# Using SNiPER in BESIII analysis

XIA Xin, LI Teng

Shandong University

June 6 2016

# Outline

# Outline

# Motivation

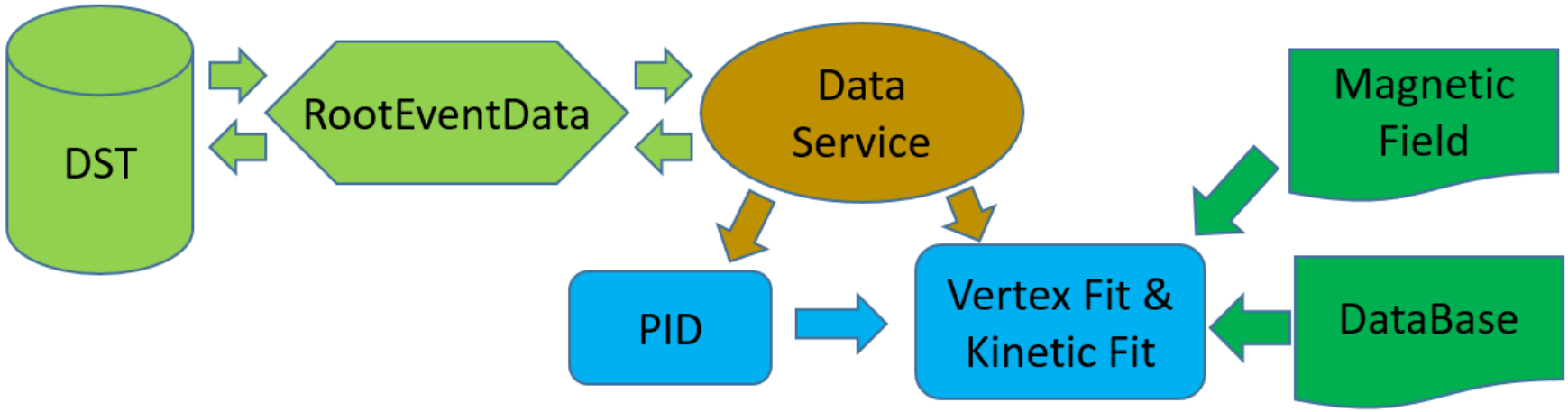
1. **Rapidly increased data samples in BESIII**
   With time increases, statistics increases rapidly in BESIII experiment. Faster data processing method will help a lot.
2. **SNiPER is light-weighted, effective and flexible Framework**
   SNiPER has been used in JUNO experiment in recent years, now also in LHAASO. It is another choice for physics data analysis.
3. **Analysis the BESIII data under SNiPER framework**
   Optimize data structure with Smart Reference.
   Transplant the BESIII analysis codes to SNiPER.
4. **Preparing for the parallel computing in the near future.**

BOSS version: 7.0.0

# Outline

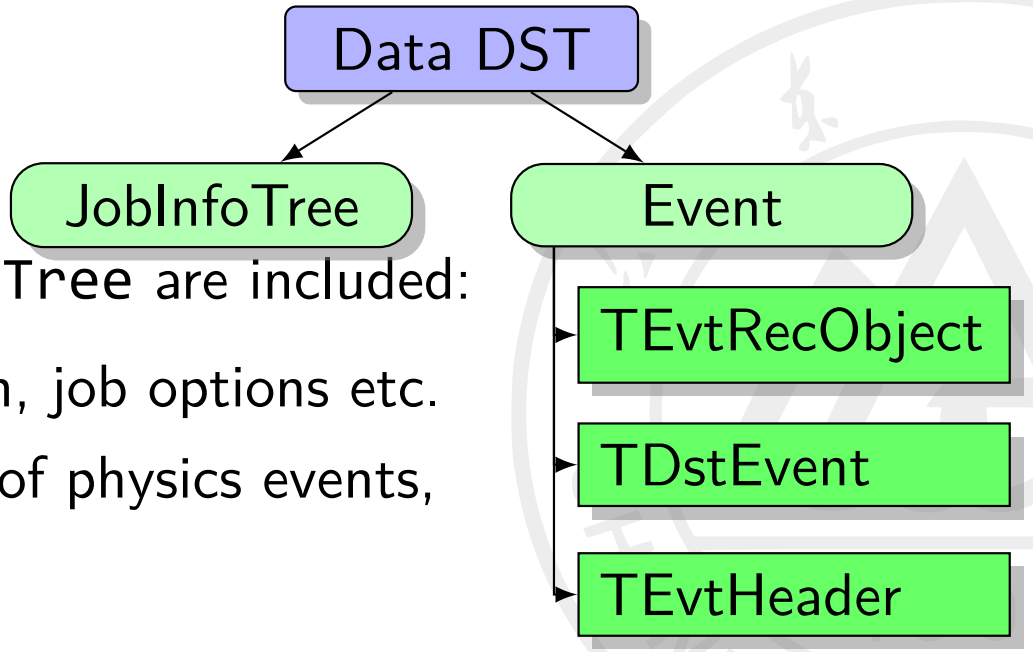
# A BESIII analysis workflow

1. Data Service retrieves data from DST via `RootEventData`
2. PID will determine the type of particle for every track
3. Vertex fit calculate the vertex position with informations of the magnetic field and primary vertex from database.



# Data model in BOSS

In BOSS, DST files are stored in the form of TTree. All the classes in branches are derived from TObject.



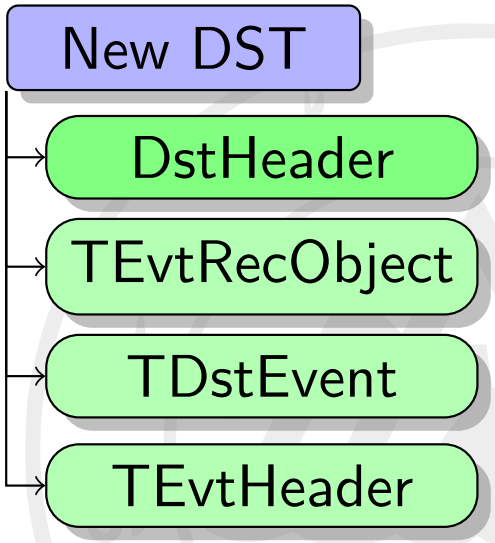

Take data files as example, two TTree are included:

1. JobInfoTree: BOSS version, job options etc.
2. Event: all the informations of physics events, part interested.

When user read the files, TObject will be converted to the EventObject by Gaudi.

# New data model under SNiPER

1. `RootEventData` is migrated to `SNiPER` for direct access to the `TObjects` in DST.
2. Use corresponding `TObjects` instead of `EventObject`.
3. Additional branch named `DstHeader` is added, consists of
   1. Parameters for fast pre-selection,
   2. Smart reference to the other trees.
4. Access to data with `DataStoreMgr`.




Smart Reference provides reference from header to events.
Pre-selection can be done with header itself for lazy-loading.

# DST conversion

Since no big changes in structure, the conversion will be easy.

1. `ReBuildDST` is developed to convert the `BESIII` DST to the `SNiPER` one.
   - Read the old DSTs in the normal `ROOT` way,
   - Add parameters to `DstHeader`,
   - Write to new DSTs with `DataStoreMgr`, it will do the rest of jobs automatically.
2. Also provide a Service `BesInputSvc` to access to the `BESIII` DST directly.
   - Parsing the DST file list,
   - Some useful functions to retrieve informations from files.

# Outline

# Particle identification (ParticleID)

`ParticleID` is the process of using informations left by one particle passing through the detectors to identify the type of particle.

- Using dE/dx, TOF, TOFQ, Emc and Muc system

1. Calculate probability for every physical track
2. In `BOSS`, `EvtRecTrack` is a logical track connecting the sub-detectors
3. `EvtRecTrack` is the input for `PID`.

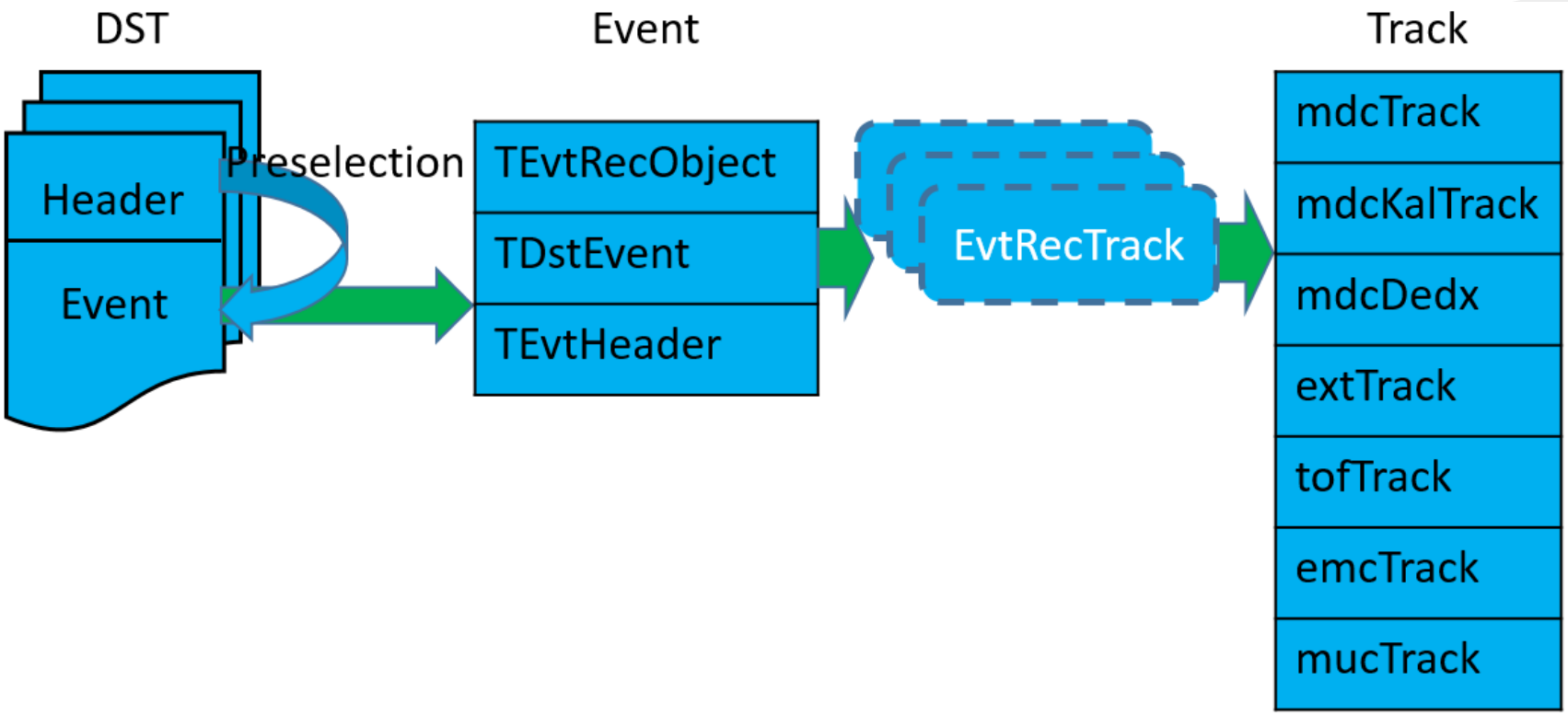
# ParticleID in SNiPER

The main problem in new framework is how to get the correct sub-detectors information in one track.

- Need `EvtRecTrack` to manager the sub-tracks
- Initialized from `TEvtRecTrack` and current scope
  - `TEvtRecTrack` stores the trackIDs of sub-detectors in one reconstructed track
  - Scope is used for getting informations like run number
- Connect the sub-tracks with trackID
- Use `TObjects` directly in `ParticleID` code
- Tool `AbsCor` is added to correct the energy of EMC Shower

# Data flow

For analysis, the data flow looks like



# Vertex fit

Vertex fit computes the best estimate of the vertex parameters (such as position) with a given set of tracks.

**Necessary parameters in `VertexFit` class,**

1. Several MDC tracks after `PID`,
2. Primary Vertex information from database,
3. Magnetic field information.

**To be used in `SNiPER`,**

1. New constructors of `WTrackParameter` are added
2. Run number is passed to `VertexFit` for database accessing
3. `MagneticField` and `DatabaseSvc` are detached from `Gaudi`, rewrote as pure C++ class

# Outline

# Performance Test

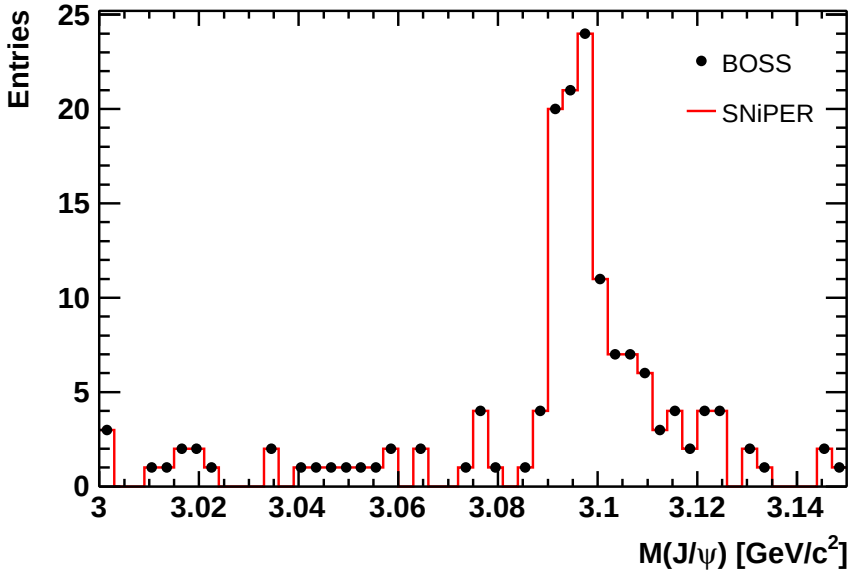
- Tested with the analysis $e^+e^- \to \pi^+\pi^- J/\psi$, $(Z_c(3900))$
- Data sample at CM energy of 4.26 GeV, 24 files used in total.

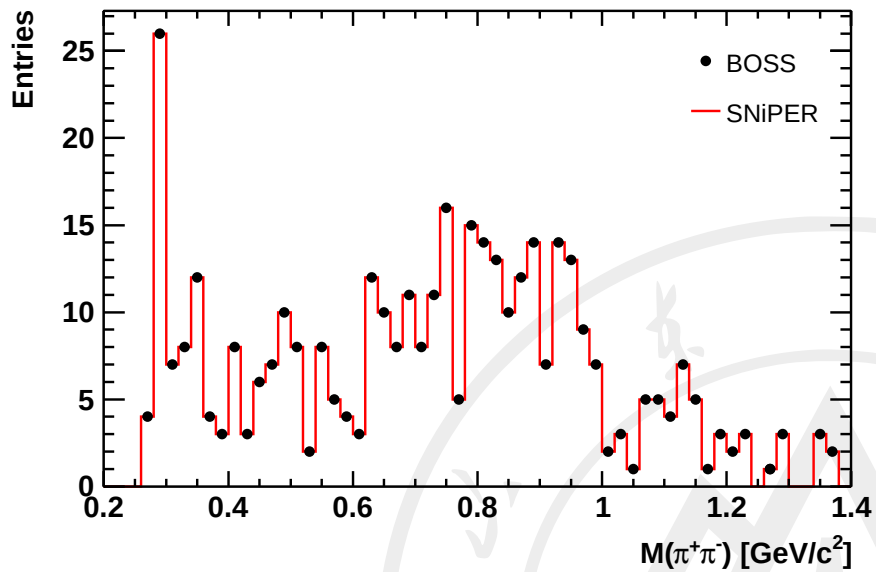
| NO. | Selection | BOSS | SNiPER |
|---|---|---|---|
| 1 | charged tracks | 4771222 | 4771222 |
| 2 | nGood==4 | 10028 | 10028 |
| 3 | bnGam>=1 | 1352 | 1352 |
| 4 | Pass pid | 1274 | 1274 |
| 5 | Pass 4C | 252 | 252 |
| 6 | save result | 27 | 27 |
| 7 | cms energy | 4.258 | 4.258 |

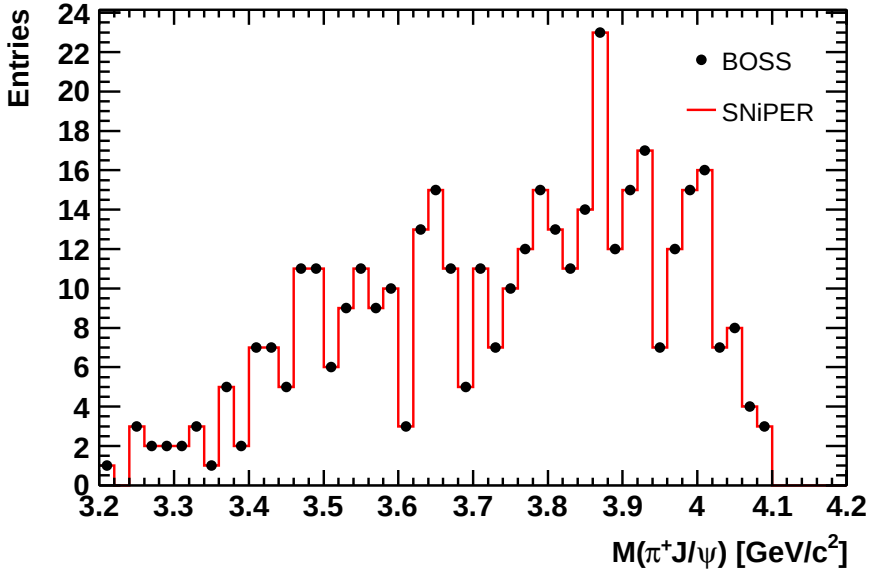
Table: The number of selected events under these two frameworks

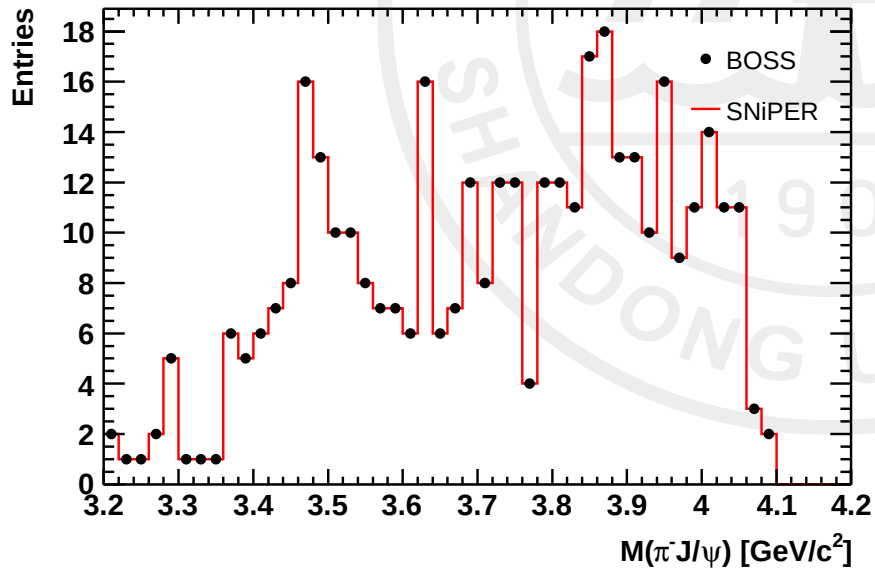
The selected results are exactly the same.
Further test is comparing the physics results.

# Mass spectrum comparison

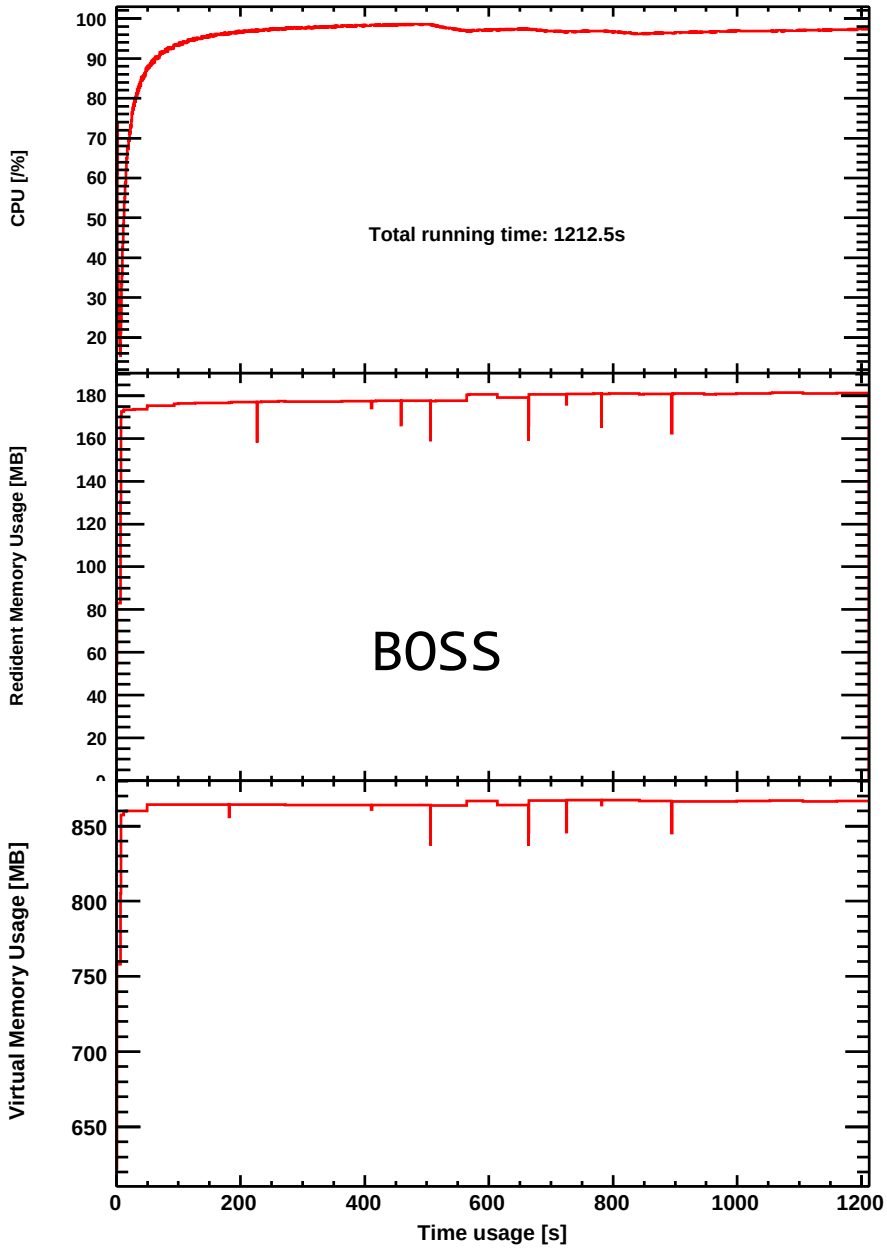


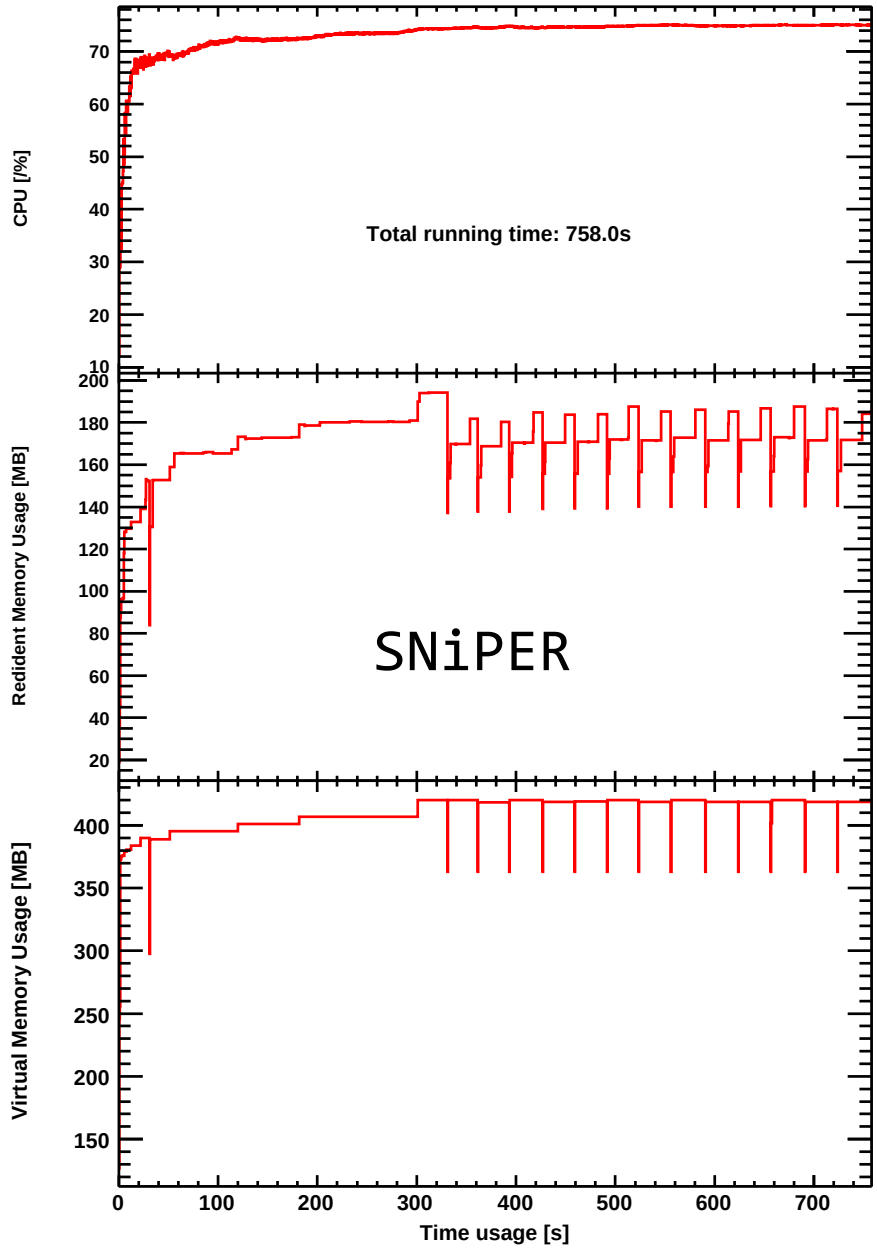



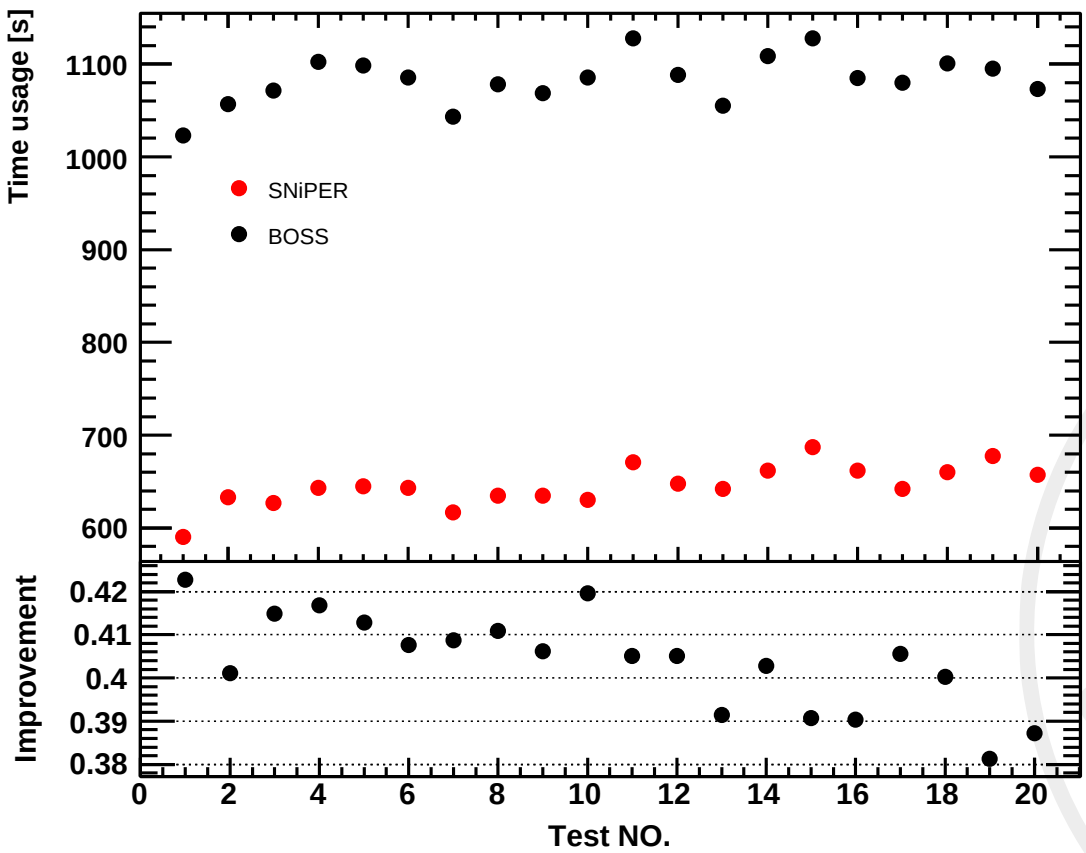



# Performance comparison



# Time consuming




- Tested 20 times, with 20 different files each time.
- SNiPER is about 40% faster than BOSS.

---

Improvement $= (T_{BOSS} - T_{SNiPER})/(T_{BOSS})$

# Outline

# Summary

1. Designed new data model for BESIII DST, which can be used under SNiPER.
2. Migrated the ParticleID and VertexFit to SNiPER for physical analysis.
3. Tested with a real analysis, the SNiPER version worked properly, and can be 40% faster than BOSS version.

# Summary

1. Designed new data model for BESIII DST, which can be used under SNiPER.
2. Migrated the ParticleID and VertexFit to SNiPER for physical analysis.
3. Tested with a real analysis, the SNiPER version worked properly, and can be 40% faster than BOSS version.

Thanks for your attention!