LCIO
Overview and Status

ECFA Workshop 2004, Durham Simulation, Sep. 3rd, 2004
Frank Gaede   DESY   -IT-
reported by Ties Behnke, DESY
Outline

• Introduction
• Overview
• Changes since Paris Workshop
• Status
• Reconstruction/Analysis Framework
• Summary
Motivation for LCIO

Generator

Java, C++, Fortran

Geometry

Simulation

Recon-

struction

Analysis

Java, C++, Fortran

Java, C++, Fortran

Java, C++, Fortran
LCIO SW-Architecture

- JAS/AIDA
- root
- hbook

**common API**
- generated from one source using AID

**Java API**
- LCIO Java implementation

**C++ API**
- LCIO C++ implementation

**f77 API**
- compressed records, pointer retrieval

*.slcio files (SIO)
Implementation - Design

- abstract event
- abstract io
- concrete classes
- persistency implementation
Data Model III

RawData and Digitization

RawCalorimeterHit
- CellID
- Amplitude
- Time (optional)

New

CalorimeterHit
- CellID
- Energy
- Time (opt.)
- Position (opt.)

SimHits

TPCHit
- CellID
- Time
- Charge

Add more specialized classes as needed -> user request!

Serve as interface classes to reconstruction

TrackerHit
- Position
- Covariance
- dEdx
- raw hit

ECFA WS 2004, Durham, 03.09.2004
Frank Gaede, DESY -IT-
The Data Model: Comments

important ingredients:

- objects (tracks, clusters, ...) are grouped into collections

- there can be several collections of the same type of objects in the event:
  - tracks at IP
  - tracks at Calo face
  - VTX tracks
  - ...

  (if this is done, documentation is essential!)

- self-referencing of the objects allows the buildup of tree structures
Changes since LCWS-2004

- added LCRelation class to store (weighted) nxm relationships between LCObjects
  - can be used to point back to MC-truth.
  - can be used to link collections
- changed track parameters, now: d0, phi, omega, z0, tanLambda
- added generic named parameters to LCRunHeader, LCEvent and LCCollection
  - use to store meta information on data
Changes since LCWS-2004  -II-

- added support for ‘generic’ user objects, that hold floats, ints and doubles:
  - can be used to store arbitrary additional data
- added RawCalorimeterHit
  - int Amplitude and int time
- added some convenient methods to the classes
- modified some classes to make the API more consistent
Reconstruction and Analysis

- need to provide a simple, lightweight environment for reconstruction and analysis
  - simple to use
  - low thresholds
  - in Europe: C++ support is essential (most people work on LHC in C++ environments)
  - no dependence on user backends (root, JAS, PAW, ... )

- Simple C++ based framework, in many ways similar to the existing LCD framework
- Developed in close collaboration with people doing actual test data analyses for TPC, Calo and physics studies

MARLIN
main author Frank Gaede
other contributors are welcome
Reconstruction and Analysis

- The LCEvent can be used as container for transient data in an application, e.g. reconstruction
- Application will call list of modules that read existing collections from the LCEvent and add resulting new Collections
- LCIO has (Event/Run)-Listener classes that can serve as base classes for modules
- define an application framework based on LCIO for reconstruction and analysis:

Modular Analysis & Reconstruction for the Linear Collider
Motivation for MARLIN

Simulation (MOKKA)

Generator

Java, C++, Fortran
Geant3, Geant4

Simulation

Java, C++, Fortran

Reconstruction

MARLIN

Analysis

Java, C++, Fortran

Persistency Framework

geometry
Implementation of MARLIN

- use LCIO as transient data model
- use C++ only (so far)
- define base classes for modules that operate on LCIO (event) data
- provide simple user steering:
  - user defined variables for each module
  - input/output files
- provide main program!
Modules and the LCIOEvent

MyInput2.slcio  MyInput1.slcio  MyInput0.slcio

MyInput.slcio

LCEvent

collection0

read and add collections

Module0

Module1

Module2

...  ModuleN

Outputmodule

MyInput2.slcio  MyInput1.slcio  MyInput0.slcio

MyInput.slcio

MyInput2.slcio  MyInput1.slcio  MyInput0.slcio
LCIOModule

- LCIOModule: base class for all user modules
- provides hooks (callbacks) for user actions:
  - `init()`
    - called once at program start
    - use to initialize histograms, counters, etc.
  - `processRunHeader(LCRunHeader* run)`
    - called for bookkeeping – new run conditions?
  - `processEvent(LCEvent* evt)`
    - the working horse – this where the analysis takes place
  - `end()`
    - called once at end of job
    - write out histos, ...
Under development in Marlin

- error handling
  - log files
  - error/warning messages
- naming convention for common parameters, e.g. InputCollectionName, OutputCollectionN.
- convention for passing user data between modules, e.g.:
  - as LCCollections of LCOBJECTS
  - as global objects (singletons)
- some logic to control execution and I/O of events, e.g. a module might want to decide that the event is not worth processing then the rest of the modules should not be called ...
- lots of additional functionality? need user feedback
MARLIN developments

Under discussion:

try to make the user hooks as similar as possible to the ones in the JAVA (LCD) framework to facilitate exchange of ideas

A problem:

The true parallel use of JAVA and C++ code to access the same LCIO even in memory is difficult

We are still far from a truly language independent frame
Status of Marlin

- very first implementation released to beta users at DESY (as LCIOFrame)
  -> see talk from J.Samson

- cvs repository with web based public access
  (will be provided by H.Vogt, Zeuthen)

- hope to have public beta release soon
  -> stay tuned
LCIO on the web

- **LCIO homepage:** [http://lcio.desy.de](http://lcio.desy.de)
  - downloads and documentation

- **LCIO forum at:** [http://forum.linearcollider.org](http://forum.linearcollider.org)
  - user/developer questions and comments
  - discussions on new developments

- **LCIO bug reports at:** [http://bugs.freehep.org](http://bugs.freehep.org)
  - bug report and new feature requests
LCIO Customers/Users

- Mokka simulation (see talk)
- Brahms reconstruction (see talk)
- JAS3
  - provides convenient file browser
  - will have LCIO-WIRED plugin -> generic event display!
- Calorimeter group (DESY)
  - has MiniCal raw data converted to LCIO files
  - to be used also for Hcal physics prototype
- TPC groups (DESY & Aachen & ...)
  - will use LCIO for prototype
- Lelaps fast Monte Carlo
- hep.lcd reconstruction
- other groups looking into using LCIO
JAS3 – LCIO

Note: JAS3 provides very nice native interfaces to LCIO: browser, code wizard, event display

Summary

LCIO:
- available since some time, stable version 1.0, beta version 1.1
- new major release very soon

MARLIN:
- first beta release available
- CVS in Zeuthen being set up (same place as other LC software)
- user feedback needed!
- http://www.desy.de/~gaede (real site to come soon)

User feedback is extremely important on all these projects!

Use the forum: http://forum.linearcollider.org
or sent e-mail to one of us
Appendix

- Extension slides, details, examples
Requirements

- need Java, C++ and f77 (!) implementation
- extendable data model for current and future simulation studies
- user code separated from concrete data format
  - -> want to be flexible for future decisions on persistency
- needed a.s.a.p.
  - -> keep it simple (lightweight)
- no dependence on other frameworks
LCIO persistency framework

LCIO

- data model
- data handling
- data format

- contents
- API
- persistency
- implementation
Data Format (persistency): SIO

- SIO: Simple Input Output
- developed at SLAC for NLC simulation
- already used in hep.lcd framework
- features:
  - on the fly data compression 😊
  - some OO capabilities, e.g. pointers 😊
  - C++ and Java implementation available 😊
  - no direct access 😞
    -> use fast skip 😊
Data Model II

Monte Carlo

**MCParticle**
- Kinematics (4Vector)
- Parents/Daughters
- Generator Status
- PID
- Vertex
- ....
- -> all of HEPEVT
- + Simulator Status
- + Endpoint

**SimCalorimeterHit**
- CellID
- Energy/Amplitude
- Position (opt.)
- MCParticle Contributions

**SimTrackerHit**
- Position
- dEdx
- MCParticle Contribution
Data Model V

Run and Event

LCEvent
- Event number
- Collections

LCRun
- Run number
- description

LCCollection
- Type
- Name
- Elements

LCOBJECT

MCParticle

SimCalorimeterHit

CalorimeterHit

SimTrackerHit

TrackerHit

TPCHit

Reconstructed Particle

Cluster

Track

LCIntVec

LCFloatVec

The LCEvent serves as a container of named collections of the various data types in LCIO (LCOBJECT subclasses)

Generic integer and float vectors for user extensions
Javadoc example

Method Summary

**int close()**
Closes the output file/stream etc.

**int open(String filename)**
Opens a file for reading (read-only).

**LCEvent readEvent(int runNumber, int evtNumber)**
Reads the specified event from file.

**LCEvent readNextEvent()**
Reads the next event from the file.

**LCEvent readNextEvent(int accessMode)**
Same as above allowing to set the access mode (LCIO.READ_ONLY is default)

**LCEvent readRunHeader()**
Reads the next run header from the file.

**int readStream()**
Reads the input stream and notifies registered listeners according to the object type found in the stream.

**void registerLCEventListener(LCEventListener l)**
Registers a listener for reading LCEvents from a stream.

**void registerLCEventListener(LCEventListener l)**
Registers a listener for reading LCEventsLCEventHeaders from a stream.

**void removeLCEventListener(LCEventListener l)**
Removes a listener for reading LCEvents from a stream.

**void removeLCEventListener(LCEventListener l)**
Removes a listener for reading LCEventHeaders from a stream.

ECFA WS 2004, Durham, 03.09.2004

Frank Gaede, DESY - IT-
Doxygen example

The generic object that is held in an LCollectionData. More...

#include <LObject.h>

Inheritance diagram for DATA::LObject:

[Diagram showing the inheritance hierarchy of DATA::LObject, including classes like DATA::CalorimeterHitData, DATA::MCParticleData, DATA::SimCalorimeterHitData, DATA::SimTrackerHitData, EVENT::CalorimeterHit, EVENT::MCParticle, EVENT::SimCalorimeterHit, EVENT::SimTrackerHit, IMPL::CalorimeterHitImpl, IMPL::MCParticleImpl, IMPL::SimCalorimeterHitImpl, IMPL::SimTrackerHitImpl, IOIMPL::CalorimeterHitIOImpl, IOIMPL::MCParticleIOImpl, IOIMPL::SimCalorimeterHitIOImpl, IOIMPL::SimTrackerHitIOImpl, with arrows indicating the inheritance relationships.]