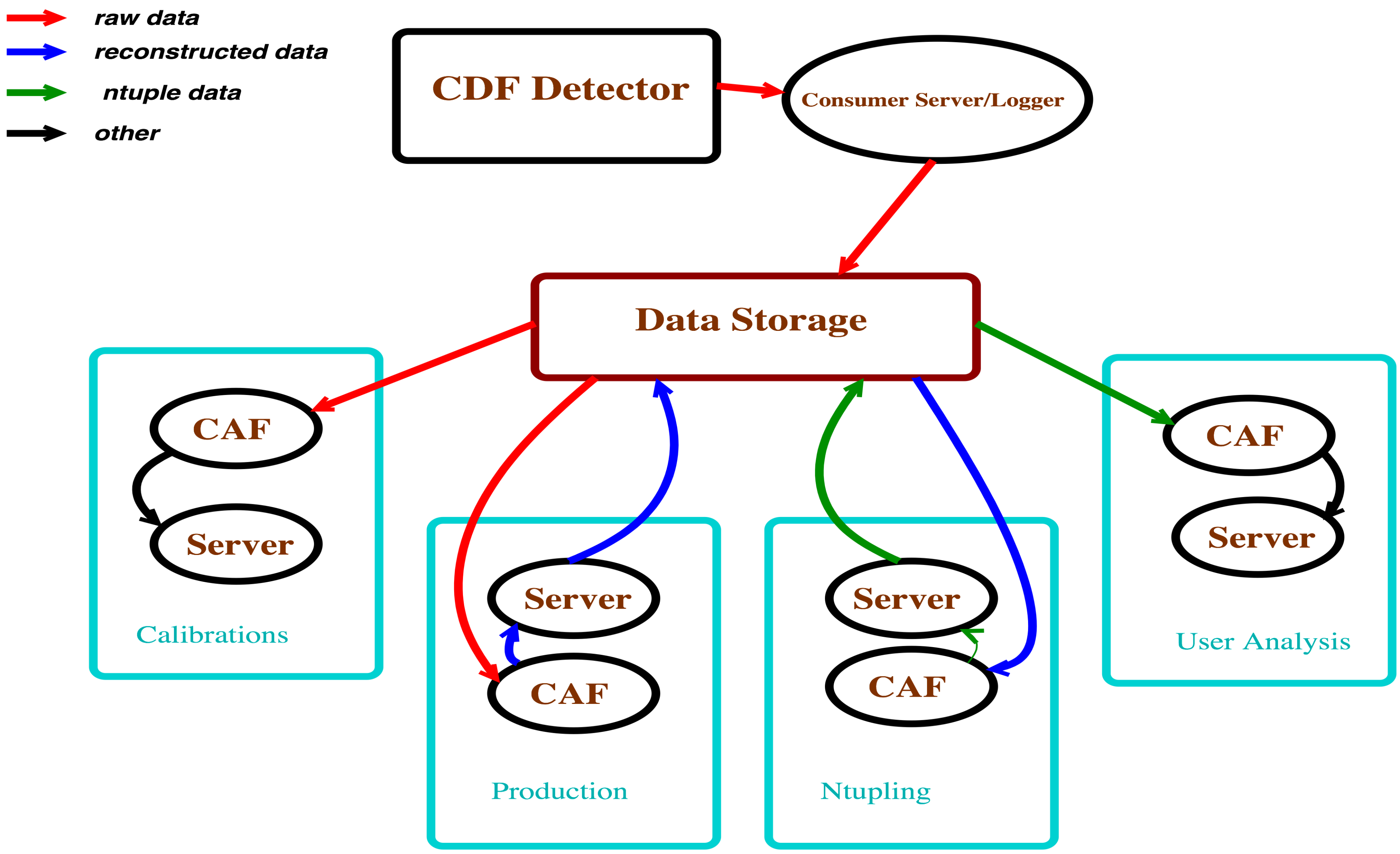
•The offline Analysis Framework at CDF

•*Elena Gerchtein, FNAL*

•

- ***Offline data processing***
- ***Software framework***
- ***Software management***
- ***Data storage***
- ***Simulation framework***
- ***Summary: CDF Offline performance***

CDF Data Flow



CSL: Consumer-Server/Logger

- **The final component in CDF Data Acquisition chain before data is archived to tapes**
- **Primary functions of the CSL**
  - –receives data from final stage trigger and sorts it into "physics streams"
  - –records meta-data information into a database
  - –sends fraction of events for real time monitoring of data quality
- **Buffering capacity:**
  - **– 24 TB of disk arrays can handle 3 days of buffering data with 80 MB/s**
- **Staging capacity:**
  - **– Stager (archiving data to FCC) can digest 100 MB/s continuously for more then 24 hours (dry-run test)**
- **CSL was upgraded in 2006**

CDF Software: AC++ framework

- **AC++ provides a unified framework for event reconstruction, post-reconstruction analysis, online triggering, calibration and monitoring**
- Used both in **online** and **offline** environment
- **Module concept**: generic operations for processing physics event
  - initialization; run dependent initialization; physics event analysis; end run; program termination; module "talk-to" processing
- **Module types:** input modules, output modules, filter modules, specialized modules
- AC++ executables allow to **configure** input for modules ("talk-to") using TCL scripts **w/o recompiling**. TCL provides complex menu structures and menu items of different types.
- Based on C++

CDF software: management

- **more then 15 million lines of code organized into 294 packages layered into fixed hierarchical tree of dependencies (managed by SoftRelTools)**
- **packages are grouped depending on place in hierarchy tree:** binaries-modules-algorithms-data objects-interfaces-infrastructures-standard utilities and services
- **great attention was paid to physical design.** Developers followed strict rules and guidelines enforced by integration leaders. CDF managed to avoid any cyclic package dependency, all CDF offline applications can be linked in one pass, with no libraries listed more then once.
- **Tools: CVS, SoftRelTools, code browser**
- Many developers leave a project after one year of service. Special attention was paid to ensure sufficient overlap between code developer and who will take over maintenance.

CDF software: development

- **Releases: major releases** (every two month during development stage, once a year at operation stage) **– integration releases – development release**
- Each release followed by well defined **validation procedures** (code and physics validation) and **regression testing**
- Release integration was managed by **code librarians** and **integration leaders**
- **C**ode performance has not been an issue. Wasteful copying was eliminated early in development. **Choosing efficient algorithms gains more performance then hand optimizing the code**
- **Problems:** memory leaks were traced with standard tools; uninitialized memory was a bigger problem, hard to debug.
- Crash rate of farm operations of the reconstruction is less then 1 crash per million events.

CDF computing facilities

- GRID enabled farms:
  - CDFGrid (5K)
  - NamGRID (300 MIT + 150 KISTI)
- Jobs running on GRIDs
  - Offline Calibration
  - Reconstruction (production)
  - Ntupling,
  - MC
  - User analyses

Data production: offline calibrations

- •Calibration constants for use at reconstruction level: offline calibrations
- •Each sub-detector system has associated calibration constants that are stored in a database. The constants may change from run to run (not implemented on runsection scale)
- •Offline calibration are calculated on specially processed physics data after data taking
- • ~1/5 of data is reconstructed to get calibration constants

Data production: reconstruction

- raw data are processed to produce higher level objects for each sub-detector applying different reconstruction algorithms:
  - tracks, jets, muons and electrons, etc.
- the whole process is called “production”
- only one executable: ProductionExe (AC++)
- only higher level objects are kept in the production output file to reduce event size

Adding user analysis layer: Ntupling

- run general analysis AC++ executable over reconstructed data and save data in simple ROOT-based format. ROOT ntuple are analyzed using ROOT-base scripts
- PRO:
  - dataset size reduced 70%
  - simplified format
  - corrections to reconstruction are done as needed
- CONS:
  - different physics groups run the same algorithms to produce different datasets: data redundancy and excessive CPU usage

User analyses

- Users access three types of ntuples and analyze data using ROOT-based scripts
  - STNtuple
  - TopNtuple
  - BSTNtuple
- unrestricted access to ntuples
- In special cases, when access to raw/produced data is justified and unavoidable for particular analysis, it takes much more coordination effort to provide such access.

CDF software: lessons learned

- **Code:**
  - **Need to work harder to enforce better coding practice**
  - **critical design pieces should be reviewed by experts**
  - develop and maintain validation plan/code in parallel with major algorithms
  - Plan for maintenance: documentation, transitions, training
- **Algorithms**
  - **have speed/size target for each algorithm, subject to review and negotiation**
- **Operations**
  - **offline software shifts (run validation, checking reported software errors, do bug fixes) were helpful**

Data storage: mass storage system at Fermilab

- tertiary storage system (Enstore)
- data media: tapes
- Enstore is integrated with disk caching software called dCache, both share namespace PNFS
- access data directly on-site
- access data through disk caching software called dCache on-site and off-site

Data storage: Enstore

- Enstore provides a generic interface for experiments to efficiently use mass storage systems as easily as if they were native file systems
- client-server architecture allows hot swapping hardware, dynamic software configuration, it is platform independent, easily extendable. Most of the operations are transparent to users.
- http://www-ccf.fnal.gov/enstore/design.htm

Data storage: PNFS namespace

•PNFS is a global namespace developed jointly by Fermilab, DESY, and NGDF

•PNFS is used by Enstore and dCache to distribute filenames and other storage metadata

•PNFS has UNIX file system like directory structure, can be mounted on on-site computers just like nfs, or accessed indirectly through various Enstore and dCache protocols

•Files are accessed directly by their PNFS name. When a user copies a data file from a local disk to Enstore or dCache system, the destination is specified in terms of PNFS name. The data file gets copied to a storage volume and a corresponding metadata entry is created in the PNFS directory.

Data storage: dCache system

•Files on tapes can be accessed via **dCache, distributed storage solution**. dCache hides the complexity of data movement from user, so the end-user sees only the large amount of storage. The flow of data in dCache is carefully controlled to minimize the impact of chaotic data access.

•When reading files from dCache,   data are retrieved from tapes and stored on local disks providing high-performance access to frequently used files.

•Files transferred to dCache are written to the tape after some time limit or data accumulation, transparently to user

Data storage: SAM

- **SAM** is another **data storage solution**
  - **store and retrieve files and associated metadata**
  - **file catalog**
  - file caching
  - **job submission, for local or grid-aware systems, which is coupled with file delivery system**
- CDF is accessing data using SAM, while file caching is done by dCache

Data storage: media type

- CDF media type: **LTO3, LTO4 tapes**
  - currently LTO-4 (800GB native capacity, 120 MB/sec max transfer rate)
  - moving toward higher density of data: recent advances in LTO technology allow to store more data in the same size cartridge. CDF started with 9940B tapes (200GB) that migrated to LTO-4 tapes

- CONS: expensive

Data storage: CDF data

- CDF using slots in the tape robotic libraries, located in FCC and GCC:
  - CDF-LTO3
  - CDF-LTO4F1
  - CDF-LTO4G1
  - CDF-9940B-D0
- CDF raw data: **1222 TB on tapes** (periods 0-27 run range 138425-287261)

Data storage: lessons

- need better plan for resource allocation/recycling
  - tapes may be need recycling in the nearest future
  - diskpool for temporary output is 50% junk data but no tools/policy on how to clean up
  - jobs waiting for files from tape and wasting CPU

Simulation at CDF

- **Simulation framework**
  - design and infrastructure
  - generators and decay packages
  - detector simulation
  - trigger simulation
- **Simulation performance**
- **Simulation: User's view**

Simulation Framework (SF): design and infrastructure

- **SF is integrated into AC++ application framework**

• design is based on a mixture generic programming and OOP
=> SF is easily extensible and time efficient

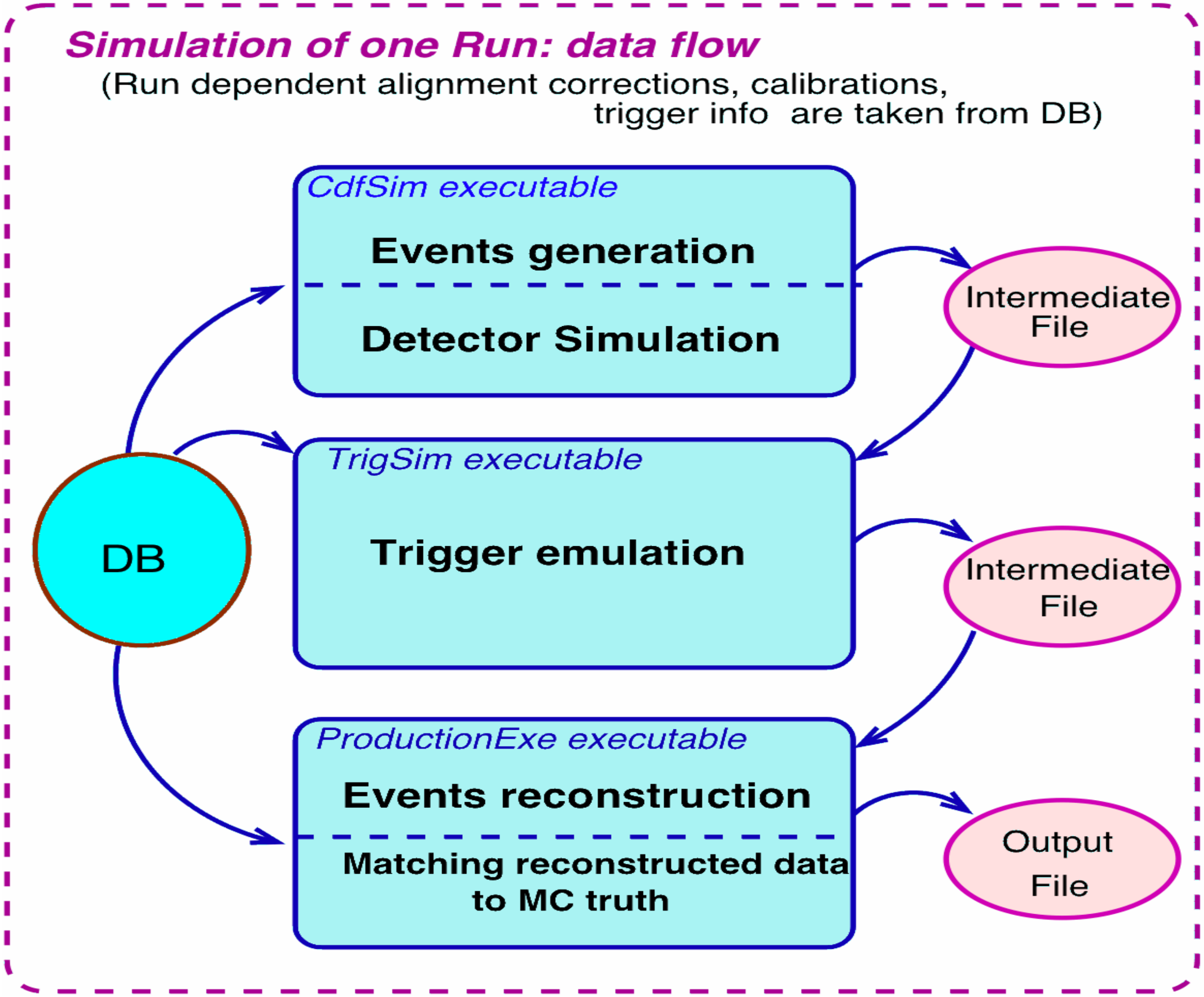
• CDF uses **the same geometry for reconstruction, simulation, and visualization**

• The same data objects are used for simulation and real data reconstruction

•Software provided to submit large scale MC jobs to GRIDs according to complicated run dependent job plan

• Use of **random number generators** is unified throughout SF (CLHEP package), each AC++ module has at least one independent random stream to ensure statistically independent production of large MC samples. Each random stream initialized with random seeds provided with user job plan.

Part of MC job plan: One run MC segment



generators: what do we need?

- **CDF is a multipurpose detector**
- broad physics program
  - –Top – precision EWK program
  - –Search for new phenomena
  - –Tests of perturbative QCD
  - –B physics
- **Need to be able to plug in ANY event generator into SF with minimum efforts by demand. Would like to read arbitrary event generator output and pass it through detector simulation.**

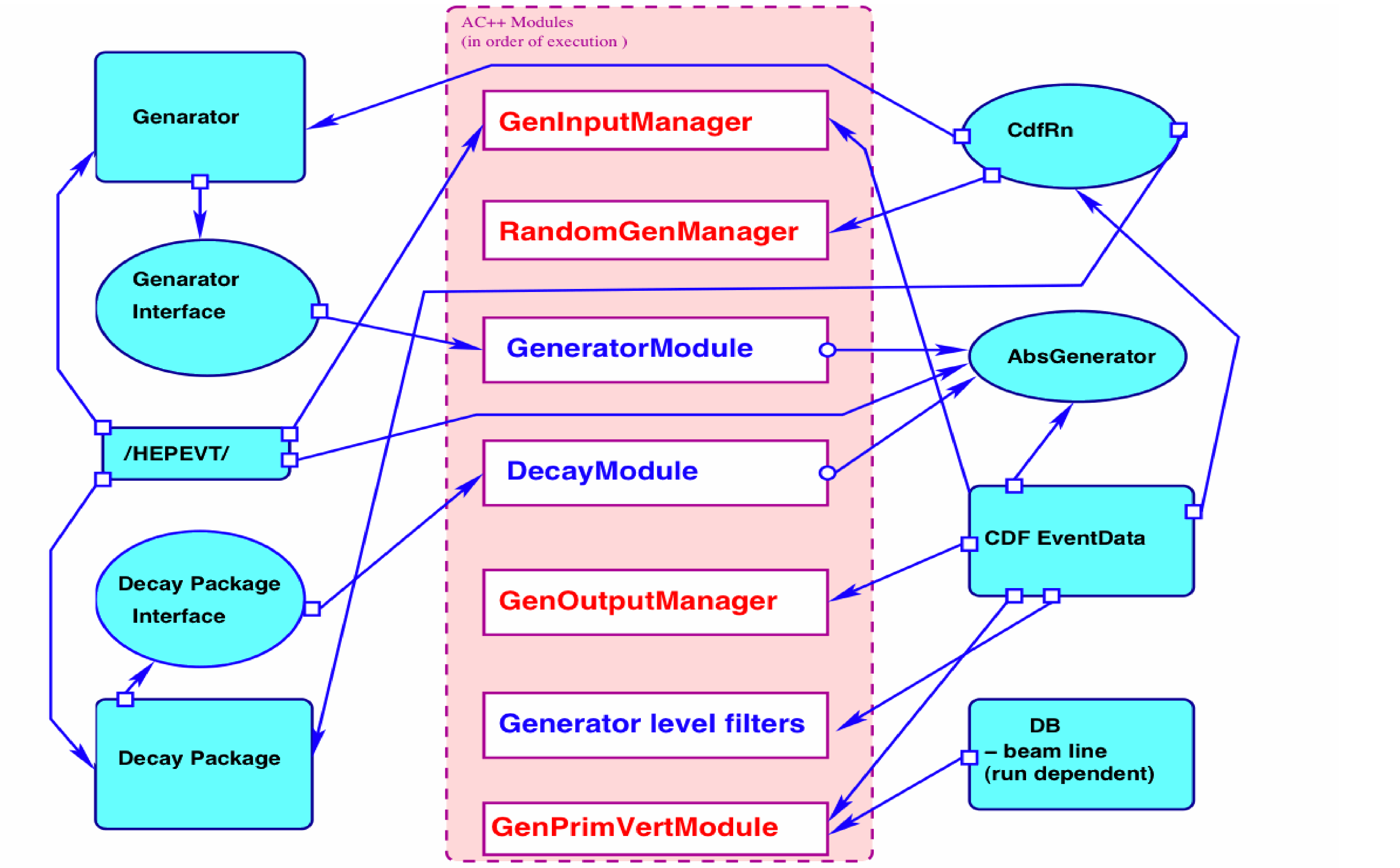
SF: generators

- **cdfSim allows to generate physics events with different generators:**
  - Herwig 6.4 – PYTHIA 6.2 – ISAJET 7.51 – WGRAD – WBBGEN – GRAPPA (GRACE for ppbar) – VECBOS – Bgenerator(quarks only) – HeavyQuarkGen – MinBiasGenerator (parametrization for diff.physics) – Single Particle
  - Les Houches Accords – universal interface between matrix element generators and MC programs – implemented in PYTHIA and GRAPPA
- **Decay packages:** QQ 9.1(obsolete) – EvtGen - Tauola

SF: Generators – implementation details

- **Generator and decayer sequence** is incorporated within AC++ framework
- Generator related AC++ module has an interface to generator/decay package controlling the access to the underlying FORTRAN routines
- Generators/decay packages communicate via **HEPEVT** common block. After event has been processed, HEPEVT is converted to persistent object and added to the event record
- can use output of stand-along generators (generation mode 3)

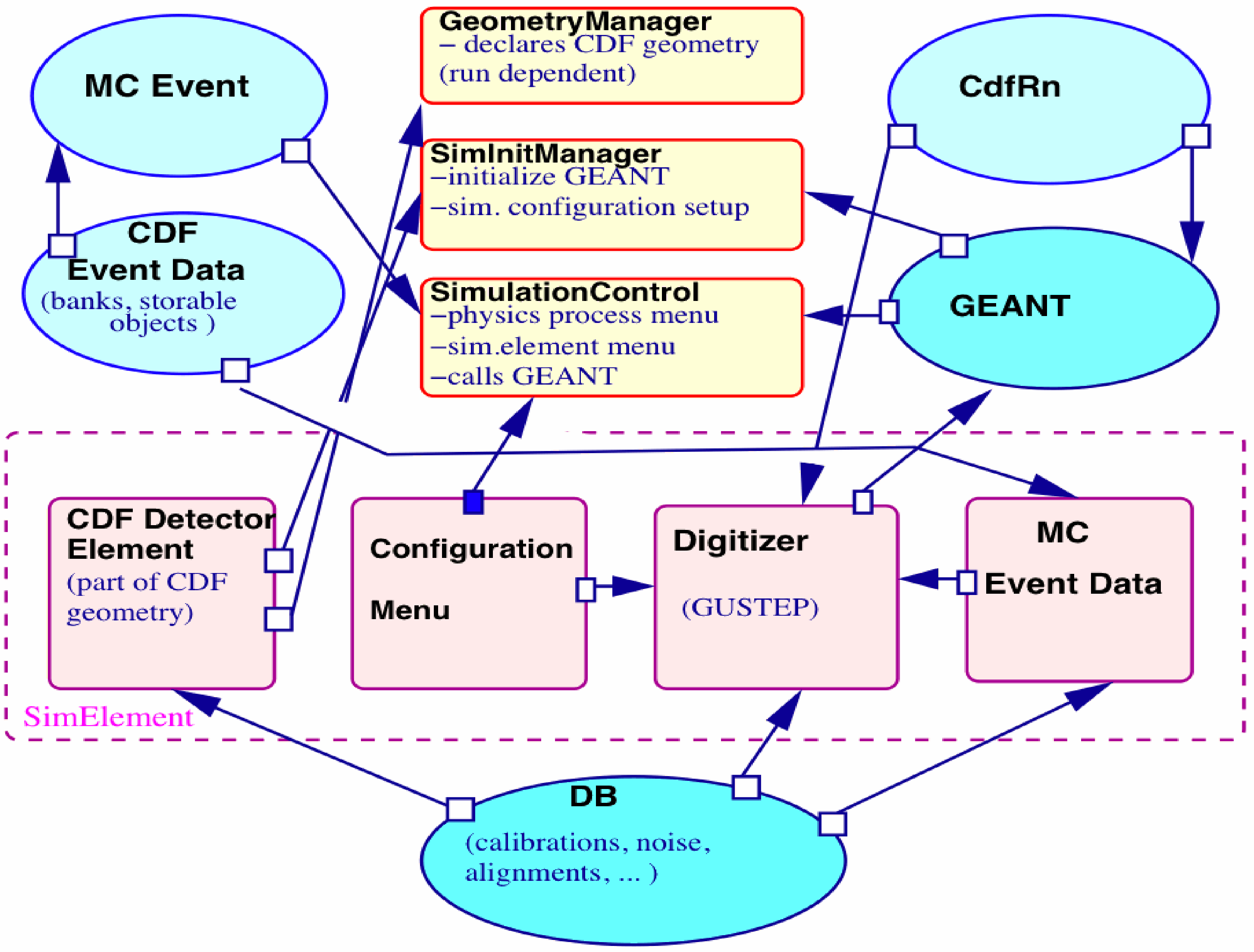
Generator modules within AC++



SF: Detector Simulation

- **Detector simulation** is integrated into **AC++ application framework**
- Implemented as an abstract factory – easy to extend
- **Detector simulation elements:**
  - geometry description
  - configuration menu
  - digitizer
  - event data object
- Tracking of particles through matter is based on **GEANT3** tracking. Fast parametric simulation available for some sub-detectors

SF: Detector Simulation

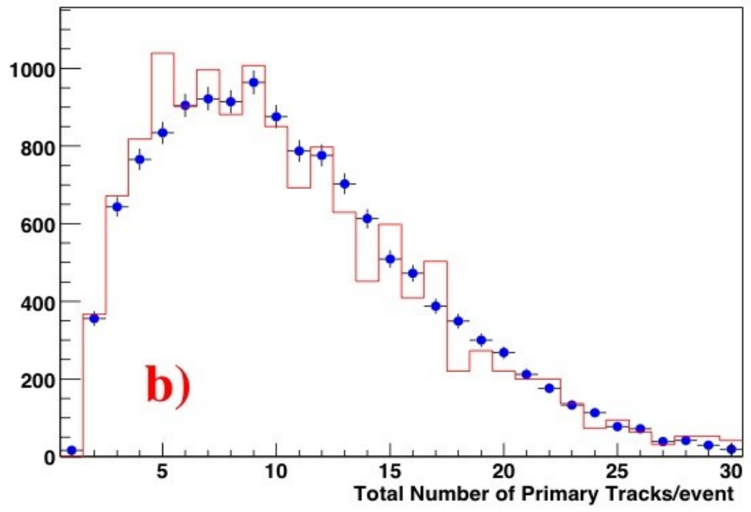


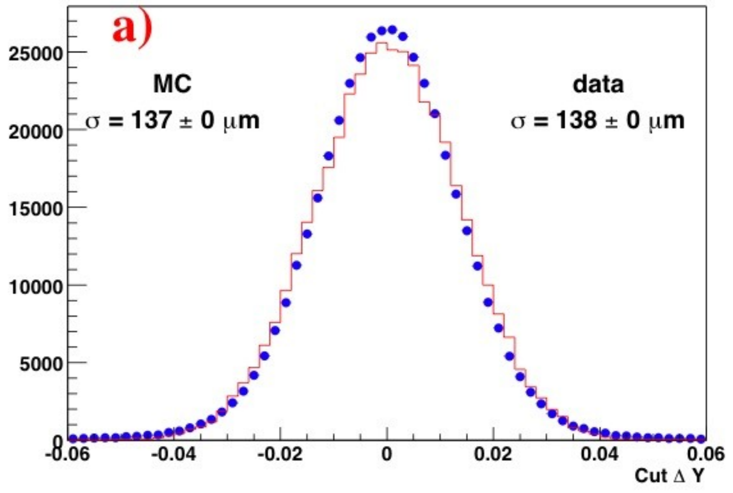
SF: Simulated subdetectors

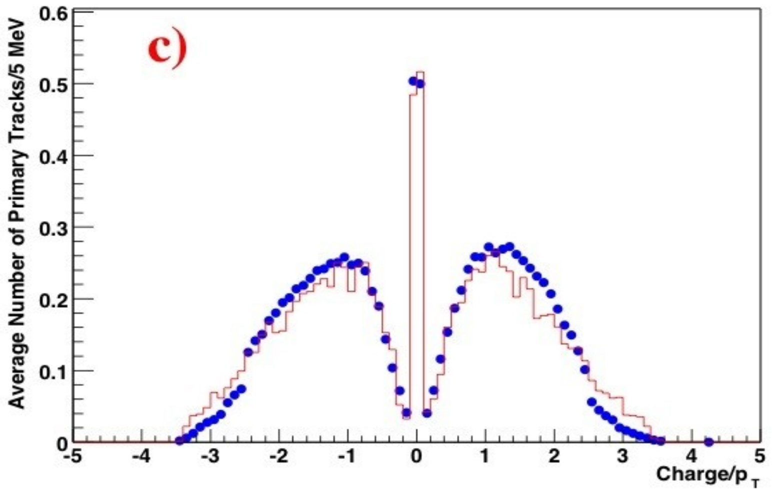
- **cdfSim allows configuration of subdetectors with different geometry levels and different physics processes, depending on desired accuracy vs. time efficiency (parametric and full simulation)**
  - –Silicon detector (SVX, ISL)
  - –Central Open-cell Tracking chamber(COT)
  - –Muon detectors (CMU,CMP,CMX,IMU)
  - –Time-of-flight system
  - –EM and HAD calorimeters
  - –Cherenkov luminosity counter(CLC)
  - –Very forward detectors(Miniplug, BSC,RPS)

SF: COT simulation

- **GARFIELD drift model**
- W-> μv
- **data(points):** high Pt muons (>18 GeV/c)
- **MC(hist):** Pythia, DM GARFIELD, tuned
- a) Residual ΔY
- b) Track multiplicity of primary tracks
- c) Charge/Pt of primary tracks

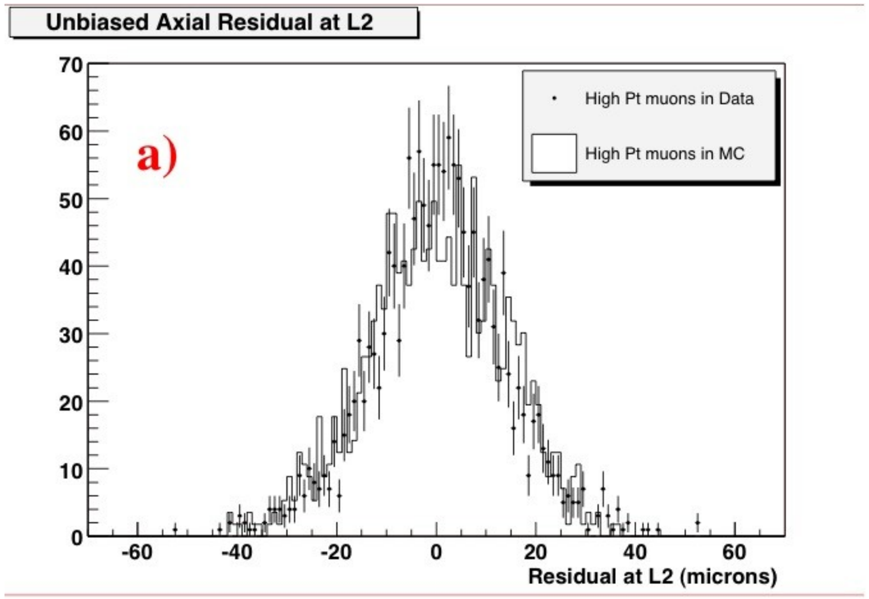


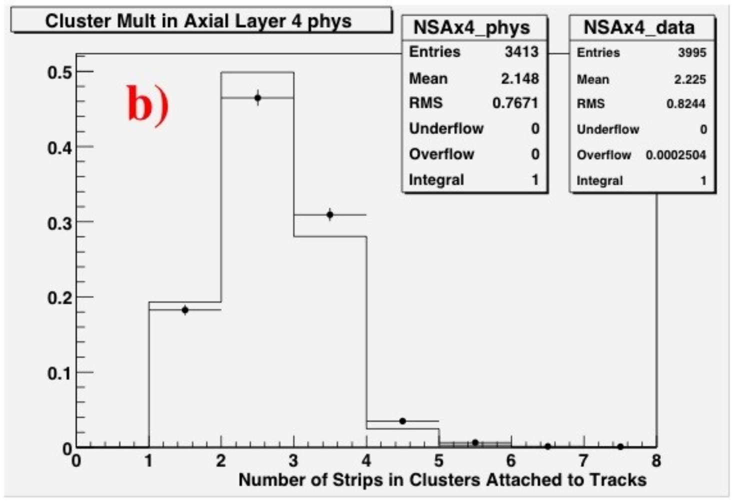




SF:Silicon detector simulation

- **Silicon microstrip detector with double sided readout**
- Charge deposition models (CDM) affect cluster efficiency and intrinsic resolution
  - geometric/parametric/physical
- To get impact paramrter right:
  - complicated geometry
  - alignment
  - beam parameters
- **MC(hist) single particles compared to 15 GeV/c muons data(points)**
- **a) Intrinsic resolution (layer 2, cluster width 2, calculated using unbinned likelihood fit)**
- **b) Cluster profile for layer 4**





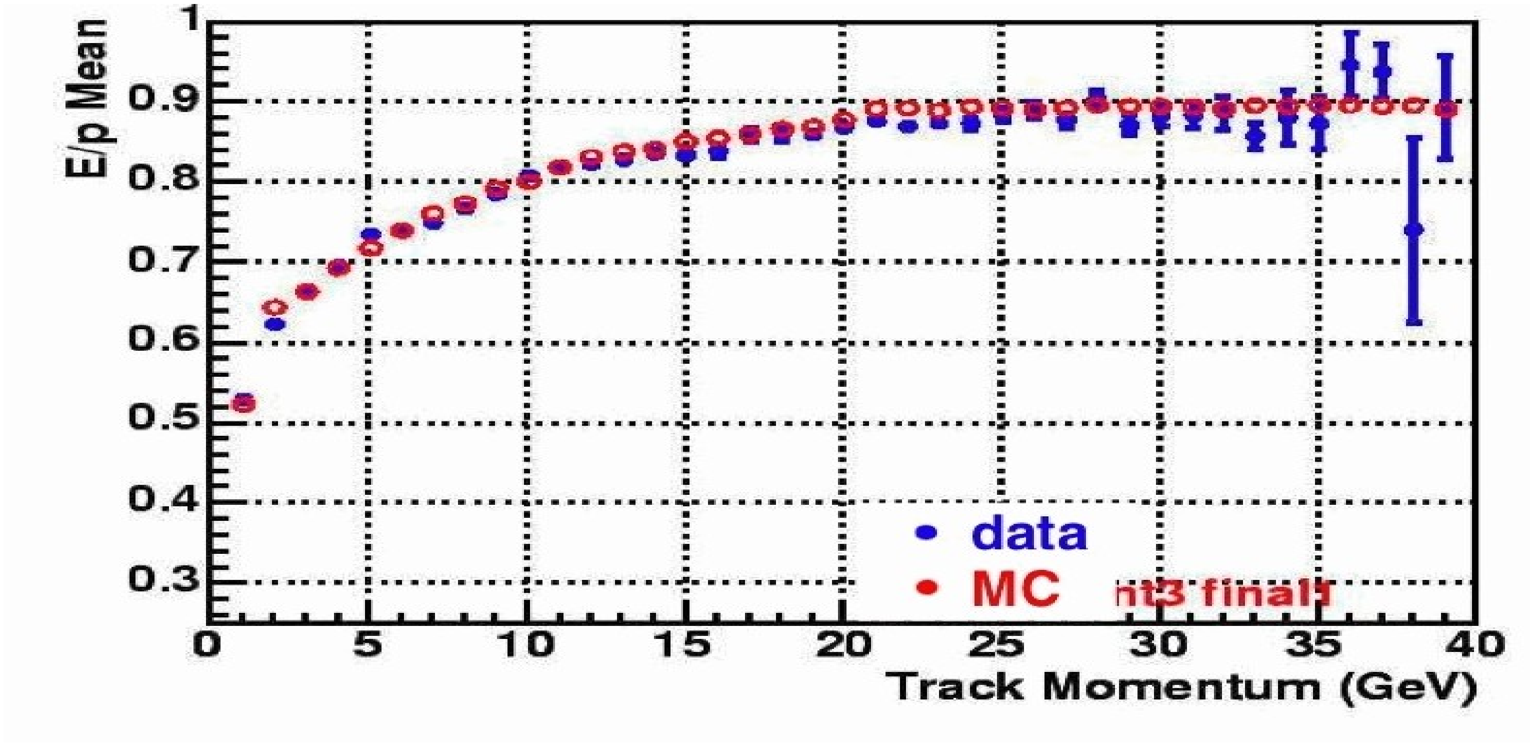
NSAx4_phys	
Entries	3413
Mean	2.148
RMS	0.7671
Underflow	0
Overflow	0
Integral	1

NSAx4_data	
Entries	3995
Mean	2.225
RMS	0.8244
Underflow	0
Overflow	0.0002504
Integral	1

SF: Calorimeter simulation

- Particle propagated from interaction point through detector volume up to first inelastic interaction, then control passed to **GFLASH** (*G.Grindhammer, M.Rudowitcz and S.Peters, NIM A290(1990)469*)
  - –**fast robust tunable simulation of EM and HAD showers** (in CDF it is 100 times faster then G3 shower), ideal for simple geometry with repetative sampling structure.
  - – uses G3 material/geometry information
  - –generates longitudinal and lateral shower profile
  - –distributes energy spots according to lateral profile and sampling fluctuations

SF: Calorimeter simulation



- High P response in central calorimeter

•MC (open circles) compared to 57 GeV testbeam data (hist)

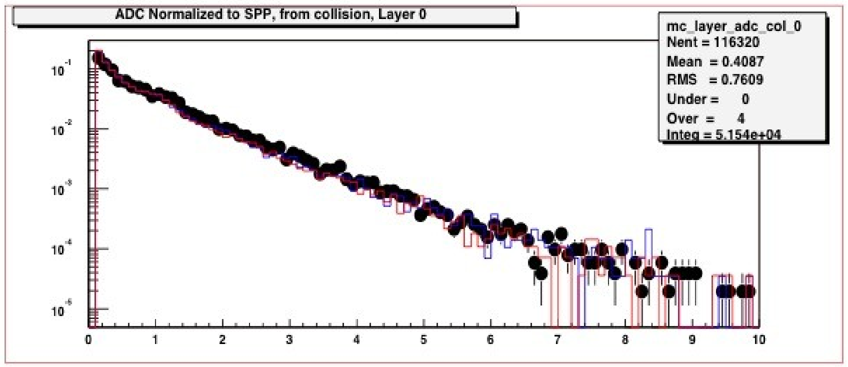
SF: Cerenkov luminosity counters

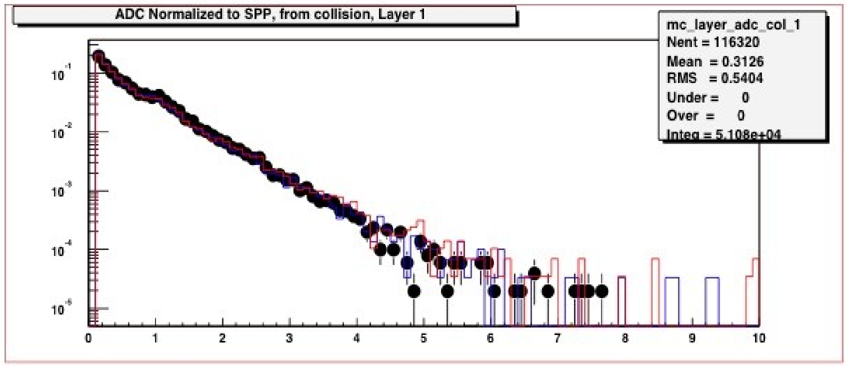
CLC is used to measure **luminosity** at CDF. CLC acceptance to ppbar inelastic process is estimated from simulation and gives major contribution to lumi uncertainty

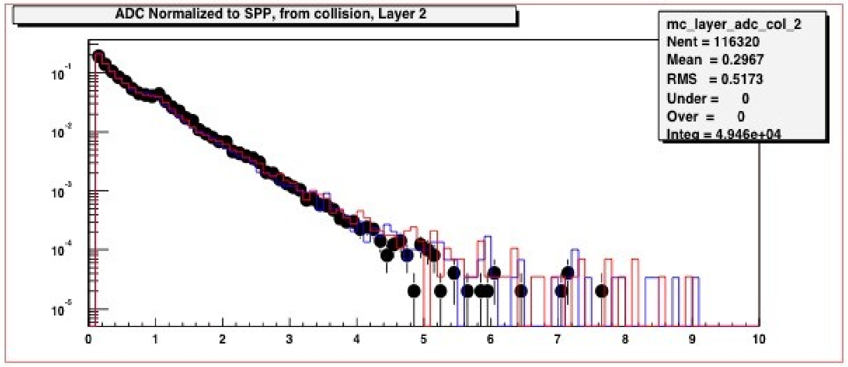
Generation and propagation of Cherenkov photons is simulated by GEANT3. Geometry in front and around CLC is described with high detail level

Uncertainty due to CLC simulation in the luminosity measurement is less then 4%

Amplitude distributions in CLC counters (3 layers); MC(dots) compared to data (red and blue hist. corresponding to west and east sides respectively)

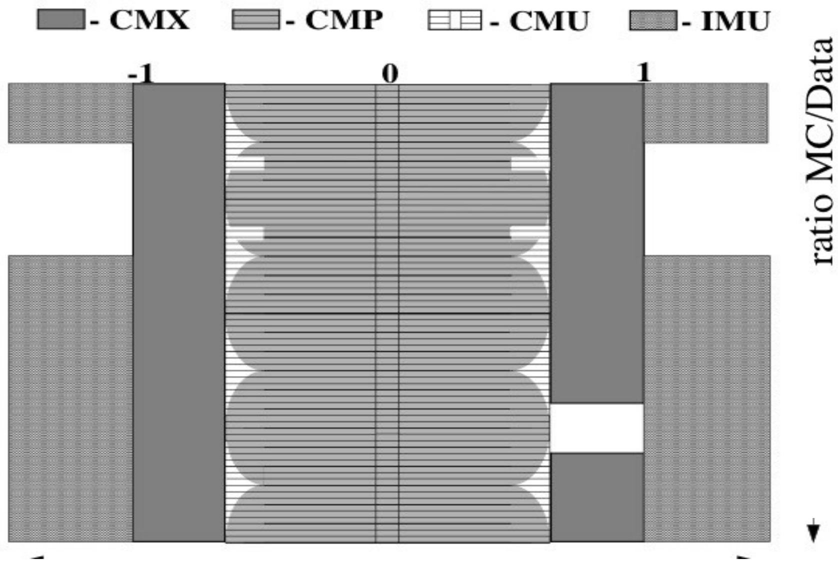


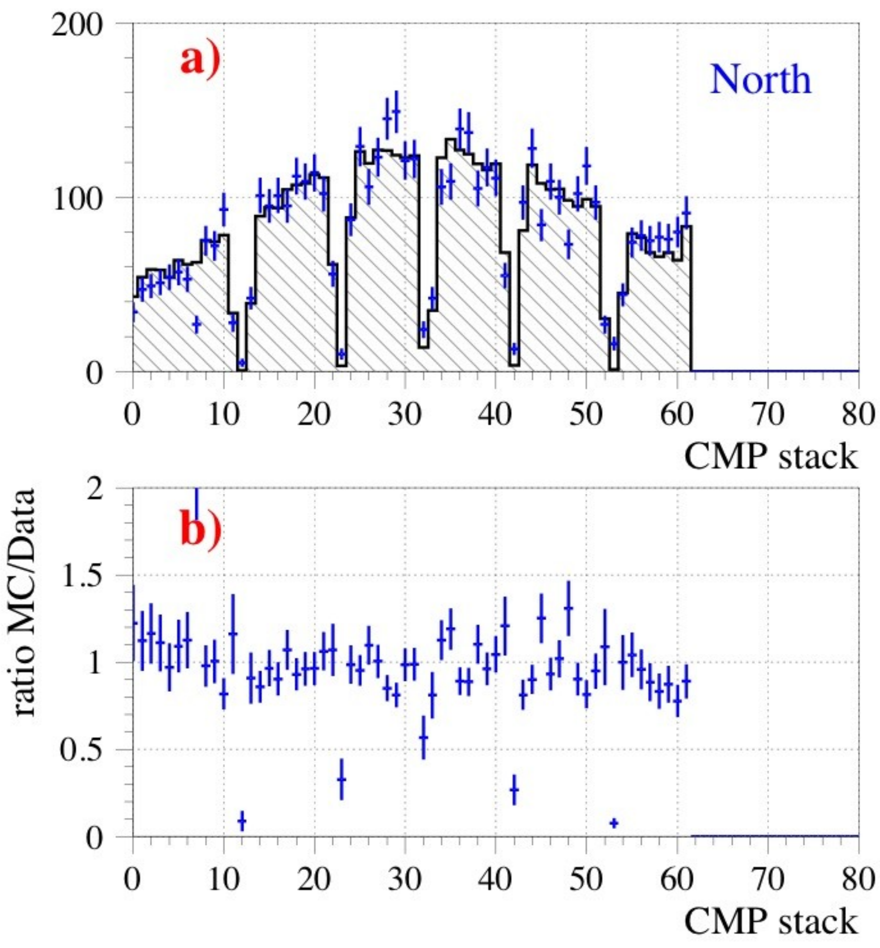




SF: Muon system simulation

- **Challenging geometry (below: location of central muon component in azimuth φ and pseudorapidity η)**
- W-> μν **data(points) compared to MC(hist.) single muons**
- **a)** number of CMUP muons for each stack in the north wall
- **b)** Ratio MC/data





SF: user's perspective

- **Running MC jobs is quite transparent to user**. There is no need to look at the code. There are infrastructure scripts that allow manipulating MC submission to GRID while taking care of complicated schema of run dependent job plans
- User actions:
  - Configure TCL script for physics processes to be generated and detectors to be simulated
  - Generate job plan of jobs that will be run on GRID depending on run range and the number of desired events
  - Submit MC job to GRID and get the output back

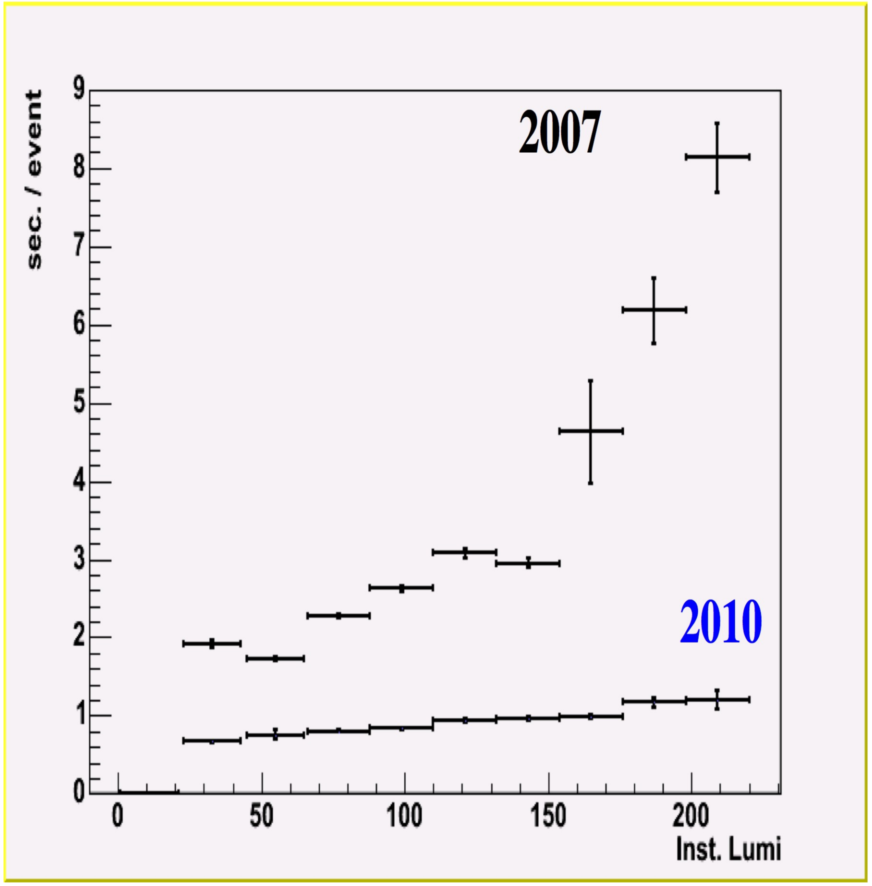
CDF offline: lessons

- Plan for scalability more then you currently expect
  - CPU gets faster
  - loads get bigger (more jobs are submitted)
  - systems run longer then expected
  - datasets get bigger
  - scalability in DB access, job control, data access
- Frequent system downtimes were unexpected, not all services are easily restored (CDF has major downtimes for security upgrades every 60 days)

CDF offline: performance

- Current processing time per event
  - reconstruction ~1 sec
  - ntupling ~2 sec
  - MC ~4 sec
- PLOT: average CPU time per event for reconstruction vs. average inst. lumi per run
- Great improvement after tracking upgrade and switching to more powerful machines

Average CPU time per event


sec. / event
Inst. Lumi