Computing in High Energy Physics

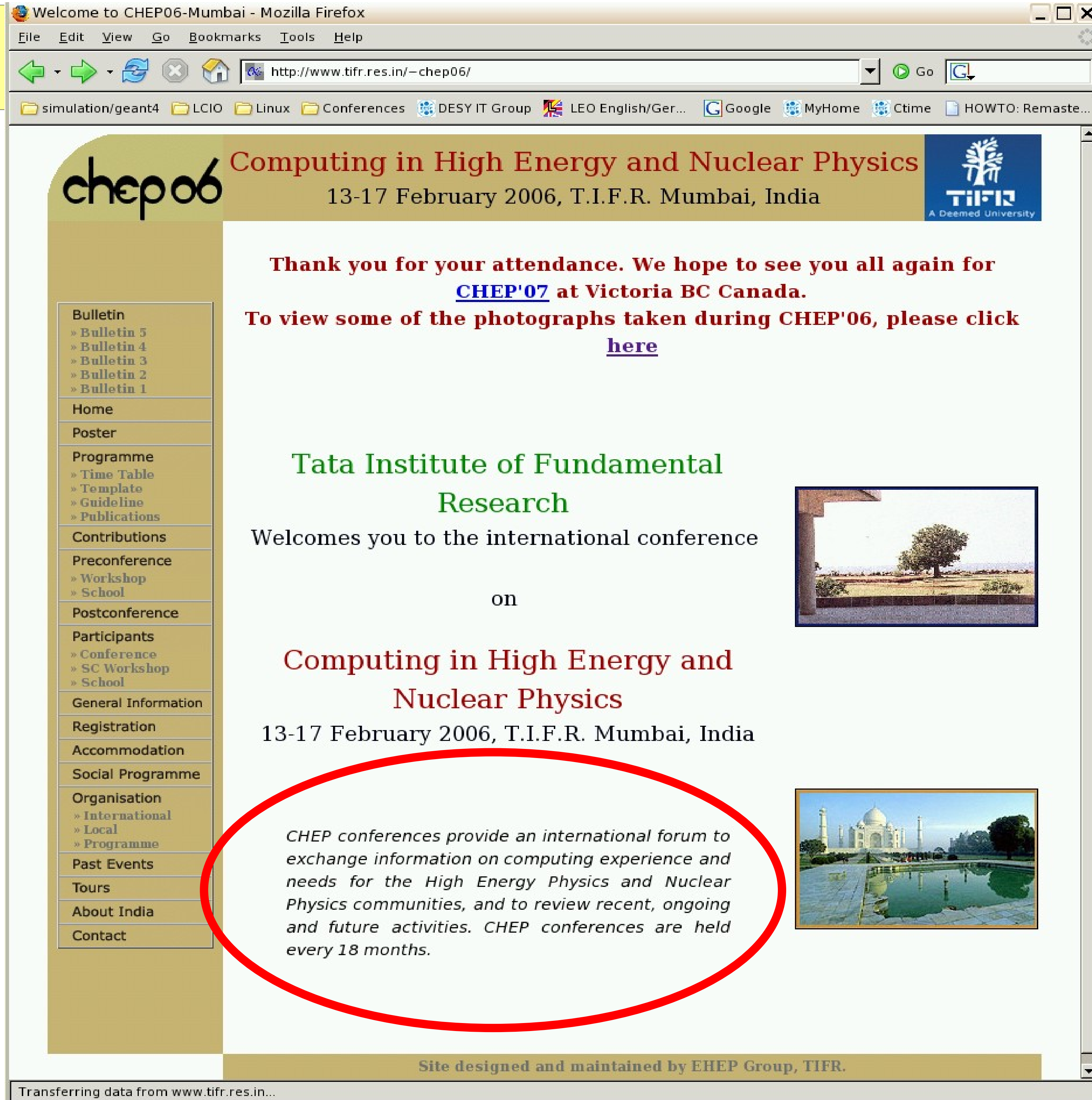
An Introductory Overview

Frank Gaede
DESY *IT Physics Computing*

Summer Student Lecture
DESY, August 20, 2008

Introduction

- The aim of this lecture is to provide an overview and some understanding of the basic concepts of ***C**omputing in **H**igh **E**nergy **P**hysics*
- the (randomly :)) chosen topics are of course a subset of possible topics
- we will just scratch the surface of this wide field
- for an overview of topics that are currently discussed and under development see the programme of the second to last **CHEP-**Conference in Mumbai(next two slides):

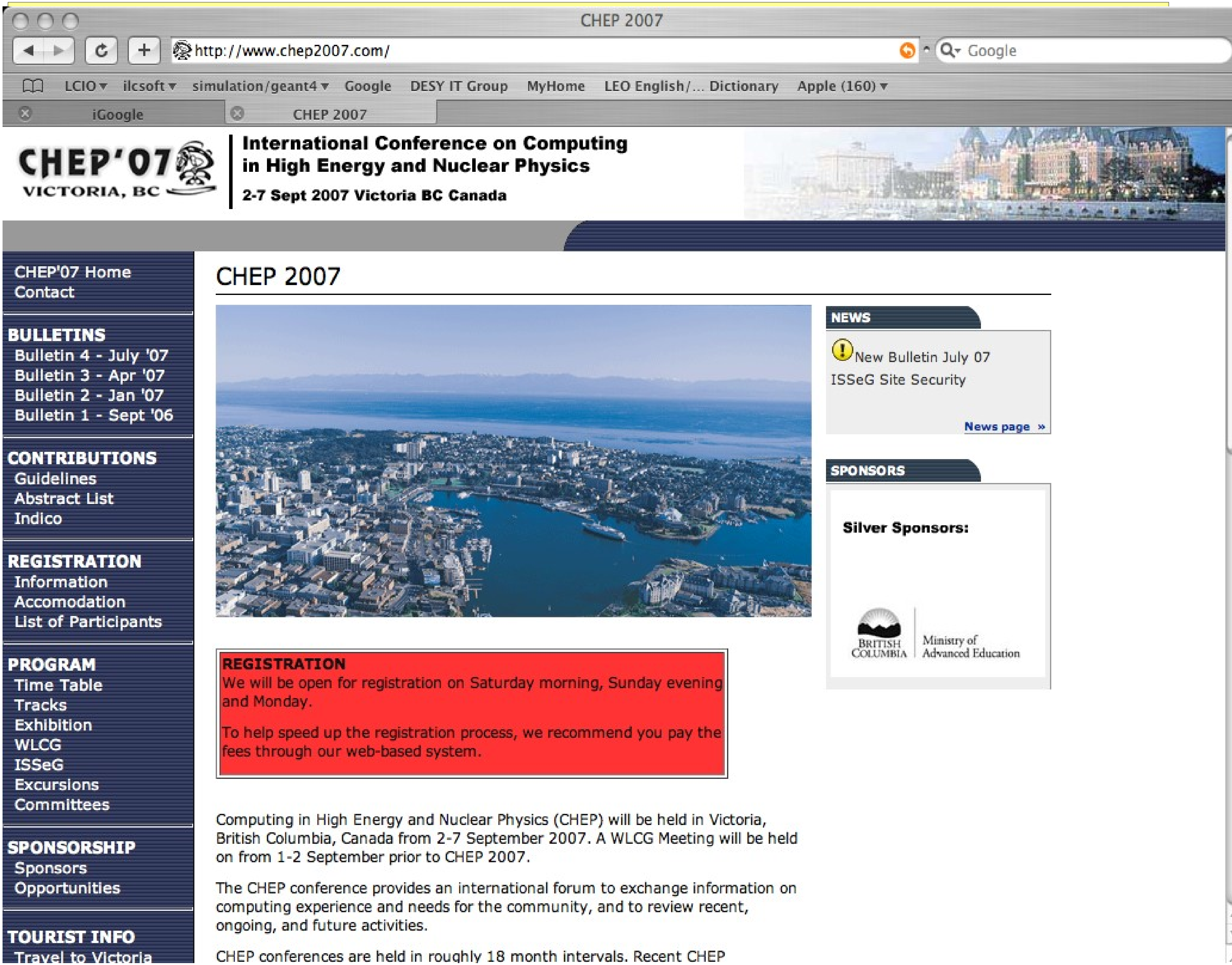

Welcome to CHEP06-Mumbai - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.tifr.res.in/~chep06/
simulation/geant4 LCIO Linux Conferences DESY IT Group LEO English/Ger... Google MyHome Ctime HOWTO: Remaste...
chep06
Computing in High Energy and Nuclear Physics
13-17 February 2006, T.I.F.R. Mumbai, India
TIFR
A Deemed University
Bulletin
» Bulletin 5
» Bulletin 4
» Bulletin 3
» Bulletin 2
» Bulletin 1
Home
Poster
Programme
» Time Table
» Template
» Guideline
» Publications
Contributions
Preconference
» Workshop
» School
Postconference
Participants
» Conference
» SC Workshop
» School
General Information
Registration
Accommodation
Social Programme
Organisation
» International
» Local
» Programme
Past Events
Tours
About India
Contact
Thank you for your attendance. We hope to see you all again for CHEP'07 at Victoria BC Canada.
To view some of the photographs taken during CHEP'06, please click here
Tata Institute of Fundamental Research
Welcomes you to the international conference
on
Computing in High Energy and Nuclear Physics
13-17 February 2006, T.I.F.R. Mumbai, India
CHEP conferences provide an international forum to exchange information on computing experience and needs for the High Energy Physics and Nuclear Physics communities, and to review recent, ongoing and future activities. CHEP conferences are held every 18 months.
Site designed and maintained by EHEP Group, TIFR.
Transferring data from www.tifr.res.in...

CHEP 06 Programme I

- Online Computing
  - CPU farms for high-level triggering; Farm configuration and run control; Describing and managing configuration data and conditions databases; Online software frameworks and tools
- Event processing applications
  - Event simulation and reconstruction; Physics analysis; Event visualisation and data presentation; Toolkits for simulation and analysis; Event data models; Detector geometry models; Specialised algorithms for event processing
- Software Components and Libraries
  - Persistency; Interactivity; Foundation and utility libraries; Mathematical libraries; Component models; Object dictionaries; Scripting; Graphics; Use of 3rd party software components (open source and commercial)
- Software Tools and Information Systems
  - Programming techniques and tools; Software testing; Configuration management; Software build, release and distribution tools; Quality assurance; Documentation

CHEP 06 Programme II

- Computing Facilities and Networking
  - Global network status and outlook; Advanced technologies and their use in applications; HENP networks and their relation to future grid systems; The digital divide and issues of access, readiness and cost; Collaborative systems, progress in technologies and applications
- Grid middleware and e-Infrastructure operation
  - Integral systems (cpu/storage) and their operation and management; Functionality and operation of regional centres; Global usage and management of resources; Grid infrastructure and its exploitation in distributed computing models.
- Distributed Event production and processing
  - Development of the distributed computing models of experiments; Real experience in prototypes and production systems; Emphasis on the early days of LHC running.
- Distributed Data Analysis
  - Large distributed data-base over wide area network; Low-latency interactive analysis over wide area network; Collaborative tools for supporting distributed analysis; Remote access to and control of data acquisition systems and experiment facilities.



CHEP 2007

http://www.chep2007.com/

LCIO ilcsoft simulation/geant4 Google DESY IT Group MyHome LEO English/... Dictionary Apple (160)

iGoogle | CHEP 2007

CHEP'07 VICTORIA, BC

International Conference on Computing in High Energy and Nuclear Physics

2-7 Sept 2007 Victoria BC Canada

CHEP'07 Home
Contact

BULLETINS
Bulletin 4 - July '07
Bulletin 3 - Apr '07
Bulletin 2 - Jan '07
Bulletin 1 - Sept '06

CONTRIBUTIONS
Guidelines
Abstract List
Indico

REGISTRATION
Information
Accomodation
List of Participants

PROGRAM
Time Table
Tracks
Exhibition
WLCG
ISSeG
Excursions
Committees

SPONSORSHIP
Sponsors
Opportunities

TOURIST INFO
Travel to Victoria

CHEP 2007

REGISTRATION
We will be open for registration on Saturday morning, Sunday evening and Monday.

To help speed up the registration process, we recommend you pay the fees through our web-based system.

Computing in High Energy and Nuclear Physics (CHEP) will be held in Victoria, British Columbia, Canada from 2-7 September 2007. A WLCG Meeting will be held on from 1-2 September prior to CHEP 2007.

The CHEP conference provides an international forum to exchange information on computing experience and needs for the community, and to review recent, ongoing, and future activities.

CHEP conferences are held in roughly 18 month intervals. Recent CHEP

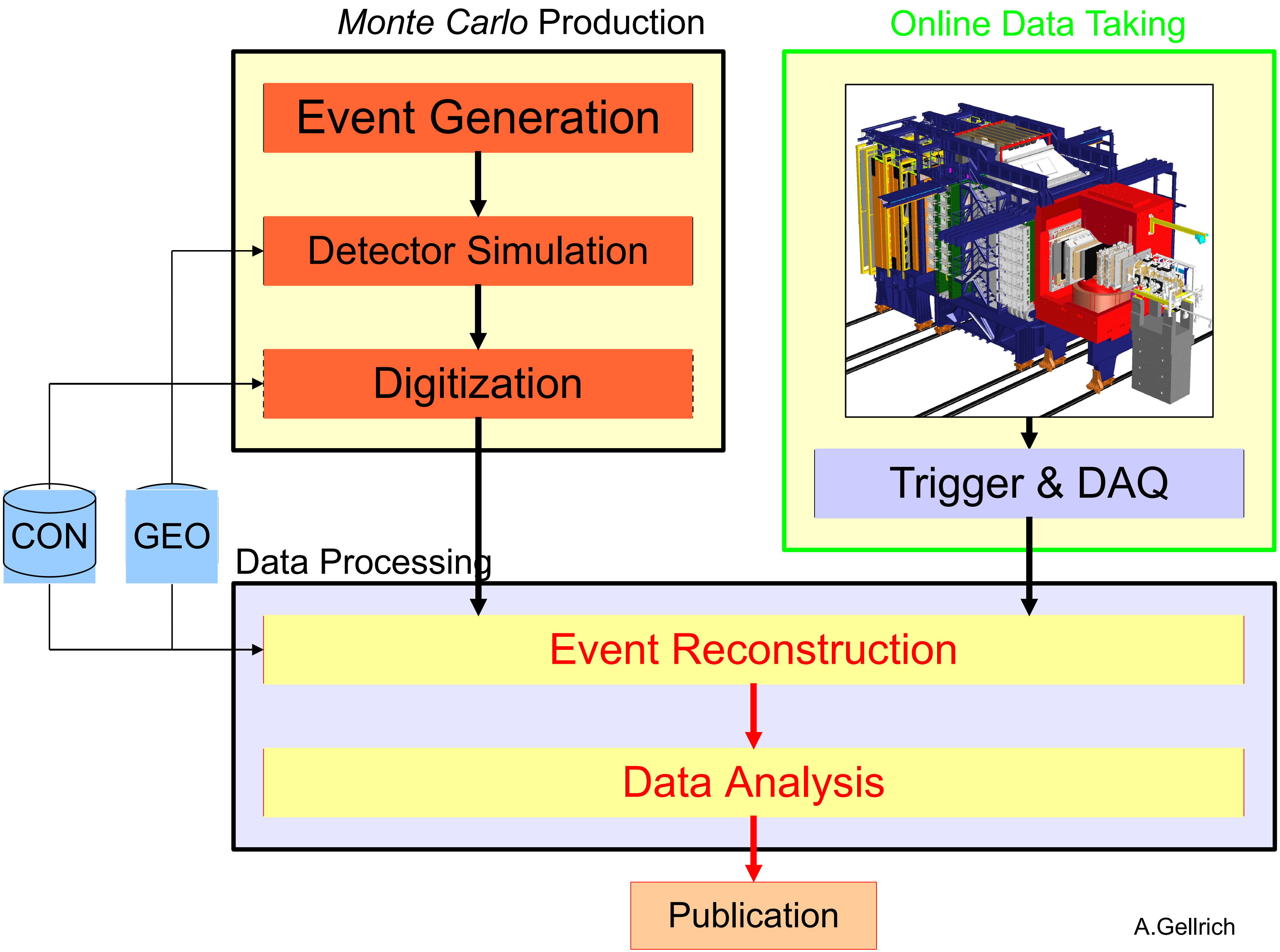
NEWS
New Bulletin July 07
ISSeG Site Security
News page »

SPONSORS
Silver Sponsors:
British Columbia | Ministry of Advanced Education

Selected Topics

- Online Computing - DAQ (data acquisition)
  - Readout software
  - Monitoring
  - Trigger
- Offline Computing
  - Monte Carlo Simulation
  - Reconstruction
  - Analysis – Software Frameworks
- Computing infrastructure (hardware)
  - large PC farms
- GRID Computing

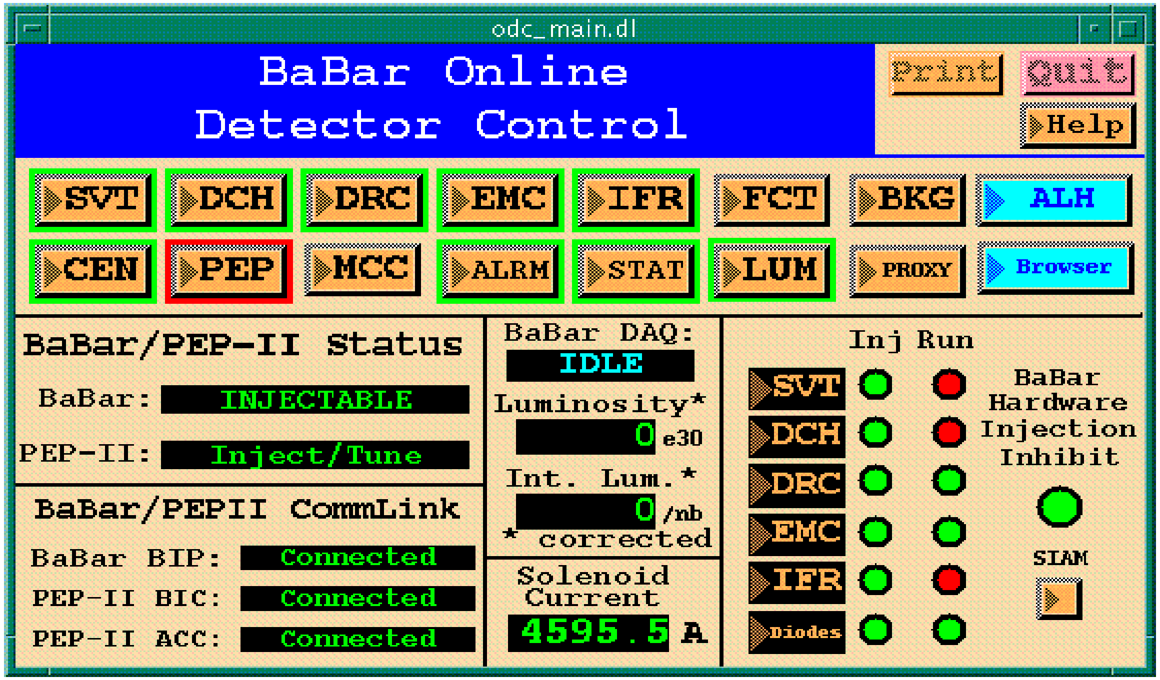
HEP Computing overview



Online - DAQ

- The Online/DAQ computing makes sure that the interesting physics data is read out from the detector and written to tape/disk (mass storage)
  - it is typically divided in three main tasks:
    - Online Monitoring (*slow control*)
      - temperature readings, high voltage, gas supplies...
      - manage the running of the detector
    - Trigger (software/hardware)
      - give signal that data needs to be read out 'coz sth. interesting happened in the detector
    - readout (data flow)
      - actual readout is tightly coupled to hardware (*front end electronics*)

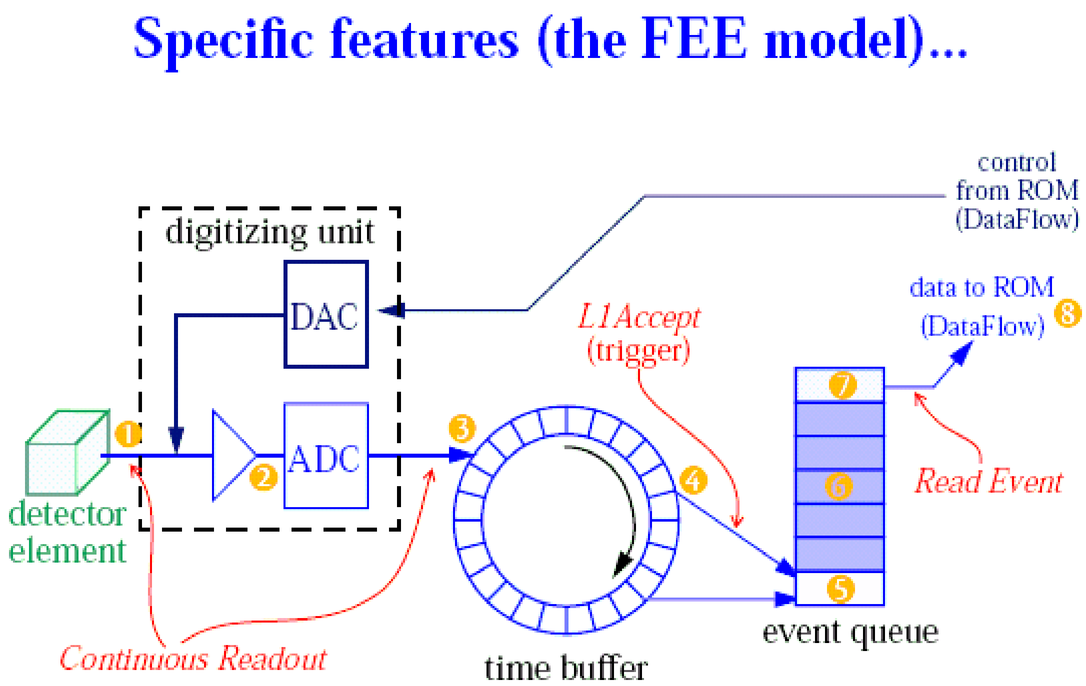
Online detector/run control



Modern particle physics detectors are run using online software tools, *example: BaBar ODC*

- Online Monitoring – Slow Control systems typically provide a GUI that allows the physicist to run and monitor the detector, by;
  - configuring the detector / online software / trigger
  - start & stop data taking runs
  - monitor temperature readings, high voltage, gas supplies...

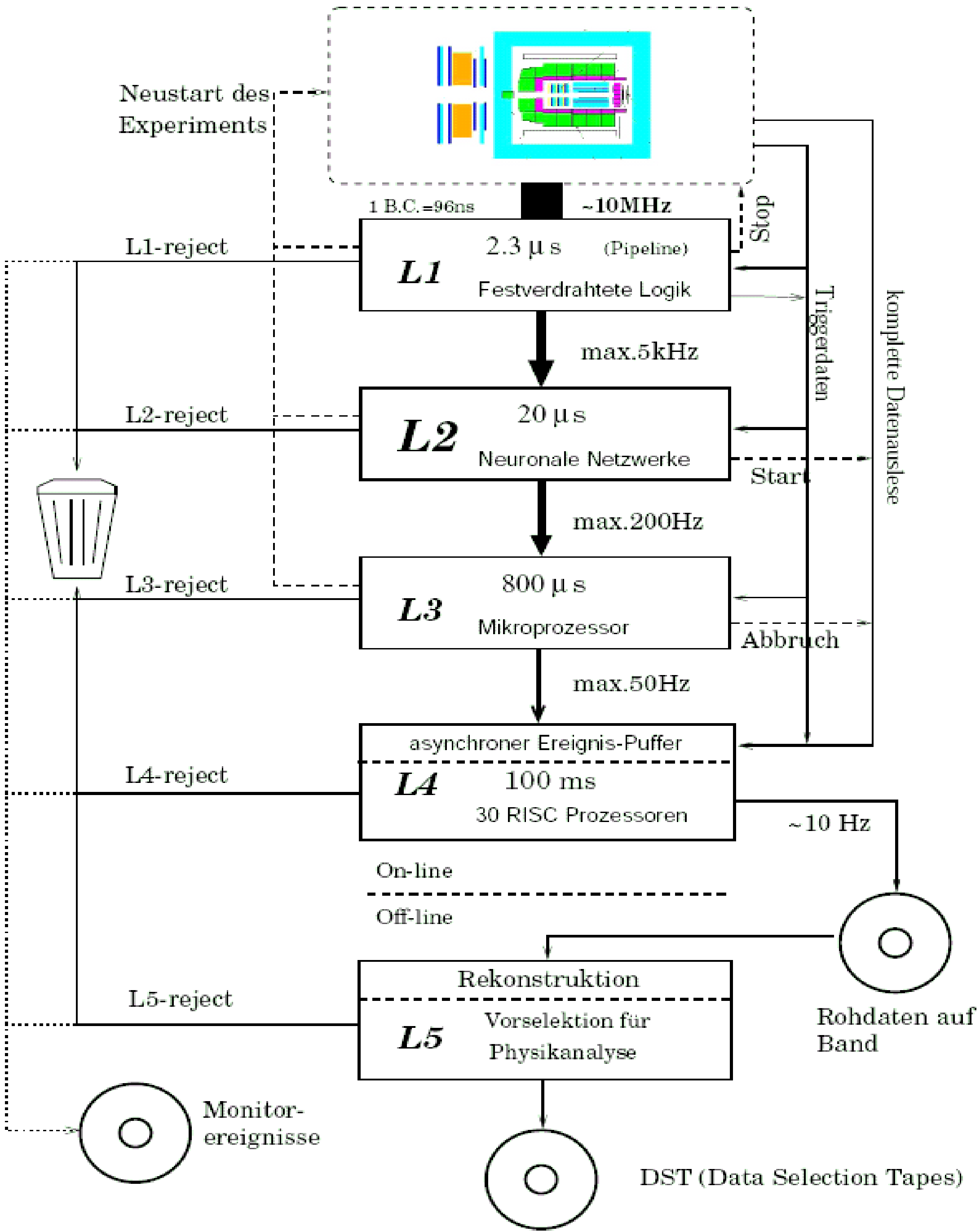
Readout software



example: front end readout software (Data Flow) of the BaBar experiment

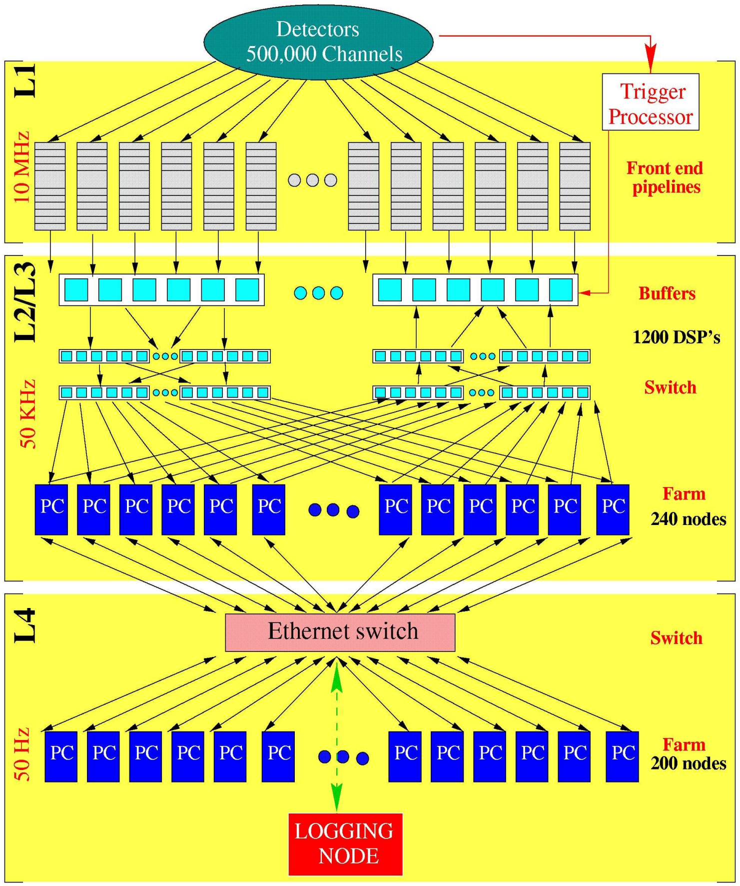
- the readout software is very tightly coupled to the hardware. ie. front end electronics and readout boards it typically involves tasks as:
  - buffering of data read out from the detector
  - feature extraction (zero suppression, integrating electronics signals, fitting of peak positions,...)

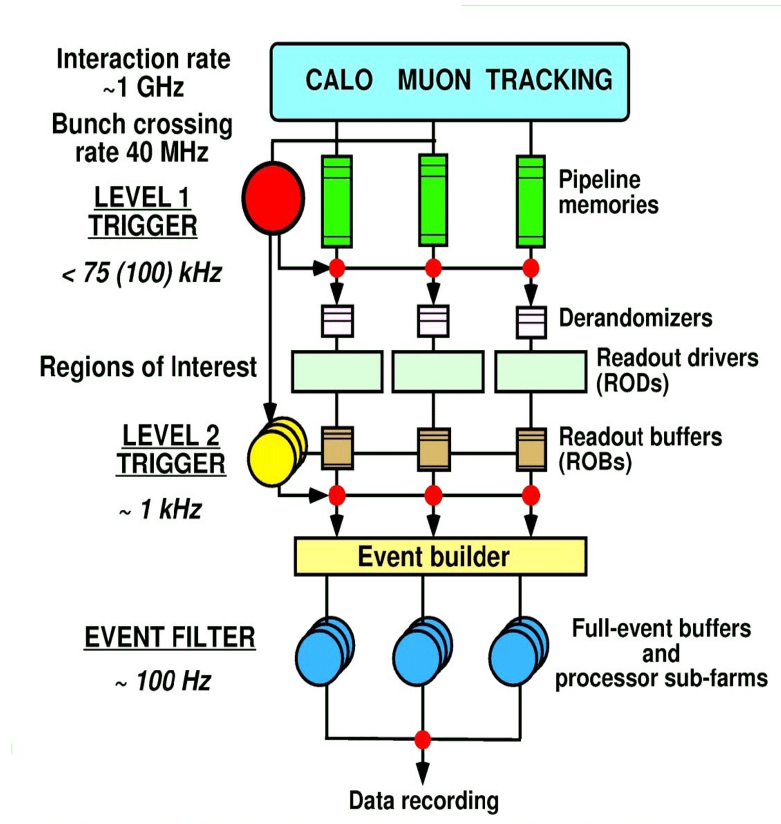
Multilevel Trigger System I



- **trigger**
- typically collider
- experiments have far more
- activity in sensitive parts
- than can be read out,
- stored or analyzed
- due to:
  - background from beam-gas interactions
  - high cross sections of (soft) relatively uninteresting physics,e.g. *photoproduction*
- multilevel trigger system reduce the rate through
  - successive application of more advanced algorithms
- buffering pipelines help to reduce the dead time

Multilevel Trigger System II



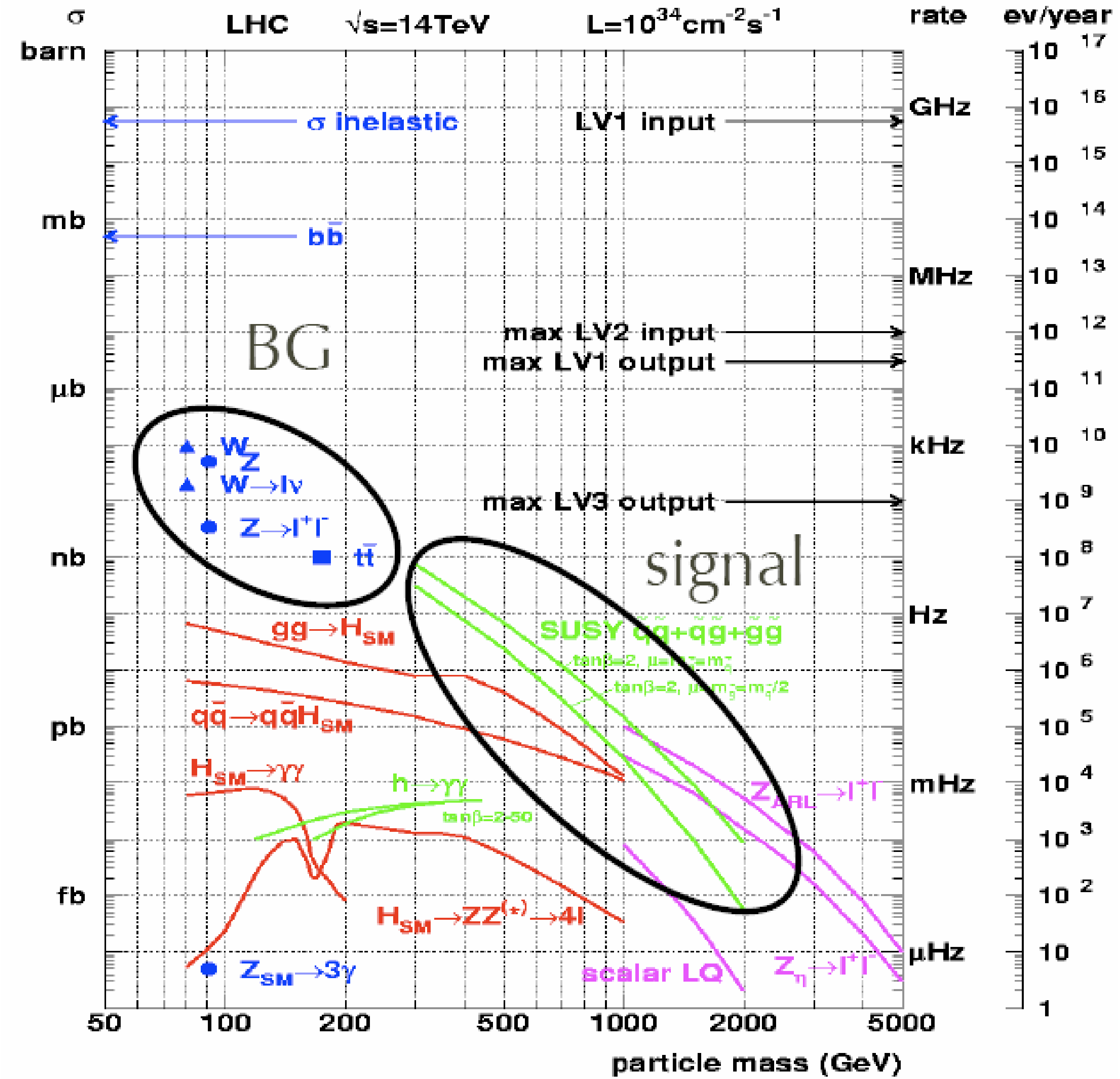


other examples:

HERA-B and ATLAS trigger systems

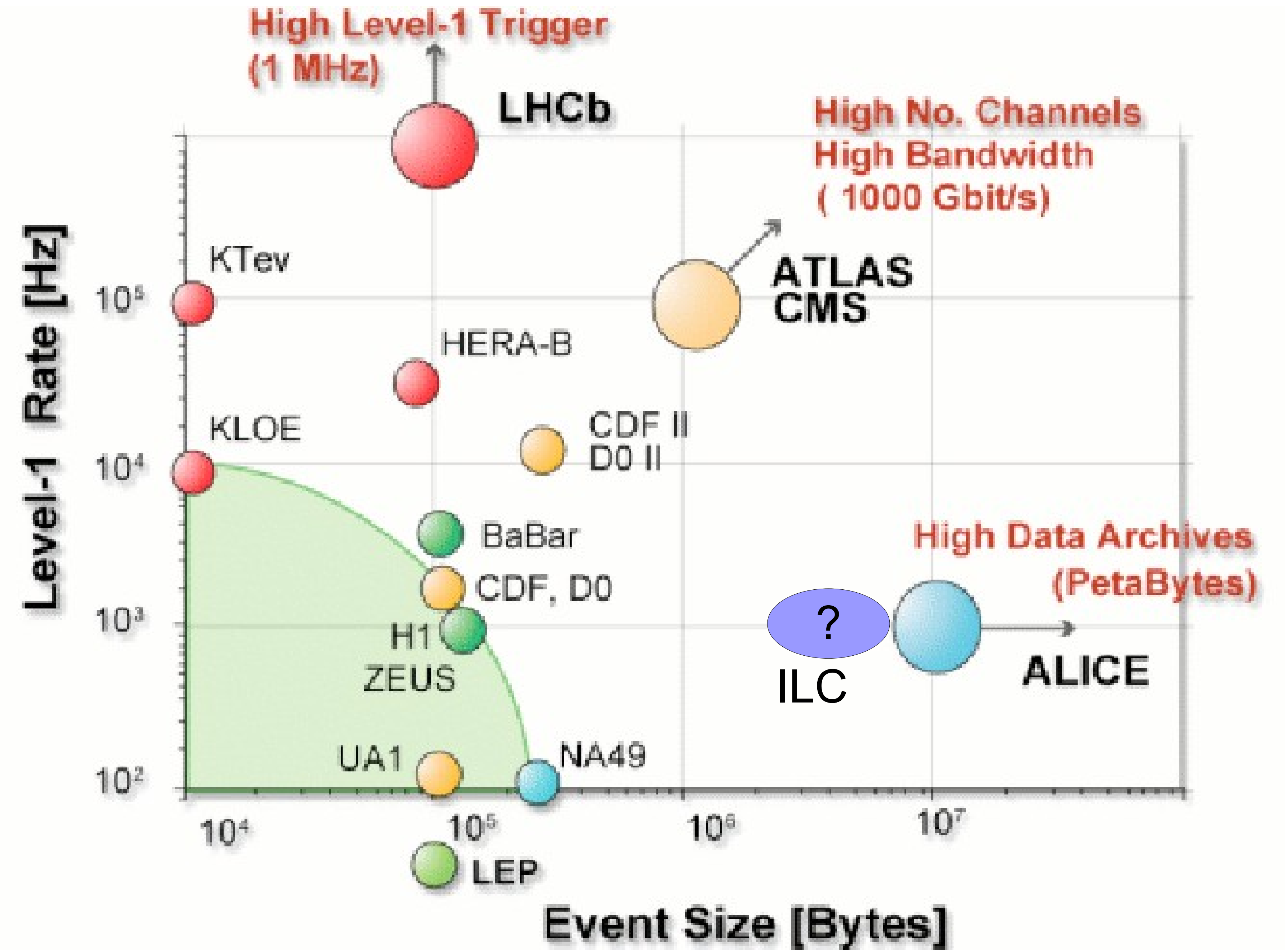
exception: planned **ILC** (Linear Collider) due to comparatively low rates and high extrapolated bandwidth (~2015?) no trigger system foreseen but **continuous read out planned** -> no dead time !

Atlas trigger and physics rates



Frank Gaede, Summer Student Lecture, DESY, August 20, 2008

Trigger rates and event sizes



Monte Carlo Simulation

Why Monte Carlo Simulations ?

- R&D phase: (*planning phase, e.g. ILC*)
  - determine the best geometry of the detector
  - study the (needed) performance of subdetectors
  - compare different designs (competition)
  - evaluate feasibility (bg-rates/radiation)

- Data taking (*running experiments*)
  - study efficiency of the detector for all the different physics channels (cross sections)
  - determine the fundamental parameters of underlying physics

Monte Carlo Simulation Programs

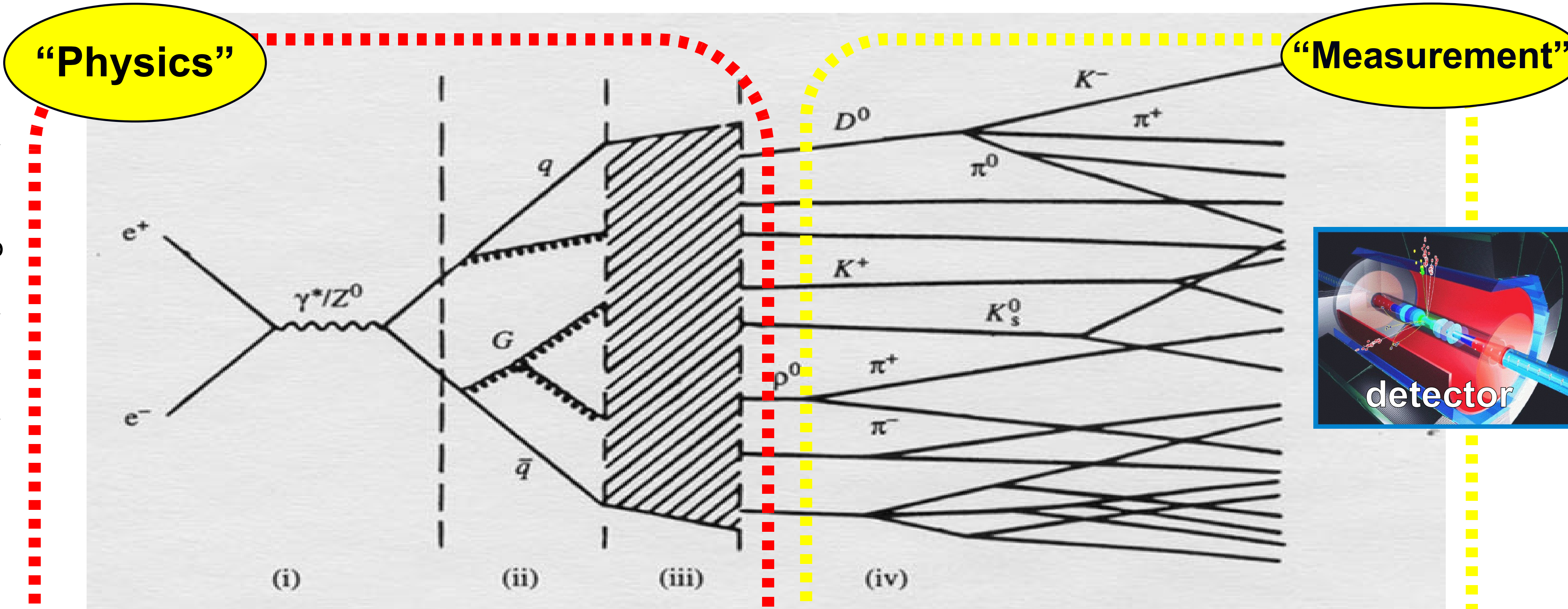
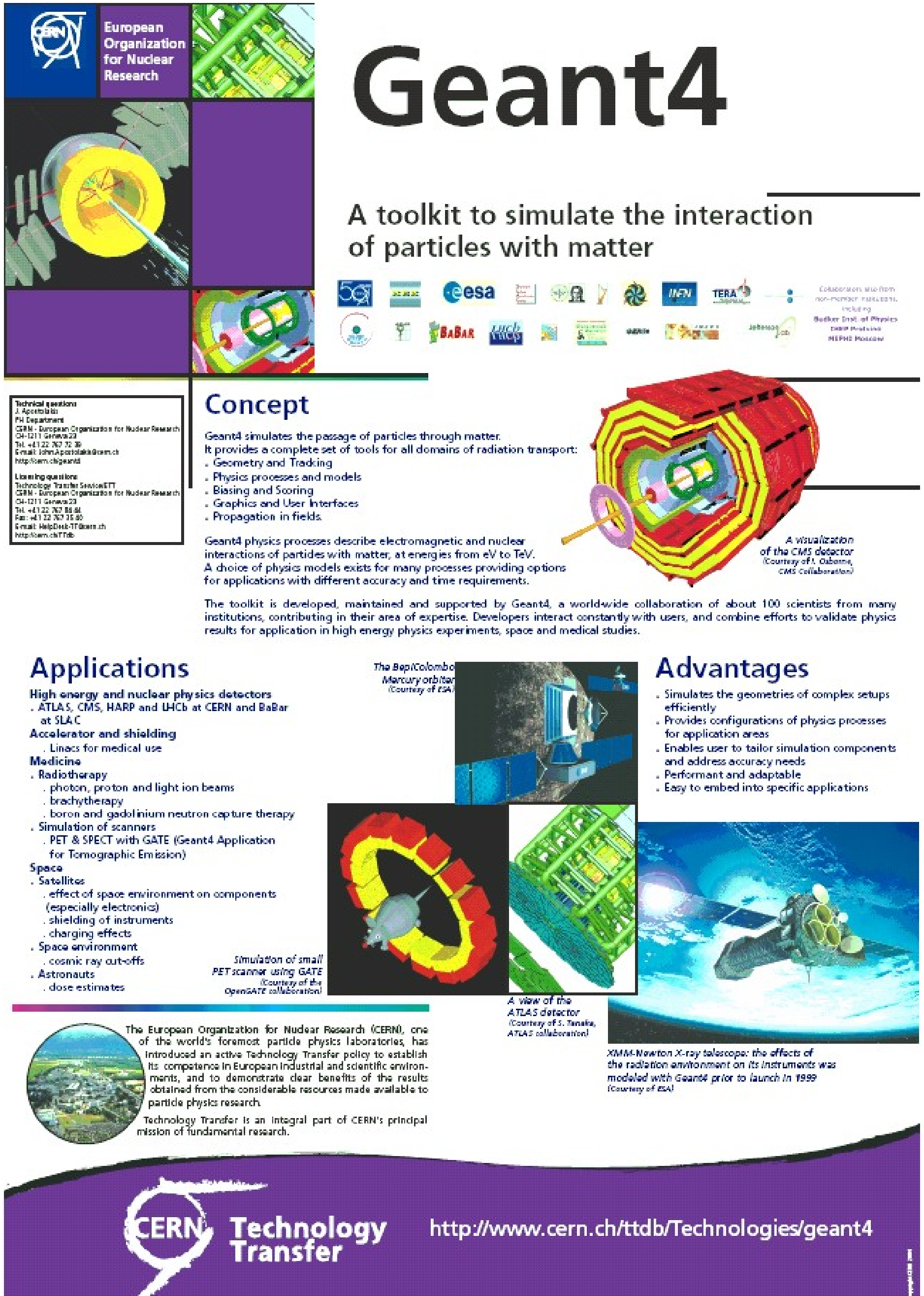


Fig. 25.15. Schematic illustration of $e^+e^- \rightarrow$ hadrons.

Generator:
generates 4-vectors of all
particles in the event
e.g. PYTHIA, Herwig

Simulation:
detector response to longlived
particles -> <u>HITS</u>
e.g. Geant4

Simulating the detector response

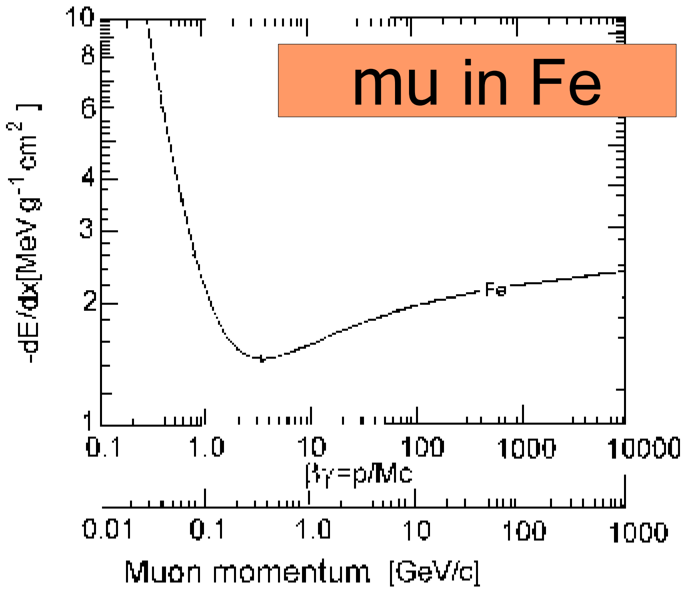


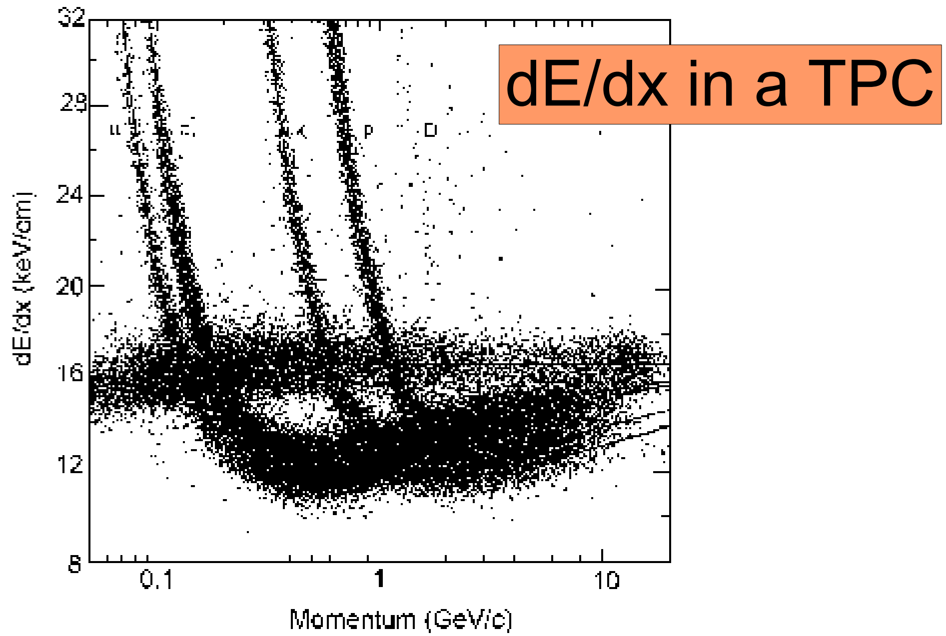
- example: **geant4** - a C++ toolkit that simulates passage of particles through matter using "known physics":
  - particle decay (lifetime/branching ratios)
  - photoelectric effect
  - Compton scattering
  - pair creation (EM-cascade)
  - energy loss due to ionization (exaltation), multiple scattering
  - Cherenkov radiation
  - Positron - Electron Annihilation
  - Bremsstrahlung
  - ~hadronic interactions
    - cross section tables
    - parameterizations
    - ... many more ...

passage of particles through matter

- simulating the detector response only meaningful if the the underlying physics is known well enough
- in general true for all electromagnetic interactions
  - ionization in tracking detectors
  - electromagnetic showers in calorimeters
    - EM-cascade due to repeating Bremsstrahlung/pair-creation
    - QED has a non divergent perturbation series
- in general not so true for hadronic interactions
  - QCD has divergent perturbation series in
    - low energy (soft) hadron interactions
    - quark-gluon coupling in hadronization
- -> need phenomenological parameterizations and measured cross sections for hadron calorimeters

Ionization energy loss

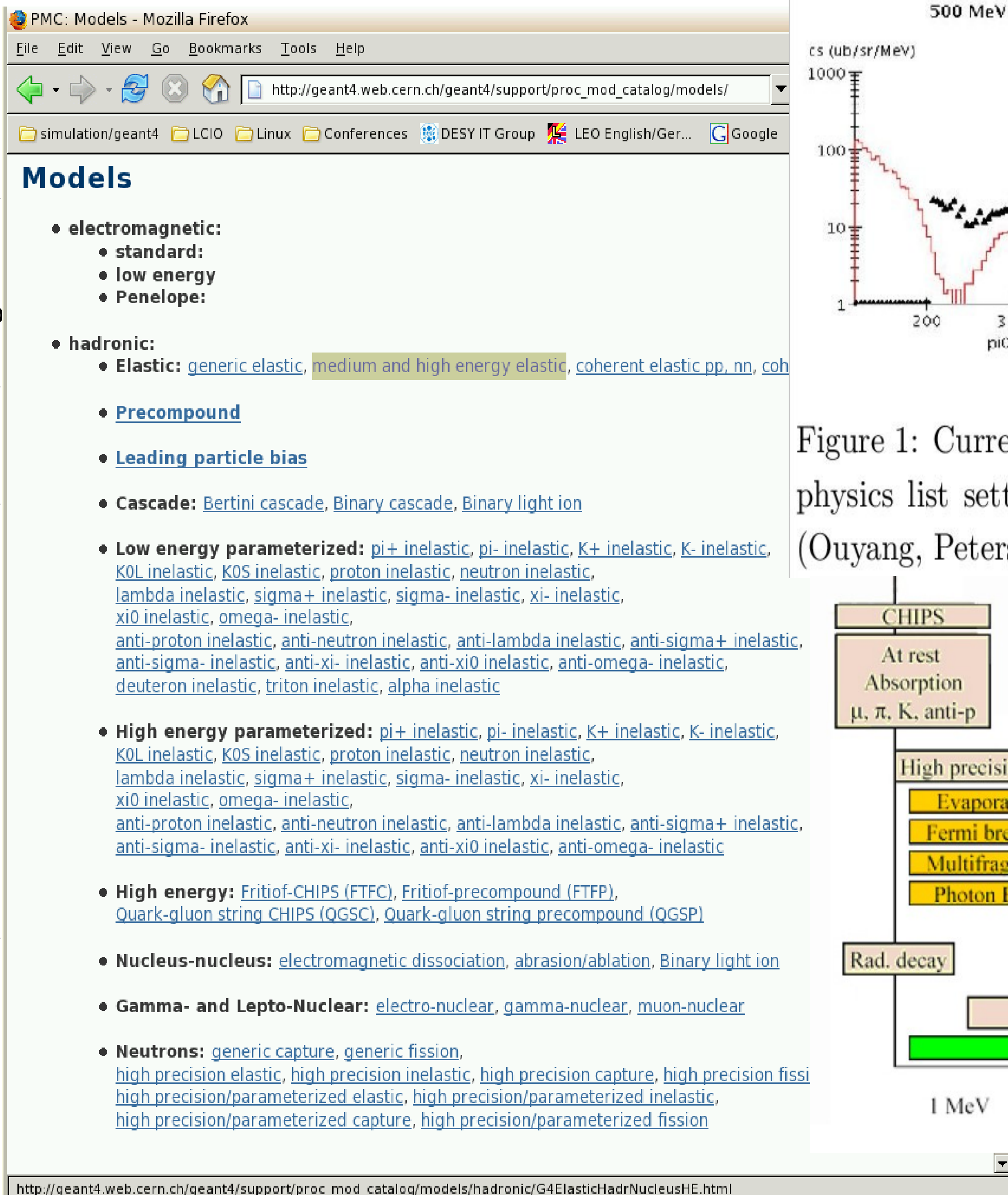




$$-\frac{\mathrm{d}E}{\mathrm{d}x} = \kappa z^2 \cdot \frac{Z}{A} \cdot \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2 m_e c^2 \gamma^2 \beta^2}{I^2} E_{\mathrm{kin}}^{\mathrm{max}} - \beta^2 - \frac{\delta}{2} \right]$$

Bethe-Bloch Formula

hadronic shower parameterization


PMC: Models - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://geant4.web.cern.ch/geant4/support/proc_mod_catalog/models/

simulation/geant4 LCIO Linux Conferences DESY IT Group LEO English/Ger... Google

Models

- **electromagnetic:**
  - **standard:**
  - **low energy**
  - **Penelope:**
- **hadronic:**
  - **Elastic:** generic elastic, medium and high energy elastic, coherent elastic pp, nn, coh
  - **Precompound**
  - **Leading particle bias**
  - **Cascade:** Bertini cascade, Binary cascade, Binary light ion
  - **Low energy parameterized:** pi+ inelastic, pi- inelastic, K+ inelastic, K- inelastic, K0L inelastic, K0S inelastic, proton inelastic, neutron inelastic, lambda inelastic, sigma+ inelastic, sigma- inelastic, xi- inelastic, xi0 inelastic, omega- inelastic, anti-proton inelastic, anti-neutron inelastic, anti-lambda inelastic, anti-sigma+ inelastic, anti-sigma- inelastic, anti-xi- inelastic, anti-xi0 inelastic, anti-omega- inelastic, deuteron inelastic, triton inelastic, alpha inelastic
  - **High energy parameterized:** pi+ inelastic, pi- inelastic, K+ inelastic, K- inelastic, K0L inelastic, K0S inelastic, proton inelastic, neutron inelastic, lambda inelastic, sigma+ inelastic, sigma- inelastic, xi- inelastic, xi0 inelastic, omega- inelastic, anti-proton inelastic, anti-neutron inelastic, anti-lambda inelastic, anti-sigma+ inelastic, anti-sigma- inelastic, anti-xi- inelastic, anti-xi0 inelastic, anti-omega- inelastic
  - **High energy:** Fritiof-CHIPS (FTFC), Fritiof-precompound (FTFP), Quark-gluon string CHIPS (QGSC), Quark-gluon string precompound (QGSP)
  - **Nucleus-nucleus:** electromagnetic dissociation, abrasion/ablation, Binary light ion
  - **Gamma- and Lepto-Nuclear:** electro-nuclear, gamma-nuclear, muon-nuclear
  - **Neutrons:** generic capture, generic fission, high precision elastic, high precision inelastic, high precision capture, high precision fissi high precision/parameterized elastic, high precision/parameterized inelastic, high precision/parameterized capture, high precision/parameterized fission

http://geant4.web.cern.ch/geant4/support/proc_mod_catalog/models/hadronic/G4ElasticHadrNucleusHE.html

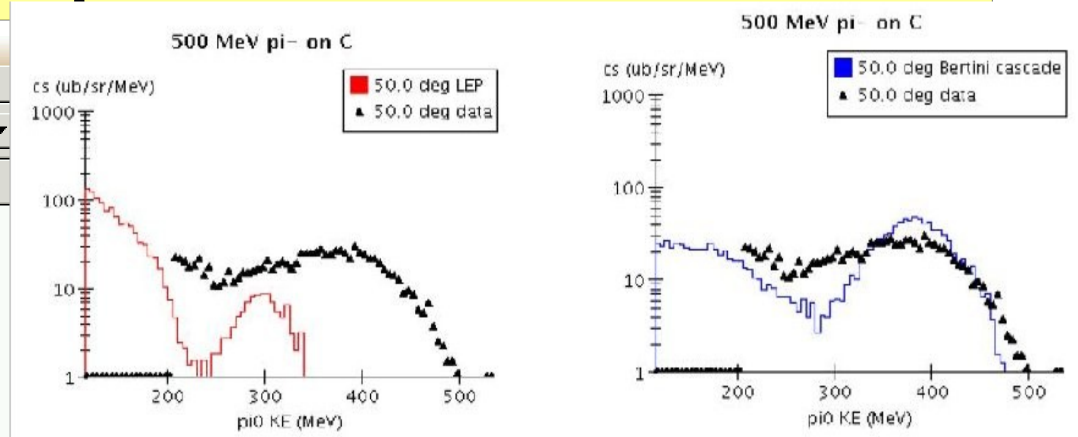
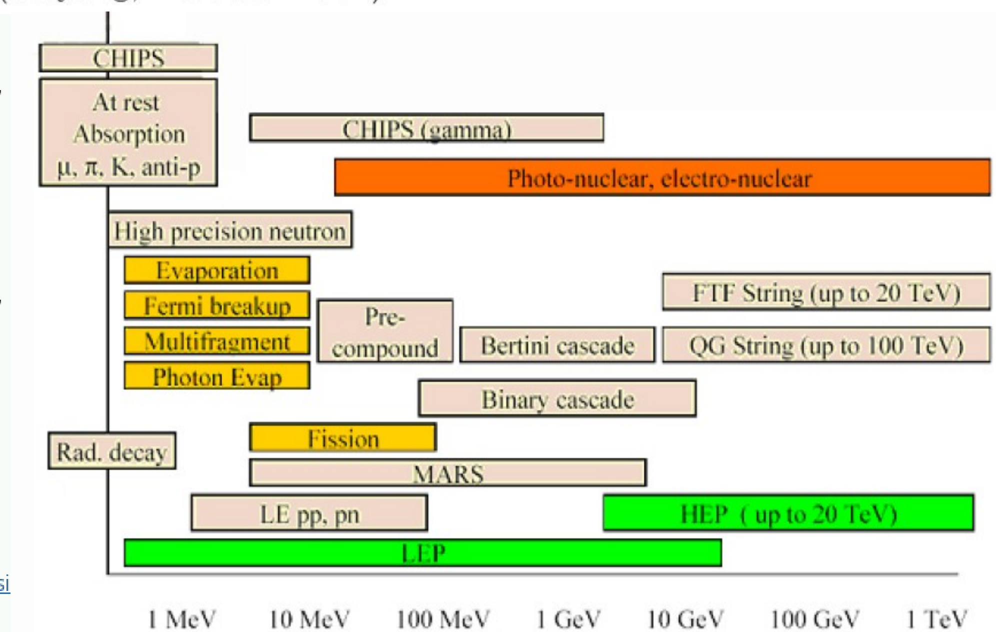


Figure 1: Current GEANT4 LEP physics list setting against data (Ouyang, Peterson 1992)

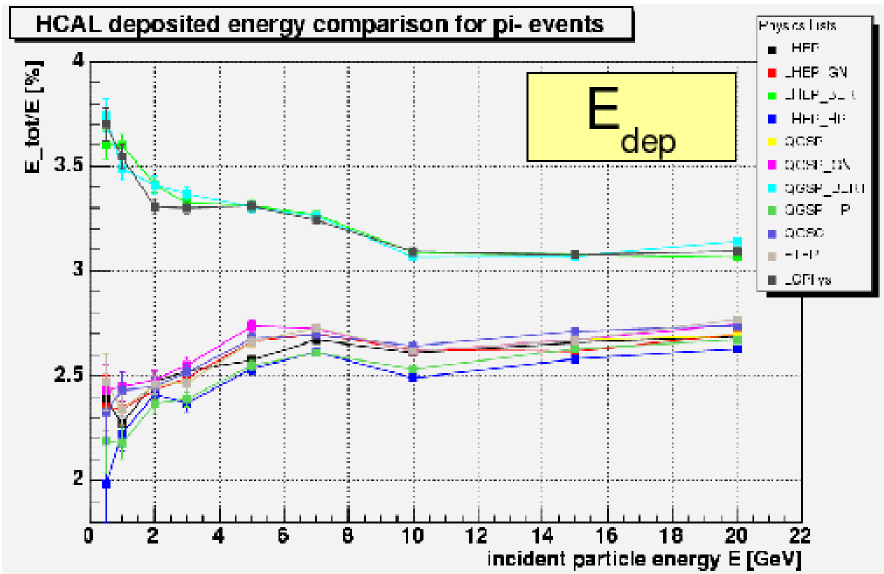
Figure 2: Bertini cascade model



hadronic showers in ILC-HCal prototype

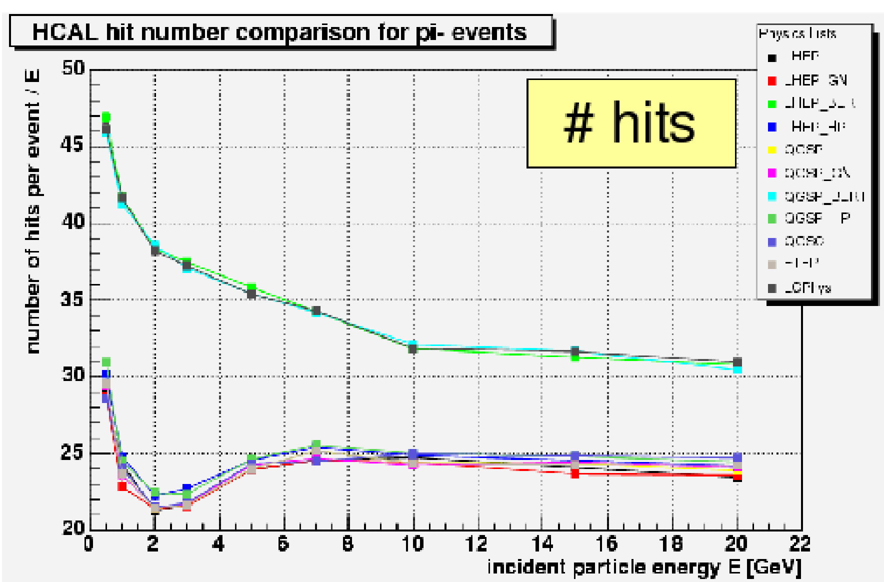
GEANT4.6.1

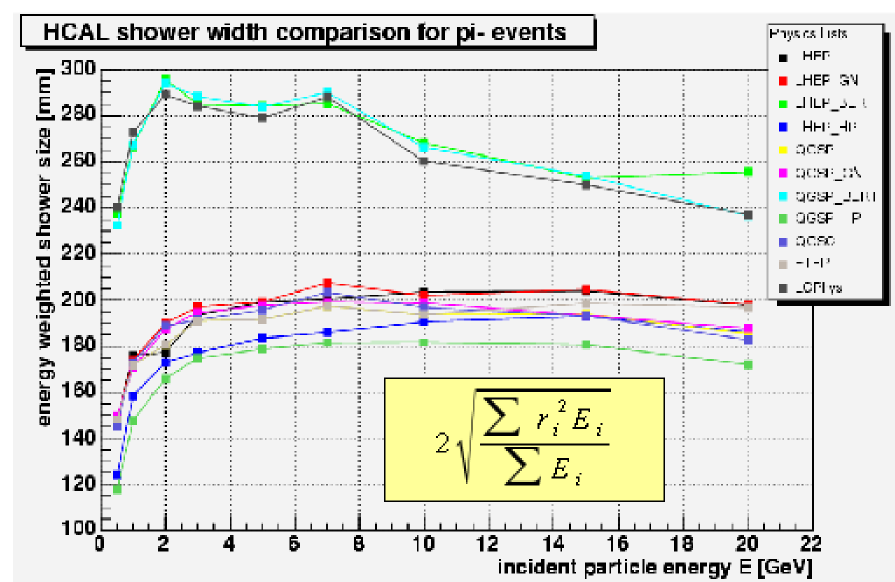
•P.Melchior, summer student programme 2004
• need verification with testbeam -> still ongoing !



=> only two classes of physics lists in given energy domain:

- LEP like parameterization
- Bertini cascade

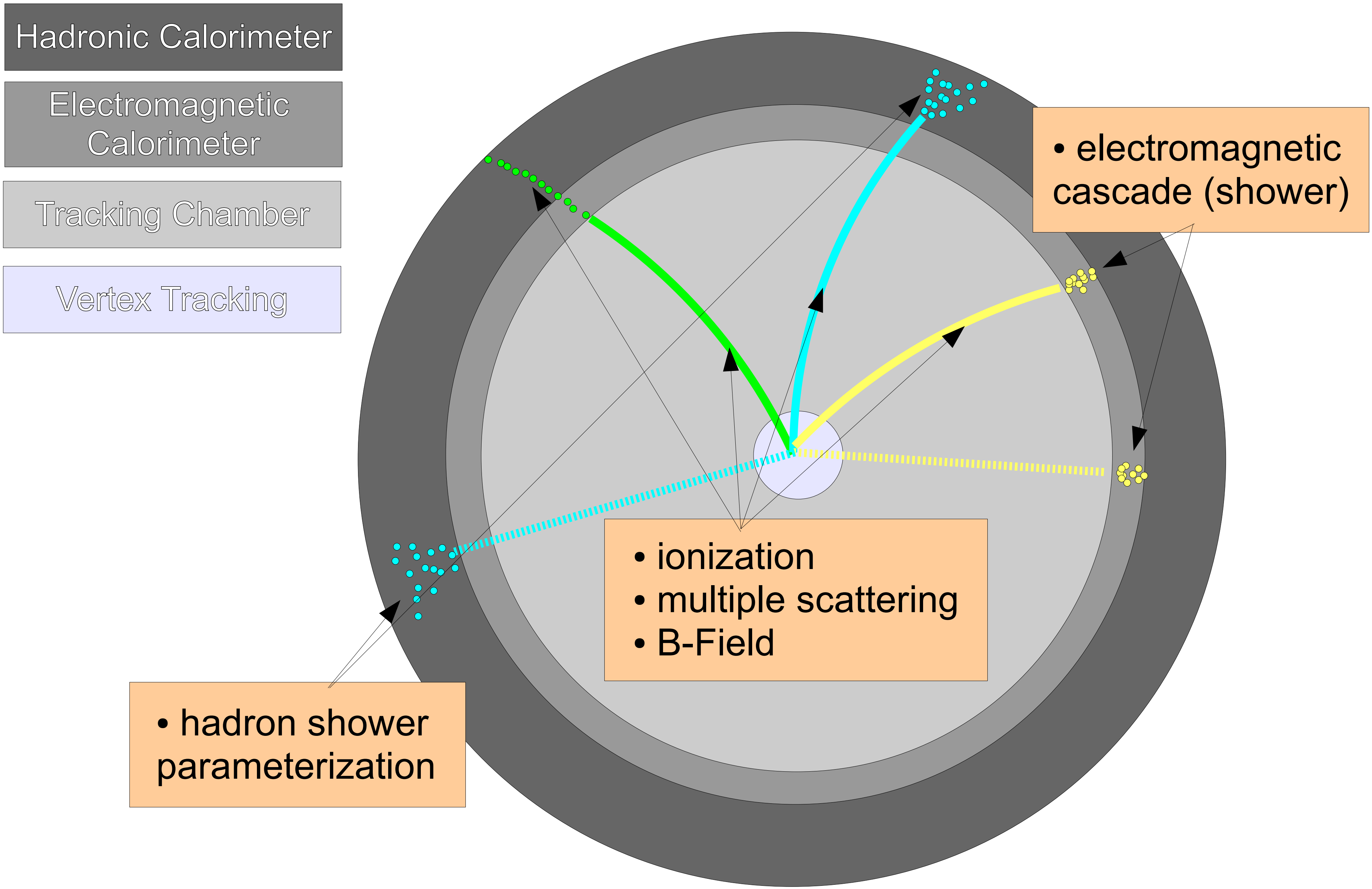




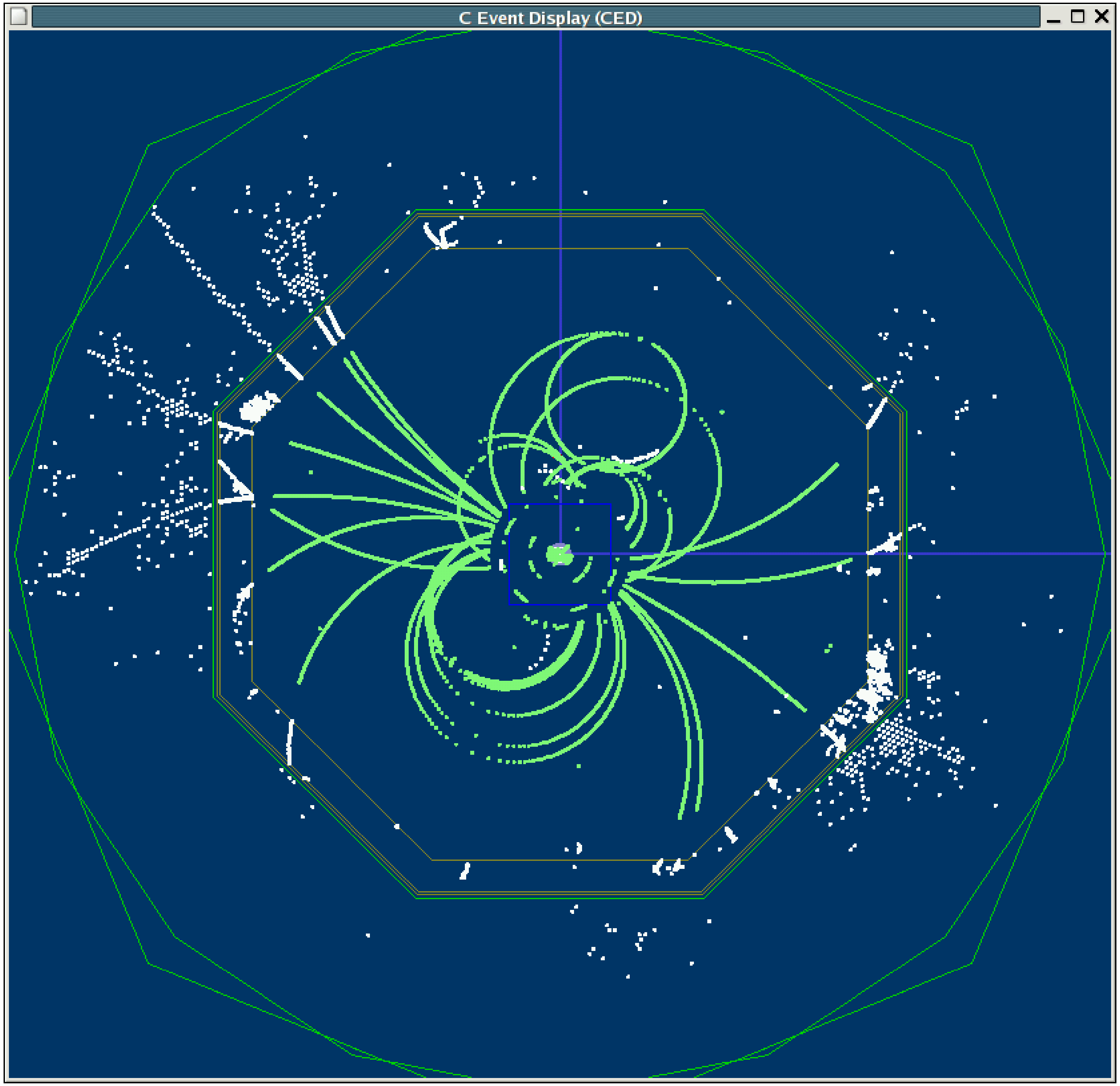
physics processes in geant4

- each particle has its own list of applicable processes
- at each step, all processes listed are invoked to get random physical interaction lengths (Monte Carlo Method!)
- the process with the shortest interaction length limits the step
- each process can have any of the following actions:
  - **AtRest** (e.g. muon decay at rest)
  - **AlongStep** - continuous process (e.g. ionization)
  - **PostStep** - discrete process (e.g. decay in flight)
- every action that is applied to a particle is defined as a process:
  - transportation (E,B fields)
  - decay
  - interactions with Material (ionization, delta-electrons,.....)
  - step length cut off

dominating processes in simulation



simulation output - hits



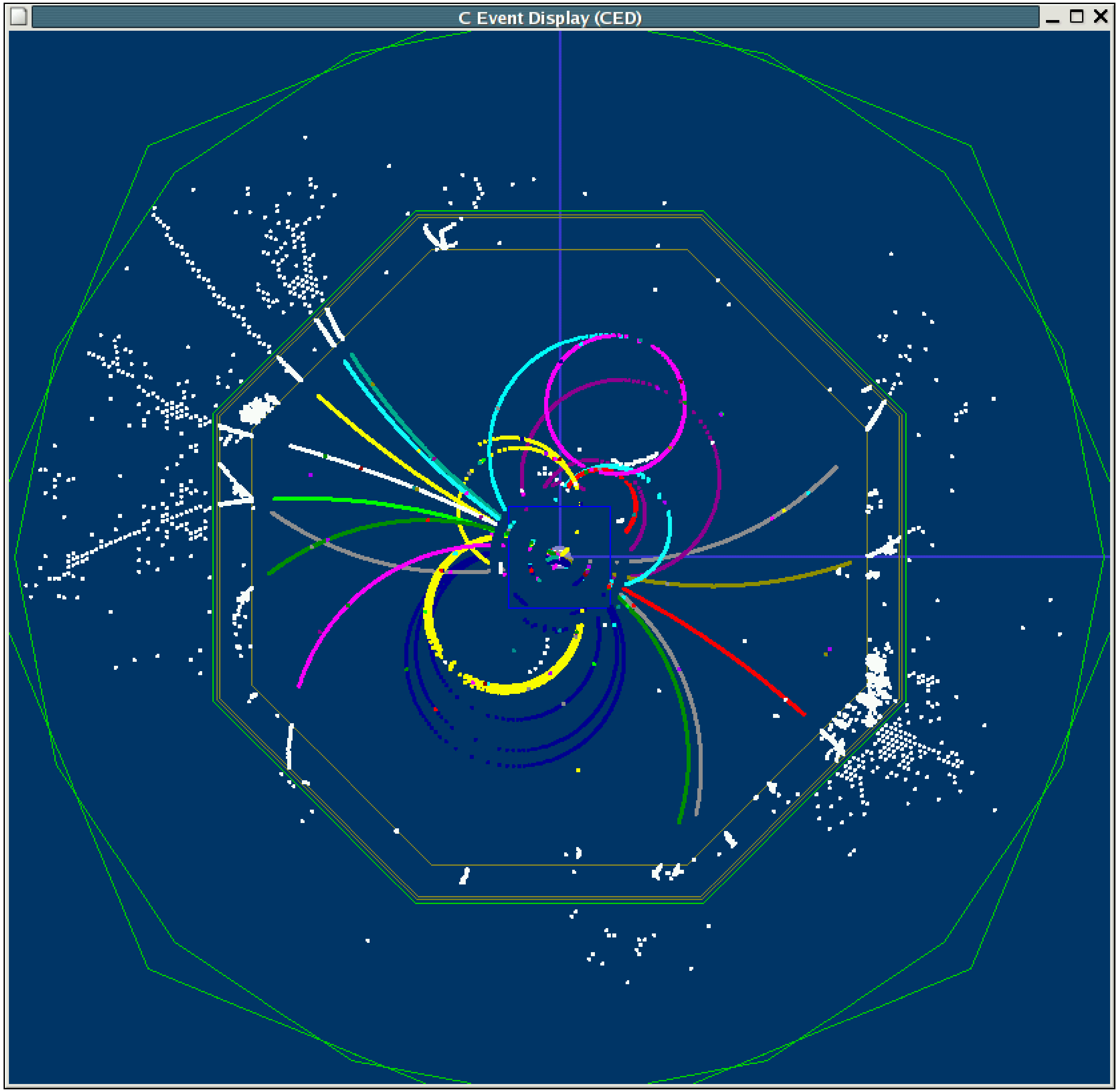
- **simulation** output:
- *calorimeter hits*
  - cell position
  - amplitude (energy)
- *tracker hits*
  - amplitude
  - dE/dx

- **digitization**
  - smear hits
  - apply noise
    - electronics
    - physics

Reconstruction

- now we have simulated the detector response (hits) to the generated event
- ideally this is *indistinguishable from real data*
  - (not true in practice as of course MC-Truth is conserved)
- next step: Reconstruction – combining hits to reconstructed particles in order to perform the actual physics analysis

reconstruction - tracking



- **tracking**
  (pattern recognition):
  - *track finding*
    - combine hits that most likely belong to one particle
  - *track fitting*
    - apply fit to all hits taking B-field into account
    - 'Helix approximation'
- Kalman Filter typically perform both steps in one, taking fit to previous points as estimate to next point

track parameters - fitting

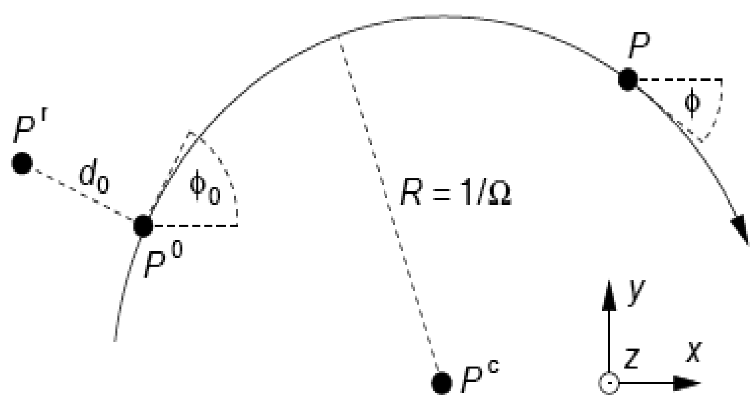


Figure 1: The projection of a helix segment in the xy plane is a part of an arc with centre P^c and radius R. The direction of the particle is shown with the arrow at the arc. All track parameters are given relative to the reference point P^r.

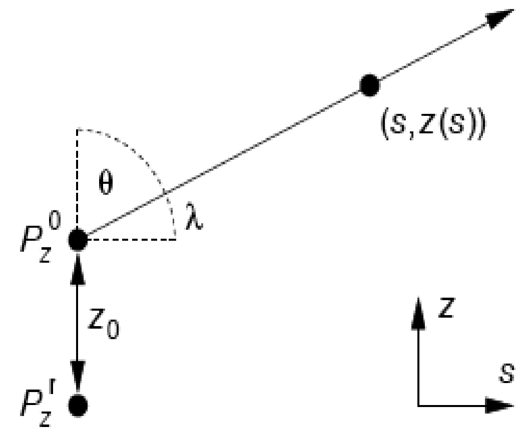
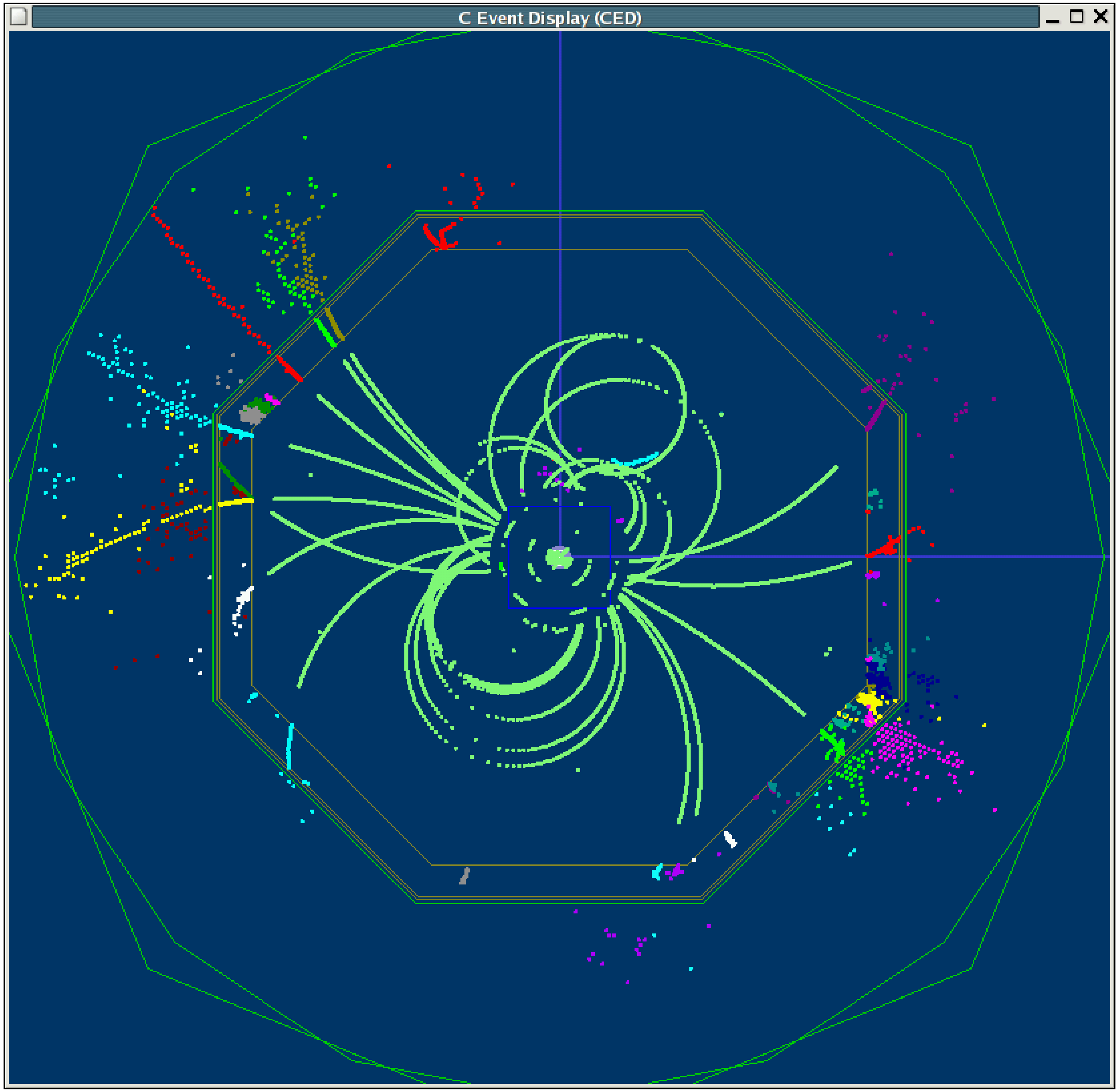


Figure 2: The projection of a helix in the sz plane is a straight line (see Eq. 10). The variable s at a point P is the arc length in the xy plane from P^0 to P. This also implies that $s = 0$, if $z = z_0$.

- a charged particle in a homogenous field describes a helix
- (except for energy loss and multiple scattering)
- a helix is described by five parameters, e.g
- **d0, phi0, omega, tan lambda, z0**
- after identifying the hits from one particle fitting the
- above parameters determines the 3-momentum (and charge):

$$p_t = a\left|\frac{B}{\Omega}\right|, p_x = p_t \sin\phi_0, p_y = p_t \cos\phi_0, p_z = p_t \tan\lambda$$

reconstruction - clustering



- **clustering**
  - combine hits that most likely belong to a particle shower
  - compute energy of shower
- typically based on some metric that links nearby hits *"Nearest-Neighbor"*
- could additionally use
  - tracks as seeds
  - hit energy amplitude

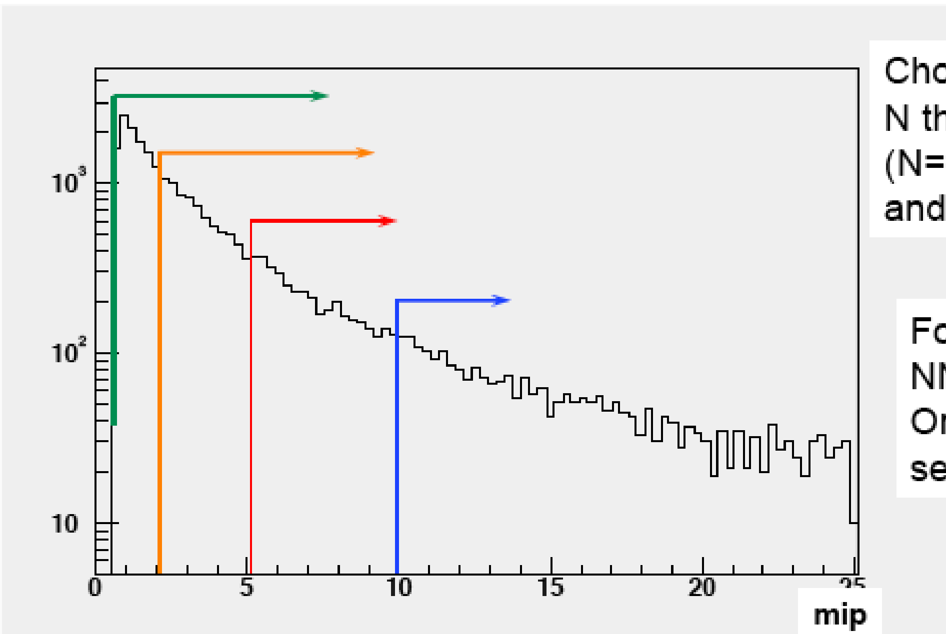
example NNClustering

```
template <class In, class Out, class Pred >
void cluster( In first, In last, Out result, Pred* pred ) {

  typedef typename In::value_type GenericHitPtr ;
  typedef typename Pred::hit_type HitType ;

  typedef std::vector< GenericCluster<HitType >* >  ClusterList ;

  ClusterList tmp ;
  tmp.reserve( 256 ) ;

  while( first != last ) {

    for( In other = first+1 ; other != last  ; other ++ ) {

      if( pred->mergeHits( (*first) , (*other) ) ) {

        if( (*first)->second == 0 && (*other)->second == 0 ) {  // no cluster exists

          GenericCluster<HitType >* cl = new GenericCluster<HitType >( (*first) ) ;

          cl->addHit( (*other) ) ;

          tmp.push_back( cl ) ;

        }
        else if( (*first)->second != 0 && (*other)->second != 0 ) { // two clusters

          (*first)->second->mergeClusters( (*other)->second ) ;

        } else {  // one cluster exists

          if( (*first)->second != 0 ) {
            (*first)->second->addHit( (*other)  ) ;
          } else {
            (*other)->second->addHit( (*first)  ) ;
          }
        }
      } // dCut
    }
    ++first ;
  }
  // remove empty clusters
```

- simplest algorithm: *nearest neighbor clustering :*
  - loop over all hit pairs
  - merge hits into one cluster if *d(h1, h2) < cut*
  - d() could be 3D-distance – typically more complicated

- in real life the NNClustering does not provide the necessary accuracy, e.g. in dense jets where showers overlap
- -> more advanced algorithms needed and under development/study, e.g.
  - *tracking like clustering*
  - *genetic algorithms*
  - *unsupervised learning, ...*

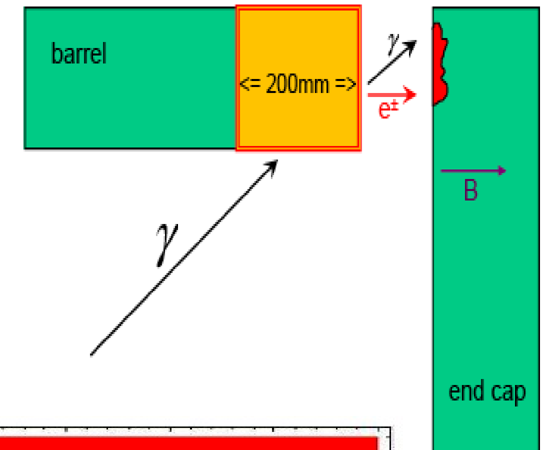
advanced example: photon shower ID

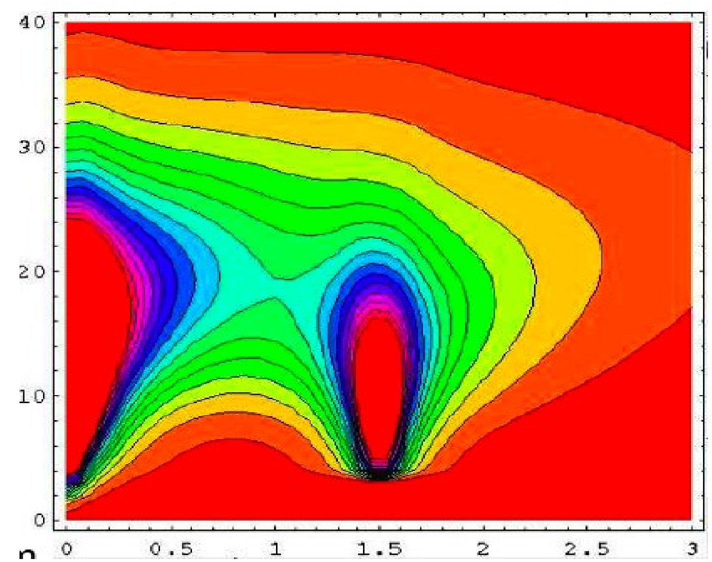


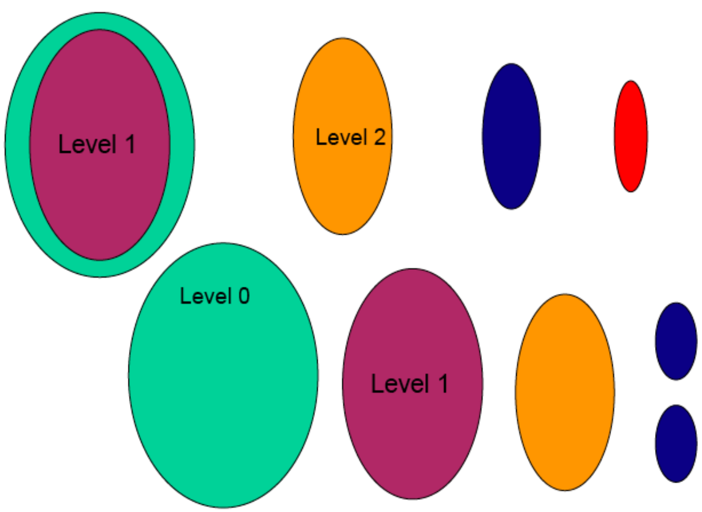
Choose
N threshold levels
(N=10 at the moment)
and get N sets of hits

For each set do a
NN clustering
Only in particular
set!!

P.Krstonosic





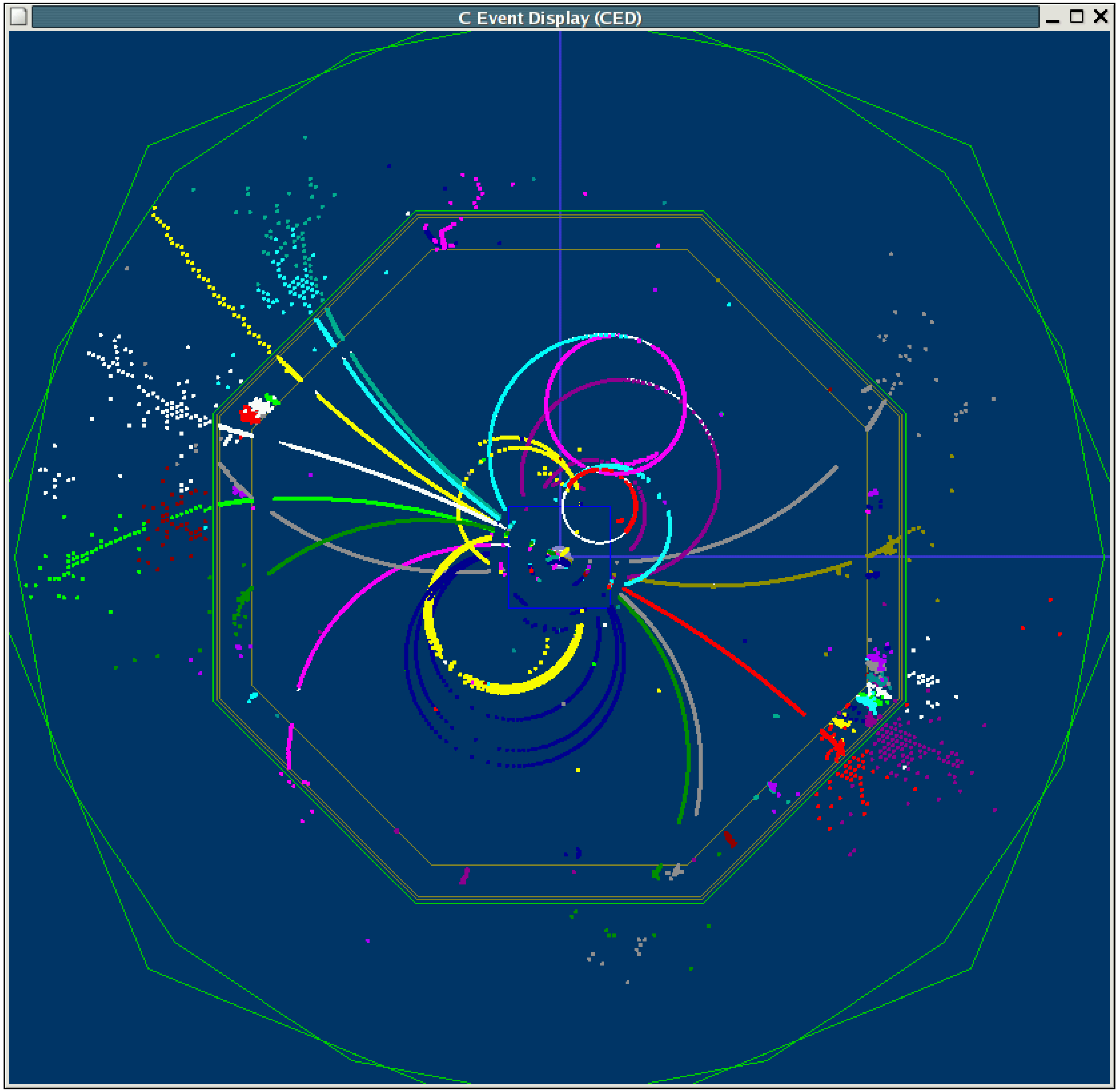


Single photon

?!

- sophisticated photon ID
- based on N clusterings with different thresholds

reconstruction - PFA

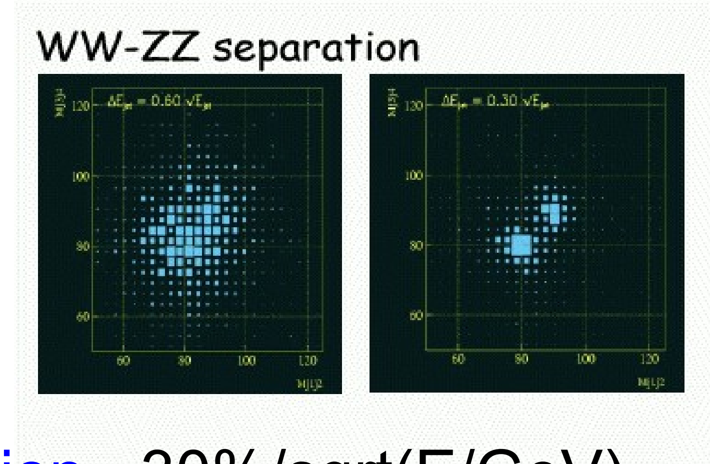


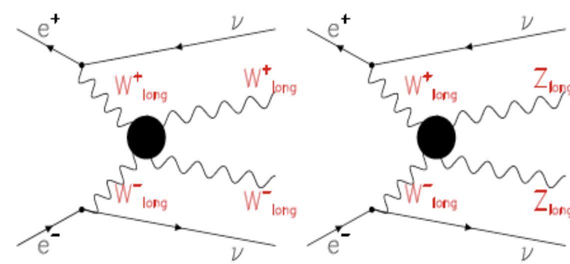
- **track cluster merging** (particle flow)

- extrapolate the tracks into the calorimeter and merge with clusters that are consistent with the momentum/direction and energy of the track
- the unmerged cluster are then the neutral particles
- ideally one would like reconstruct every single particle (PFA)

example: reconstruction @ the ILC

- general ILC detector features:
  - precision tracking
  - precision vertexing
  - high granularity in calorimeters
    - (Ecal ~1cm, Hcal ~1-5cm)
- important: very high jet-mass resolution ~30%/sqrt(E/GeV)





Particle Flow

- reconstruct all single particles
- use **tracker** for **charged particles**
- use **Ecal** for **photons**
- use **Hcal** for **neutral hadrons**

- dominant contribution (E<50 GeV):
  - Hcal resolution
  - confusion term

$$\sigma_{E_{jet}}^{2} = \epsilon_{trk}^{2} \sum_{i} E_{trk,i}^{4} + \epsilon_{ECal}^{2} E_{ECal} + \epsilon_{HCal}^{2} E_{HCal} + \sigma_{confusion}^{2}$$

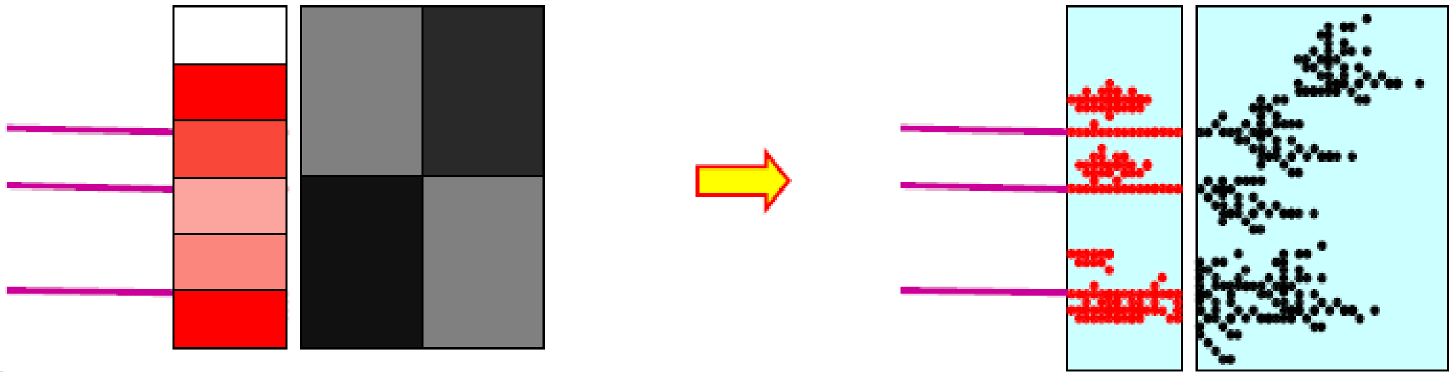
$$\epsilon_{trk} = \delta(1/p) \approx 5 \cdot 10^{-5}, \quad \epsilon_{ECal} = \frac{\delta E}{\sqrt{E}} \approx 0.1, \quad \epsilon_{HCal} \approx 0.5$$

Particle flow calorimetry @ ILC

Hardware:

★ Need to be able to resolve energy deposits from different particles

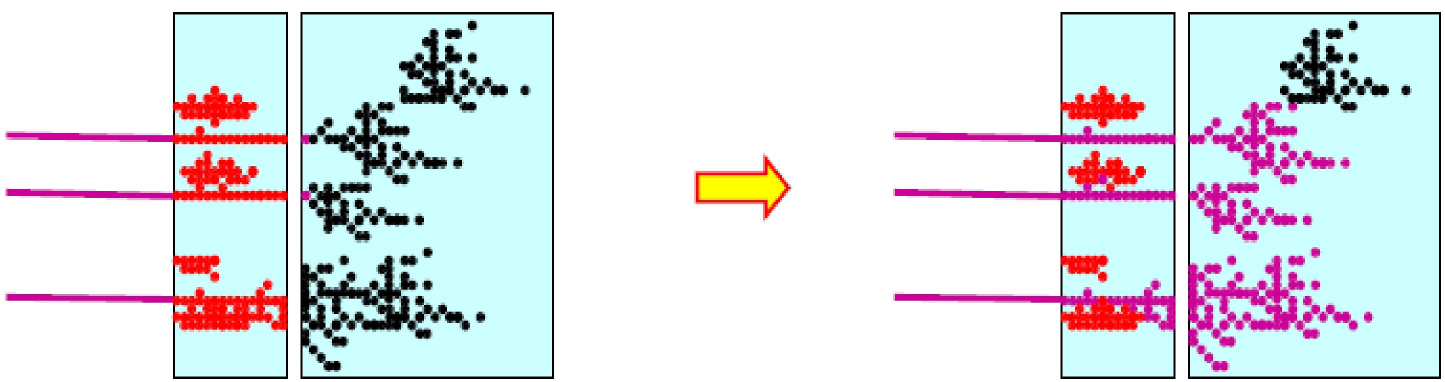
→ **Highly granular detectors (as studied in CALICE)**



Software:

★ Need to be able to identify energy deposits from each individual particle !

→ **Sophisticated reconstruction software**



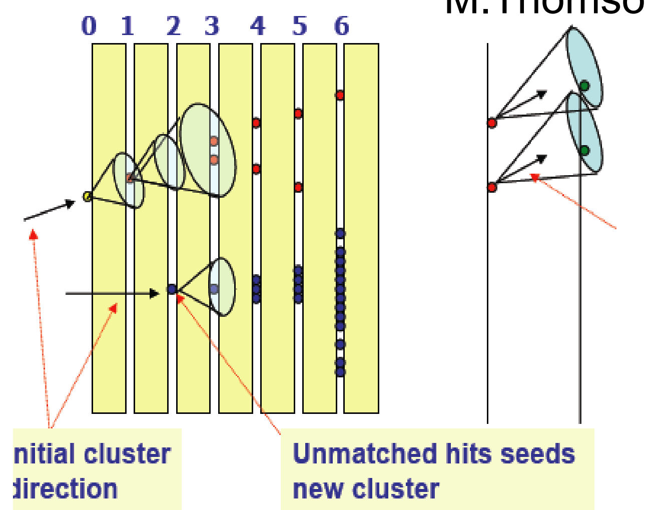
★ **Particle Flow Calorimetry = HARDWARE + SOFTWARE**

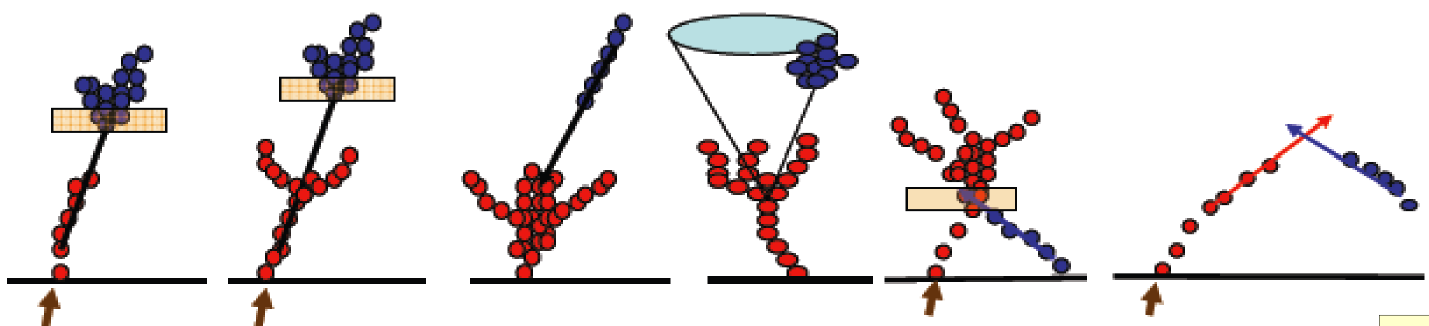
example PandoraPFA clustering

M.Thomson

i. Preparation (MIP hit ID, isolation, tracking)
ii. Loose clustering in ECAL and HCAL
iii. Topological linking of clearly associated clusters
iv. Courser grouping of clusters
v. Iterative reclustering
vi. Photon Recovery (NEW)
vii. Fragment Removal (NEW)
viii. Formation of final Particle Flow Objects (reconstructed particles) – not very sophisticated

Order inter-changable



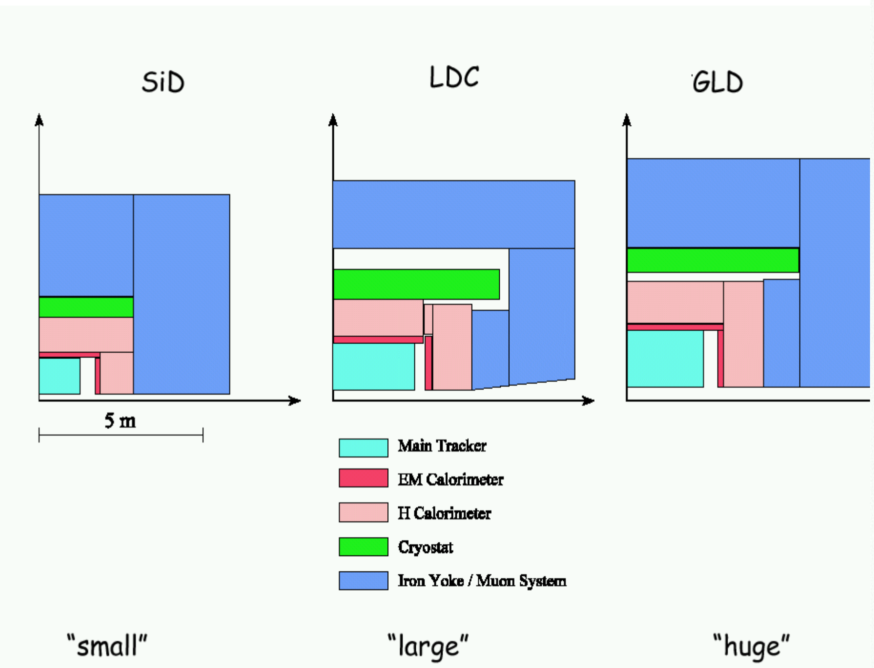


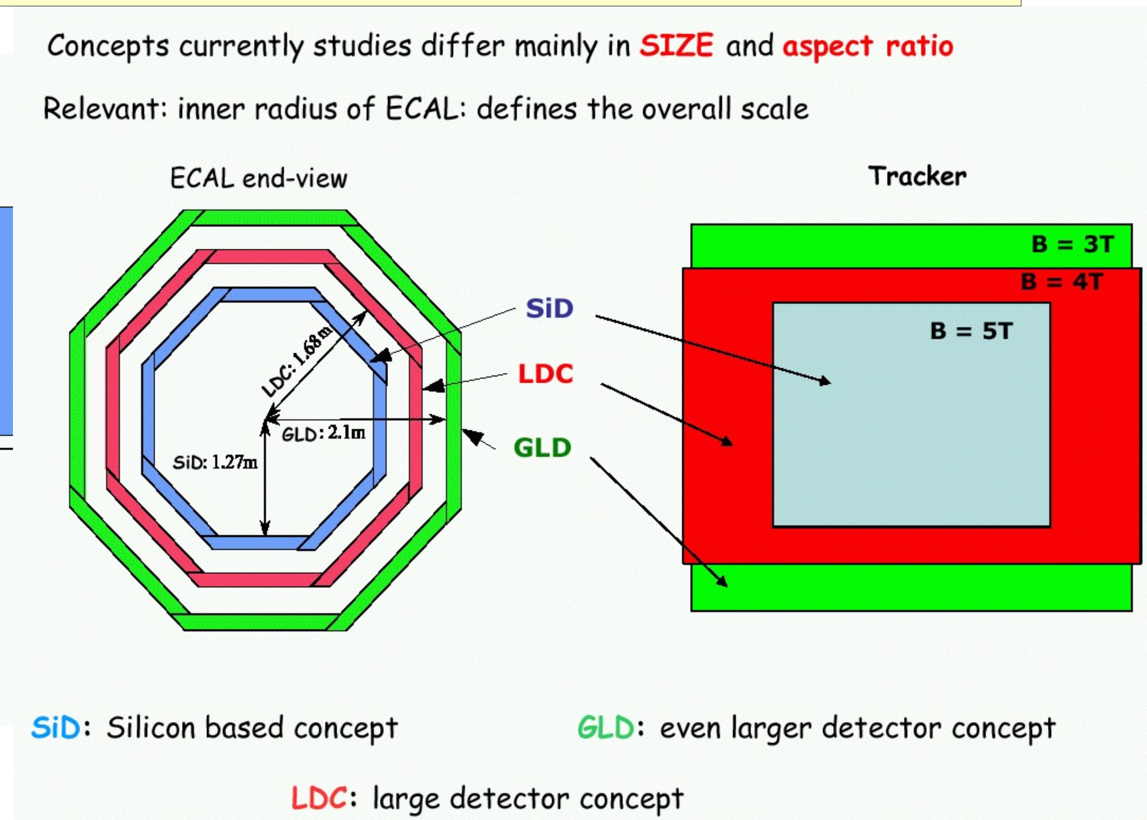
If track momentum and cluster energy inconsistent : RECLUSTER

Pandora is the most sophisticated and best performing PFA to date

example: ILC - Detector Concept Study

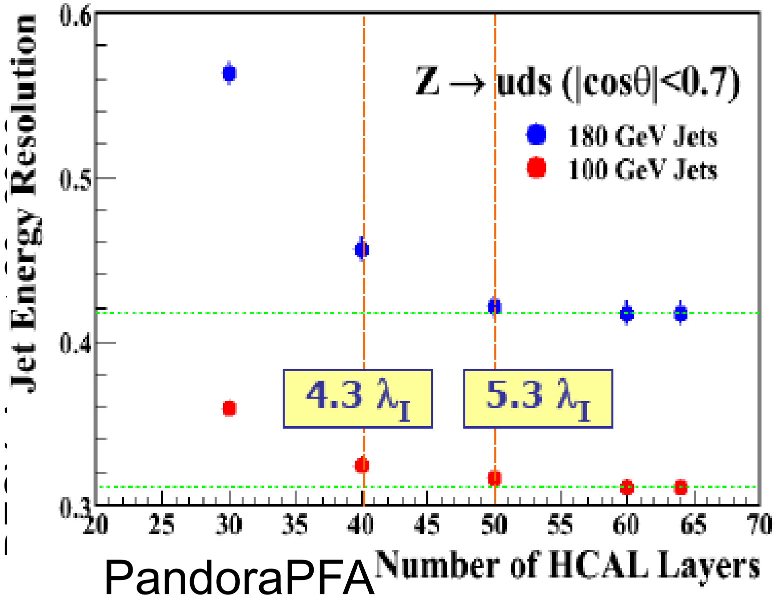
currently three international detector concepts in R&D



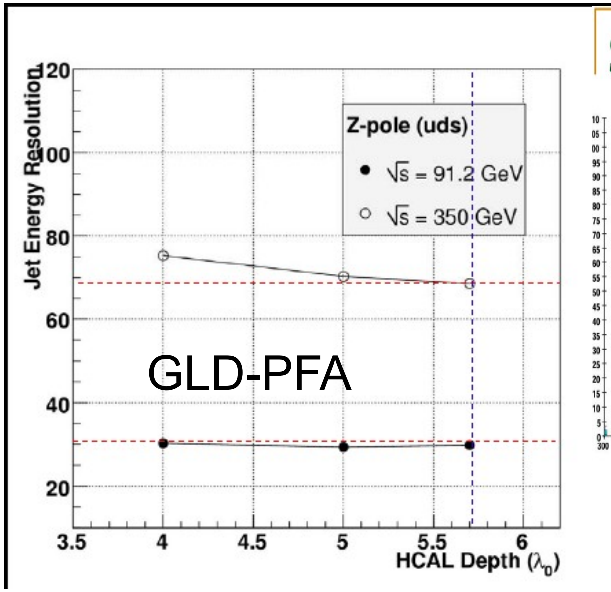


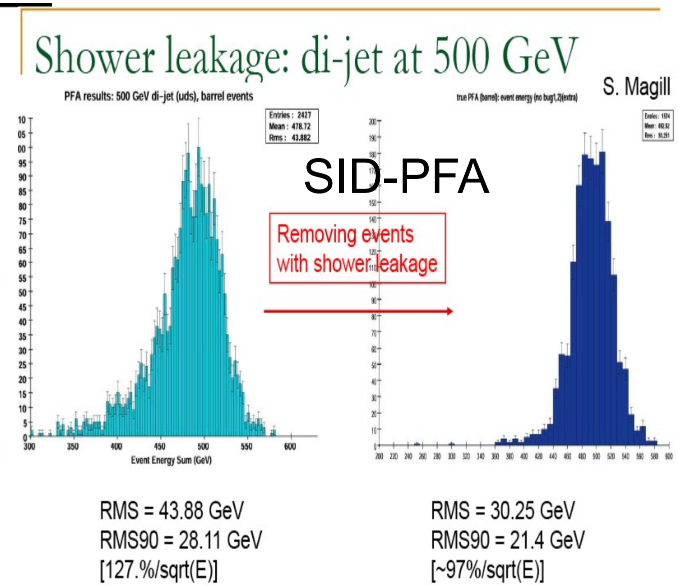
need of sophisticated **Monte Carlo Simulation** programs as well as full **reconstruction** tools to improve and compare the different detector concepts

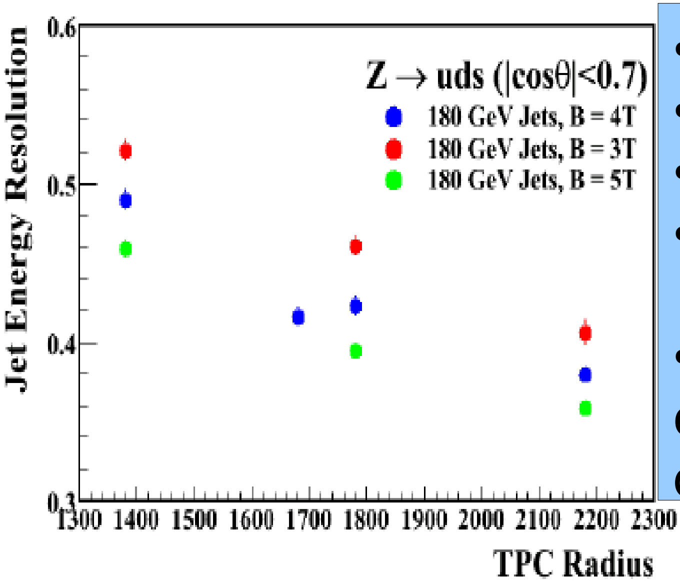
Monte Carlo for detector optimization



PandoraPFA



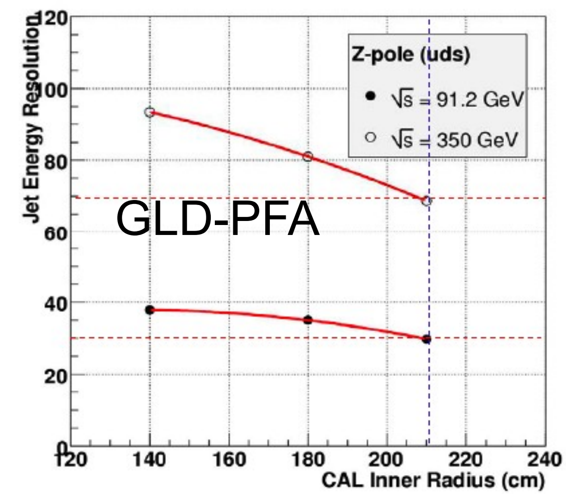




- vary detector parameters,eg:
- Hcal thickness
- Tracking radius
- B-field strength

....

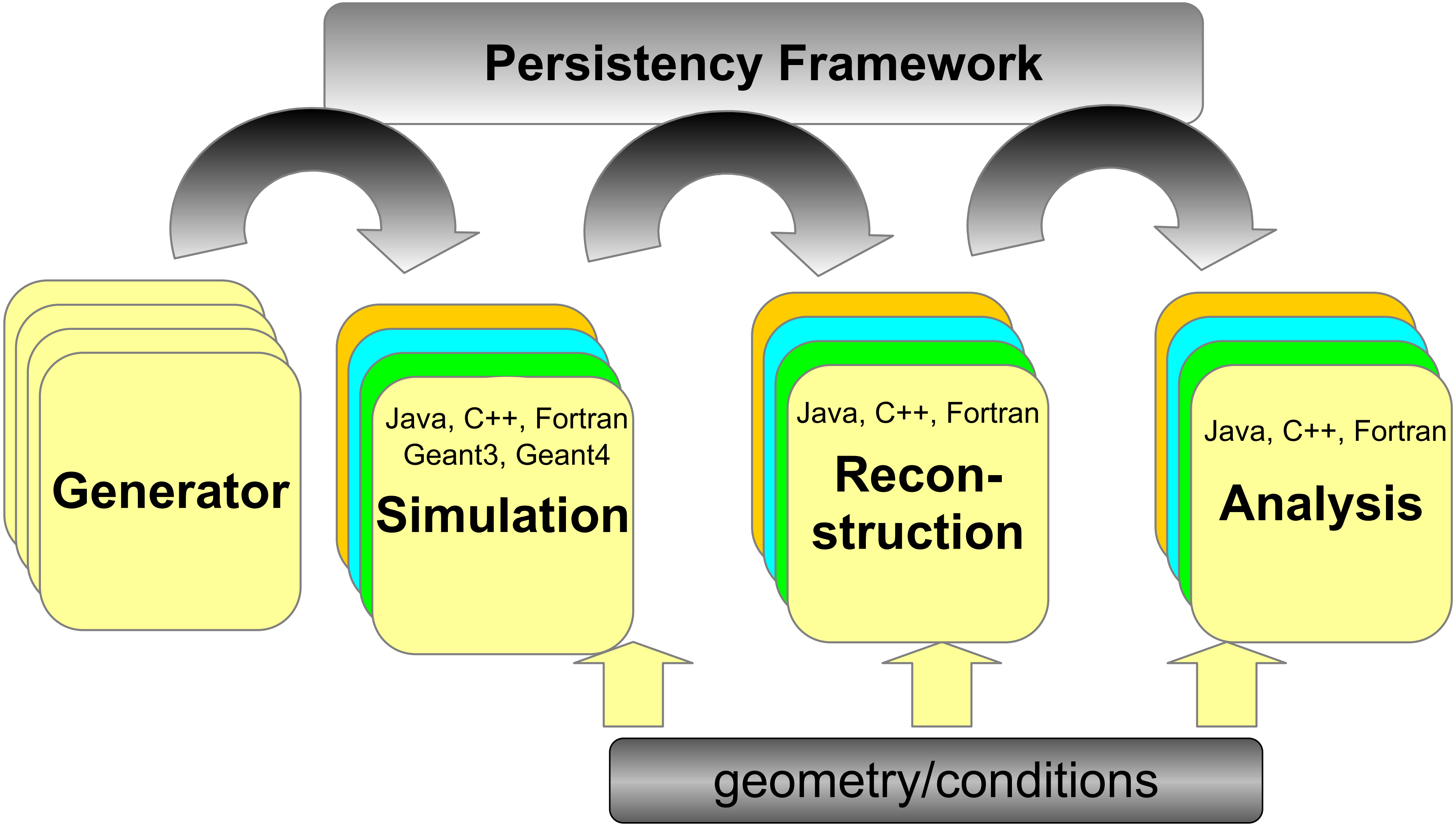
- use Monte Carlo for (cost conscious) optimization of detector



HEP Software Frameworks

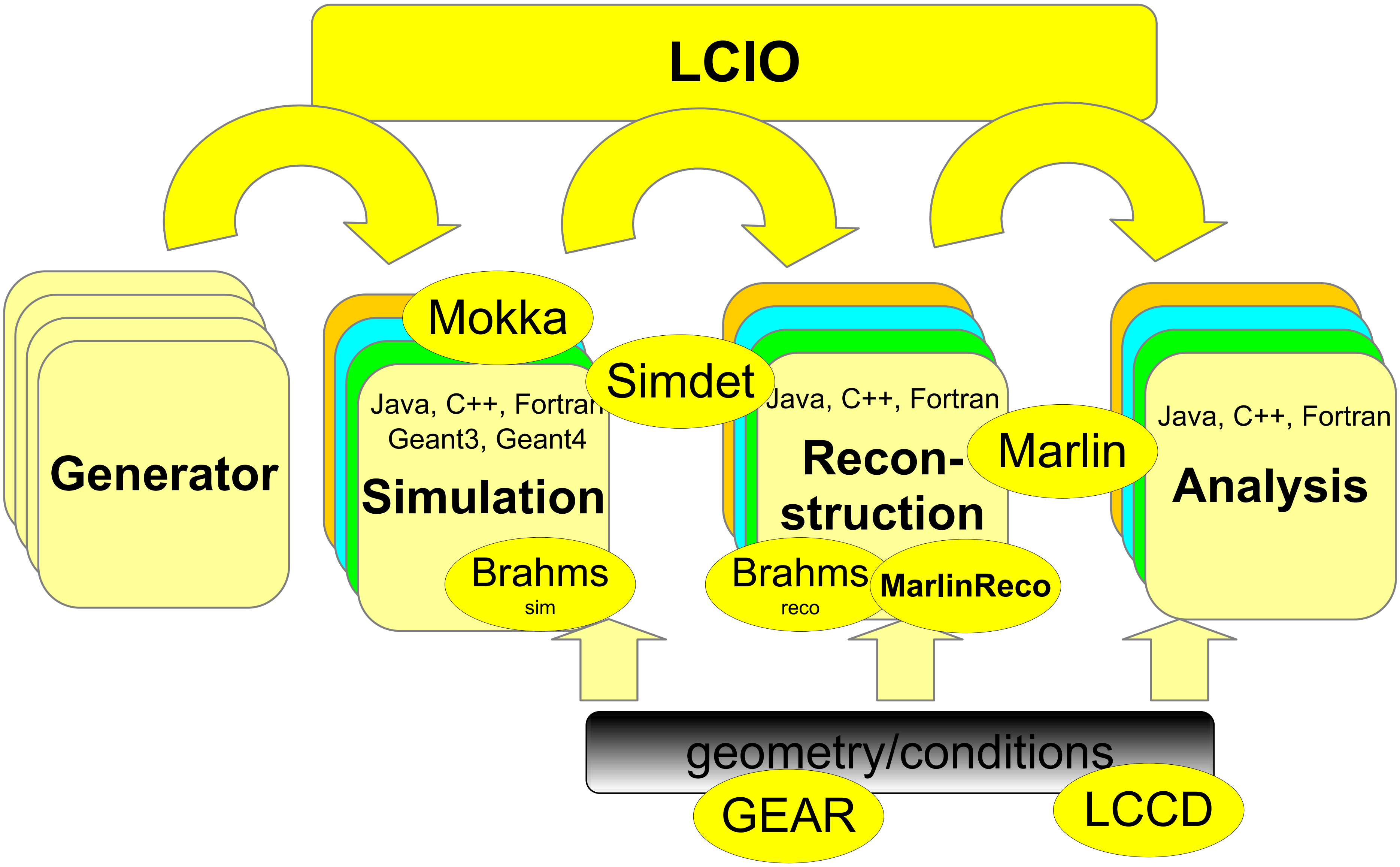
From generated 4-vectors and/or data
to published histograms

ILC Monte Carlo software chain



Frank Gaede, Summer Student Lecture, DESY, August 20, 2008

ILC Monte Carlo software chain



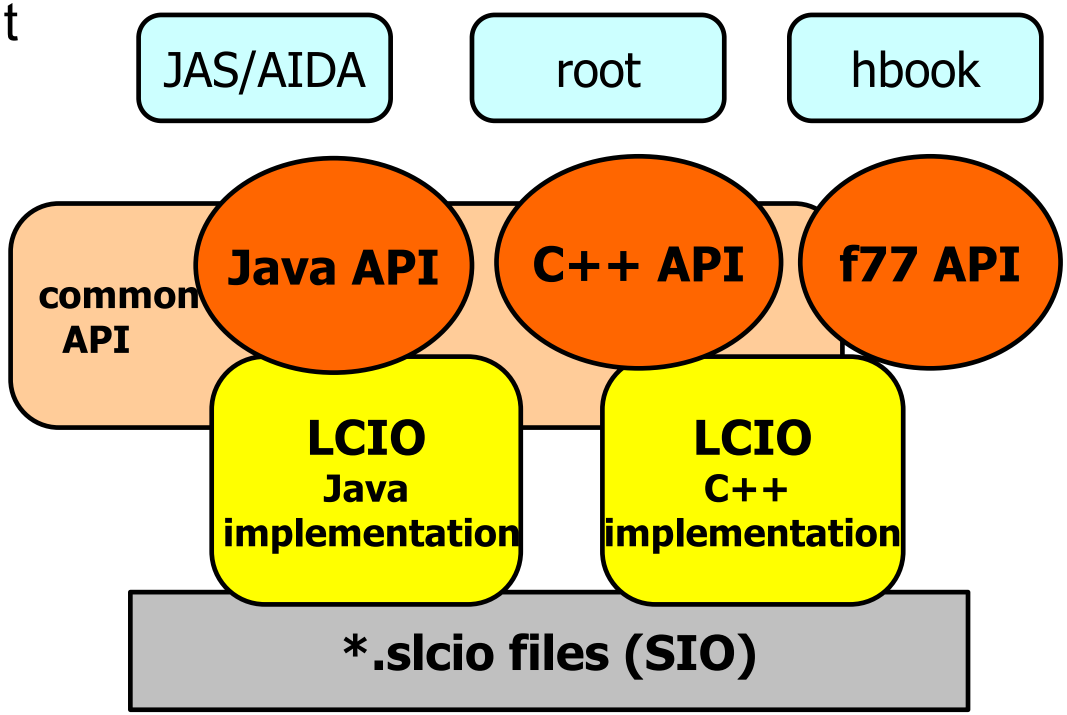
LCIO overview

- DESY and SLAC joined project:
  - provide common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependency on other frameworks

simple & lightweight

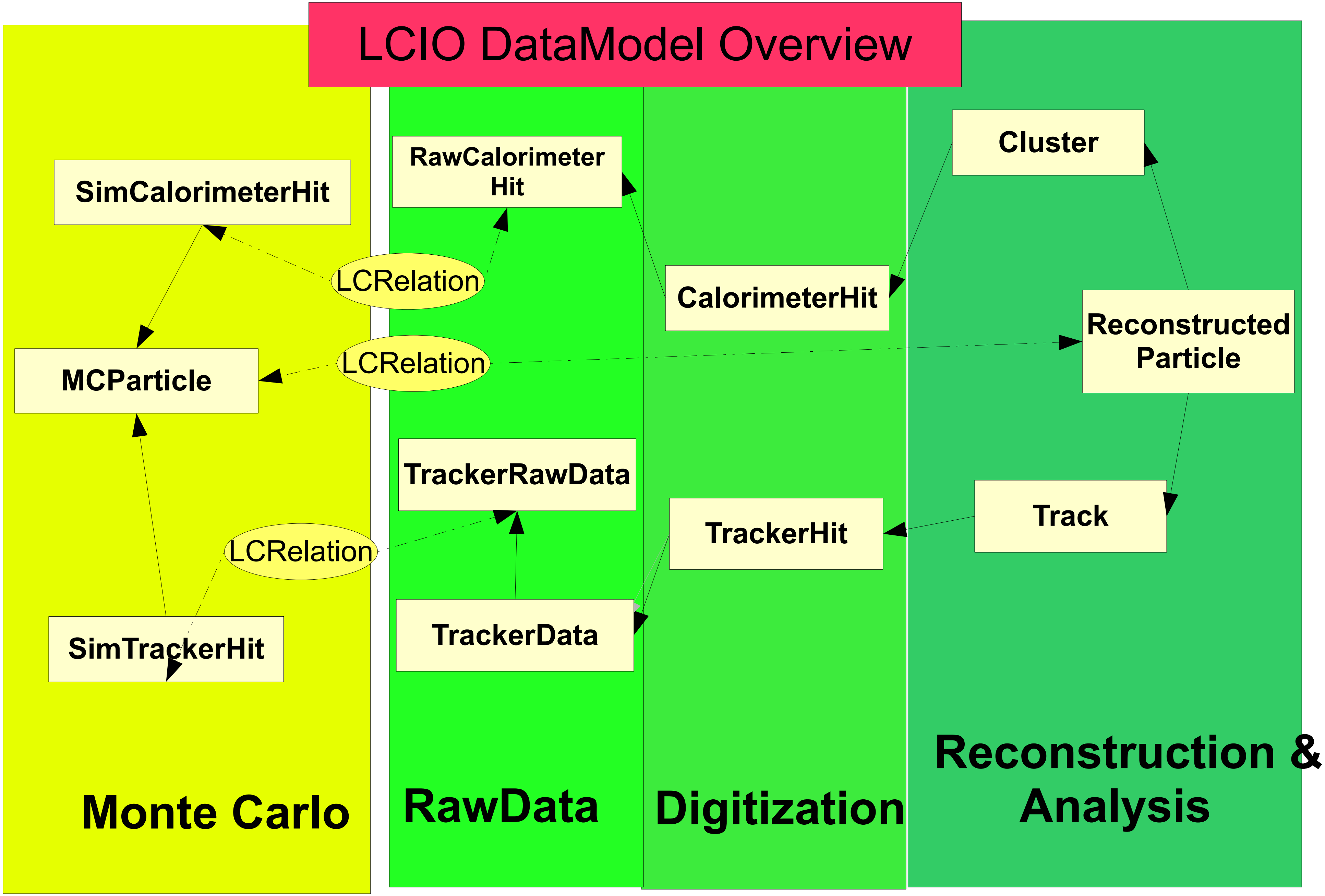
international standard persistency & datamodel for ILC software

SW-Architecture



current release: **v01-10**

event data model



example MCParticle data class

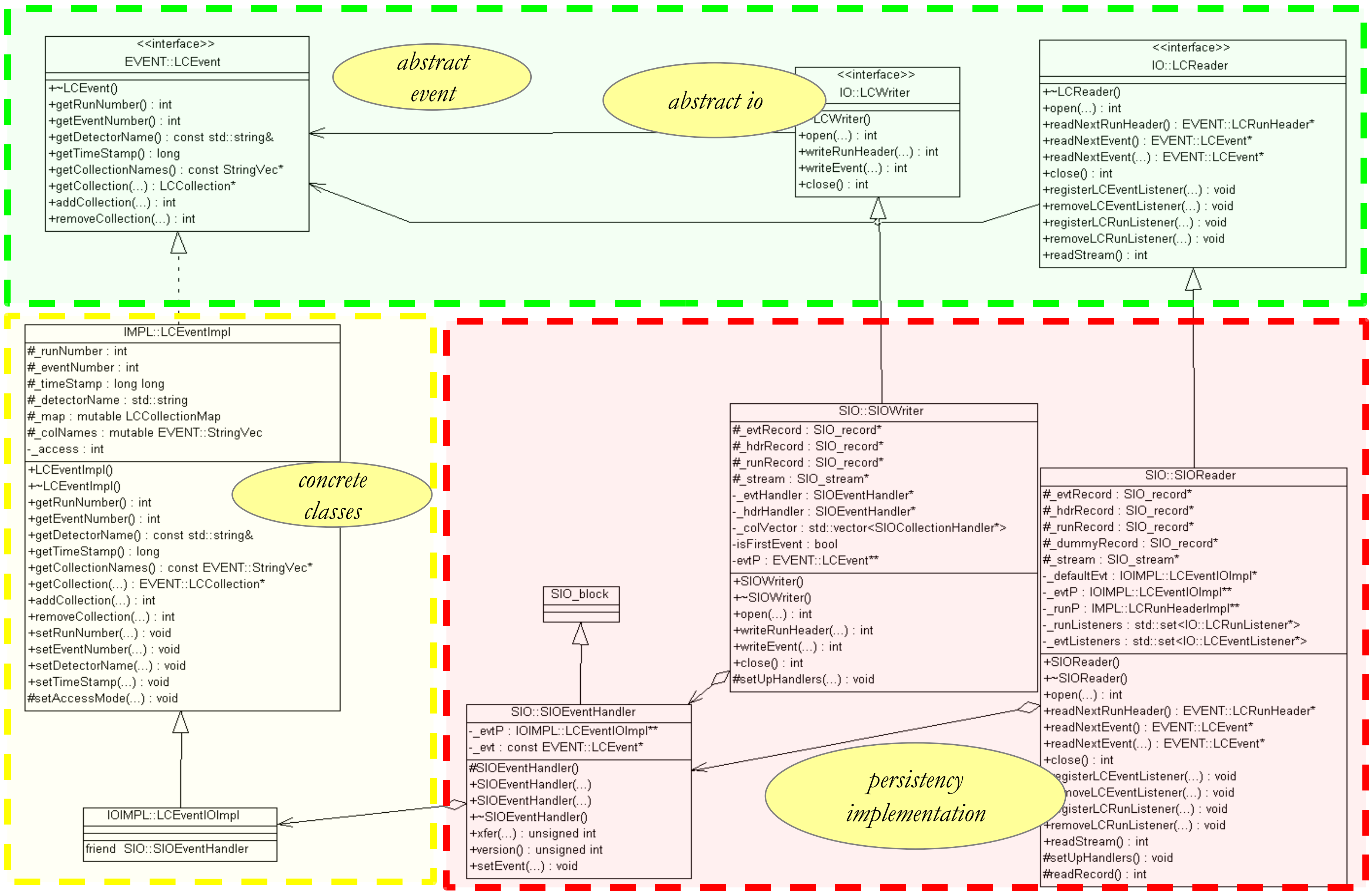
virtual	**~MCParticle** () *Destructor.*
virtual double	**getEnergy** () const=0 *Returns the energy of the particle (at the vertex) in [GeV] computed fro particle's momentum and mass - only float used in files.*
virtual const **MCParticleVec** &	**getParents** () const=0 *Returns the parents of this particle.*
virtual const **MCParticleVec** &	**getDaughters** () const=0 *Returns the daughters of this particle.*
virtual int	**getNumberOfParents** () const=0 *Returns the number of parents of this particle - 0 if mother.*
virtual **MCParticle** *	**getParent** (int i) const=0 *Returns the i-th parent of this particle.*
virtual int	**getPDG** () const=0 *Returns the PDG code of the particle.*
virtual int	**getGeneratorStatus** () const=0 *Returns the status for particles as defined by the generator, typically* *0 empty line* *1 undecayed particle, stable in the generator* *2 particle decayed in the generator* *3 documentation line.*
virtual int	**getSimulatorStatus** () const=0 *Returns the status for particles from the simulation, e.g.*
virtual bool	**isCreatedInSimulation** () const=0 *True if the particle has been created by the simulation program (rather the generator).*
virtual bool	**isBackscatter** () const=0 *True if the particle was created by the simulator as a result of an intera decay in non-tracking region, e.g.*
virtual bool	**vertexIsNotEndpointOfParent** () const=0 *True if the particle was created as a result of a continuous process where the parent particle continues, i.e.*

virtual bool	**isDecayedInTracker** () const=0 *True if the particle decayed or interacted in a tracking region.*
virtual bool	**isDecayedInCalorimeter** () const=0 *True if the particle decayed or interacted (non-continuous interaction, particle terminated) in non-tracking region.*
virtual bool	**hasLeftDetector** () const=0 *True if the particle left the world volume undecayed.*
virtual bool	**isStopped** () const=0 *True if the particle lost all kinetic energy inside the world volume and did not decay.*
virtual const double *	**getVertex** () const=0 *Returns the production vertex of the particle in [mm].*
virtual float	**getTime** () const=0 *The creation time of the particle in [ns] wrt.*
virtual const double *	**getEndpoint** () const=0 *Returns the endpoint of the particle in [mm] if the endpoint has been set explicetly.*
virtual const double *	**getMomentum** () const=0 *Returns the particle's 3-momentum at the production vertex in [GeV] only float used in files.*
virtual double	**getMass** () const=0 *Returns the mass of the particle in [GeV] - only float used in files.*
virtual float	**getCharge** () const=0 *Returns the particle's charge.*
virtual int	**getNumberOfDaughters** () const=0 *Returns the number of daughters of this particle.*
virtual **MCParticle** *	**getDaughter** (int i) const=0 *Returns the i-th daughter of this particle.*

Persistency – file formats I

- most OO-languages such as C++ don't have a built in persistency mechanism
- typically experiments defines their own binary format, due to convenience and efficiency reasons
- -> need tools/code to 'persist' class contents to files
- preferences and requirements change
- -> decouple data model from actual streaming code
  - needs to be taken into account in software design
  - (BaBar objectivity crisis)

LCIO class design



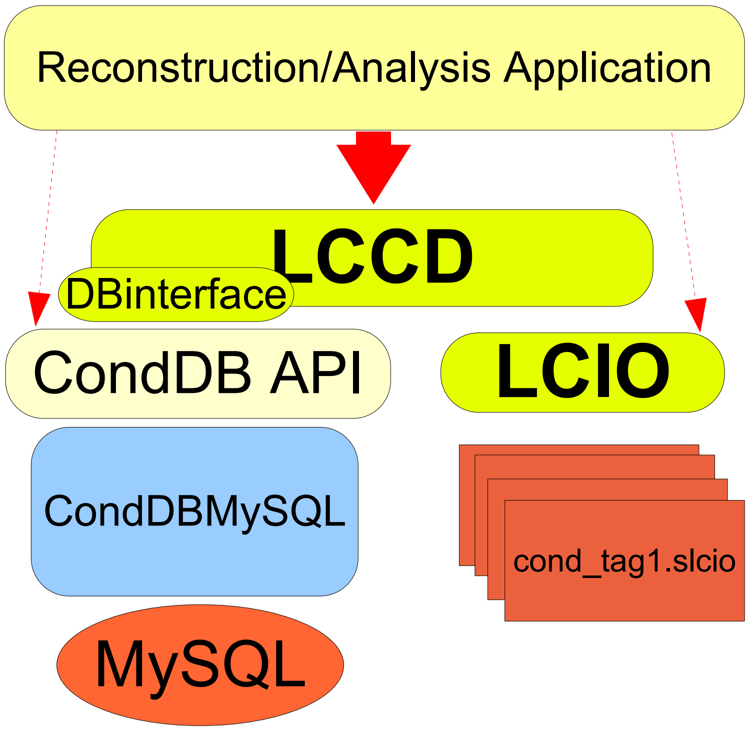
Persistency – file formats II

- file formats need to flexible:
- not allways same information stored
- different tasks need different level of detail, ie. different amount of data, eg.
  - calibration needs all detector signals (Hits)
  - physics analyses need only output of reconstruction, ie. high level objects such as reconstructed particles or even jets
- -> typically several levels of data files exist with increased condensation of information and decreased file sizes
- simplest case: full raw data & reconstructed data,

ATLAS multilevel persistency scheme

EDM Level	Contents	Primary Intent	Size/ Event (KB)	Max Ideal Input rate (Hz)	Access- ibility
Raw Data Objects	Raw Channels	Reconstruction (calibration)	1600	N/A	Central Reco/ Reprocessing: Tier 0/1
Event Summary Data	Cells, Hits, Clusters, Tracks, MET, Electron, Jet, Muon, Tau, Truth	Derive calibrations, Re-reconstruction, Re-calibration	500		CERN CAF (access limited), Tier 1 (on tape)
Analysis Object Data	Lepton Cells, Hits, Clusters, Tracks, MET, Electron, Jet, Muon, Tau, Slimmed Truth	Limited Re-reconstruction (eg Jets, b-tag), limited re-calibration, Analysis	100	1000	Full: Tier 1,2 (disk) Subset: Tier 3
Derived Physics Data	Any of the above + composites (eg top) + derived quantities (sphericity)	Interactive Analysis: Making plots, performing studies	Typically ~10	106	Tier 3: eg your laptop
TAG	Summary. Ex: p_T, η of 4 best e, γ, μ, τ, jet	Selection Events for analysis	1	108	Everywhere

conditions database



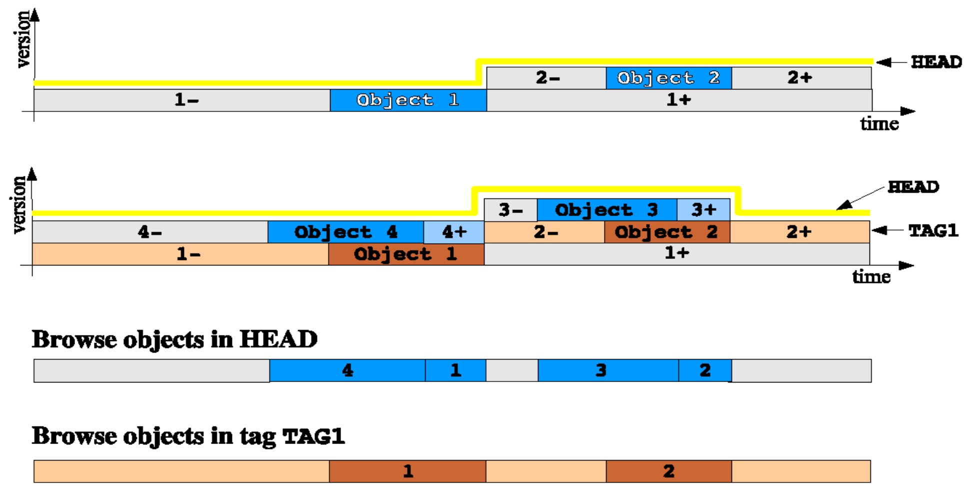


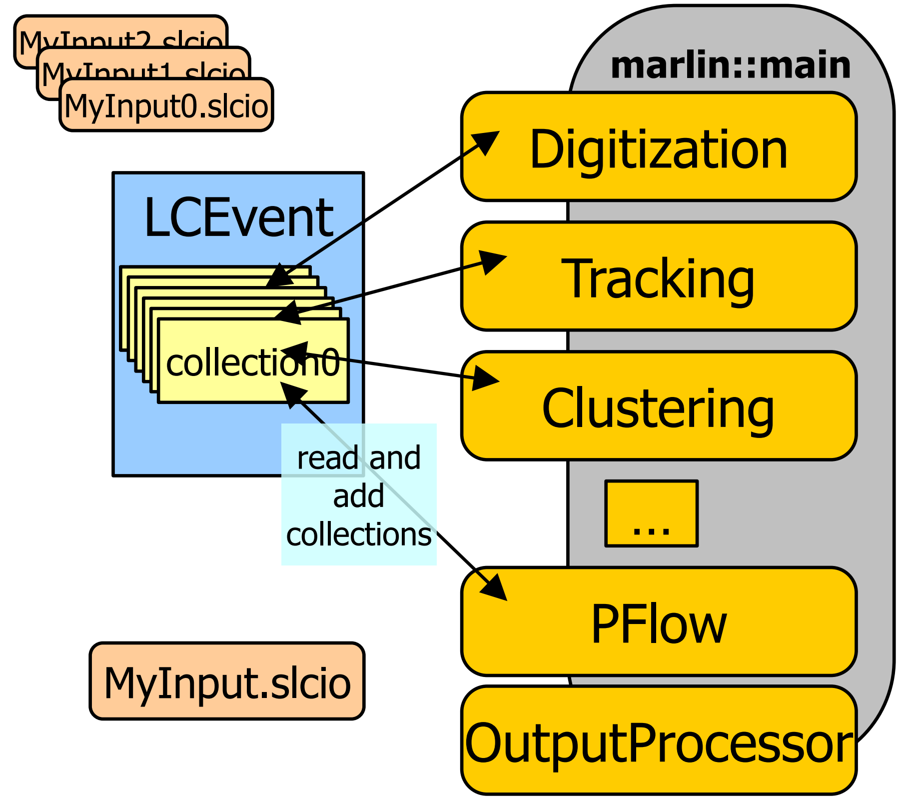
Figure 3: tagging and browsing example in the ConditionsDB mySQL's implementation.

- Conditions Data:
  - all data that is needed for analysis/reconstruction besides the actual event data
  - typically has lifetime (validity range) longer than one event
    - can change on various timescales, e.g. seconds to years
    - need for tagging mechanism, e.g. for calibration constants
- example: trigger configuration, temperature readings, gas pressures,
- calibration constants, electronic channels mapping,...

example analysis/reconstruction framework: Marlin

Modular**A**nalysis & **R**econstruction for the **L I N**ear Collider

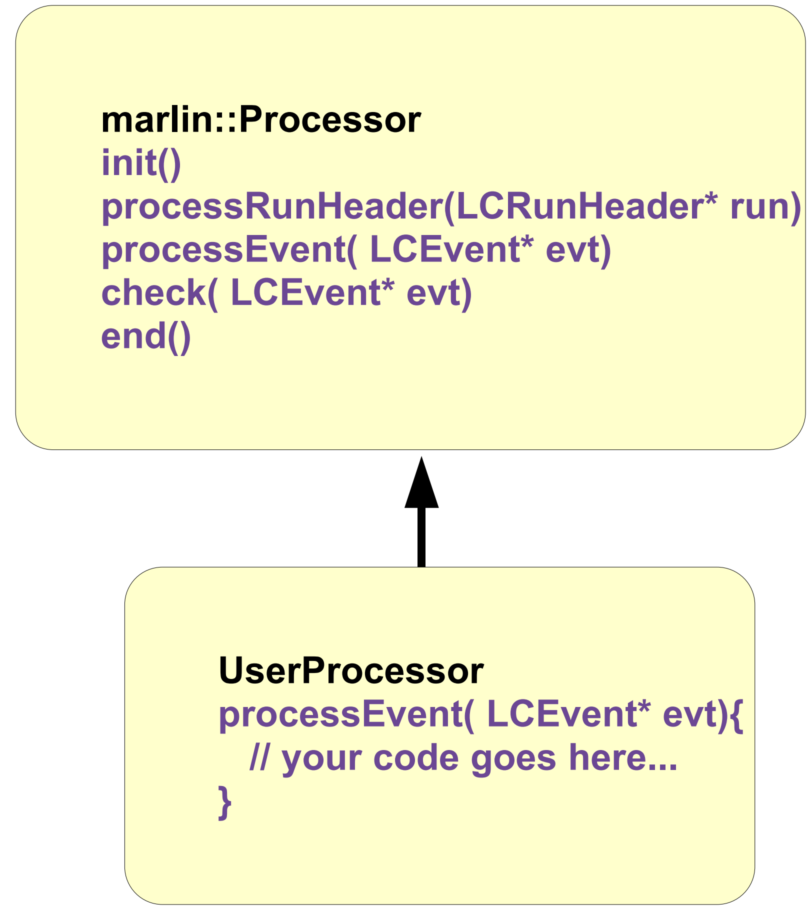
- modular C++ **application framework** for the analysis and reconstruction of LCIO data
- uses LCIO as transient data model
- software modules called Processors
- provides main program !
- provides simple user steering:
  - program flow (active processors)
  - user defined variables
    - per processor and global
  - input/output files
  - **Plug&Play** of processors



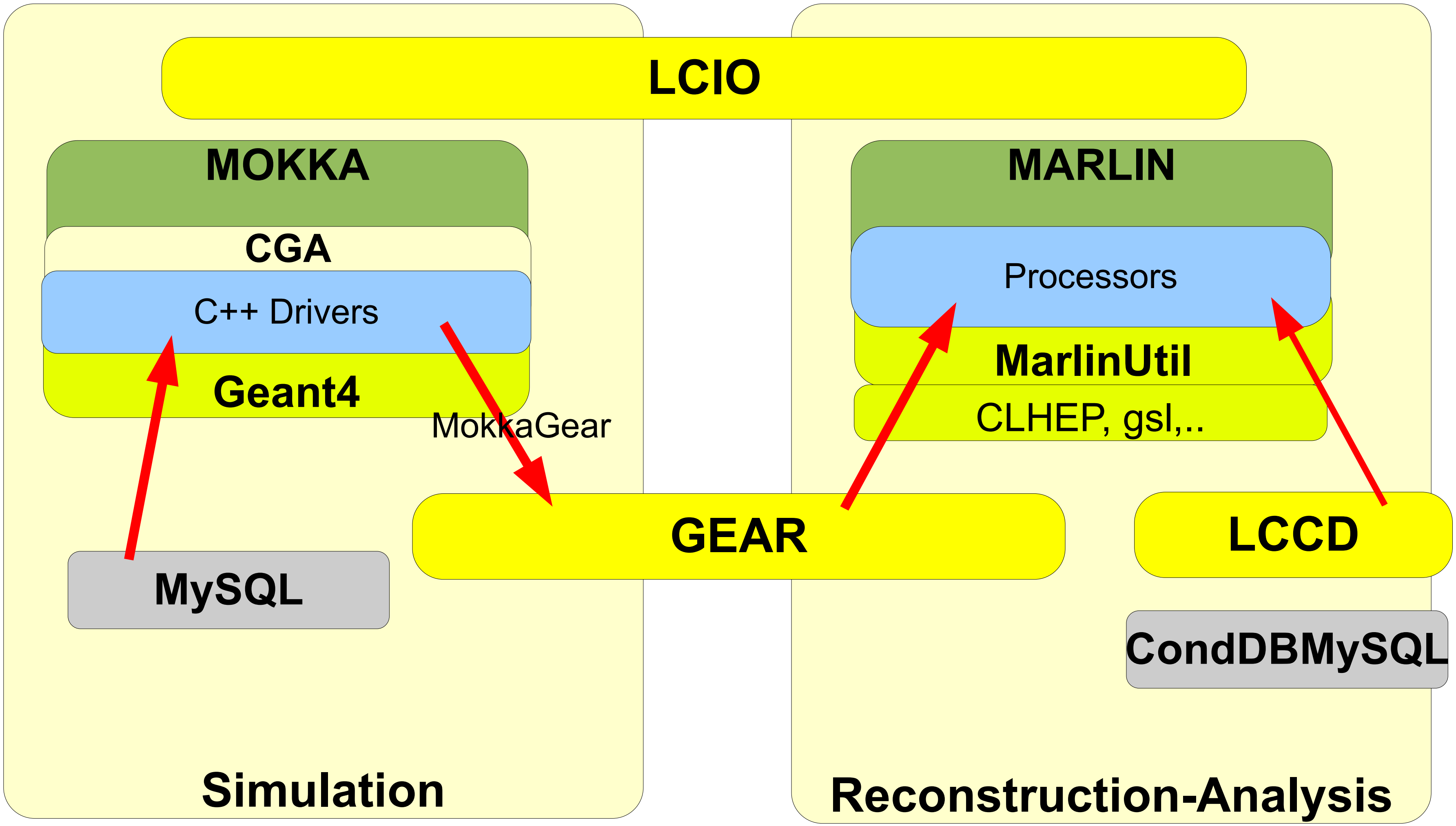
Frank Gaede, Summer Student Lecture, DESY, August 20, 2008

Marlin Processor

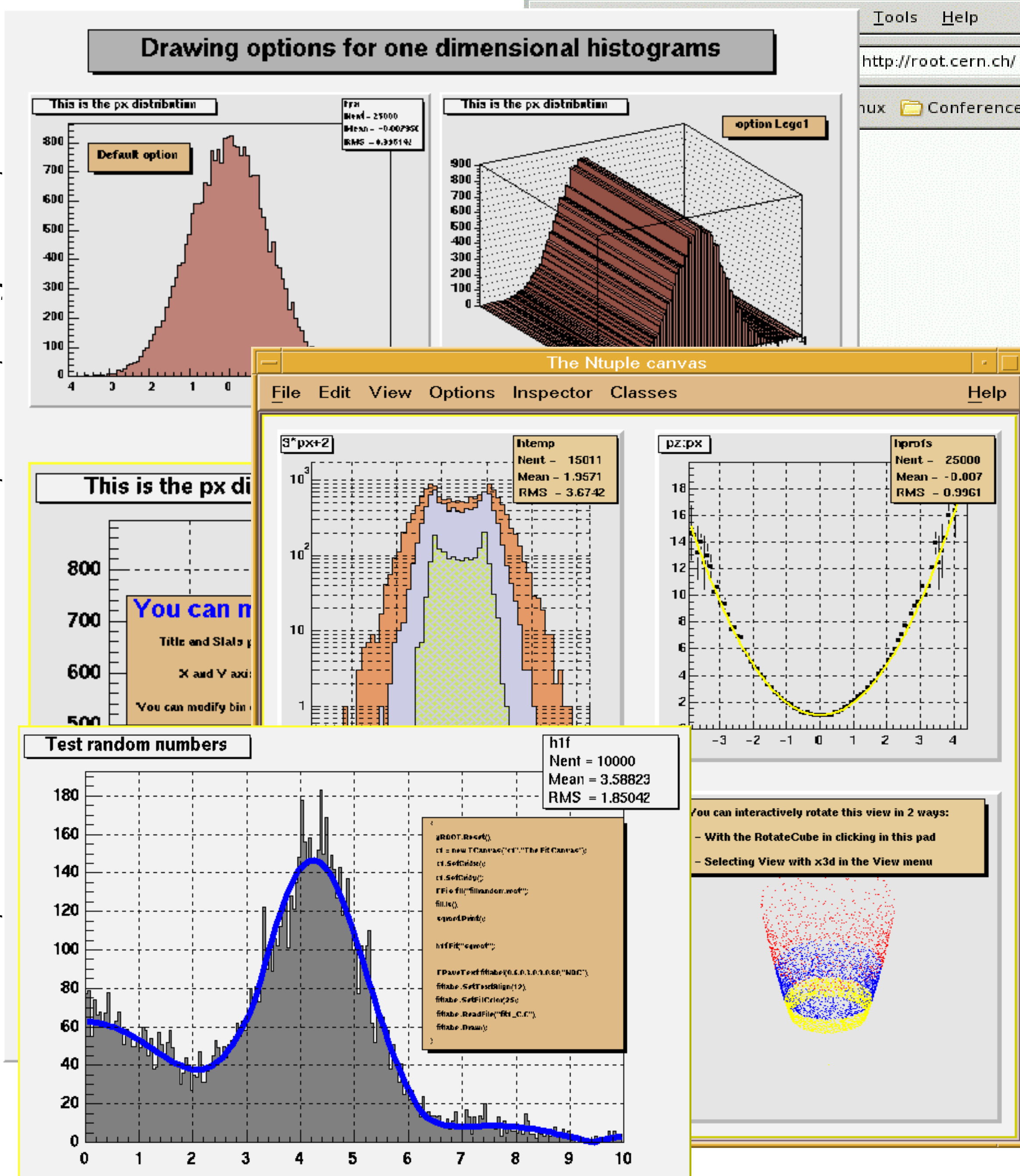
- provides main user callbacks
- has own set of input parameters
  - int, float, string (single and arrays)
  - parameter description
- naturally modularizes the application
- order of processors is defined via steering file:
  - easy to exchange one or several modules w/o recompiling
  - can run the same processor with different parameter set in one job
- processor task can be as simple as creating one histogram or as complex as track finding and fitting in the central tracker

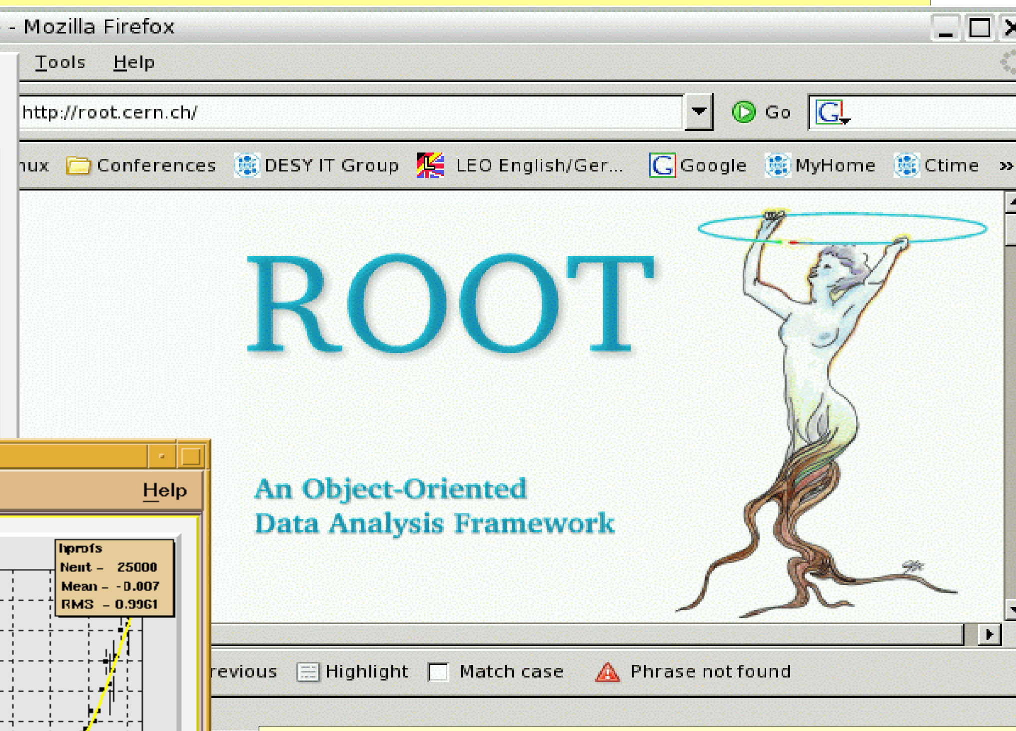


LDC simulation framework



Analysis Tools – example root



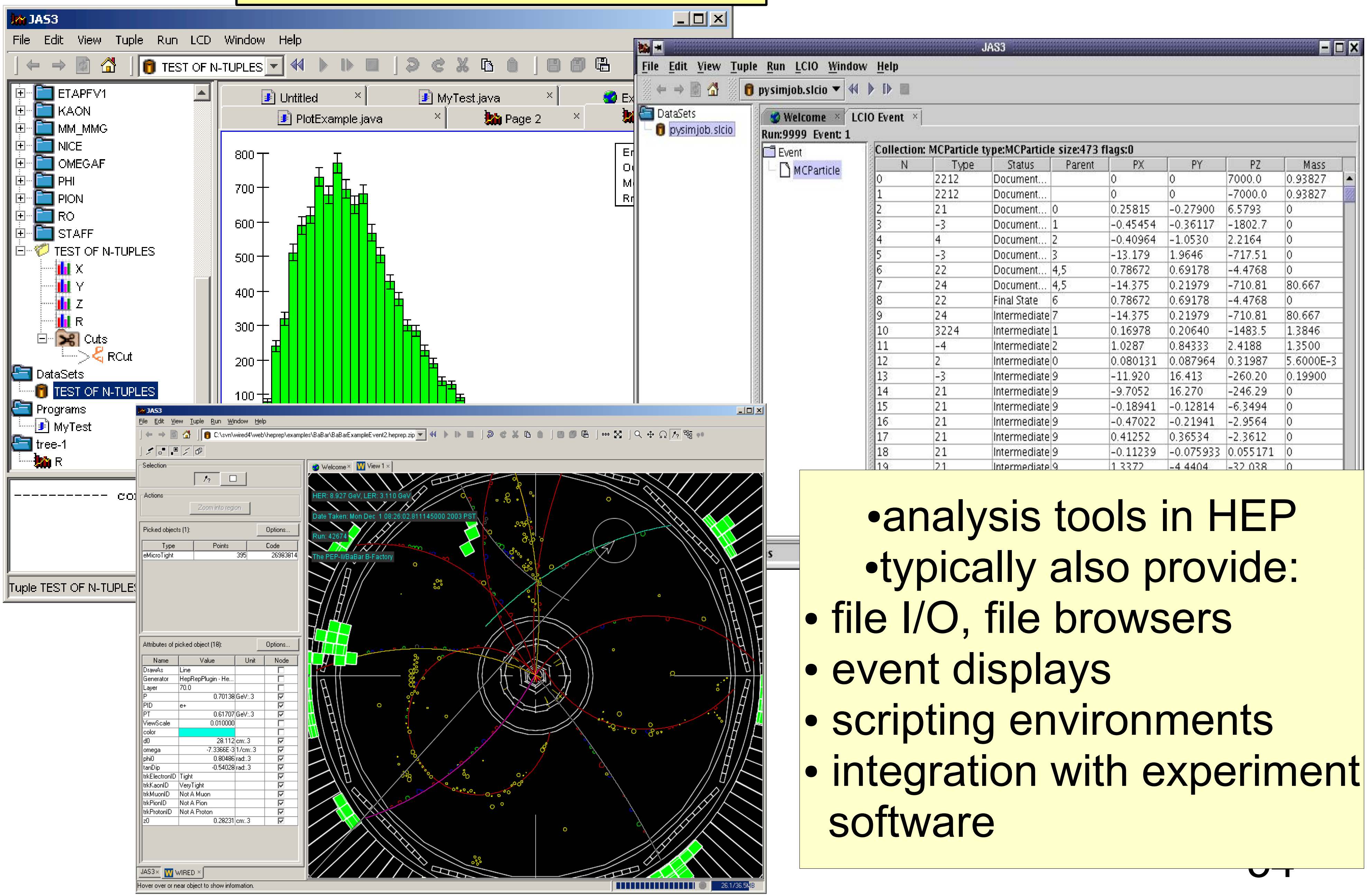


- analysis tools in HEP
- provide core features:
- ntuples
- histograms (1D/2D)
- fitting
- plotting /publishing

note: the root C++ framework provides much more functionality not discussed here

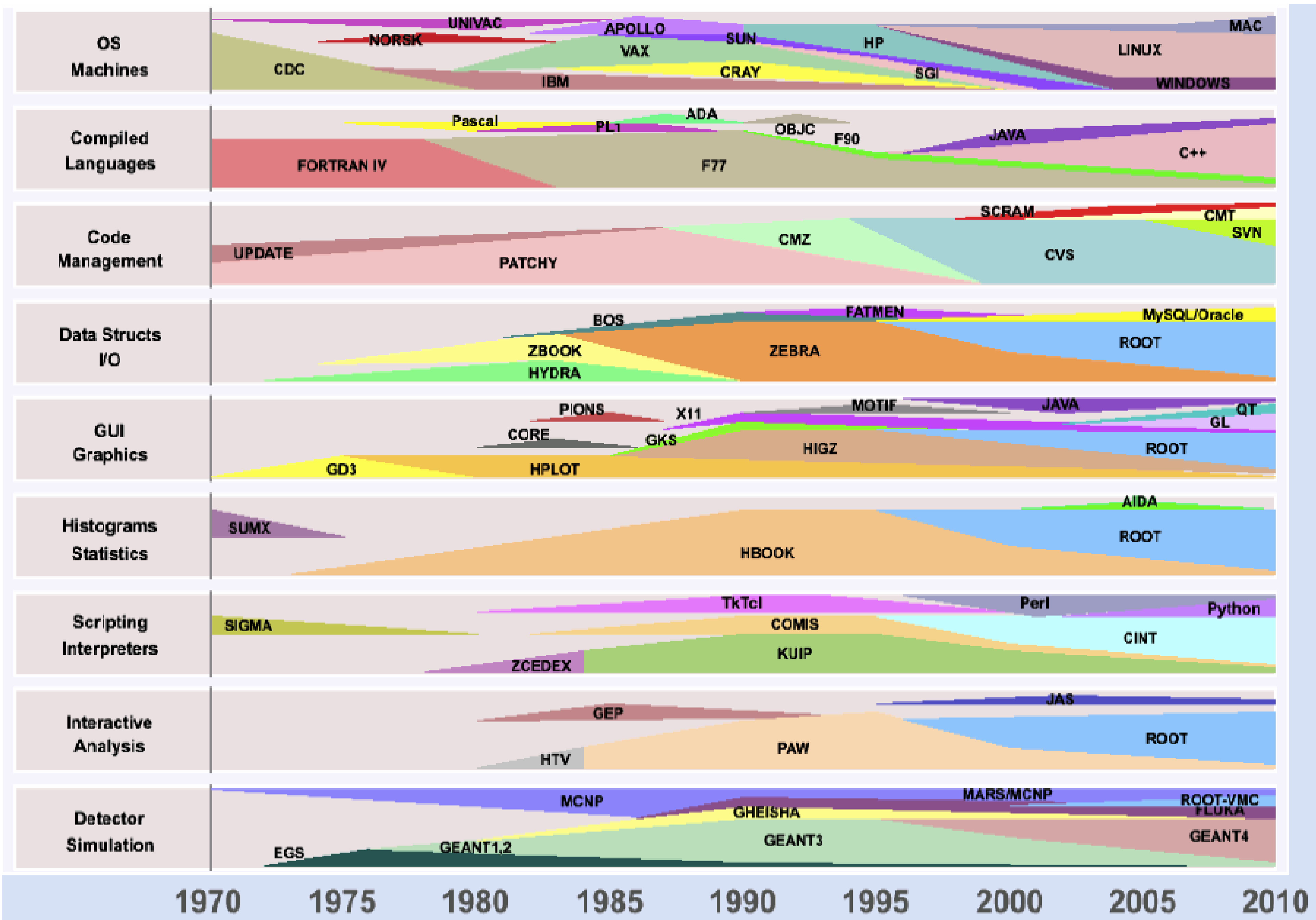
Analysis Tools – example JAS3

http://jas.freehep.org/jas3/index.html



- analysis tools in HEP
- typically also provide:
- file I/O, file browsers
- event displays
- scripting environments
- integration with experiment software

tools/languages used in HEP

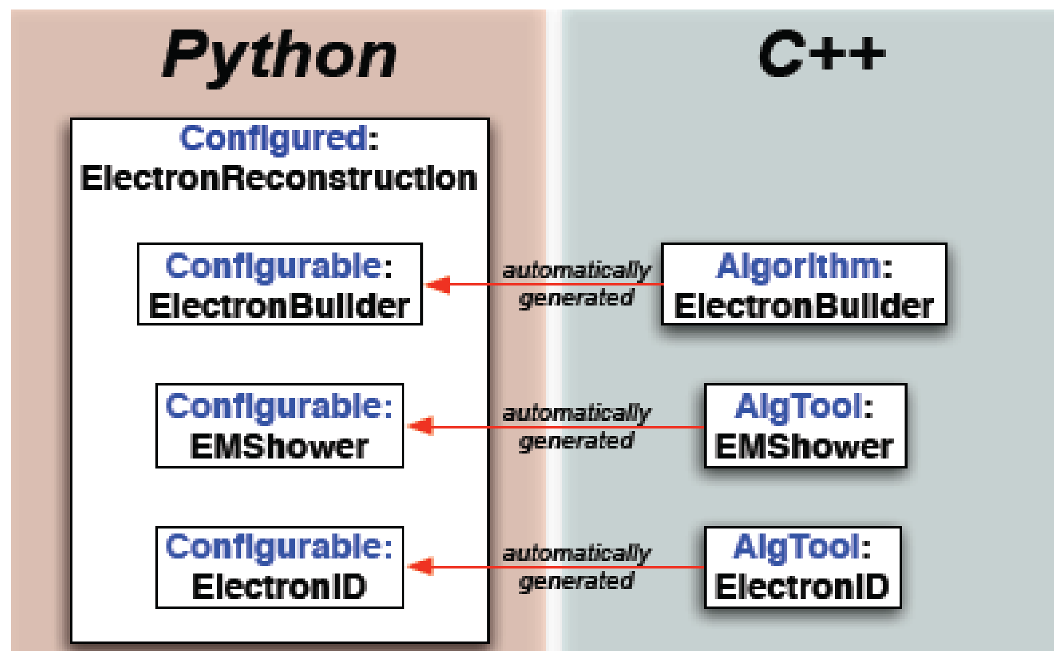


Frank Gaede, Summer Student Lecture, DESY, August 20, 2008

multi language frameworks

- different languages have different advantages in different application domains
- programmers have different skills (and preferences !)
  - Atlas and CMS: more than 2000 physicists
- multi language frameworks are a possible way to 'keep everyone happy'

example:
Atlas sw framework:
- core sw in C++
- python binding for modules
- main application incl. configuration and simple analyses in python



Computing Infrastructure - Hardware

Mass Storage

- mass storage of HEP data and Monte Carlo is typically done on tapes
- e.g. @ DESY we have (8/2006)
  - 22 000 tapes
  - 46 tape drives
  - 4 robots
  - ~ 1.6 PetaByte data (mostly HERA experiments and Monte Carlo)

access to data on tape fairly slow
-> need smart disk caching system

dCache

SRM provides interface to grid

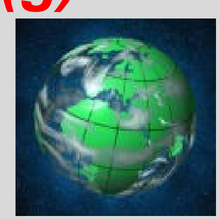
SRM for storage management

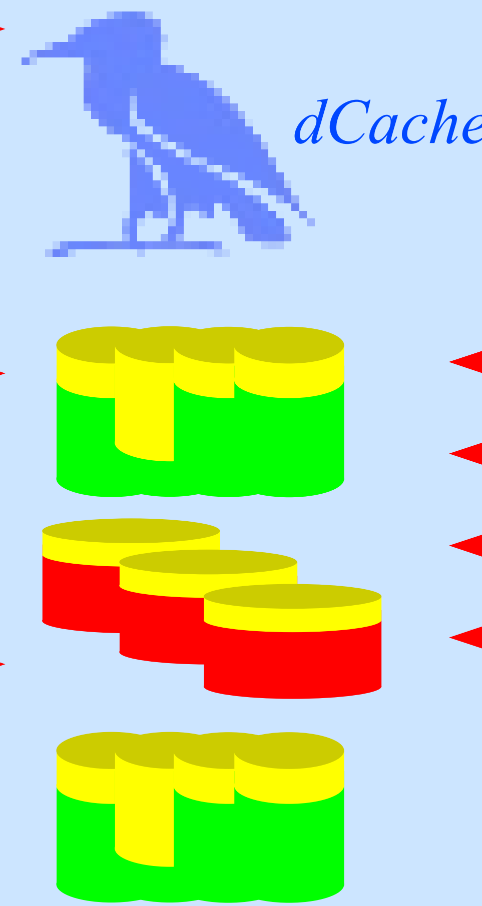
nfs2/3 for name space

/pnfs/<site>/<VO>/...

dCap, xRoot for random LAN

gsiFtp, http(g) for random WAN





Osm, Enstore, Tsm, Hpss, ...

- dCache developed @ DESY&FNAL
- transparent access to files on tape via pnfs file system
- various protocols for worldwide access and file transfer

Computing plattforms I

- the main working horse for HEP computing today are large PC clusters or farms, which are mostly operated with *linux*
- typically the high level trigger operates on a *dedicated experiment specific farms* in order to guarantee the throughput needed
- Monte Carlo production, reconstruction and analysis run on often on shared farms
  - shared between *tasks*
  - shared between *experiments* (-> institute batch system)
  - shared between *institutes* (-> grid)
  - shared between *communities* (-> grid)

Computing plattforms II

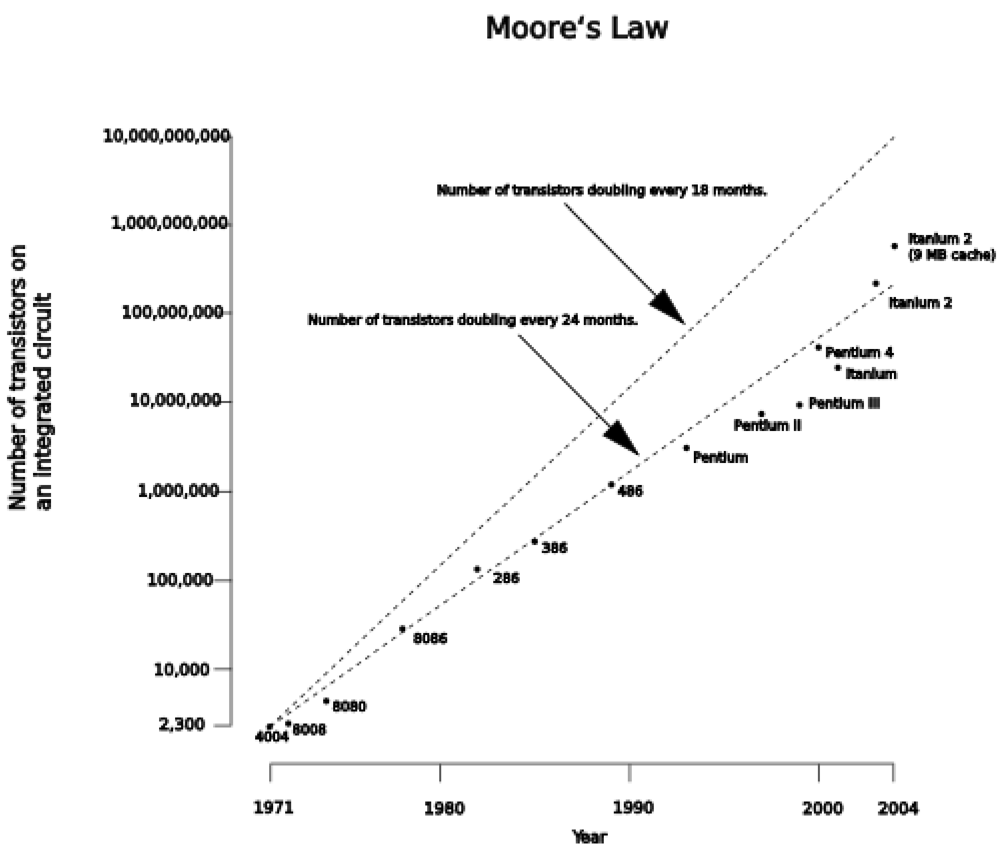
Computing requirements

Application	Input	Output	CPU
MC generator	*none*	*small*	*little*
MC simulation	*small*	*large*	*huge*
MC digitization	*large/huge*	*large*	*little*
Reconstruction	*large*	*large*	*large*
Analysis	*large*	*small*	*small*

example:

- simulating (geant4) an ILC event takes ~200s on a standard PC (2007) (an Atlas event takes ~600s !)
- O(10^6) events will take 10 CPU years !
- need hundreds of CPUs to get events in reasonable time

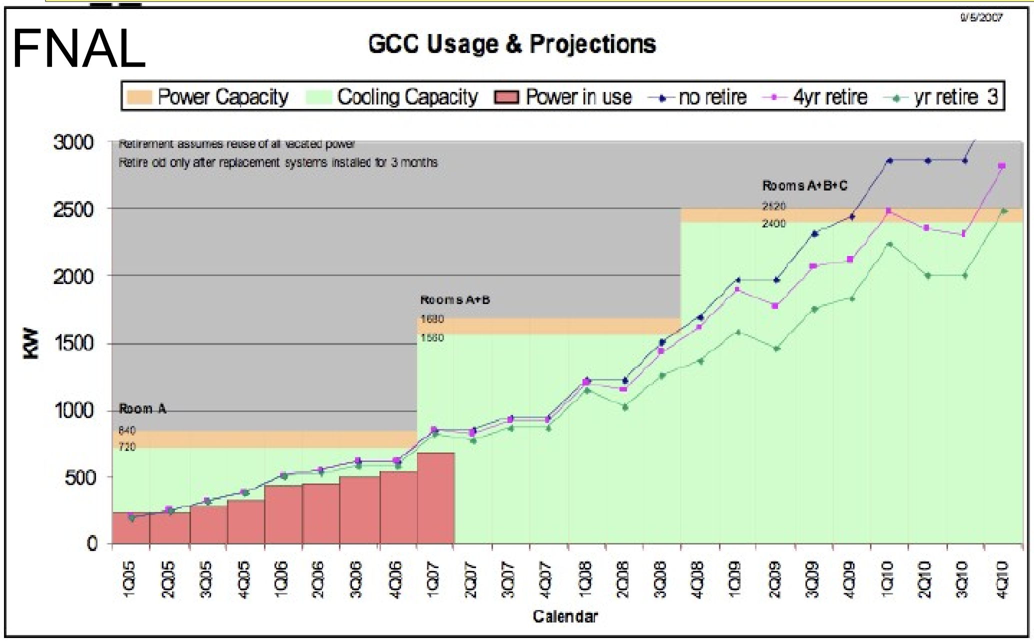
increase of computing power

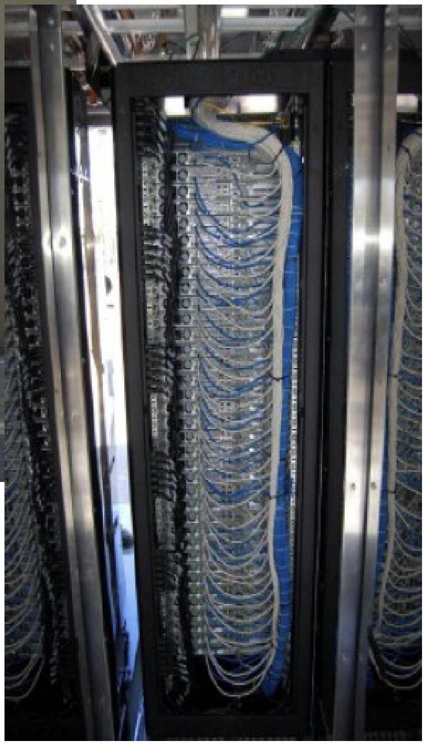


- **Moore's law:**
- number of transistors per chip doubles every ~2y
- ~true for >40 years !
- however:
  computing power does not grow at the same speed
  -> cache, memory, architecture
- requirements grow even faster:
  - more and larger data sets
  - complexity of algorithms
  - 'sloppiness' of programmers
  (why bother writing efficient code – the next generation of CPUs will run the slower/larger algorithm as fast ...)

- the ever increasing need for more
- computing power leads to larger and
- larger farms (thousands of CPUs)
- problem (cost):
  - **power and cooling**
  - need for new **facilities**

power, cooling and facilities





SLAC

- replace old 'junk'
  - hardware older than 2-4 years is to expensive to operate -> replace w./ new hardware
- use water cooled racks (more efficient)
- increase density of CPUs per rack
  - -> multi core CPUs
- SUN's black box: container solution

electricity bills for computing will become a large fraction of labs' total budgets !

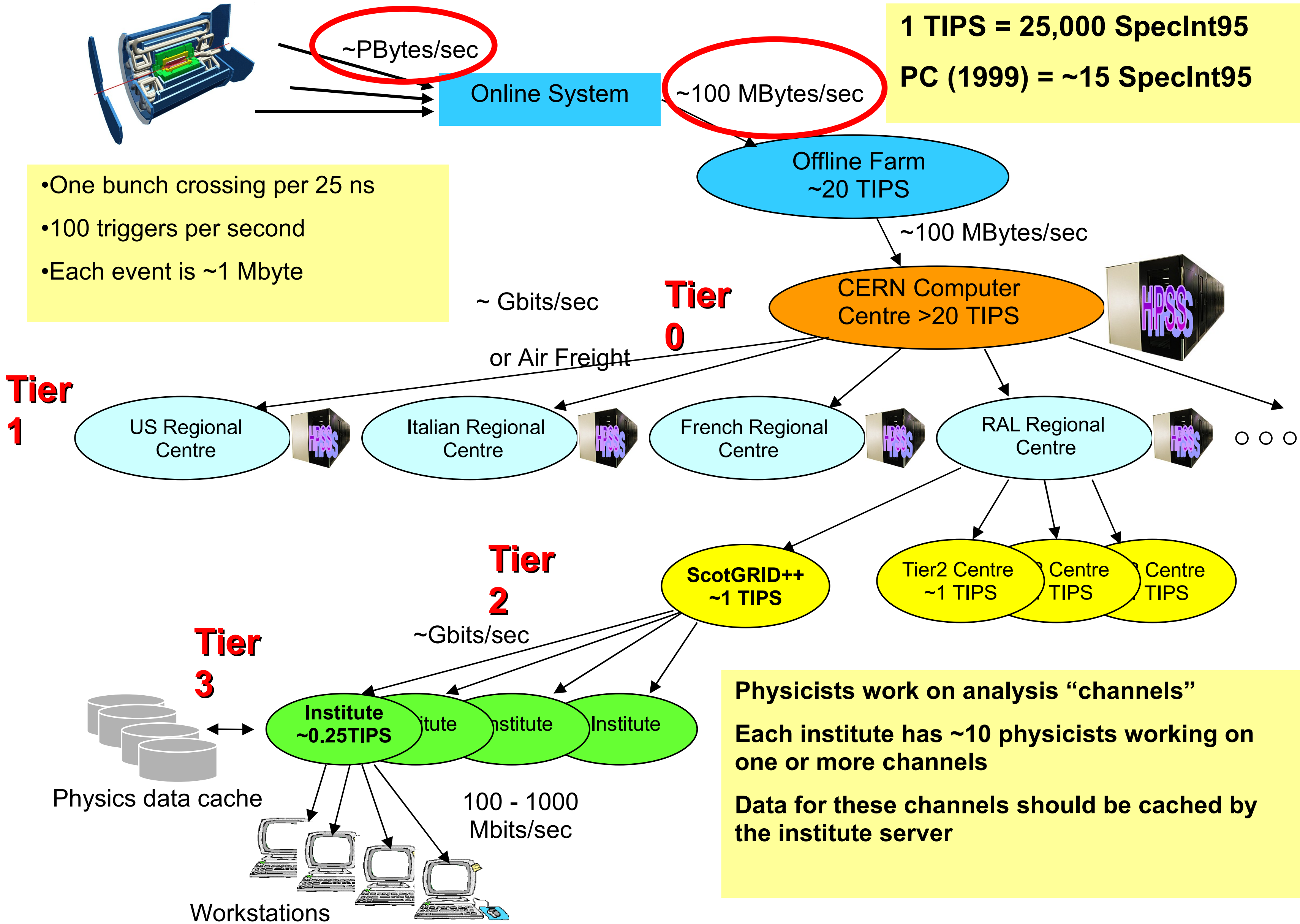
multi core CPUs

- HEP computing is 'embarrassingly parallel':
  - process one event at a time on one computer
  - -> total processing power scales w/ number of CPUs
  - HEP computing paradigm of 90s and 0s
- current trend: multi core CPUs
  - memory (RAM and L2 cache) does not scale w. # cores
  - processing one event per core is inefficient
  - -> need new computing paradigms and tools
    - multithreading (non trivial)
    - dedicated compilers that do multithreading for you
    - smaller memory footprint of programs
    - learn from 'embedded' programmers !?
  - ongoing exiting developments ...

Grid Computing

“Sharing resources within Virtual Organizations in a global world.”

LHC Computing model



1 TIPS = 25,000 SpecInt95

PC (1999) = ~15 SpecInt95

- One bunch crossing per 25 ns
- 100 triggers per second
- Each event is ~1 Mbyte

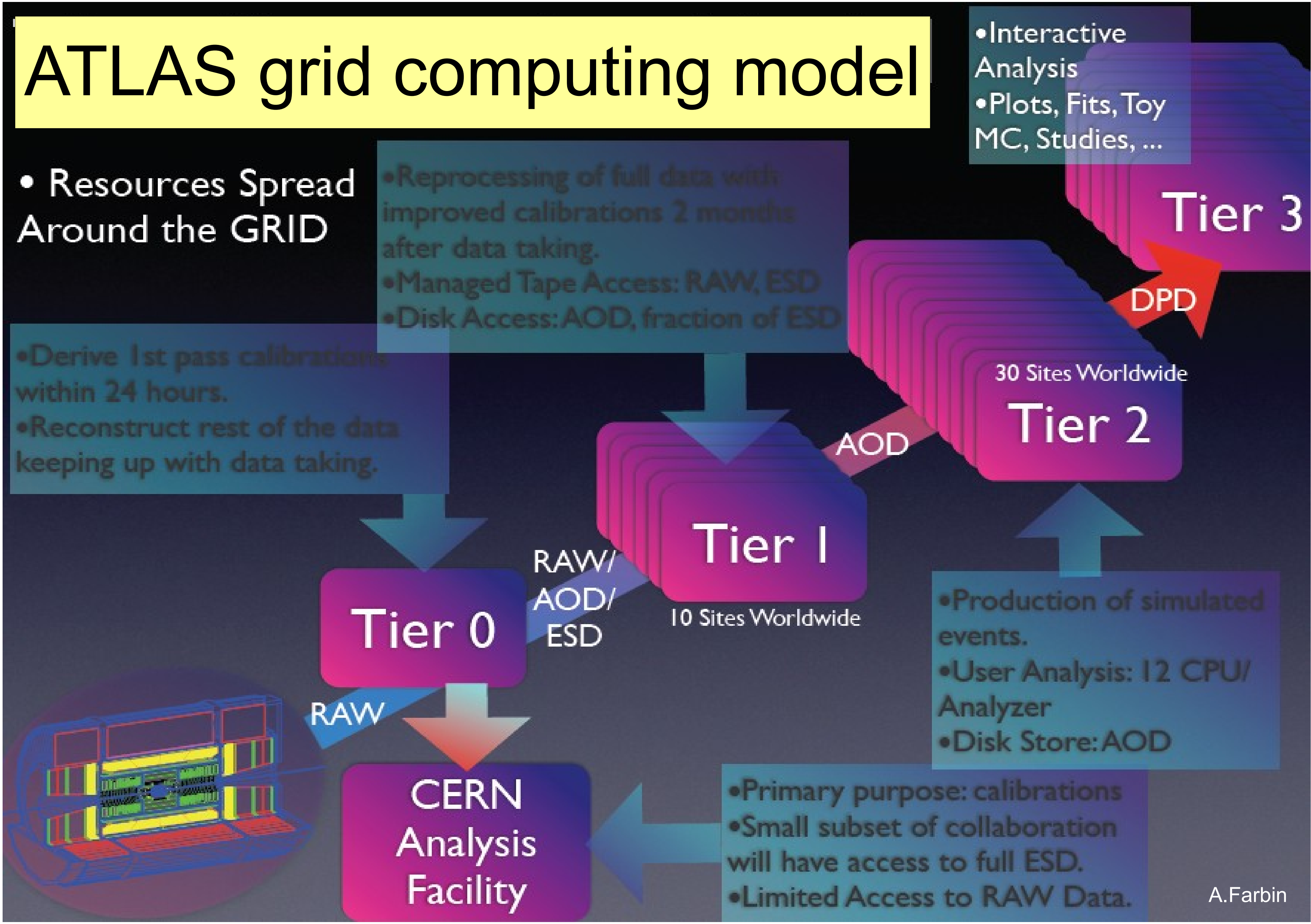
Physicists work on analysis “channels”

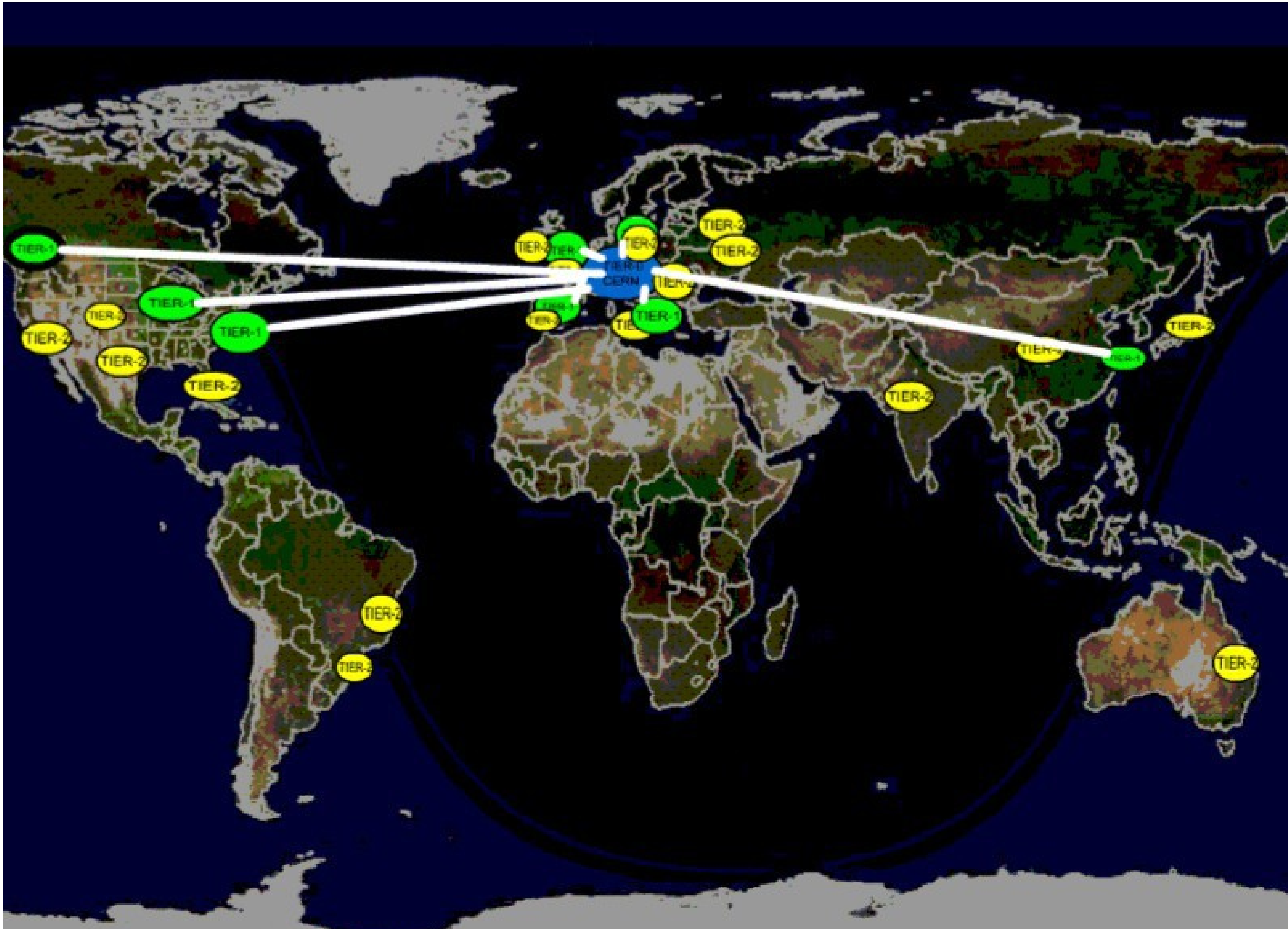
Each institute has ~10 physicists working on one or more channels

Data for these channels should be cached by the institute server

ATLAS grid computing model

- Resources Spread Around the GRID

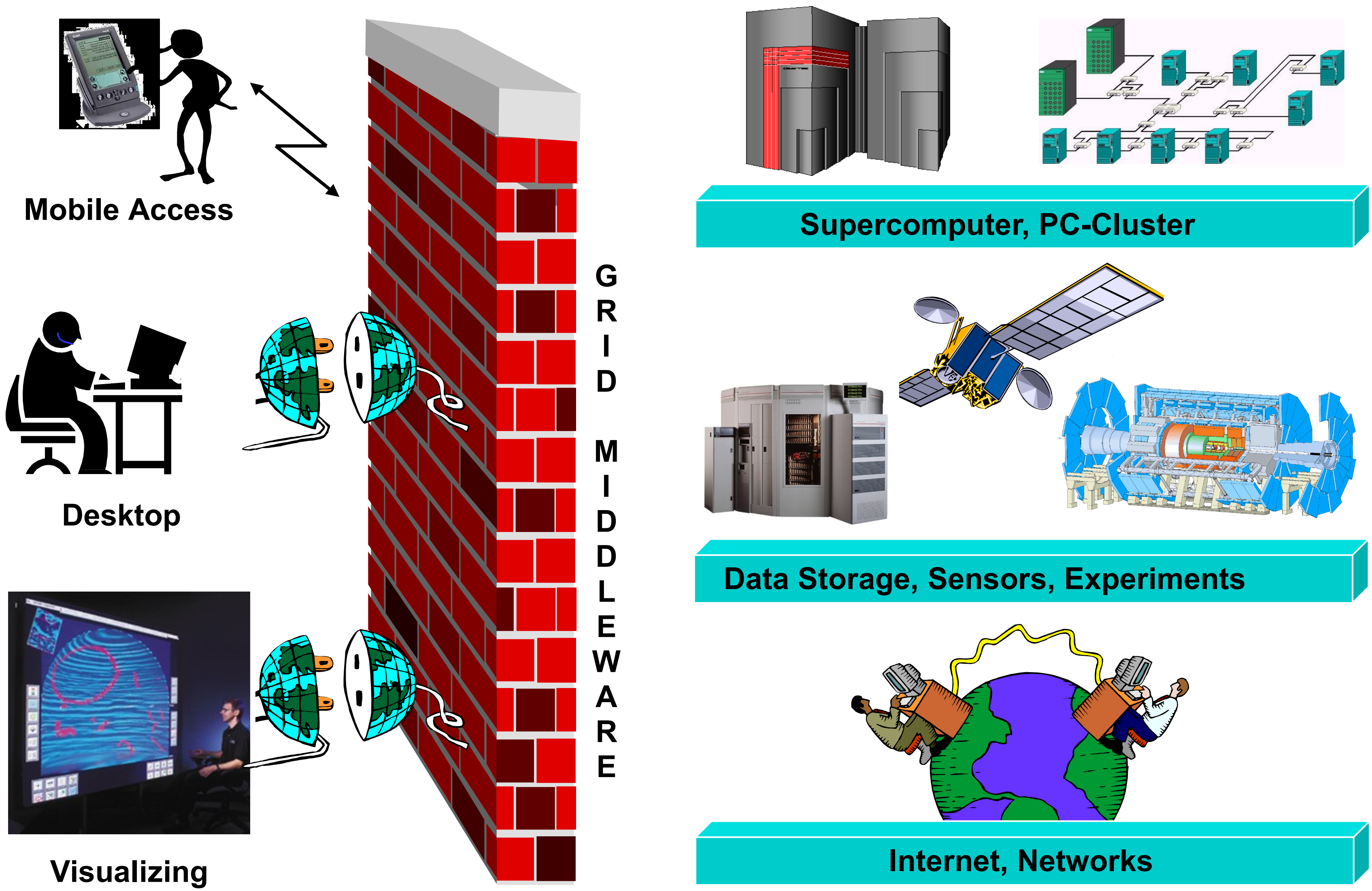



TIER-1
CERN
TIER-1
TIER-2

Grid Definition

- I. Foster: What is the Grid? A Three Point Checklist (2002)
- “A Grid is a system that:
- coordinates resources which are not subject to centralized controls …
  - integration and coordination of resources and users of different domains vs. local management systems (batch systems)
- … using standard, open, general-purpose protocols and interfaces …
  - standard and open multi-purpose protocols vs. application specific system
- … to deliver nontrivial qualities of services.”
  - coordinated use of resources vs. uncoordinated approach (world wide web)

The Grid dream



The Fuzz about Grids

DataGRID

GGF

ILDG

crossGrid

egee Enabling Grids for E-sciencE

LCG

TERAGRID

NORDUGRID Nordic Testbed for Wide Area Computing and Data Handling

GriPhyN Data Intensive Science

GRIDPP UK Particle Physics

INFN GRID

DataTAG

PPDG

iVDgL

HEP Grids Worldwide

Data GRID

LCG

Trillium GriPhyN iVDGL PPDG

DataTAG

Grid Types

- Data Grids:
  - Provisioning of transparent access to data which can be physically distributed within Virtual Organizations (VO)
- Computational Grids:
  - allow for large-scale compute resource sharing within Virtual Organizations (VO)
- Information Grids:
  - Provisioning of information and data exchange, using well defined standards and web services

Grid Ingredients

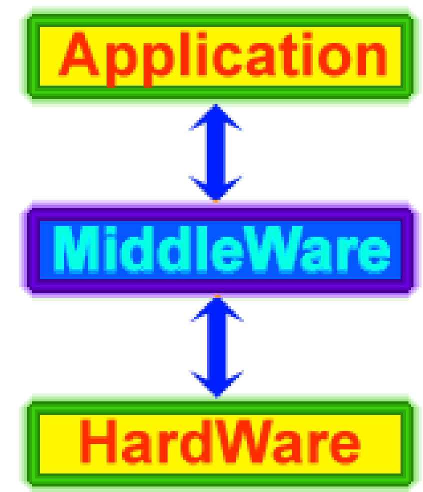
- Authorization:
  - Users must be registered in a Virtual Organization (VO)
- Information Service:
  - Provide a system which keeps track of the available resources
- Resource Management:
  - Manage and exploit the available computing resources
- Data Management:
  - Manage and exploit the data

Grid: Authentification & Authorization

- a user is uniquely identified through a *certificate*
  - an encrypted electronic document, digitally signed by a Certification Authority (CA)
  - a certificate is your passport to enter the grid world
  - example: /O=GermanGrid/OU=DESY/CN=Frank Gaede

- access to resources is provided (controlled) via membership in a Virtual Organization
  - a dynamic collection of individuals, institutions, and resources which is defined by certain sharing rules
  - the VO a user belongs to is not part of the certificate.
  - a VO is defined in a central list, e.g. a LDAP tree.
  - DESY maintains VOs for experiments and groups, e.g. *hone, zeus, ilc, ...*

Grid Middleware



Globus:

- Toolkit
- Argonne, U Chicago

EDG (EU DataGrid):

- Project to develop Grid middleware
- Uses parts of Globus
- Funded for 3 years (01.04. 2001 - 31.03.2004)

LCG (LHC Computing Grid):

- Grid infrastructure for LHC production
- Based on stable EDG versions plus VDT etc.
- LCG-2 for Data Challenges

EGEE (Enabling Grids for E-Science in Europe)

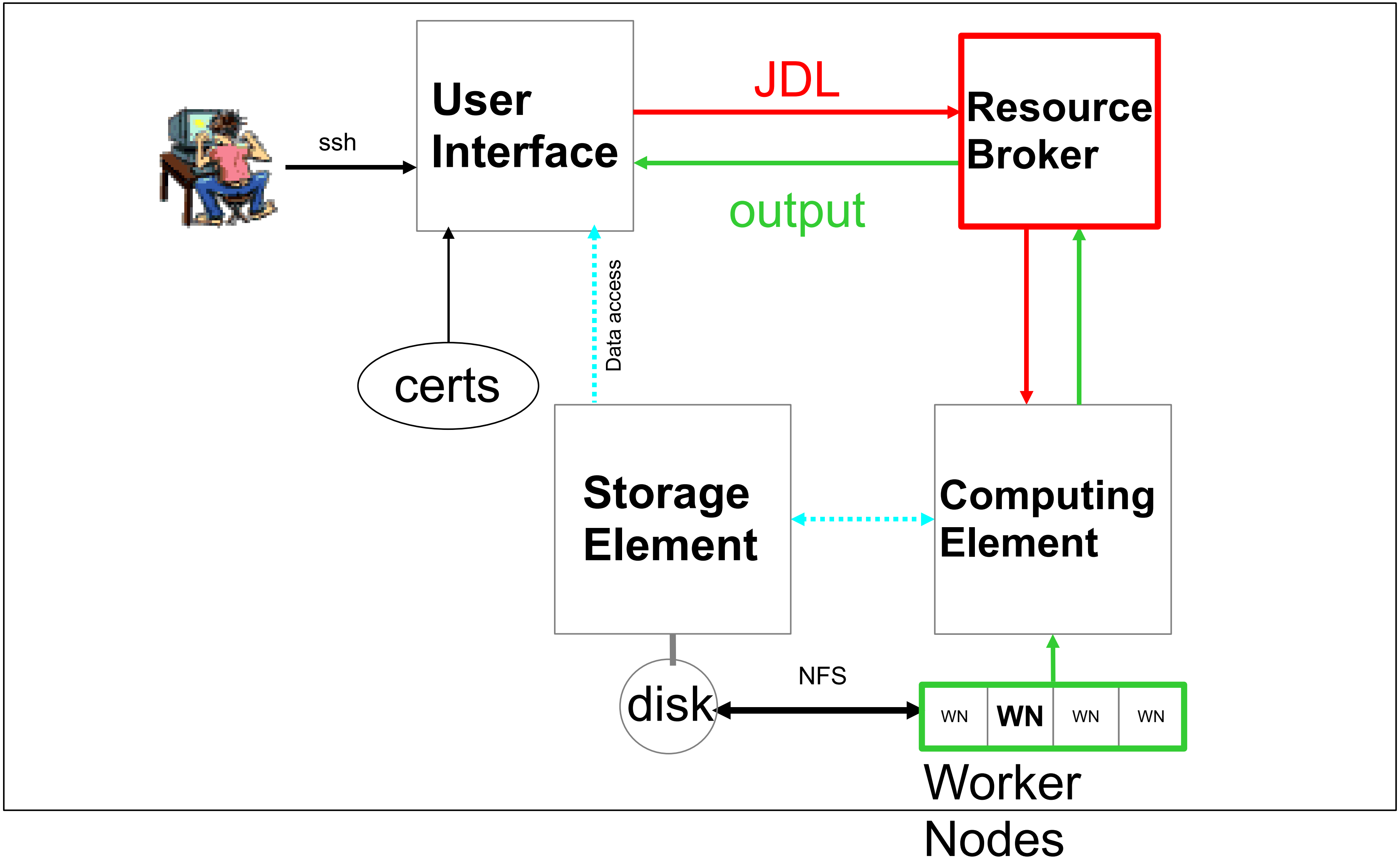
- Started 01.04.2004 for 2 + 2 years
- developed **gLite** as successor of LCG middleware

Job submission to the Grid

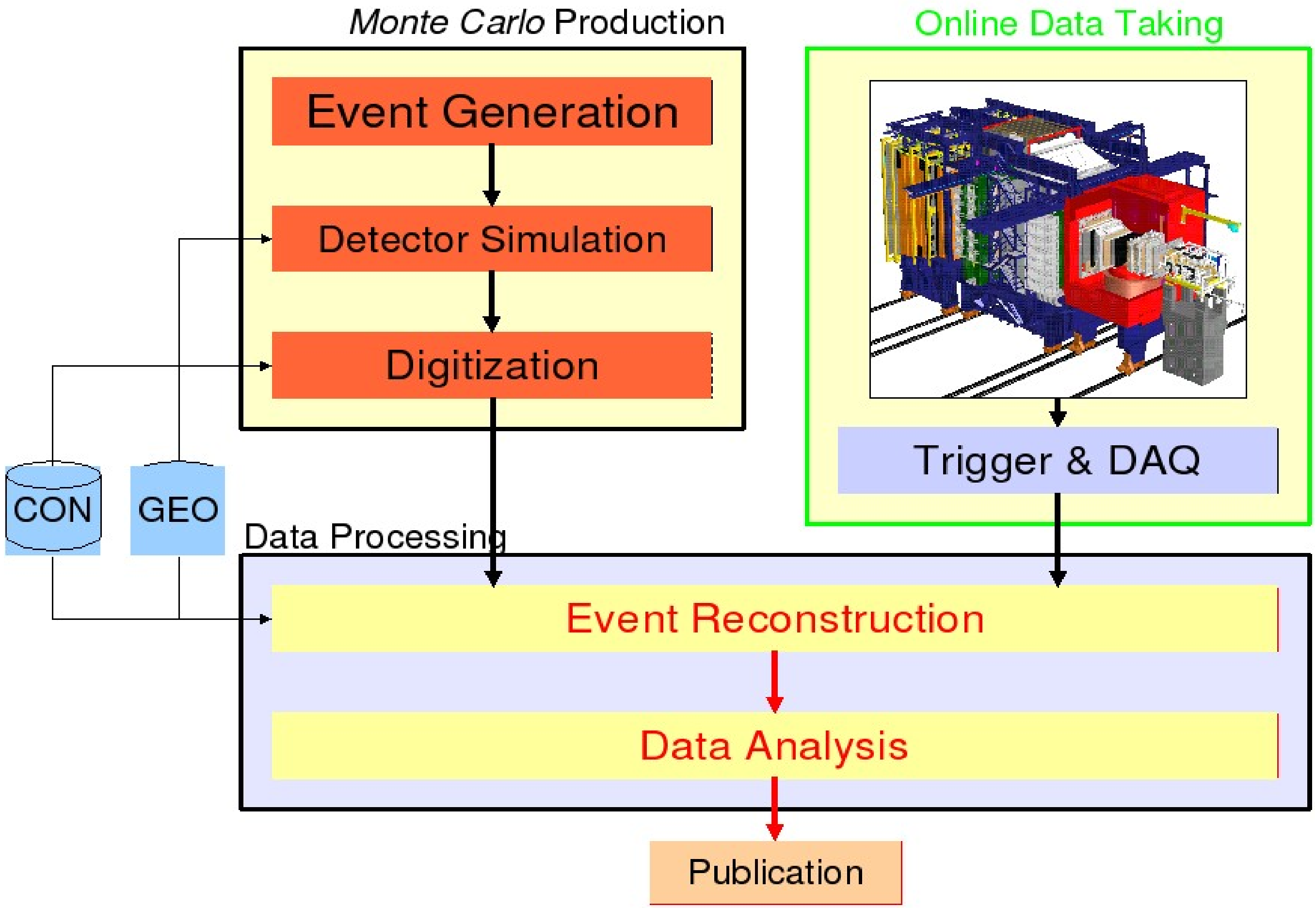
- requirements:
  - grid certificate
  - VO membership
  - all files (input, binary, libraries,…) on SE
  - jobscript (JDL) that:
    - retrieves all needed files from SE onto WN
    - sets custom environment
    - executes binary/script
    - stores output on SE

- submission:
- start your proxy (secured interface)
- put all job depended files on SE
- write your JDL script specifying:
  - name of jobscript
  - arguments
  - input/output Sandbox
  - VO
- edg-job-submit *your-jdl-file*
- check status via job-ID

Grid schematic view



'summary'



Questions ?

thanks to A.Gellrich for Grid material