FERMILAB-Conf-98/315-E
CDF

Unified Design of the CDF Calorimeter Reconstruction and Simulation

J. Lamoureux
For the CDF Collaboration
Brandeis University

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

October 1998

Published Proceedings of the *International Conference on Computing in High Energy Physics (CHEP '98)*, Chicago, Illinois, August 31-September 4, 1998

CDF/PHYS/JET/PUBLIC/4772
October 10, 1998

Unified Design of the CDF Calorimeter Reconstruction and Simulation

Jodi Lamoureux
Brandeis University
Fermi National Accelerator Laboratory
Batavia, IL 60510 USA

We have integrated the CDF calorimeter reconstruction and simulation into a cohesive software package. The design includes layers of indirection which provide a flexible physics analysis environment. New algorithms are easy to implement in the design: including legacy code in fortran. Finally, we describe how the new jet clustering code can be transported outside of the CDF software environment making it possible for the first time at CDF for theorists to study our jet algorithm.

1 Introduction

CDF has chosen C++ as the primary computing language for the analysis software in the upcoming Tevetron run at Fermilab. Upgrading the calorimeter software affords us an opportunity to improve the design. In the following paper, I will show the OO code design classes for calorimeter tower reconstruction and simulation as well as the jet reconstruction. The new design integrates the reconstruction and simulation. It includes abstract interface classes to improve flexibility for physics analyses. These layers of abstraction allow one to isolate the CDF jet algorithm from the CDF framework - making it possible to run in a standalone mode as well as in the CDF code environment. Since reconstruction tasks have been assigned to well defined classes, the new code is readable and new algorithms are easily incorporated. Furthermore, the new design can be interfaced to legacy (fortran) algorithms.

A fundamental property of sampling calorimeters is that electromagnetic (hadronic) showers occur early (late): their longitudinal and transverse shape depending on the radiation (interaction) length of the detector material. Showers are detected by amplifying and digitizing signals from a localized region. Signal detection is expensive and usually limits the detector segmentation to fairly course electromagnetic and hadronic towers.

2 Calorimeter Reconstruction and Simulation

We begin to reconstruct the data by defining a HardwareKey class which can contains the address of each photo-multiplier tube (PMT) readout by the frontend electronics at CDf. We have found the enumeration relating the calorimeter acronyms to an integer to be particularly useful. This key also contains an isValid method to determine the validity of its internal data based on the geometry database.

For reconstruction, we define the RecoKey class which treats the detector hermetically as a grid of towers uniquely identified by a pair of integers (ieta,iphi). This grid is not necessarily a simple grid. For our purposes, sometimes the iphi ranges from 0 to 23 and at others it ranges from 0 to 47. Again an isValid method is provided to validate the internal data based on the geometry database. In addition to knowing the mapping of hardware keys onto reconstruction keys, this key can return a list of its neighboring towers.

The calorimeter tower attributes are maintained by the CalTower class. The energy is measured in the calorimeter PMT's. The position of this energy is determined from the geometry database and the measured event vertex. The combined information is a four-vector. Towers are assumed to be massless ($|p| == E$).

An architecture diagram of the CDF calorimeter reconstruction and simulation is shown in figure 1. It centers around a CalBankMaker class which reads raw persistent data off of a file somewhere, retrieves the hardware key for a piece of data, fetches the relevant calibration constant, converts the data from digitized counts into energy (GeV), converts the hardware key into a reconstruction key and then writes the energy out to the element bank based on the reconstruction key. The same mechanism is used by the simulation to convert the element data into raw data. The simulation also shares the information in the geometry database with the reconstruction, unifying the calorimeter code design.

3 Jet Reconstruction

In a mathematical sense, jets can be defined as non-overlapping subsets of calorimeter towers. The algorithm which separates the list of towers into these subsets is more or less irrelevant to the code architecture. An architecture diagram of the CDF jet reconstruction code is shown in figure 2. The jet reconstruction uses the same element bank and reconstruction key described earlier. Other than these, it is completely independent of the calorimeter reconstruction.

The input set of CalTowers is maintained by a TowerCollection class. This class provides an iterator for the collection as well as random access based on a reconstruction key. It is desirable to make the tower collection based on either the calorimeter data stored in the element bank, or from monte-carlo generated particles via the generated particle bank. An abstract InputSource class is provided to add this flexibility. In addition, the current CDF jet algorithm sums towers according to the Snow Mass Convention,[1] but another sensible method for summing towers is to treat each like a four-vector. To facilitate studying the physics impact on tower summing algorithms, an abstract Calculator class has been defined and the concrete calculation can be chosen at run time.

The jet clustering algorithm itself is driven from a set of parameters. Each algorithm is required to have a name, input source, and calculator. Cone clustered jets have specific parameters such as thresholds and cone radius. Kt clustered jets have thresholds and a ymin cut. The output of the clustering routine is a list of jets, which are associated with a particular set of parameters. To make the jets persistent, a makeBanks method is included. To access the jets from the persistent data, a readBanks method is included. When the jets are recorded, they are recorded with their parameters. When they are read back in, only those with the current ConeParms are included in the JetCandidates class. Thus at any time, the list of jets in an instance of the ConeJets class is unique and self consistent.

Two related aspects of the code design remain. The first is our ability to export the CDF jet clustering algorithm outside the CDF framework. The second is our ability to import well-documented legacy clustering algorithms (mostly in fortran).

The current jet clustering algorithms can be accessed in stand-alone mode if a few criteria are met. The user must provide a reconstruction Key describing at least a minimal set of the desired calorimeter geometry. This separates the external user from the CDF geometry data base. The input source can be separated from the CDF persistent data, by providing a concrete InputSource which depends only on the HepEvt common block or the C++ equivalent structure.[2] Finally, the jets will be provided to the user without the makeBanks and readBanks methods which depend on the CDF persistent data. The towerCollection and output jets will be extremely useful for studying calorimeter-based physics.

A separate issue is encorporating well-documented legacy algorithms into the jet analysis at CDF. Examples of existing jet clustering algorithms are PYTHIA's lucell,[3] and the Kt algorithm[4]. In all cases, these algorithms use the HepEvt common block or an array of four-vectors for their input. In addition, a list of parameters must be passed into the algorithm. Only three steps are necessary to interface these algorithms to the CDF software. The calorimeter tower information must be converted to a collection of four-vectors. The external fortran clustering routine must be called passing in parameters and the four-vector array as necessary. And, finally, the output must be converted into JetCandidates and made persistent. The legacy code fits nicely into the jet reconstruction design with a minimum of data-copying.

4 Conclusions

We have integrated the CDF calorimeter reconstruction and simulation. At the same time, the jet reconstruction has been isolated from the low level tower reconstruction. Tasks have been assigned to well defined classes. The design includes layers of indirection to provide a flexible physics analysis environment. Inputs, parameters and algorithms may be varied with relative ease. New algorithms are easy to implement - including legacy code. And finally, the new jet clustering code can be transported outside of the CDF software environment making it possible for the first time for theorists to study the CDF jet algorithm.

References

1. J.E. Huth *et al.*, Proceedings of the Summer Study on High Energy Physics, Snowmass, Colorado, 1990, p. 134.
2. G.R.Lynch and T.G. Trippe, “Monte Carlo Particle Numbering Scheme”, Phys Rev. D54(1996)170.
3. T. Sjostrand, Computer Physics Commun. 82(1994)74; hep-ph/9508391.
4. S.Catani, Y.L. Dokshitzer, M.H. Seymour and B.R. Webber, Nucl. Phys. B406:187, 1993;
   S.D. Ellis and D.E. Soper, Phys. Rev. D48(1993)3160.

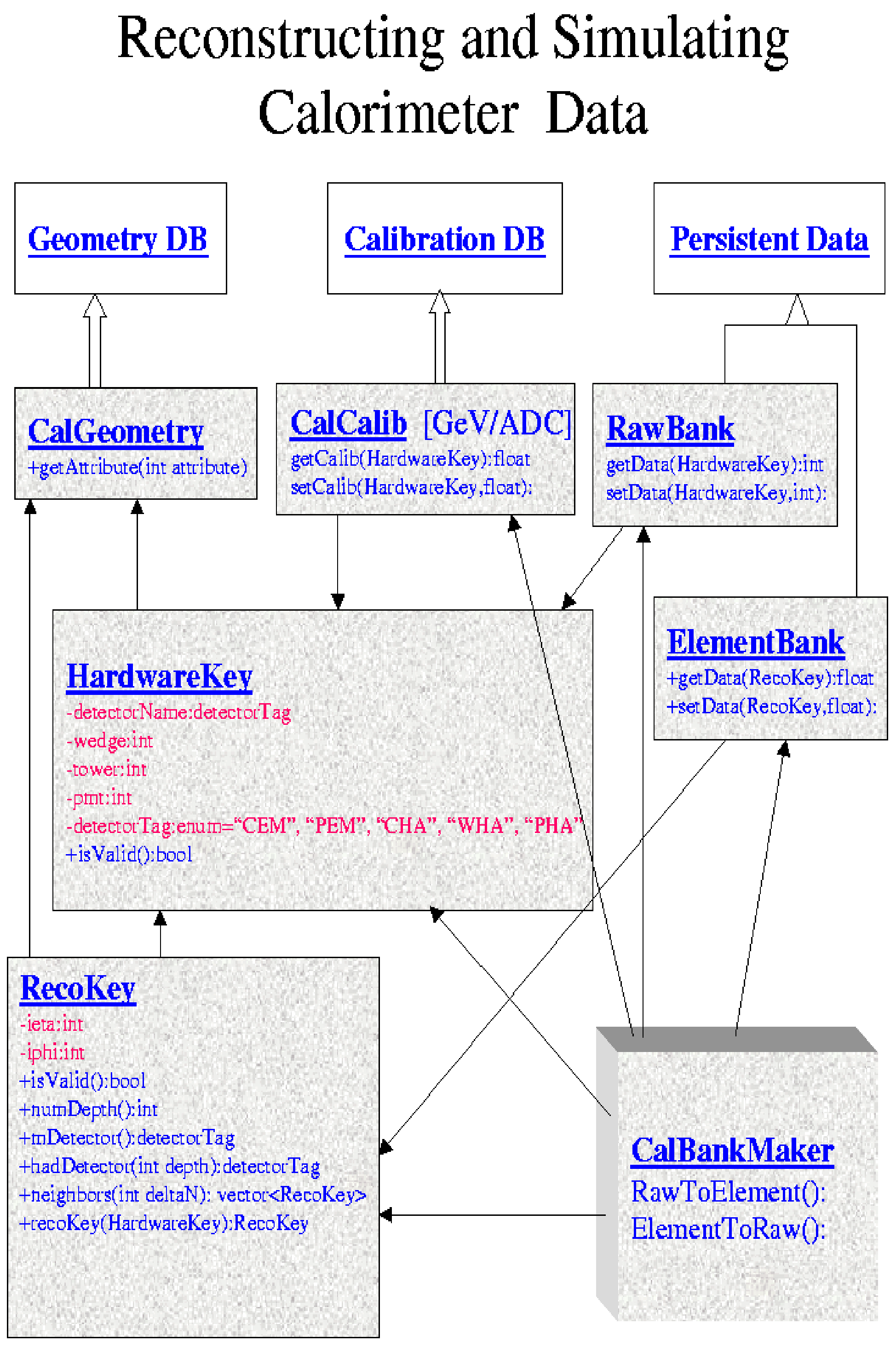
Reconstructing and Simulating Calorimeter Data
Geometry DB
Calibration DB
Persistent Data
CalGeometry
+getAttribute(int attribute)
CalCalib [GeV/ADC]
getCalib(HardwareKey):float
setCalib(HardwareKey,float):
RawBank
getData(HardwareKey):int
setData(HardwareKey,int):
ElementBank
+getData(RecoKey):float
+setData(RecoKey,float):
HardwareKey
-detectorName:detectorTag
-wedge:int
-tower:int
-pmt:int
-detectorTag:enum="CEM", "PEM", "CHA", "WHA", "PHA"
+isValid():bool
RecoKey
-ieta:int
-iphi:int
+isValid():bool
+numDepth():int
+mDetector():detectorTag
+hadDetector(int depth):detectorTag
+neighbors(int deltaN): vector<RecoKey>
+recoKey(HardwareKey):RecoKey
CalBankMaker
RawToElement():
ElementToRaw():

Figure 1: Architecture diagram for the CDF calorimeter reconstruction and simulation code.

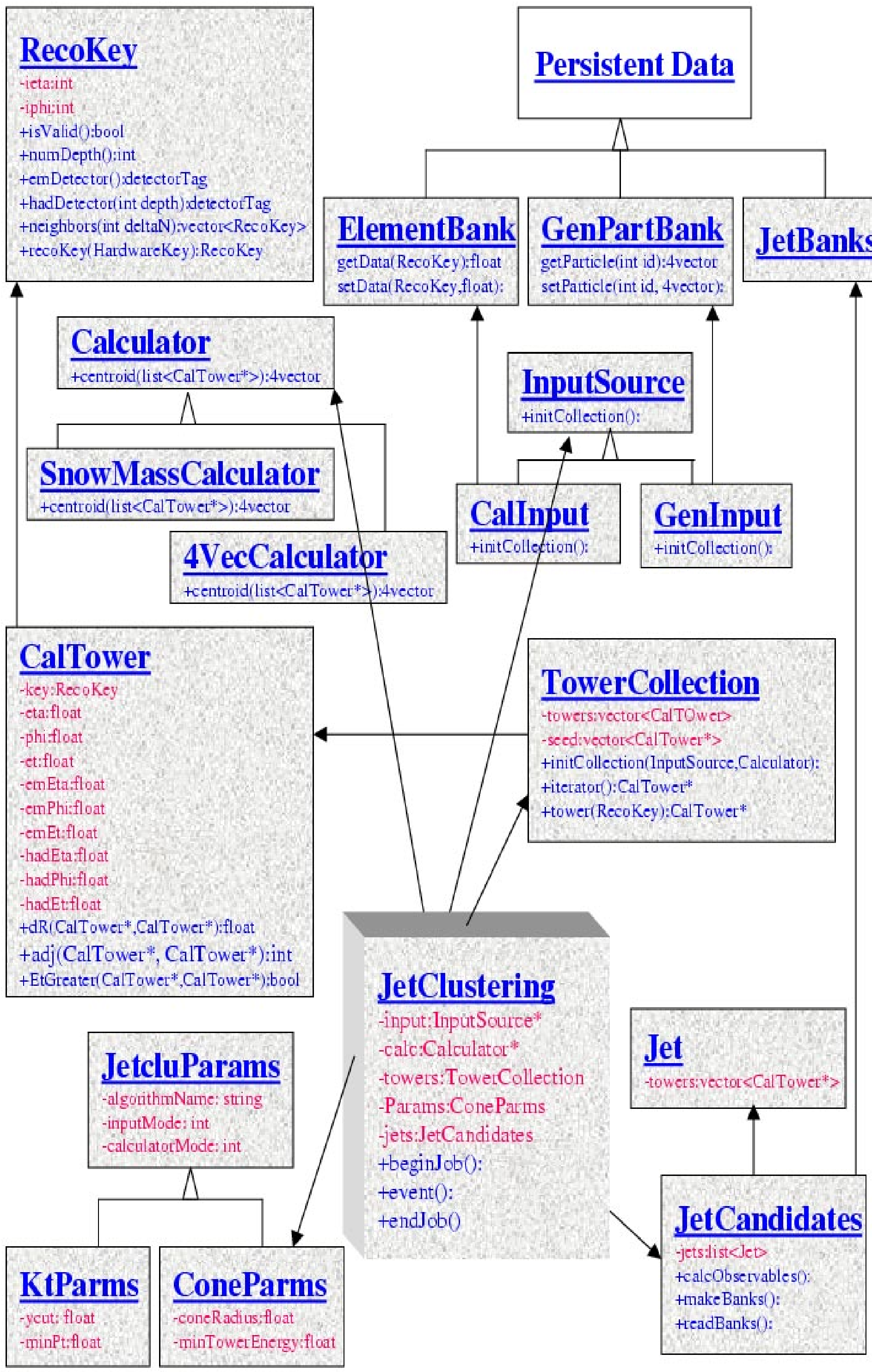

Figure 2: Architecture diagram for the CDF jet reconstruction code.