro_value WorkArea
/besfs5/users/lint/boss-practices/7.0.6 # This value should be same to
                                        # your working directory
$ ls $BesArea/TestRelease # Please remember the tag of TestRelease
$ echo $PATH | tr ':' '\n'
$ echo $LD_LIBRARY_PATH | tr ':' '\n'
$ echo $CMTPATH | tr ':' '\n' # your working directory should be on the top

$ which root # You will get the system root, but it is not expected.
/usr/bin/root
```

- However, this basic environment does not include the externals and BOSS yet. As you seen in the example, the ROOT is not from BOSS.

Practice: the first time you setup BOSS TestRelease

- In order to let all the necessary libraries available

step 1: go to your working directory

```
$ cd /besfs5/users/${USER}/boss-practices/7.0.6
```

step 2: checkout the TestRelease in your working directory

```
$ ls $BesArea/TestRelease # Confirm the tag of TestRelease from the output
$ cmt co -r TestRelease-00-00-95 TestRelease
```

- After you checkout the TestRelease, you need to source the setup scripts.

step 3: setup the TestRelease

```
$ cd TestRelease/TestRelease-00-00-95/cmt
$ cmt config
$ source setup.sh
```

step 4: Check the software environments

```
$ which root # You will see the right root
$ echo $LD_LIBRARY_PATH | tr ':' '\n' # You will see more libraries available
```

Practice: the next time you setup BOSS

- When you already setup such environment for a dedicated version, you don't need to repeat them every time.
- What you need is to `source` these existing scripts.
- You can test it with a new shell.

An example to setup an existing environment

```
cd /besfs5/users/${USER}/boss-practices/cmthome/cmthome-7.0.6-Slc6Centos7Compat
source setupCMT.sh # enable GCC and CMT
source setupCVS.sh # enable CVS
source setup.sh    # basic environments
cd /besfs5/users/${USER}/boss-practices/7.0.6/TestRelease/TestRelease-00-00-95/cmt/
source setup.sh    # TestRelease
```

- For convenient, you can put them into a shell script and source it.

A complete data processing chain in BOSS (1)

- As you already setup the BOSS software environment, then you can play with BOSS.
- A good start point is a complete data processing chain including physics generator, detector simulation, reconstruction and analysis.
- For the details in these job options, you can learn from other talks.

Practice: run the simulation, reconstruction and analysis

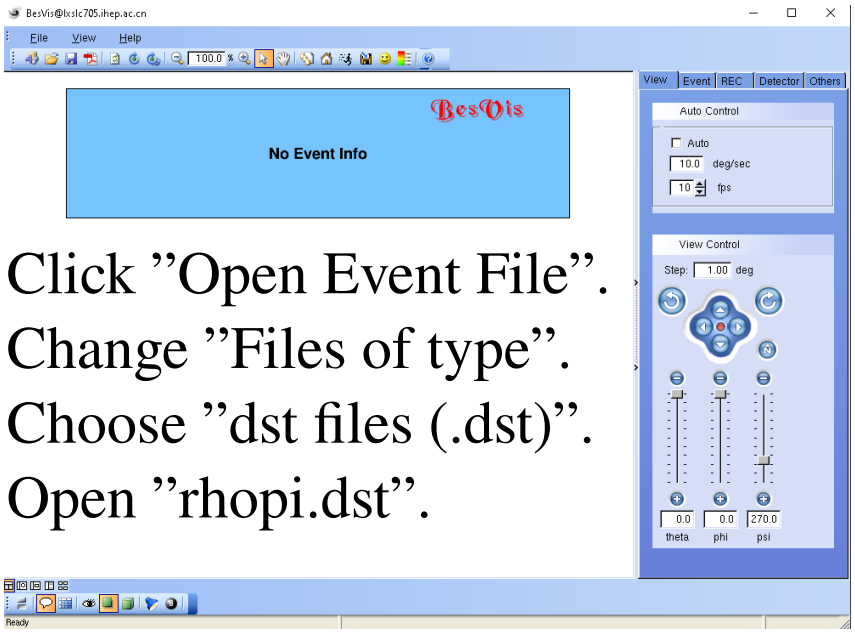
```
$ cd $TESTRELEASEROOT/run  # the necessary job options could be found here
$ boss.exe jobOptions_sim.txt # simulation
$ ls rhopi.rtraw # this is the output of simulaiton
$ $ boss.exe jobOptions_rec.txt # reconstruction
$ ls rhopi.dst # output of reconstruction
$ boss.exe jobOptions_ana_rhopi.txt
$ ls rhopi_ana.root # output of analysis
```

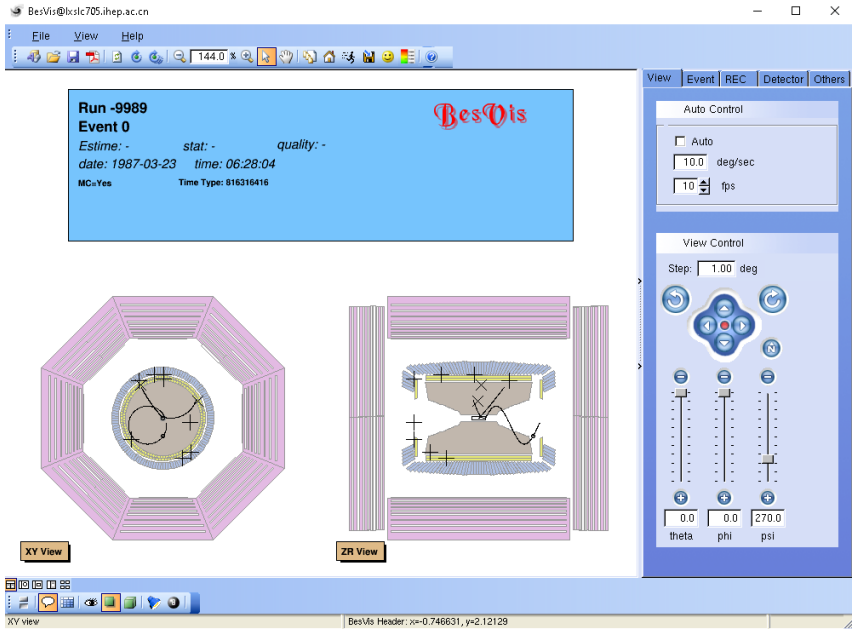
A complete data processing chain in BOSS (2)

- Another powerful tool is the event display, called `besvis.exe`.
- It supports the file format of rtraw (simulation) and rec/dst (reconstruction).

Practice: run the event display

```
$ besvis.exe
```





Job options: configuration of jobs

- In the previous practice, you already run several commands to do the simulation, reconstruction and analysis with three job options.
- A job option file configures what algorithms to be run and what services to be used.
- The syntax of job option is similar to C++.

An example: job option (you can save following as `myhello.txt`)

```
// This is a comment
// Include the other job options
#include "HelloWorldOptions.txt"
// Then you could modify their properties.
// The values could be scalar or vector. Change them as you like
HelloWorld.MyInt = 42;
HelloWorld.MyBool = true;
HelloWorld.MyDouble = 3.14159;
HelloWorld.MyStringVec = { "Welcome", "to", "Boss", "Framework", "Tutorial" };
```

Run the `myhello.txt`

```
$ boss.exe myjob.txt
```

Practice: Develop a Gaudi Algorithm

- In the previous practice, you should already know how to configure job options.
- In this practice, you will learn the underlying: how to develop an algorithm.
  - How to create a new package.
  - How to develop a Hello World Algorithm.
  - How to use Message Service in your algorithm.
  - How to access RAW data.
  - How to access DST data.

Create a new package in your working area

```
$ cd /besfs5/users/${USER}/boss-practices/7.0.6
$ cmt create MyAlg MyAlg-00-00-01 # MyAlg is the package name
                                  # MyAlg-00-00-01 is the tag name
$ cd MyAlg/MyAlg-00-00-01/cmt
```

Practice: write the `requirements` file

- After we create a new package, the first step is modifying the `requirements` file under `cmt`.

The content of `requirements`

```
package MyAlg # The package name

# The dependencies of this package
use BesPolicy    BesPolicy-01-*
use GaudiInterface GaudiInterface-01-* External

# The search path of include directories
include_path none
include_dirs $(MyAlg_root)/src $(MyAlg_root)/src/components

# Build a library called libMyAlg.so
# The source files are the following .cxx files under "src/"
library MyAlg MyAlg.cxx components/MyAlg_entries.cxx components/MyAlg_load.cxx
apply_pattern component_library
```

Practice: prepare the source code file

- The next step is creating the source code files under `src/`.

The files should be copied to your `src`

```
/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/src/MyAlg.h
/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/src/MyAlg.cxx
/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/src/components/*.cxx
```

You can copy them with following commands

```
$ cd /besfs5/users/${USER}/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/src
$ cp -r /besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/src/* .
```

Build the library and setup the environment

```
$ cd /besfs5/users/${USER}/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/cmt
$ cmt config
$ cmt make
$ source setup.sh
```

Practice: MyAlg.h

- In the header file, the class MyAlg is defined.

MyAlg.h

```
#ifndef MyAlg_h
#define MyAlg_h

#include "GaudiKernel/Algorithm.h"

class MyAlg: public Algorithm {

public:
    MyAlg(const std::string& name, ISvcLocator* pSvcLocator);

    StatusCode initialize();
    StatusCode execute();
    StatusCode finalize();

private:
    int m_myInt;
};

#endif
```

Practice: MyAlg.cxx

- In the file, the class MyAlg is implemented.

MyAlg.cxx

```
#include "MyAlg.h"
#include "GaudiKernel/MsgStream.h"

MyAlg::MyAlg(const std::string& name, ISvcLocator* pSvcLocator)
    : Algorithm(name, pSvcLocator), m_myInt(0)
{
    declareProperty("MyInt", m_myInt);
}

StatusCode MyAlg::initialize(){
    return StatusCode::SUCCESS;
}

StatusCode MyAlg::execute(){
    return StatusCode::SUCCESS;
}

StatusCode MyAlg::finalize(){
    return StatusCode::SUCCESS;
}
```

Practice: job option

- You can copy the job option from my working directory

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/share/MyAlgOptions.txt`

MyAlgOptions.txt

```
//load relevant libraries
ApplicationMgr.DLLs += { "MyAlg" };
//top algorithms to be run
ApplicationMgr.TopAlg = { "MyAlg/myAlgInstance" };

MessageSvc.OutputLevel      = 2;
MessageSvc.useColors        = true;

ApplicationMgr.EvtSel = "NONE";

// Number of events to be processed (default is 10)
ApplicationMgr.EvtMax = 10;

myAlgInstance.MyInt = 42;
```

Practice: print information using Message Service (1)

- Please update your `MyAlg.cxx`.

Update the `initialize` method

```
StatusCode MyAlg::initialize(){
    // Part 1: Get the messaging service, print where you are
    MsgStream log(msgSvc(), name());
    log << MSG::INFO << " MyAlg initialize()" << endreq;

    // Part 2: Print out the property values
    log << MSG::INFO << " * MyInt =    " << m_myInt << endreq;

    return StatusCode::SUCCESS;
}
```

Practice: print information using Message Service (2)

Update the `execute` method

```
StatusCode MyAlg::execute(){
    // Part 1: Get the messaging service, print where you are
    MsgStream log(msgSvc(), name());
    log << MSG::INFO << "MyAlg execute()" << endreq;

    // Part 2: Print out the different levels of messages
    log << MSG::DEBUG << "A DEBUG message" << endreq;
    log << MSG::INFO << "An INFO message" << endreq;
    log << MSG::WARNING << "A WARNING message" << endreq;
    log << MSG::ERROR << "An ERROR message" << endreq;
    log << MSG::FATAL << "A FATAL error message" << endreq;

    return StatusCode::SUCCESS;
}
```

- If you have any troubles with the code, you can have a look at my working area.
- I create a new tag for this practice.

  `/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-02`

Practice: print information using Message Service (3)

- After you modify the two methods, you need to rebuild the library.

Re-Build the library and setup the environment

```
$ cd /besfs5/users/${USER}/boss-practices/7.0.6/MyAlg/MyAlg-00-00-01/cmt
$ cmt config
$ cmt make
$ source setup.sh
```

- You can play with the job option and see what happens when you change the output level.

Configure the job option

```
//MessageSvc.OutputLevel      = 2;
MessageSvc.OutputLevel      = 5;

//MessageSvc.useColors        = true;
MessageSvc.useColors        = false;
```

Practice: Access Raw data (1)

- In the previous practices, you should be familiar with the Gaudi framework.
- The next step is reading raw data in the framework.
- Actually the BOSS already help you read the raw data and convert them to C++ objects.
- What you need to do is accessing these objects from the event data store.

Step 1: Add two lines in the `requirements`

```
# The dependencies of this package
use BesPolicy    BesPolicy-01-*
use GaudiInterface GaudiInterface-01-* External
# Following are NEW
use RawDataProviderSvc  RawDataProviderSvc-*    Event
use Identifier Identifier-* DetectorDescription
```

Practice: Access Raw data (2)

Step 2: Update the job options

```
//input data
#include "$RAWDATACNVROOT/share/ReadRawDatajobOptions_dataValid.txt"
#include "$OFFLINEEVENTLOOPMGRROOT/share/OfflineEventLoopMgr_Option.txt"

//input data file
RawDataInputSvc.InputFiles={
    "/bes3fs/offline/data/raw/round02/090307/run_0008093_All_file040_SFO-1.raw"};

//load relevant libraries
ApplicationMgr.DLLs += { "MyAlg" };

//top algorithms to be run
ApplicationMgr.TopAlg = { "MyAlg/myAlgInstance" };

MessageSvc.OutputLevel      = 3;

ApplicationMgr.EvtMax = 10;
```

- For your convenience, you can copy it from my working directory.

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-03/share/MyAlgOptions.txt`

Practice: Access Raw data (3)

Step 3: Add necessary headers in the `MyAlg.cxx`

```
#include "EventModel/EventHeader.h"
#include "RawEvent/RawDataUtil.h"
#include "MdcRawEvent/MdcDigi.h"
#include "TofRawEvent/TofDigi.h"
#include "EmcRawEvent/EmcDigi.h"
#include "Identifier/MdcID.h"
```

- For your convenience, you can copy it from my working directory.

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-03/src/MyAlg.cxx`

Practice: Access Raw data (4)

Step 4: Acces event headers in the execute

```
StatusCode MyAlg::execute(){
    MsgStream log(msgSvc(), name());
    log << MSG::INFO << "MyAlg execute()" << endreq;

    // Part 3: Get the event header, print out event and run number
    SmartDataPtr<Event::EventHeader> eventHeader(eventSvc(),"/Event/EventHeader");
    if (!eventHeader) {
        log << MSG::FATAL << "Could not find Event Header" << endreq;
        return( StatusCode::FAILURE);
    }
    log << MSG::INFO
        << " retrieved event: " << eventHeader->eventNumber()
        << " run: " <<  eventHeader->runNumber() << endreq;
```

- For your convenience, you can copy it from my working directory.

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-03/src/MyAlg.cxx`

Practice: Access Raw data (5)

Step 4: Acces MDC hits in the execute

```
// Part 4: Retrieve MDC digi
SmartDataPtr<MdcDigiCol> mdcDigiCol(eventSvc(),"/Event/Digi/MdcDigiCol");
if (!mdcDigiCol) {
    log << MSG::FATAL << "Could not find MDC digi" << endreq;
    return( StatusCode::FAILURE);
}

for (MdcDigiCol::iterator iter1 = mdcDigiCol->begin();
     iter1 != mdcDigiCol->end(); ++iter1) {

    if ((*iter1)->getTimeChannel() == 0x7FFFFFFF
     || (*iter1)->getChargeChannel() == 0x7FFFFFFF) { continue; }

    Identifier id = (*iter1)->identify();
    int layer = MdcID::layer(id);
    int wire  = MdcID::wire(id);

    log << MSG::INFO << " layer id: " << layer << " wire id: " << wire
        << " time_channel = "   << RawDataUtil::MdcTime((*iter1)->getTimeChannel())
        << " charge_channel = " << RawDataUtil::MdcCharge((*iter1)->getChargeChannel
        << endreq;
}
```

Access DST data

- DST data is a reduced reconstructed event data suitable for analysis.
- A good example is the `RhopiAlg`.
  - You already produce the `rhopi.dst` and analyze the data in TestRelease.
- Try to read the source code.

Option a: read the source code in the official BOSS

```
$ echo $RHOPIALGROOT
/cvmfs/bes3.ihep.ac.cn/bes3sw/Boss/7.0.6/Analysis/Physics/RhopiAlg/RhopiAlg-00-00-23
```

Option b: Checkout the source code

```
$ cd /besfs5/users/${USER}/boss-practices/7.0.6/
$ cmt co -r RhopiAlg-00-00-23 Analysis/Physics/RhopiAlg
$ cd Analysis/Physics/RhopiAlg/RhopiAlg-00-00-23/cmt
$ cmt config
$ cmt make
$ source setup.sh
```

Run the analysis in TestRelease

```
$ cd $TESTRELEASEROOT/run
$ boss.exe jobOptions_ana_rhopi.txt
```

Practice: Access DST/REC data (1)

- In order to practice on the access of the DST/REC data, you need to update the `MyAlg`.
- The `rhopi.dst` is the input of this practice.

Step 1: Add three lines in the `requirements`

```
use RawDataProviderSvc  RawDataProviderSvc-*    Event
use Identifier Identifier-* DetectorDescription
# Following are NEW
use DstEvent DstEvent-* Event
use EventModel   EventModel-*  Event
use EvtRecEvent EvtRecEvent-* Event
```

- For your convenience, you can copy it from my working directory.

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-04/cmt/requirements`

Practice: Access DST/REC data (2)

Step 2: Update the job options

```
//input data
// #include "$RAWDATACNVROOT/share/ReadRawDatajobOptions_dataValid.txt"
#include "$OFFLINEEVENTLOOPMGRROOT/share/OfflineEventLoopMgr_Option.txt"
#include "$ROOTIOROOT/share/jobOptions_ReadRec.txt"

//input data file
// RawDataInputSvc.InputFiles={
//     "/bes3fs/offline/data/raw/round02/090307/run_0008093_All_file040_SFO-1.raw"};

EventCnvSvc.digiRootInputFile = {"$TESTRELEASEROOT/run/rhopi.dst"};

//load relevant libraries
ApplicationMgr.DLLs += { "MyAlg" };
ApplicationMgr.TopAlg = { "MyAlg/myAlgInstance" };
ApplicationMgr.EvtMax = 10;
MessageSvc.OutputLevel = 3;
```

- For your convenience, you can copy it from my working directory.

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-04/share/MyAlgOptions.txt`

Practice: Access DST/REC data (3)

Step 3: Add necessary headers in the `MyAlg.cxx`

```
#include "EvtRecEvent/EvtRecEvent.h"
#include "EvtRecEvent/EvtRecTrack.h"
```

Step 4: Remove/Comment the MDC digi in the `MyAlg.cxx`

```
    // // Part 4: Retrieve MDC digi
    // SmartDataPtr<MdcDigiCol> mdcDigiCol(eventSvc(),"/Event/Digi/MdcDigiCol");
    // if (!mdcDigiCol) {
    //     log << MSG::FATAL << "Could not find MDC digi" << endreq;
    //     return( StatusCode::FAILURE);
    // }
    // for (MdcDigiCol::iterator iter1 = mdcDigiCol->begin();
    //      iter1 != mdcDigiCol->end(); ++iter1) {
    //     if ((*iter1)->getTimeChannel() == 0x7FFFFFFF
    //         || (*iter1)->getChargeChannel() == 0x7FFFFFFF) {
    //         continue;
    //     }
    // }
```

- For your convenience, you can copy it from my working directory.

`/besfs5/users/lint/boss-practices/7.0.6/MyAlg/MyAlg-00-00-04/src/MyAlg.cxx`

Practice: Access DST/REC data (4)

Step 5: Access Dst/Rec Event in the execute

```
SmartDataPtr<EvtRecEvent> evtRecEvent(eventSvc(), EventModel::EvtRec::EvtRecEvent);
if (!evtRecEvent) {
    log << MSG::FATAL << "Could not find "
                      << EventModel::EvtRec::EvtRecEvent << endreq;
    return StatusCode::FAILURE;
}
log << MSG::INFO <<"ncharg, nneu, tottks = "
    << evtRecEvent->totalCharged() << " , "
    << evtRecEvent->totalNeutral() << " , "
    << evtRecEvent->totalTracks() <<endreq;
```

Practice: Access DST/REC data (5)

Step 5: Access Dst/Rec Track in the execute

```
// Part 6: Retrieve Dst/Rec Track
SmartDataPtr<EvtRecTrackCol> evtRecTrkCol(eventSvc(),
                                          EventModel::EvtRec::EvtRecTrackCol);
if (!evtRecTrkCol) {
    log << MSG::FATAL << "Could not find "
                      << EventModel::EvtRec::EvtRecTrackCol << endreq;
    return StatusCode::FAILURE;
}

for (EvtRecTrackCol::iterator itertrk = evtRecTrkCol->begin();
     itertrk != evtRecTrkCol->end(); ++itertrk) {
    log << MSG::INFO << "track: "
        << " isMdcValid:       " << (*itertrk)->isMdcTrackValid()
        << " isTofTrackValid:  " << (*itertrk)->isTofTrackValid()
        << " isEmcShowerValid: " << (*itertrk)->isEmcShowerValid()
        << " isMucTrackValid:  " << (*itertrk)->isMucTrackValid()
        << endreq;
}
```

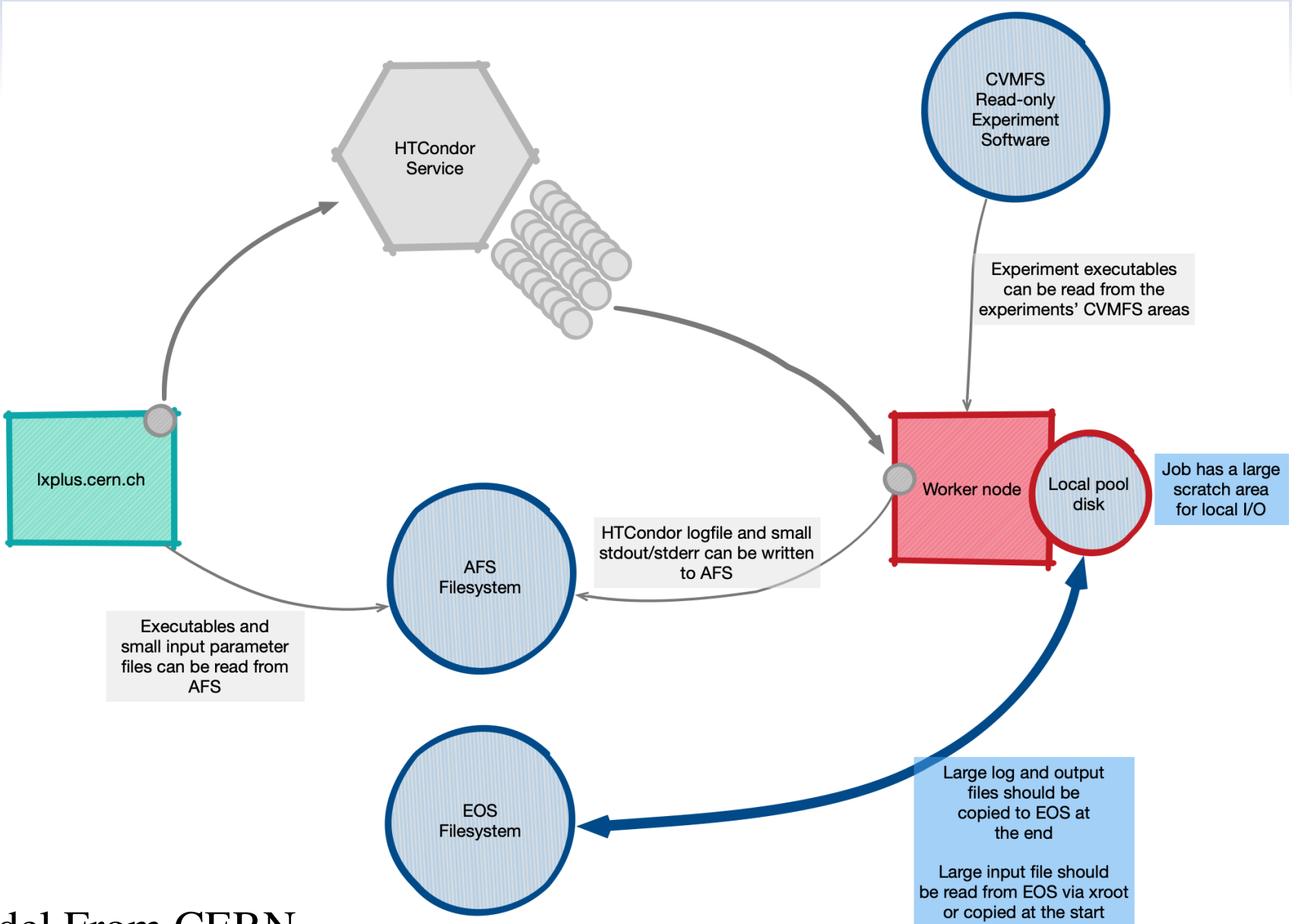
Batch jobs in computer cluster

- In the previous sections, we already learn the software part.
- Now, let's move to another part: computing.
- Computing is the underlying system providing the CPUs, Storage, Network and other IT services.
- Any bottleneck in the sub-system will cause the jobs stop working.
- In this section, you will learn
  - Understand how data flow of the job.
  - Know how to submit a simple shell script.
  - Know how to submit a BOSS job.

Batch processing

- It is not possible to use a single machine to process all the data.
- Batch processing: jobs are run without user interaction. Users submit jobs to the scheduler and the jobs are processed in the worker nodes.

The data flows in computer cluster (1)



Model From CERN

The data flows in computer cluster (2)

- At IHEP cluster, the data flows are quite similar.
- Nodes:
  - Login nodes: `lxslc7.ihep.ac.cn`
  - Work nodes: the jobs are scheduled by HTCondor and run in the work nodes.
- Storage (usually shared file systems):
  - `/afs`: your home directory. It is read-only in the worker nodes.
  - `/workfs2`: user directory with backup. 5GB per user. It is ready-only in the worker nodes.
  - `/besfs5`: user directory without backup. 50GB per user. It is read-write in the worker nodes.
- Note: submit jobs in the read-write storage. Otherwise the jobs will be failed.
- For more details, see `http://afsapply.ihep.ac.cn/cchelp/zh/`

Job management with HepJob

- Computing Center provides a set of tools to manage the jobs.
- HepJob supports following features:
  - Job submission
  - Job query
  - Job deletion
- HepJob also supports:
  - Specify the OS using container technologies.
  - Specify the required memory of job. This is very useful when you need a large memory to run jobs.
  - Re-schedule a hold jobs. For example, when the memory of a job is not enough, the job will be hold. You can edit the requirements and re-schedule the job again.
  - Submit similar jobs at once.

Setup HepJob

```
export PATH=/afs/ihep.ac.cn/soft/common/sysgroup/hep_job/bin:$PATH
```

Practice: Submit jobs

- Let's prepare a simple shell script first. Note, if you are using a shell script, make sure the shabang is correct. Otherwise, HTCondor will treat the script as a "sh" script, not "bash".

Prepare script `myjob.sh` under `TestRelease/run`

```
$ cd $TESTRELEASEROOT/run # assume it is under /besfs5
$ touch myjob.sh
$ chmod +x myjob.sh
```

The script `myjob.sh`

```
#!/bin/bash

hostname
pwd
```

Submit the script `myjob.sh`

```
$ hep_sub -g physics myjob.sh
1 job(s) submitted to cluster 40427430.
```

Practice: Query jobs

- To know the status of your submitted jobs, you can query them.

Query jobs via Job ID

```
$ hep_q -g physics -i 40427430 # this ID is from previous
```

Query jobs belong to you

```
$ hep_q -g physics -u $USER
```

Query the hold jobs (if you want to know the hold reason)

```
$ hep_q -g physics -hold
```

Practice: Reschedule jobs

- Only if your jobs are hold, you need following commands.

Edit and reschedule job via Job ID

```
$ hep_edit -g physics -m 4000 40427430 # the required memory is 4G
$ hep_release -g physics 40427430
```

Reschedule all the job belong to you

```
$ hep_release -g physics -a
```

Practice: Delete jobs

Delete job via Job ID

```
$ hep_rm -g physics 40427430
```

Delete all the job belong to you

```
$ hep_rm -g physics -a
```

Practice: Submit a BOSS job or a ROOT job

- There is a wrapper for the BOSS job.

Submit a BOSS job

```
$ boss.condor -g physics jobOptions_sim.txt
1 job(s) submitted to cluster 40427947.
```

- For the ROOT job, make sure add options "`-b -q`" to the root command.

An example of ROOT job in the script `myrootjob.sh`

```
#!/bin/bash

root -l -b -q $ROOTSYS/tutorials/hist/fillrandom.C
```

Submit a ROOT job

```
$ hep_sub -g physics myrootjob.sh
1 job(s) submitted to cluster 40427975.
```

Summary

Summary

- This is a quick start for you. Please try all the examples to have a better understanding.
- Please refer to the documentations when you want to know the details.
- If you find any problems in the slides, just let me know.

References

- BOSS wiki page:

  `https://docbes3.ihep.ac.cn/~offlinesoftware/index.php/Main_Page`
- The old Gaudi documentation:

  `https://gaudi-framework.readthedocs.io/en/latest/old/GDG.html`
- Computing Center Documentation: `http://afsapply.ihep.ac.cn/cchelp/zh/`

Thank you!