Physics Analysis with BPAT

Mao Yajun, Wang Siguang
Peking Univeersity

Zhang Xueyao*, Ding Weimin
Shandong Univeersity

Outline

- Physics Analysis in General
- BPAT（BESIII Physics Ananlysis ToolKit）
- Summary

Physics Analysis in General

- **Charged tracks** — *Helix parameters and error matrix*
- **Neutral tracks** — *EMC shower information*
- **PIDs** — *Tof, dE/dX, MuID, …*
- **Tracks in Sub-detectors** — *Partial tracks in sub-det…*

Physics Analysis

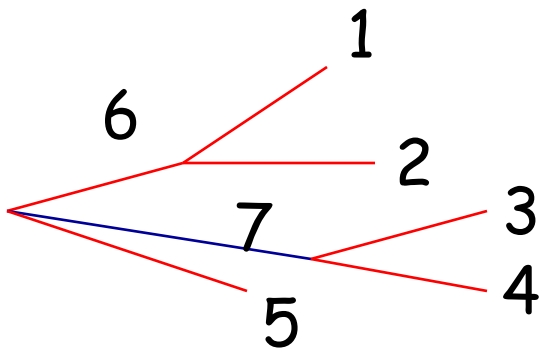
NTUPLE

- Run#
- Event#
- … …
- Particle#1 info
- Particle#2 info
- … …
- Particle#N info

- Mother particle info
- Grand-Mother particle info
- … …

What's the Problem?

- **Straightforward between DST & NTUPLE**
  - *Only good for simple case,*
  - *what if we need some additional information?*
  - *what if we want to check other accompany particles?*
  - *No good means to find some deep hiding bugs*



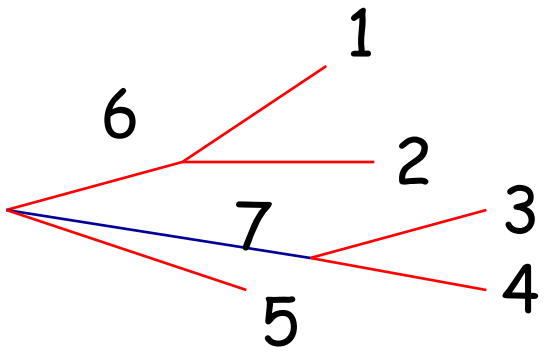
- **Broken data structure**
  - *Easy to make mistake*
  - *Hard to share analysis code, low efficiency…*

A Possible Solution

- **Split the analysis to several steps**
  - *Take the analysis as event filtering*
  - *Keep the whole event in each steps, not only those selected tracks*
- **Use a consistent data structure**
  - *Code re-usable to other analysis*
  - *Less bugs for more people's use*
  - *Higher efficiency, could concentrate in physics, rather than programming*

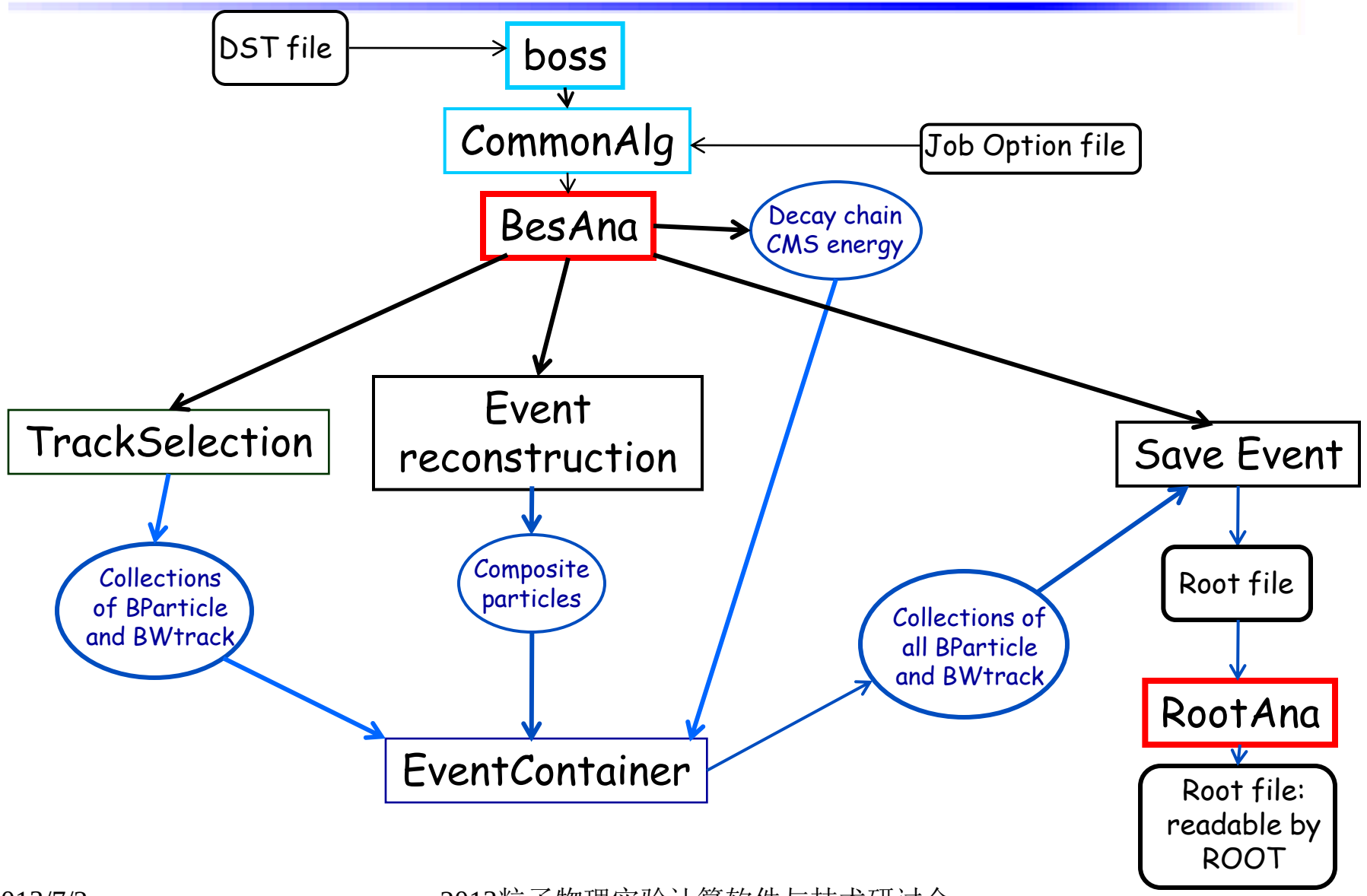
From Track To Particle?

- **A track is a measured particle**
  - *such as particle 1, 2 ,3 ,4 ,5*
  - *but a particle doesn't to be a track for example, particle 6, 7*

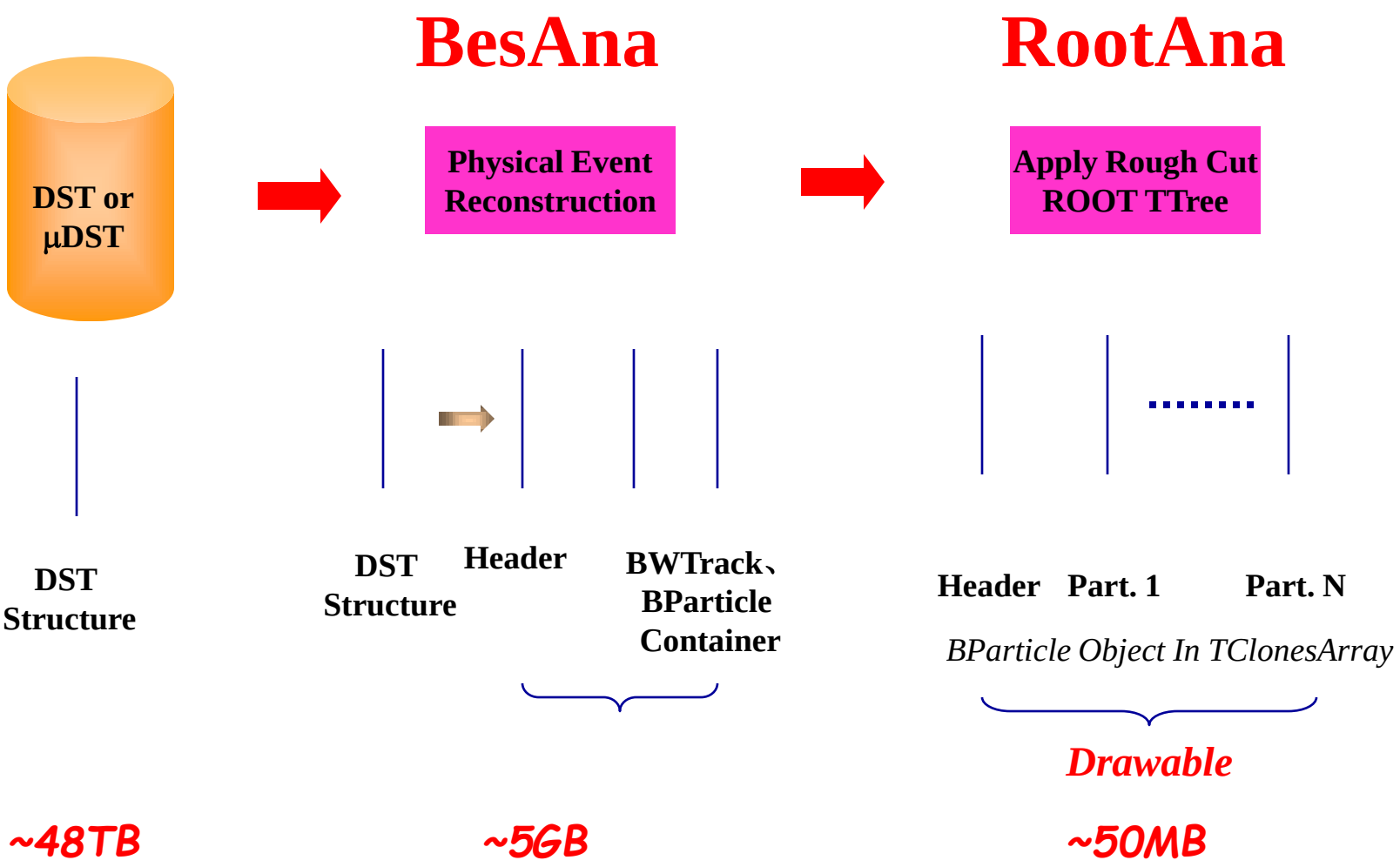


- **Define a class for particle**
  - *BParticle : represents measured particle and reconstructed particle;*
  - *Using index to describe relationship between different particles.*

BESIII Physics Analysis Toolkit (BPAT)



Data Flow with BPAT



The BParticle Class

Class represents a particle in BESIII

```
class BParticle : public TObject{
 protected:
  Int_t             fIndex;          // index of this particle in current event
  Int_t             fPdgCode;        //  PDG code of the particle
  Int_t             fMask;           // bit mask to handle the particle status and reconstruction options
  Int_t             fChannelNo;      // decay channel number
  Int_t             fTrackID;        // track id of this particle in dst track list
  TLorentzVector    fP;              //  4-momentum of this particle
  TLorentzVector    fV ;             //  primary vertex if it's a measured particle
  Double_t          fProbability;    // probability to be identified as particle with current type
  Double_t          fDecayLength;    //  proper decay length of the particle
  Double_t          fChisqKF;        //  Chisquare of kinematic fit
  Double_t          fChisqVF;        //  Chisquare of vertex fit

   std::vector<Double_t>  fUserVar;          // User Defined Variables;
   std::vector<Int_t>  fDaughterIndex;       // indices of Daughters;
   std::vector<Int_t>  fDaughterAllIndex;  // indices including grand daughters;
   std::vector<Int_t>  fMotherIndex;         // indices of mothers

   std::map<Int_t, SelectionCut *> fSelectionCut;  //!

 // use PDG database to handle constants of elementary particles
  TParticlePDG*      fParticlePDG;                 //! reference to the particle record in PDG database
```

The BWTrack Class

wrapping of WTrackParameter used for fitting and track update

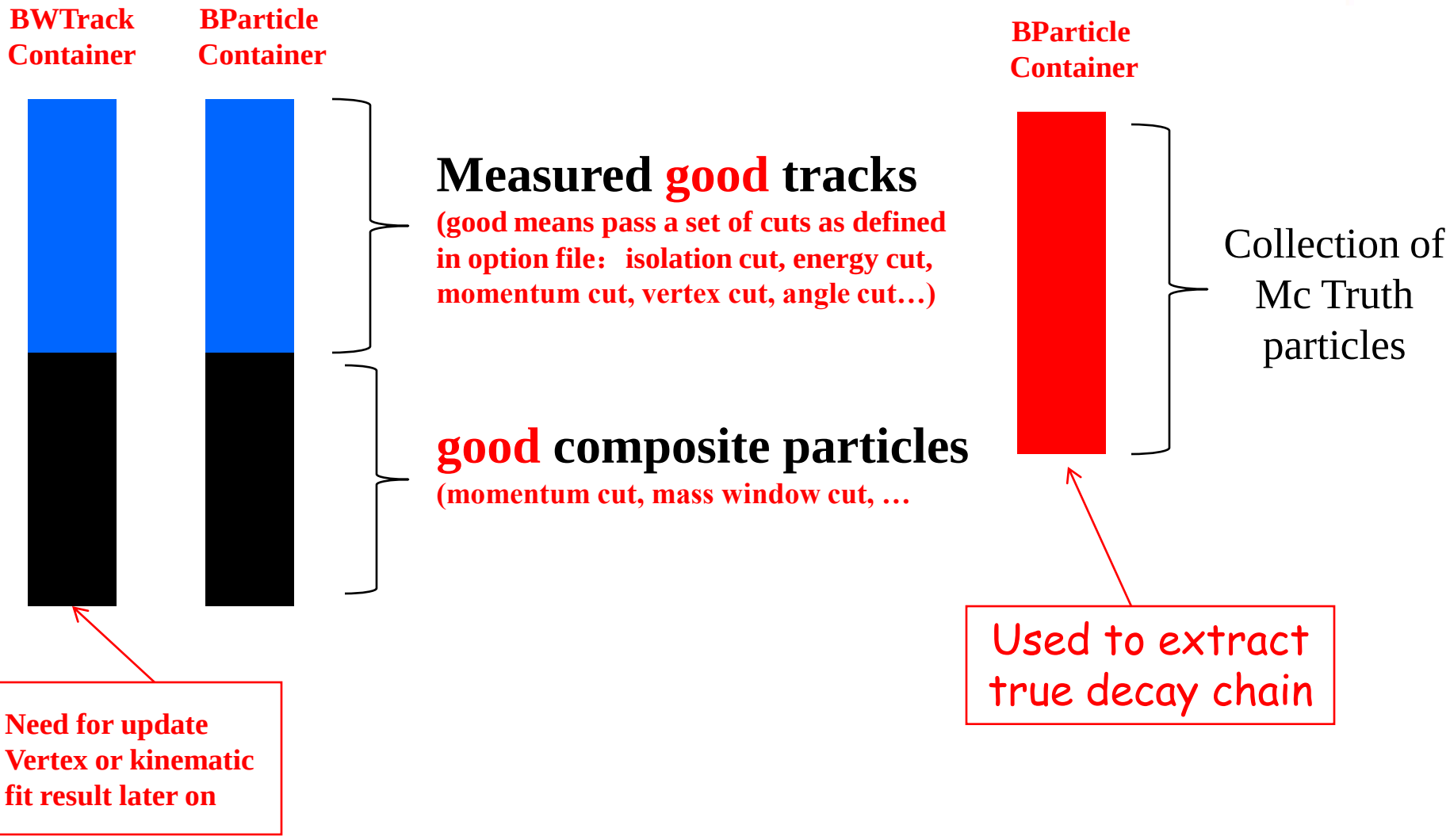
```
class BWTrack : public TObject{
 protected:
 // data members in WTrackParameter class
    Int_t    m_charge;          // charge of particle
    Int_t    m_type;            // type of particle

    Double_t m_w[7];            // W parameter
    Double_t m_Ew[7][7];        // error matrix

    Double_t m_plmp[7];         // W parameter in phi lambda format
    Double_t m_Vplm[7][7];      // error matrix

    Double_t m_mass;            // mass of particle
    Bool_t   m_massInvariable; // Is mass invarible

    // extra information for this track, added by maoyj on Jun 13, 2009
    Int_t m_Index;              // index of this track
    Int_t m_TypeFit;            // fit type of this track; 0 = vertex fit, 1 = kinematic fit

    Double_t m_Pid[5];          // probability of pid for this track

    vector<BWTrack *>   m_Daughters;    //-> daughters of this tracks after fitting
    vector<BHelixPar *>  m_HelixPar;    //-> helix parameters for 5 type of particles
```

EventContainer

Classes to handle collections of BParticle and BWTrack, decay chain, job options, CMS energy…

- DecayChannel: represents a decay channel;
- BDecayChain: Handles a list of channels and the relationship between channels;
- CMS : 4-momentum of collision beam;
- CheckEvent : check to see a event satisfy some criteria for processing and saving;
- EventContainer: manages the collection of BParticle and BWTrack, decay chain, CMS energy, …

EventContainer :Particle Collection



EventContainer :Decay Chain Def.

- BDecayChain class manages a list of decay channels represented by DecayChannel class;
- A decay chain is defined by joboption:

Example: $J/\psi \to \Lambda\bar{\Lambda}, \quad \Lambda \to p\pi^-, \quad \bar{\Lambda} \to \bar{p}\pi^+$

```
Common.DecayMode ={" J/psi  ->  Lambda0  +  Lambda0_bar ,
                     J/psi   masswin  2.0  4.0,
                      kmfit  proton pi- antiproton pi+  ! cms ;
Lambda0  ->  proton  +  pi-,
                     Lambda0  masswin 0.6 1.4 vtxfit 2,  proton pid 0;
Lambda0_bar -> antiproton + pi+,
                      Lambda0_bar masswin 0.6 1.4 vtxfit 2, antiproton pid 0; "};
```

Basic rules: ① use ; to separate a decay
② use , to separate a particle or KF option
③ use ! to separate option within KF
④ use @ to tell KF applied to which particles
⑤ use key to tell which option to be set

EventContainer : CMS energy

- CMS class manages a the 4-momentum of collision beam;
- Set by joboption:

```
Common.CMS = {"ecms 3.773 ESpread 0.00015"};
```

Or read from data base for real data

EventContainer : CheckEvent

- CheckEvent class checks is a event satisfies some criteria for processing and saving:

  If a event is to be processed, it must meet the requirement:

  - Number of good charged and neutral tracks
  - The required particle types

  If a event is to be saved, it must meet the requirement:

  - The specified particle is reconstructed, or
  - the specified channels are reconstructed

- Criteria set by joboption and decay chain

```
Common.CheckEventOption = {"
  numExtraTrkCharged: 0;     /* # of extra charged tracks allowed */
  numExtraTrkNeutral: 10;    /* # of extra neutral tracks allowed */
  finalTracks: pi+ pi- proton antiproton gamma gamma;
  SaveParticles: psipp;
  SaveChannels:0;
"};
```

BesAna

Convert track to BParticle and BWTrack、combine particles to composite particles by performing vertex fit and kinematic fit etc)

- Select good charged and neutral tracks; identify charged particle type; convert tracks to BParticle and BWTrack and store them into event container;
- Check if the event satisfies good tracks and PID requirement;
- Do event reconstruction: reconstruct each channel in defined decay chain;
- Save the event if it satisfies the save event criteria.

BesAna : Track selection

DoTrackSelection class performs the good track selection and PID and converts the track to BParticle and BWTrack, and store them in event container.

```
Common.TrkSelOption = {"
  isoAngle: 10;
  betweenGoodCharged:0;
  applyCosThetaCut:1;
  BarrelEmcEcut: 0.025;
  EndCapEmcEcut: 0.050;
  EmcTDCTimeCut: 14;
  cosThetaCut: 0.93;
  deltaRCut: 1.0;
  deltaZCut: 10.0;
  useSwamHelix: 0;
  storeHelixPar:0;
"};
```

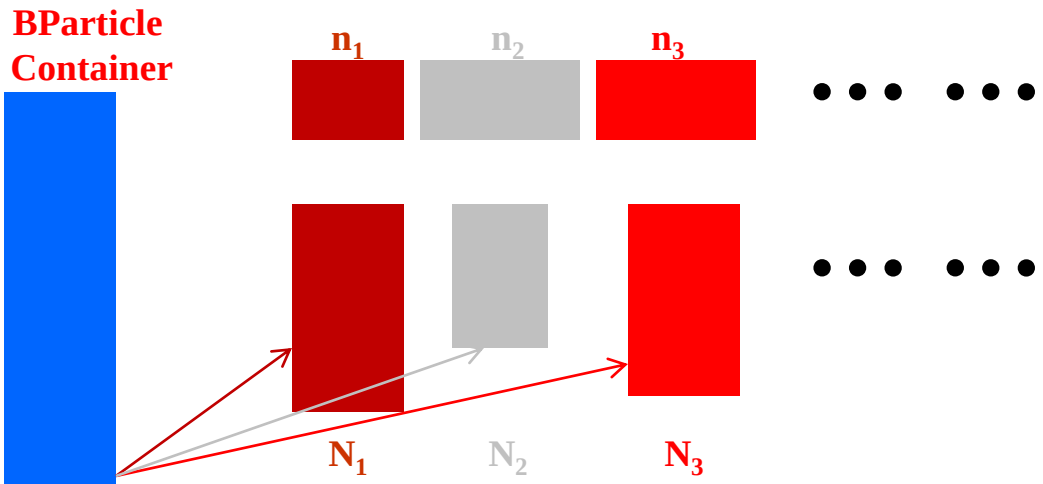
```
Common.PidOption = {"
   PidSys: useTof1 useTof2 useDedx;
   PidCase: onlyPion onlyKaon onlyProton;
   PidMethod: Probability;
   ChisqCut: 4;
   posiType: -1;
  negaType: -1;
  unidType: -2;
  minProbCut: 0.0
"};
```

BesAna : Event reconstruction

Event reconstruction is done by function BesAna::Reconstruction() by reconstruct each channel defined in decay chain

```
// event reconstruction option
Common.EvtReconOption = {"
  PreSelect: 0;             /* Flag to perform pre-selection */
  massCutBeforeVtxFit:1;     /* Cut on the invariant mass before doing vertex fit */
  vtxChisqCut: 1000.0;       /* Cut on the chi squared of the vertex fit */
  secondVtxChisqCut: 1000.0; /* Cut on the chi squared of the second vertext fit */
  vtxUpdateTrk:0;           /* flag to update track parameters after second vertex fit */
  maxChisqKF: 500.0;        /* Cut on the chi squared of kinematic fit */
  useUpdatedTrk: 1;         /* Flag to use updated track parameters in KF */
  useKalmanKmFit:1;         /* Flag to use Kalman KF */
  setBEspread:1;           /* Flag to set beam energy spread in KF */
  setBPosition:0;          /* Flag to set interaction point (IP) in KF */
  setVBPosition:0;          /* Flag to set error of IP in KF */
  useIPFromVF:1;            /* Flag to use IP from vertex fit */
  skipUnSelectedParticle:1;  /* Flag to skip un-selected particles */
  skipAlreadyUsedParticle:1; /* Flag to skip already used particles */
"};
```

How to combine particles



daughter particles classified by type

Particle lists with expected types

Total combinations: $N_{tot} = C_{n_1}^{N_1} \times C_{n_2}^{N_2} \times C_{n_3}^{N_3} \times \cdots$

Remove double countings:

① one track used twice due to composite particles

② re-combinations due to composite particles

Composite particles

① No Vertex Fit $4\text{-momenta}\ P = \sum P_i$ $\text{Vertex} = \text{IP}$

② Vertext Fit

psi' -> gamma + chi_0c , psi' masswin 2.0 4.0 vtxfit 1

vertex fit will be automatically applied to all final charged tracks. In above case, if chi_0c → ωω, ω → 3π, vertex will be applied to 4 charged pions. The χ^2 is saved to the mother particle (chi_0c.fChisqVF), the fitted tracks are kept as daughters of BWTracks, which are one-to-one association to BParticles.

③ Secondary Vertext Fit

psi' -> Lambda0 + Lambda0_bar , psi' masswin 2.0 4.0 ; Lambda0 -> proton + pi-, Lambda0 vtxfit 2

secondary vertex fit will automatically require vertex fit first, the results will be saved similar to that in vertex fit cases

Composite particles

① No Kinematic Fit, check cuts for the composite particle

② Kinematic Fit

```
psi' -> gamma + chi_0c , psi' masswin 2.0 4.0, kmfit gamma pi+ pi- gamma gamma pi+ pi- gamma gamma | 4mom
0.0 0.0 0.0 3.686 @ 0 1 2 3 4 5 6 7 8 | reson 0.13498 @ 7 8 | reson 0.134998 @ 3 4 ;
```

kinematic fit is required by key of "kmfit" with following format

kmfit track list | fit type @ list | fit type @ list …

vertex for kinematic fit:

a) get from vertex fit for this composite particle if there exists

b) get from vertex fit for daughters of this composite particle if a) fails

c) use run average value which is stored in Header of this events

Highlights of BPAT

1) can handle any type of decay chain

2) easily switch on or off vertex / kinematic fitting

3) Cut could be applied to each particle

4) pre-selection with kinematic fitting

5) Atomatic naming branch with type of particles

6) Can handle multi-chain simultaniously

7) All you need is to provide a proper option file

8) Make one could spend less of his time on coding and debugging

Summary

- A new analysis procedure was developed based on BParticle class
- Could handle very complicated physical process in a rather clear/simply way
- Same data structure for different processes
- Could check data in each step with ROOT browser, easy to find bugs
- Easy to add additional info in the events
- Most of the codes can be shared
- Simple/clear naming scheme for variables

Summary

Source code:

/ihepbatch/bes/zhangxy/workarea/6.6.2/AnalysisCommon