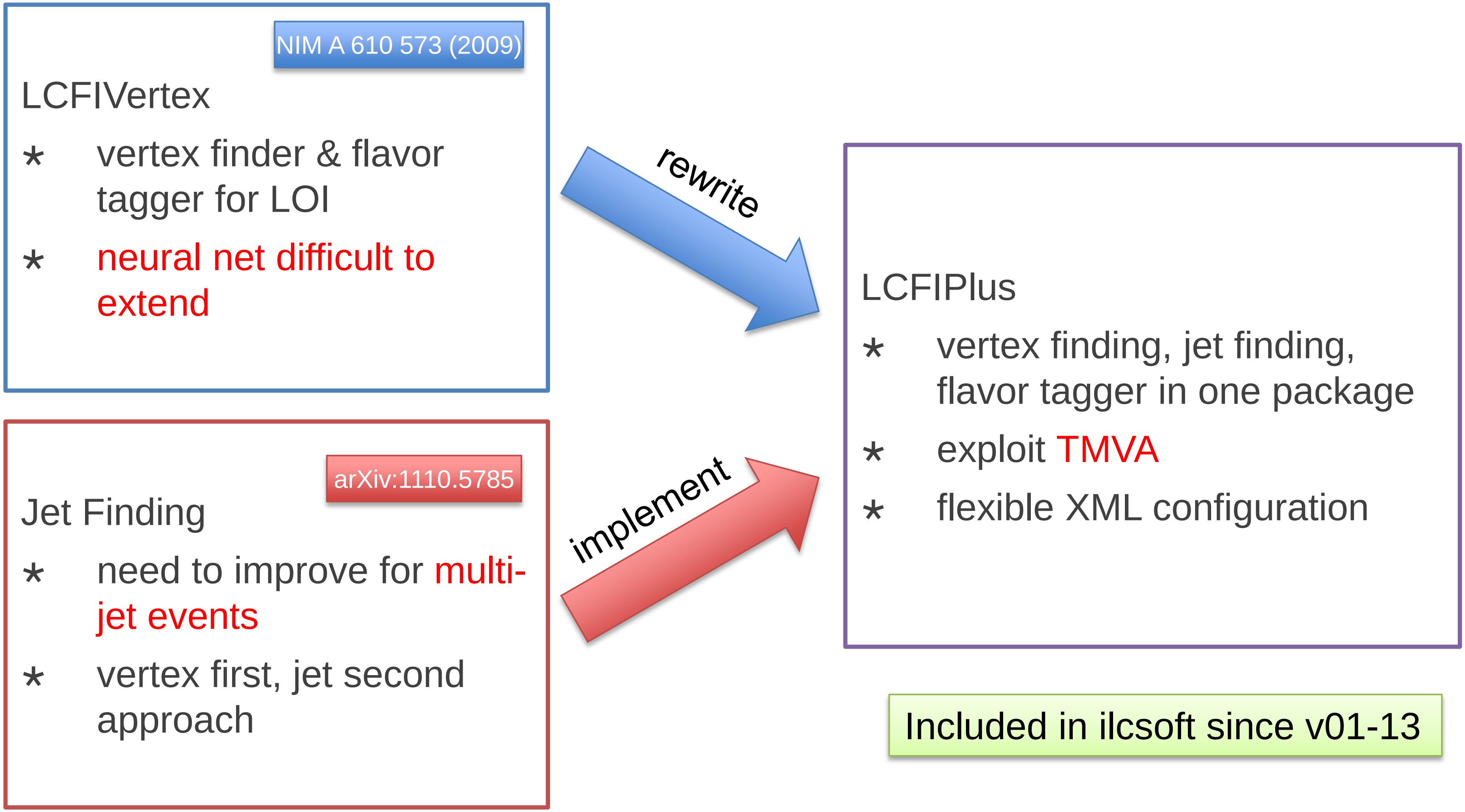
LCFIPlus

T. Tanabe, T. Suehara
ICEPP, The University of Tokyo

May 23, 2012
ILD Workshop @ Kyushu University

Introduction



NIM A 610 573 (2009)

LCFIVertex

- ⋆ vertex finder & flavor tagger for LOI
- ⋆ neural net difficult to extend

arXiv:1110.5785

Jet Finding

- ⋆ need to improve for multi-jet events
- ⋆ vertex first, jet second approach

rewrite

implement

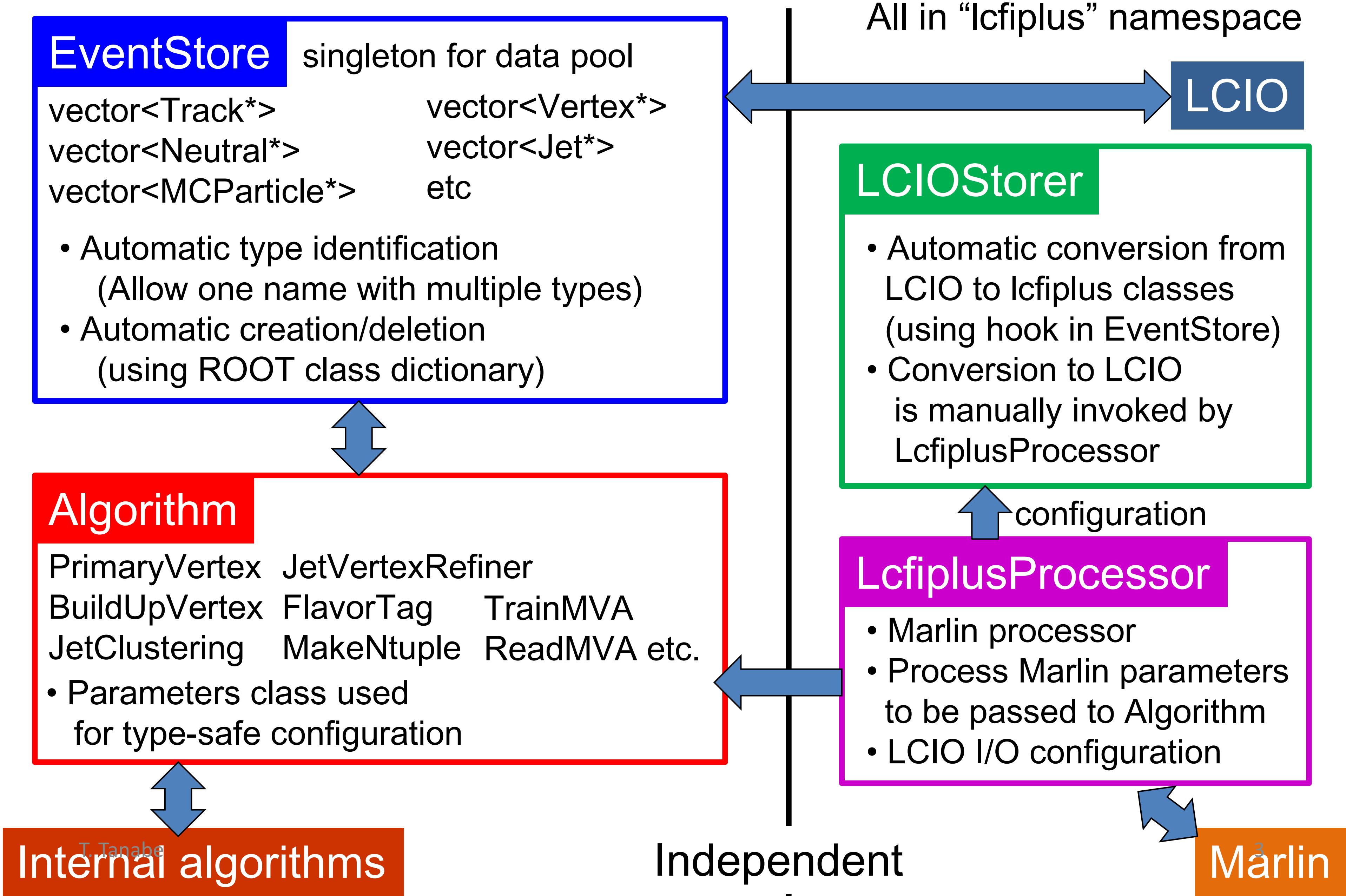
LCFIPlus

- ⋆ vertex finding, jet finding, flavor tagger in one package
- ⋆ exploit TMVA
- ⋆ flexible XML configuration

Included in ilcsoft since v01-13

Development driven by need to improve Zhh analysis

Data Flow



All in “lcfiplus” namespace

EventStore singleton for data pool

vector<Track*> vector<Vertex*>
vector<Neutral*> vector<Jet*>
vector<MCParticle*> etc

- Automatic type identification (Allow one name with multiple types)
- Automatic creation/deletion (using ROOT class dictionary)

LCIO

LCIOStorer

- Automatic conversion from LCIO to lcfiplus classes (using hook in EventStore)
- Conversion to LCIO is manually invoked by LcfiplusProcessor

configuration

Algorithm

PrimaryVertex JetVertexRefiner
BuildUpVertex FlavorTag TrainMVA
JetClustering MakeNtuple ReadMVA etc.

- Parameters class used for type-safe configuration

LcfiplusProcessor

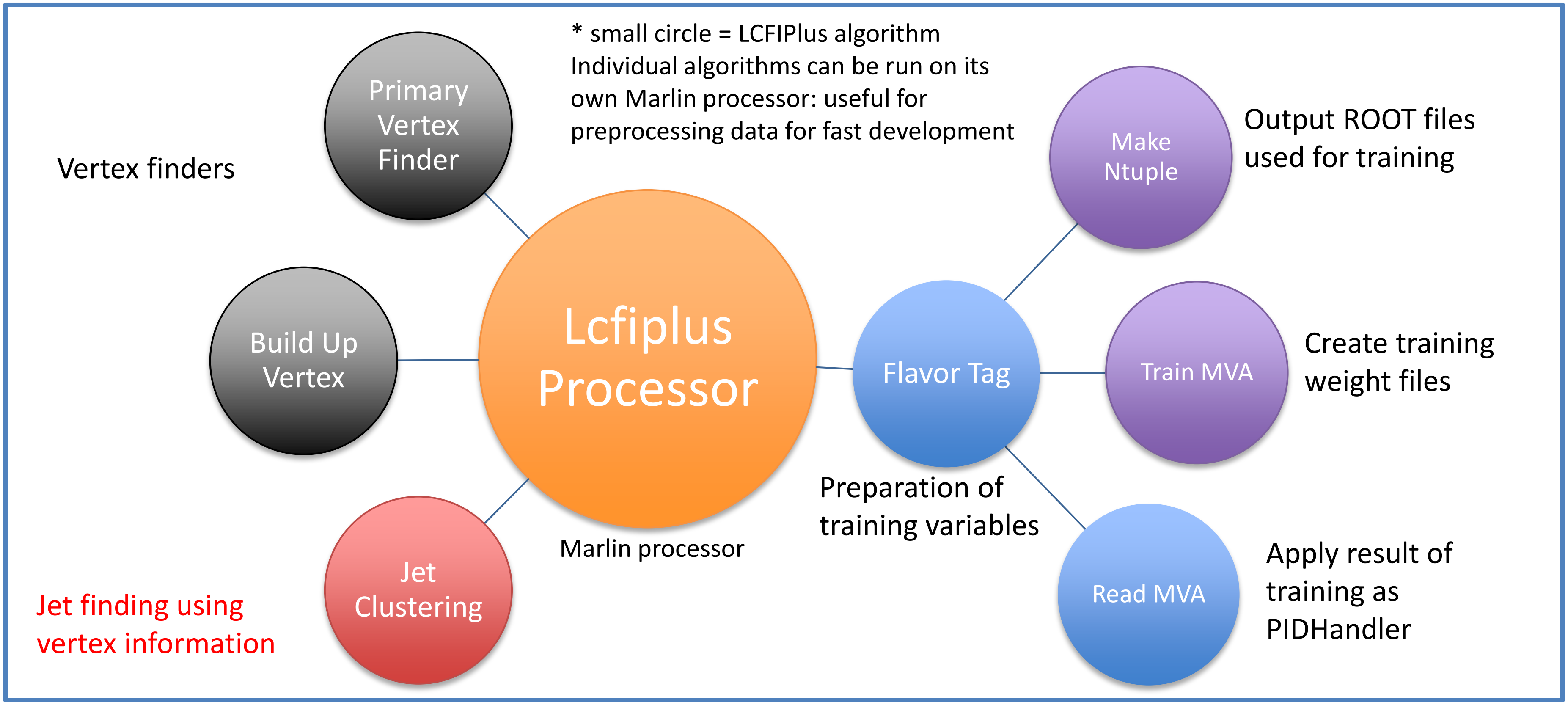
- Marlin processor
- Process Marlin parameters to be passed to Algorithm
- LCIO I/O configuration

Internal algorithms

Independent

Marlin

LCFIPlus Algorithms



* small circle = LCFIPlus algorithm
Individual algorithms can be run on its own Marlin processor: useful for preprocessing data for fast development

Vertex finders

Primary Vertex Finder

Build Up Vertex

Jet Clustering

Jet finding using vertex information

Lcfiplus Processor

Marlin processor

Flavor Tag

Preparation of training variables

Make Ntuple

Output ROOT files used for training

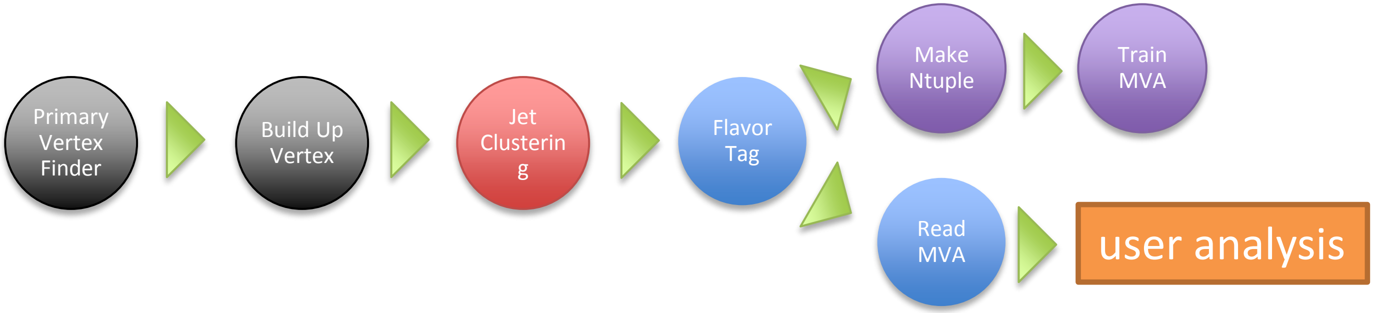
Train MVA

Create training weight files

Read MVA

Apply result of training as PIDHandler

Procedure:



Primary Vertex Finder

Build Up Vertex

Jet Clusterin g

Flavor Tag

Make Ntuple

Train MVA

Read MVA

user analysis

Vertex Finding

Primary Vertex Finder

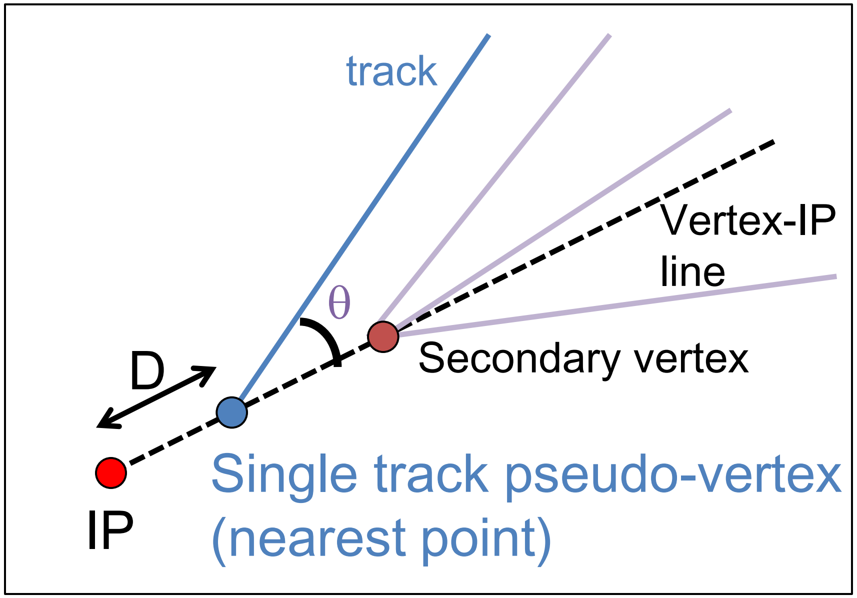
- Two kernels are implemented:
  - Kalman filter - calls LCFIVertex
  - Teardown type
- Uses beamspot constraint

Secondary Vertex Finder

- Implemented kernels are:
  - ZVTOP (LCFIVertex) - needs jet direction and cannot be used
  - Build-up type - computationally intensive
- Build-up VF has been tuned to be as efficient as ZVTOP and with higher purity
- V0 finding applied (outputs dedicated list)

- Some problems were observed in the covariance matrix:
  - Lacking floating point precision (KF)
  - Indication of convergence problems (TD)
  - Vertex positions look OK
  - Need to be fixed a.s.a.p. before mass production starts
- Need to decide on the behavior when no primary vertices are found (due to too few tracks passing the quality selection)
  - Return "beamspot" vertex?
  - How to pass on the information to LCIO? PIDHandlers on Vertex?

Single Track Pseudo-Vertex

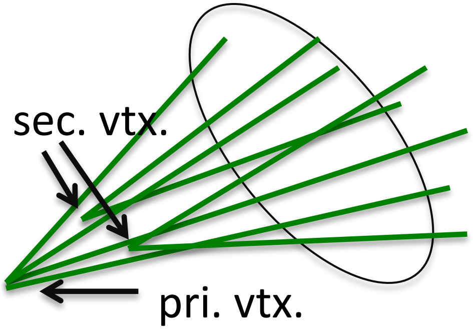


- Normal vertex finder needs at least 2 tracks – information of 1 track decay is lost.
- Given a secondary vertex, look for a single track pseudo-vertex.
- This has been shown to improve b-tagging.

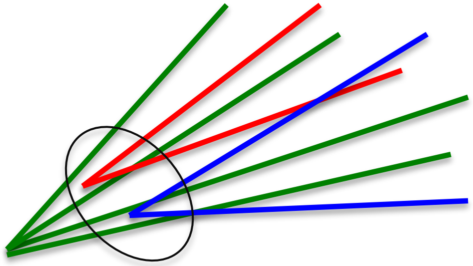
Event	0 vtx	1 vtx	>= 2 vtx
bb normal	322	1052	426(24%)
bb +single	322	459	1019(57%)
cc normal	1003	779	18(1.0%)
cc +single	1003	715	82(4.6%)

Jet Finding (Vertex-Assisted)

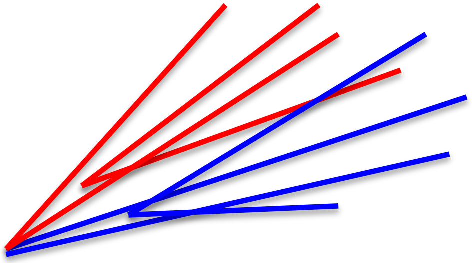
arXiv:1110.5785



1. Difficult to separate two b-jets which are close. Ordinary kt algorithm tends to merge them.



2. To overcome this, find secondary vertices first using all tracks in the event, and use them as *seeds* for jet finding.



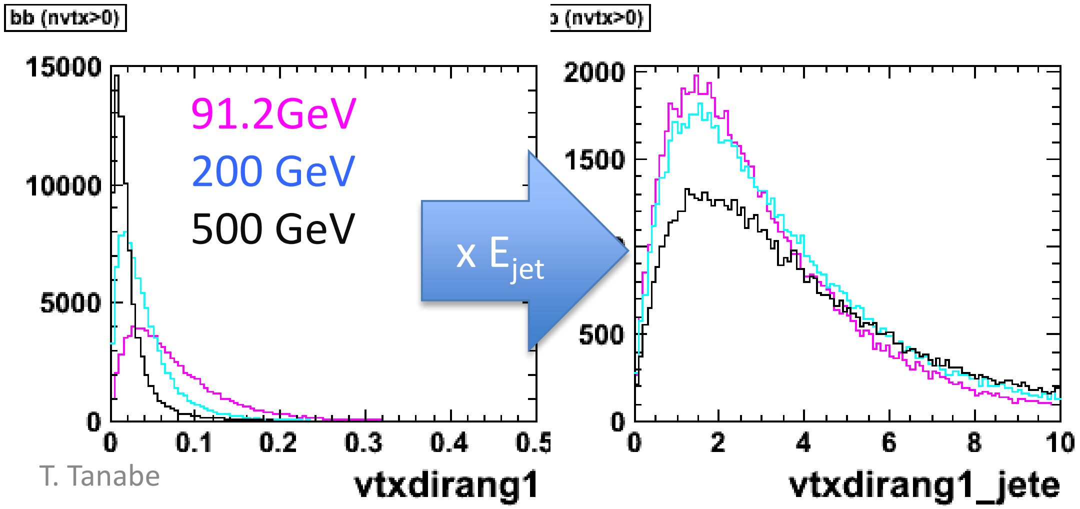
3. Results in an increased chance of correct jet separation. This effect is pronounced in final states with many b jets.
Vertex/Jet association are further refined after the jets are identified.

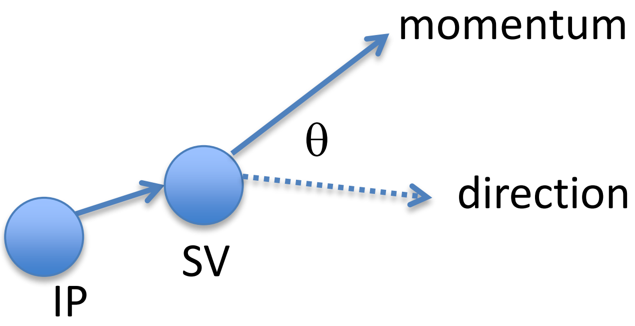
CAUTION: Be careful when applying this algorithm to backgrounds with different number of fermions, as it can enhance the background! (e.g. with gluon emissions g* → bb) Consider using multiple number of jets and/or conventional kt algorithm as complementary information.

Flavor Tagging

- Essentially an interface to TMVA
  - multiclass training → get c-tag for free!
  - boosted decision trees (BDT) with gradient boost gives nice output classifiers
- Normalization of input variables → less dependent of jet energy
- Example list of input variables (can be configured by Marlin steering file)

nvtx=0	trk1d0sig trk2d0sig trk1z0sig trk2z0sig trk1pt_**jete** trk2pt_**jete** jprobr jprob
nvtx=1	vtxlen1_**jete** vtxsig1_**jete** vtxdirang1_**jete** vtxmom1_**jete** vtxmass1 vtxmult1 vtxmasspc vtxprob (+ above)
nvtx>=2	vtxlen2_**jete** vtxsig2_**jete** vtxdirang2_**jete** vtxmom2_**jete** vtxmass2 vtxmult2 vtxlen12_**jete** vtxsig12_**jete** vtxdirang12_**jete** vtxmom_jete vtxmass vtxmult (+ above)





$\theta^* = \theta/\gamma \propto \theta/E_{jet}$

Marlin steering XML

```
<processor name="JetClusteringAndFlavorTag" type="LcfiplusProcessor”>
  <parameter name="Algorithms" type="stringVec"> JetClustering JetVertexRefiner FlavorTag ReadMVA</parameter>
  <parameter name="PFOCollection" type="string" value="PandoraPFOs" />
  <parameter name="JetClustering.InputVertexCollectionName" type="string" value="BuildUpVertex" />
  <parameter name="JetClustering.OutputJetCollectionName" type="stringVec" value="VertexJets" />
  <parameter name="JetClustering.NJetsRequested" type="intVec" value="6" />
  <parameter name="PrimaryVertexCollectionName" type="string" value="PrimaryVertex" />
  <parameter name="FlavorTag.JetCollectionName" type="string" value="RefinedJets" />
  <parameter name="FlavorTag.WeightsDirectory" type="string" value="lcfiweights" />
  <!-- include flavor tagging definitions here, must match the weight files -->
</processor>
```

Download weight files and template XML from repository

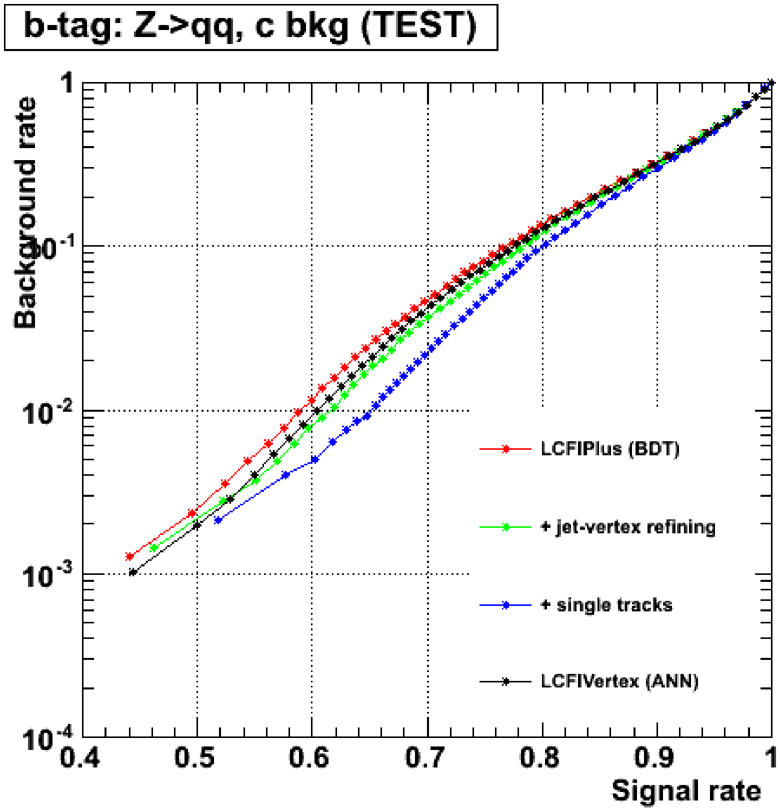
User analysis code (Marlin processor)

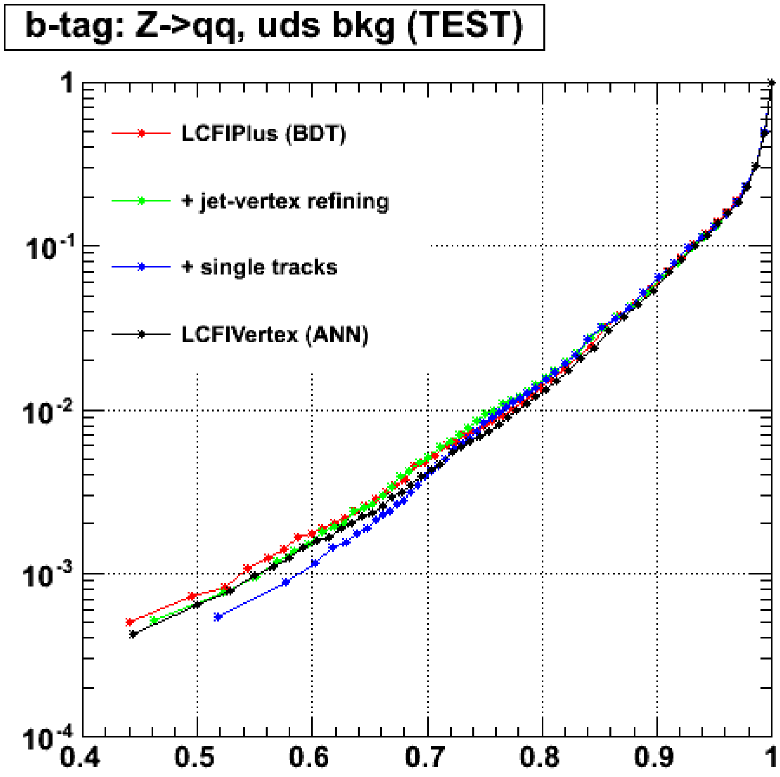
```
LCCollection* colJet = evt->getCollection(”RefinedJets");
PIDHandler pidh( colJet ); // get PIDHandler associated with jet collection
int algo = pidh.getAlgorithmID( "lcfiplus" ); // get algorithm ID
int ibtag = pidh.getParameterIndex(algo, "BTag"); // similarly for CTag

// loop over jets to extract flavor tagging information
for(int i=0; i < colJet->getNumberOfElements(); i++) {
  ReconstructedParticle *part =
    dynamic_cast<ReconstructedParticle*>( colJet->getElementAt( i ) );
  const ParticleID &pid = pidh.getParticleID(part, algo);
  cout << "btag = " << pid.getParameters()[ibtag] << endl;
}
```

Contains result of flavor tagging

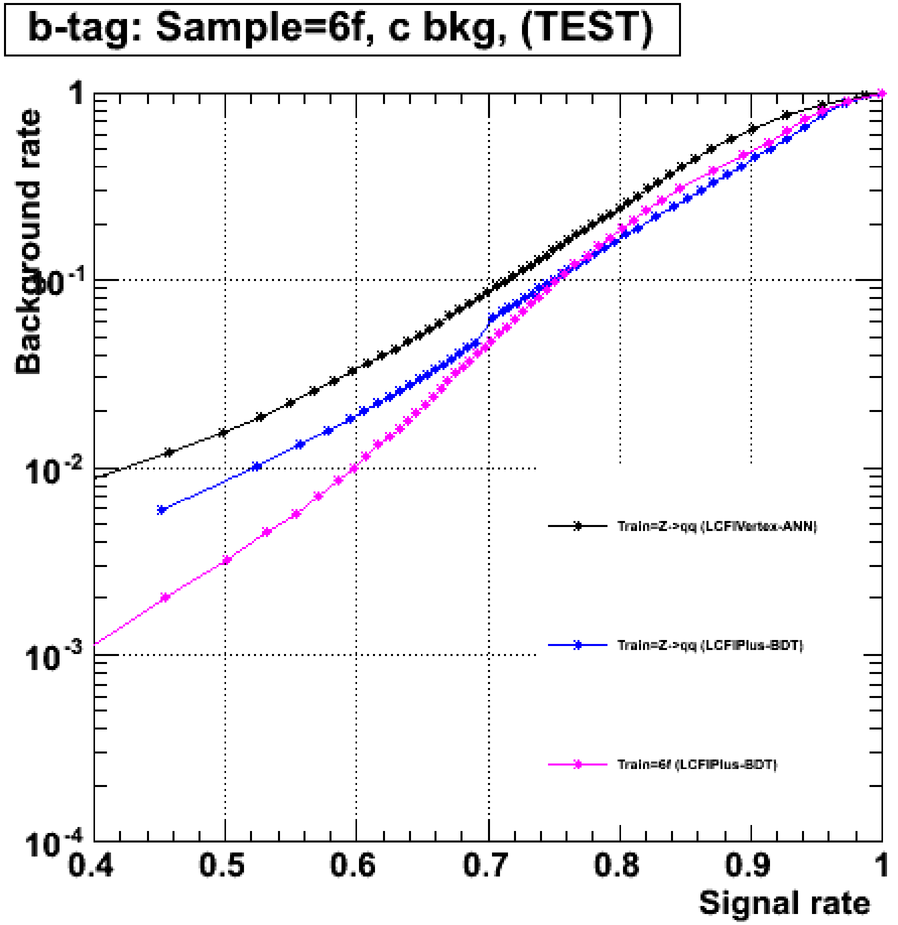
b-tag and c-tag


b-tag: Z->qq, c bkg (TEST)


b-tag: Z->qq, uds bkg (TEST)

This is for Z->qq sample at Ecm=91.2GeV.
Improvement of b/c separation in all efficiency range
Performance of b/uds separation still needs to be understood

Performance in 6-jet environment



LCFIVertex
2 jet training on 6 jet sample

LCFIPlus
2 jet training on 6 jet sample

LCFIPlus
6 jet training on 6 jet sample

Training and testing performed using 6f samples with 6b, 6c, and 6q with q=uds.

Improvement over old algorithm seen in all regions.
Performance in high efficiency region still needs to be understood.

Documentation & Feedback

- Doxygen class reference
- User feedback + documentation system hosted at SLAC (J. Strube + N. Graf):
  - Documentation wiki hosted at SLAC
    - bug tracker (JIRA) also available
  - https://confluence.slac.stanford.edu/display/ilc/LCFIPlus
    - some documentation present
- Early bug reports (J. Engels, F. Gaede, J. Strube, A. Sailer)
- Nightly builds and check input variables at CERN (J. Strube)
- Feedback and support from LC community has been very helpful during initial deployment of LCFIPlus

Summary and Outlook

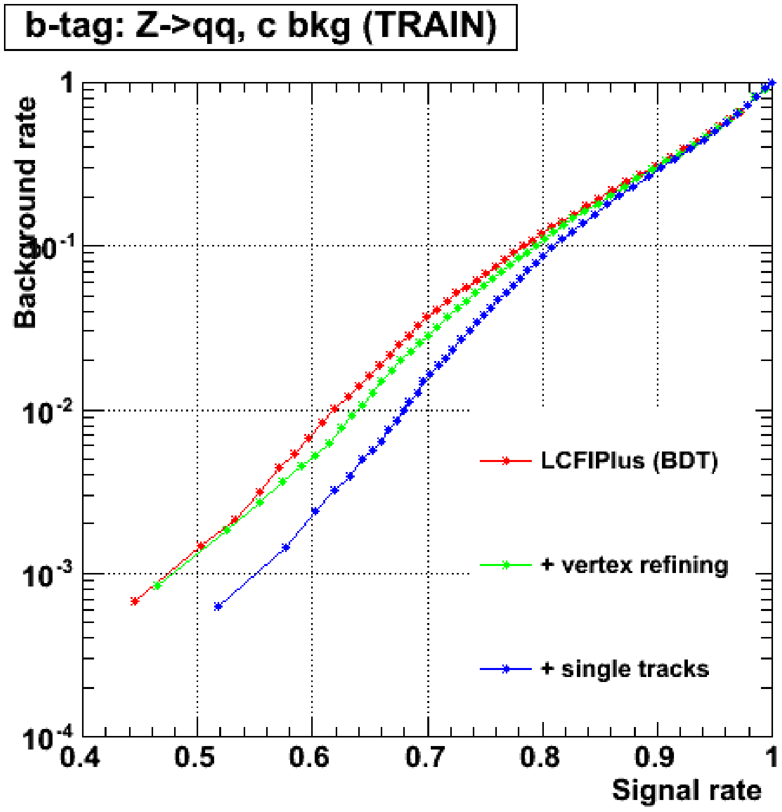
- Software infrastructure now in place for ILD DBD production
  - complied with technical requests, included in the latest ilcsoft v01-13-06
- Some issues still need to be ironed out
  - repository for training weight files
  - covariance matrix of vertex fitter
  - better understanding of TMVA behavior
  - optimization for 1 TeV
- Continue working with whole LC community for a smooth transition from LCFIVertex to LCFIPlus
- Vertex charge: next target, needed e.g. by ttbar analysis

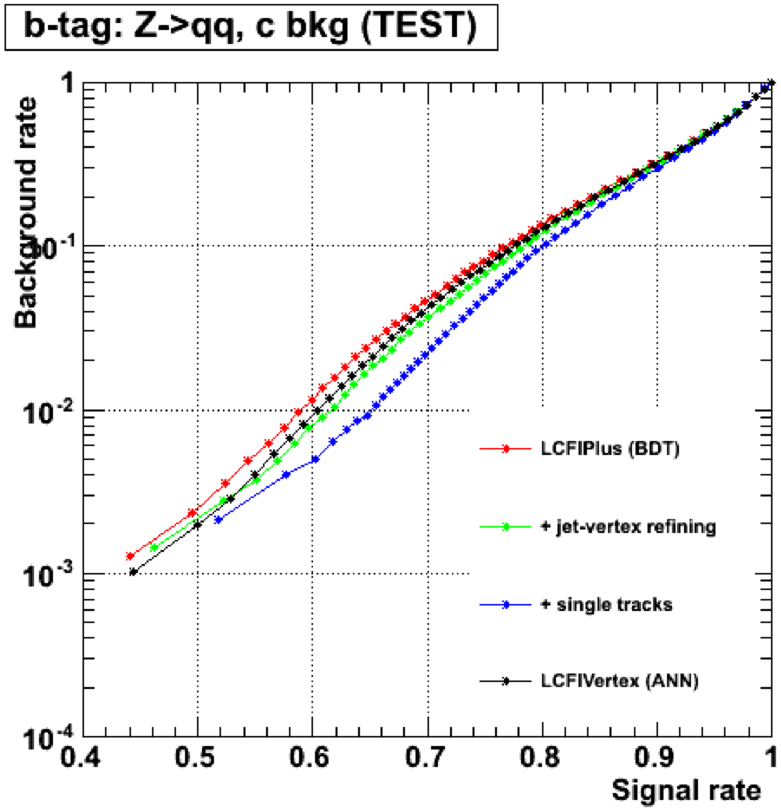
backup slides

Jet/Vertex Refining Strategy

- Apply V0 rejection on secondary vertices
  - K-short, Lambda0, photon conversions
    - properly computed using track parameters at the vertex
    - BuildUpVertex produces V0 vertex list
- Vertex clustering
  - no more than two vertices per jet (excluding V0)
  - if too many vertices are present, they get combined by using measures based on angle/distance
- Refit all vertices as a single vertex, merge them if the fit is good

Training vs. Testing





We use independent samples to evaluate the performance of the training.