Centre of Excellence MIUR - Università di Padova

"Science and Applications of Advanced Computing Paradigms"

Workshop

Aula Rostagni, Dipartimento di Fisica "G. Galilei", Padova,

May, 16^{th} 2003

The LHC data simulation production performed on the processors farm of the Padua Centre of Excellence for CMS HLT studies

Stefano Lacaprara, I.N.F.N. and Padova University

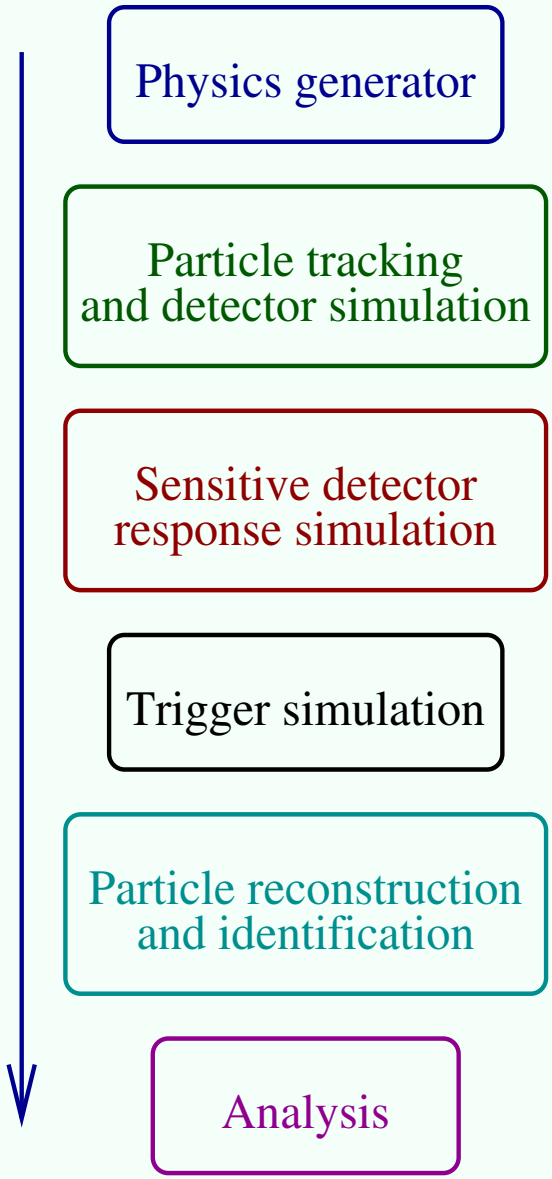
Outline

- Introduction,
- Data simulation steps,
- CMS production data flow,
- Farm architecture,
- Perspective for DC04,

Introduction

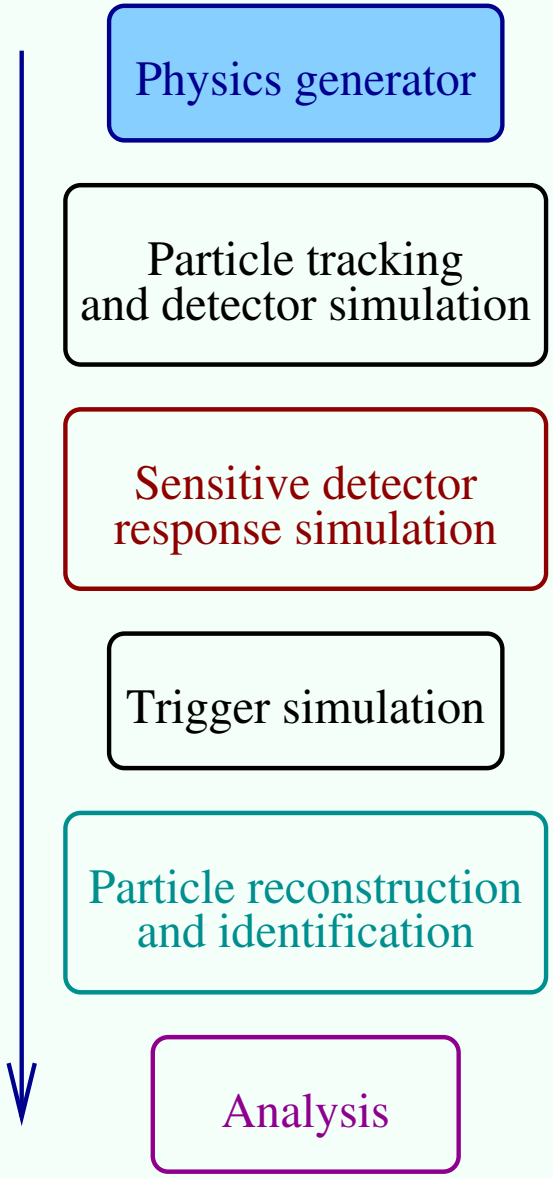
- Goal of data production is to simulate CMS detector physics performance, development of reconstruction, analysis software
- Develop CMS computing and analysis model
- **Critical issue for success of CMS**: collaboration is worldwide, need to develop know-how and skills to allow optimal usage of computing resources in a distributed environment
- Data simulation production is worldwide: need tight syncronization, organization, coordination and data access
- Data production is “easy” (just one master), data analysis much harder, given the *chaotic* pattern
- Developing the former gives invaluable experience on how to deal with the latter

Data Simulation Steps



- ◇ To get final result (as shown by Ugo's talk) many following steps are needed
- ◇ First simulation steps, then reconstruction, identification, then analysis
- ◇ The goal of first steps is to simulate, with high accuracy, what will be the real physics outcome of true CMS detector
- ◇ Each step provides the output for the following one
- ◇ Each step make different use of computing resources (I/O, cpu, storage . . .), so computing facilities must be flexible to get good efficiency for all of them
- ◇ Many different output/input formats

Physics Generator



- ◇ HEP library used Pythia
- ◇ Simulate particle interaction (pp for CMS @ LHC)
- ◇ "Knows" kinematic and dynamic of $p - p$ collision
- ◇ Produce as output the 4-momentum of particles produced in primary interaction
- ◇ Many of processes and final states
- ◇ **Input "cards"**: user parameters for simulation (pp energy, process and final state selection, physics model parameters, . . .)
- ◇ **Output is small** `HBOOK` **ntuple,** `ZEBRA` **format**: a standard for HEP community for ~ 30 years
- ◇ CPU very process dependent: $\mathcal{O}(s) \div \mathcal{O}(100s)$/event on PIII $1\,GHz$ cpu
- ◇ Code is `FORTRAN`

Detector Simulation

Physics generator

Particle tracking and detector simulation

Sensitive detector response simulation

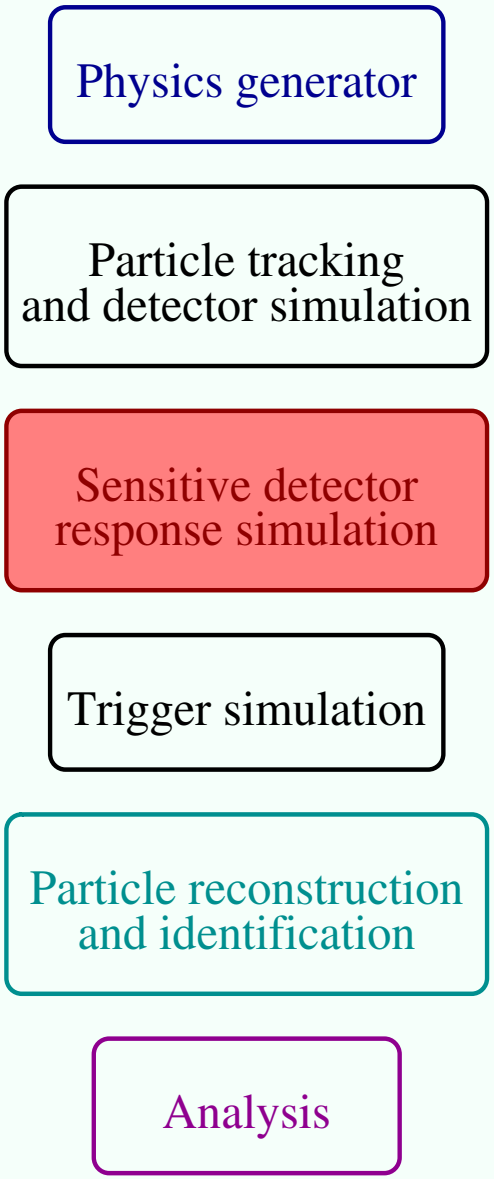
Trigger simulation

Particle reconstruction and identification

Analysis

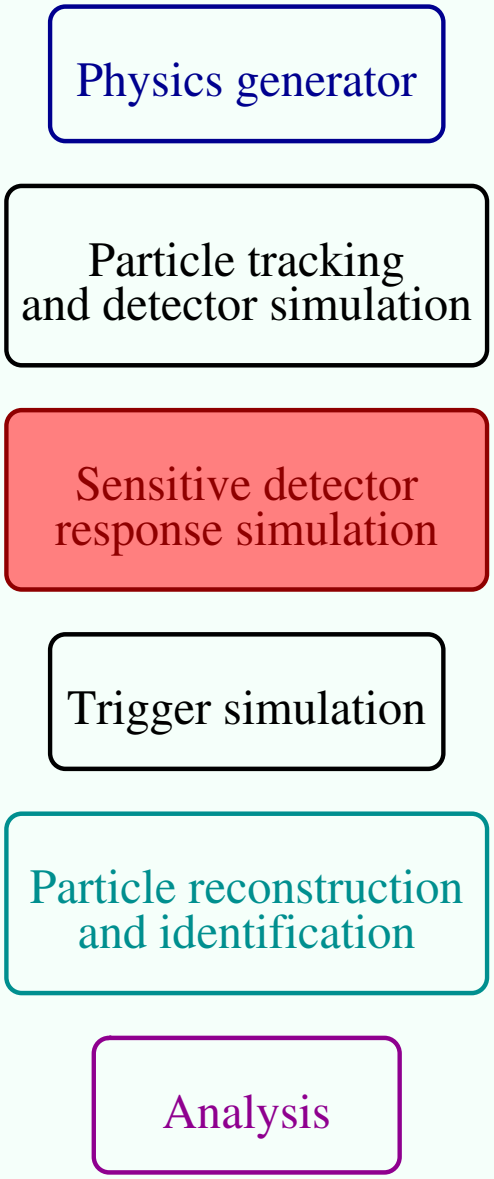
- ◇ CMSIM using GEANT3 libs (HEP)
- ◇ Tracks of all particle produced by primary interaction in CMS detector
- ◇ $\vec{B}$ field, interaction with material (energy loss, multiple scattering, . . .), particle decays, . . .
- ◇ Produce as output a complex tree of all particles tracks
- ◇ For sensitive volumes produce SimHits containing info about entry-exit point, energy loss, ToF, . . .
- ◇ **Input Pythia ntuple** and user *cards*
- ◇ **Output is** `FZ` **file,** `ZEBRA` **format**: single file, size $0.5 \div 1.5\, GB$
- ◇ CPU time long! $\sim 200\ s/ev$ on PIII $1\ GHz$ cpu

Detector Response



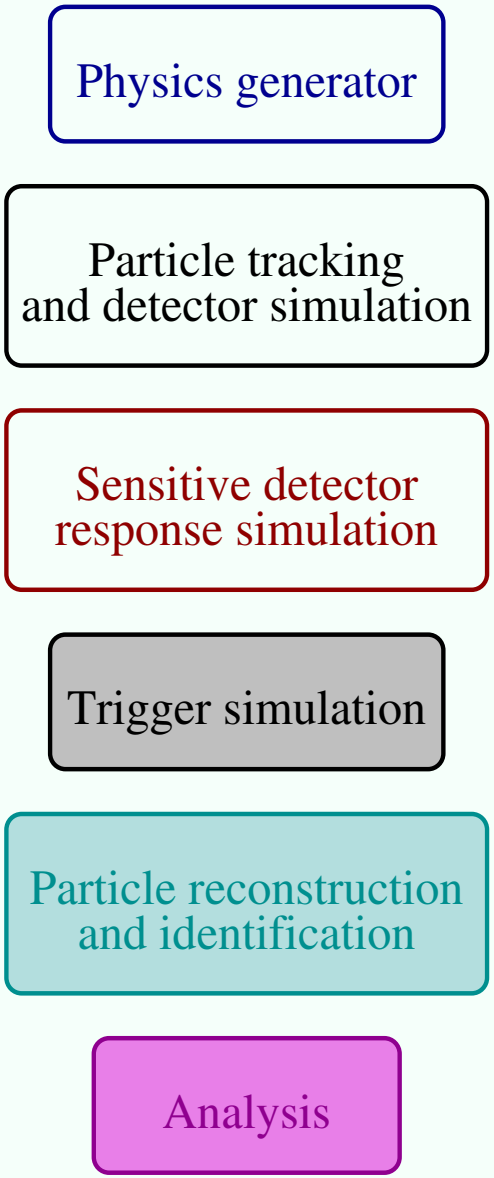
- ORCA full custom
- From SimHits simulate electronic response of real detectors to passing particles
- **Input FZ file** and user parameters
- Need to include suitable number of "Pile Up" events, for complete simulation
- PU events produced independently with same chain
- Must add $\langle n_{PU} \rangle = 3.4 \div 17.3$ (different LHC running condition) for each signal event
- Choose randomly PU events from same big sample for all signal events
- Need efficient random access to PU collection: FZ access sequential

Detector Response (2)



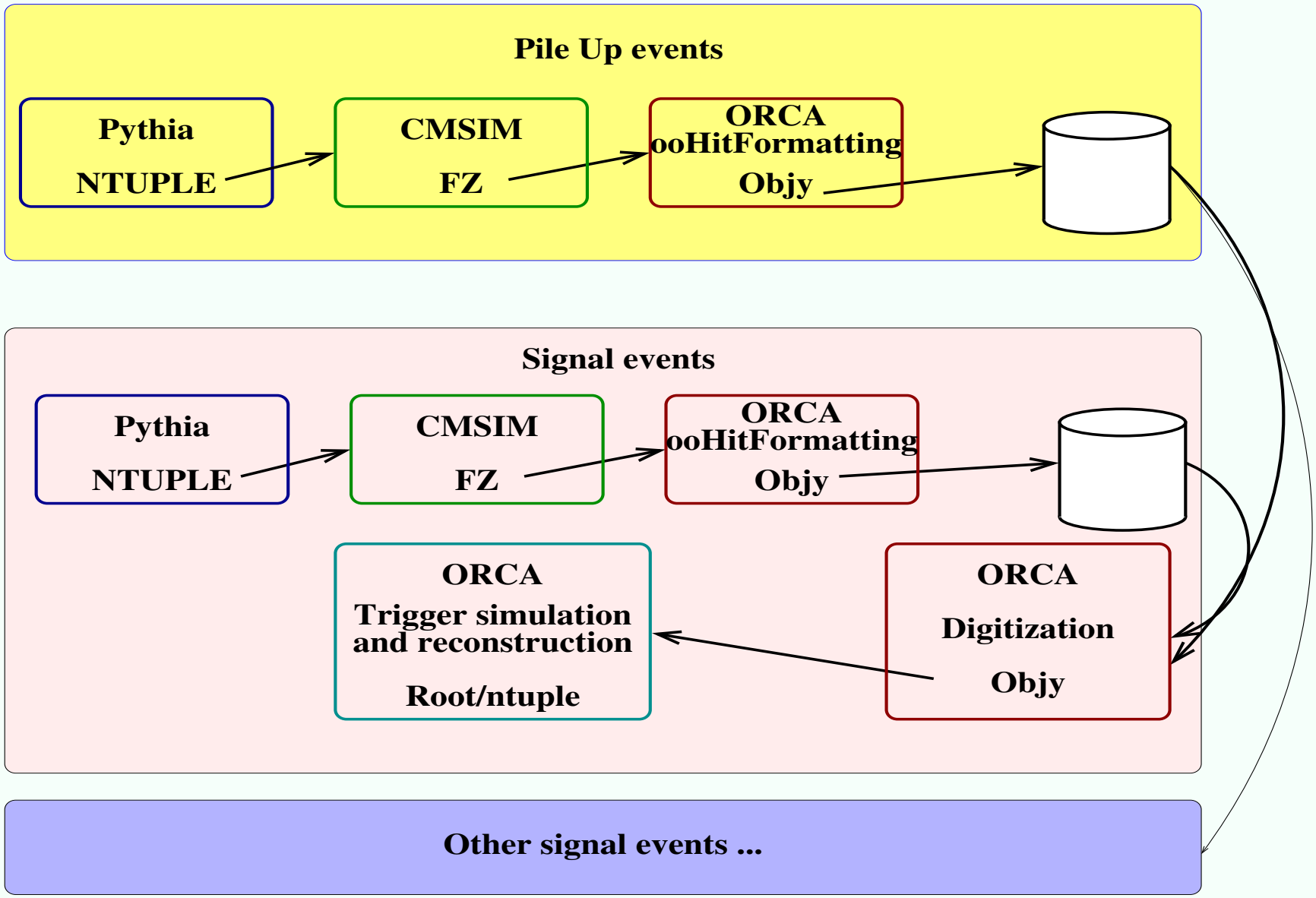
- ◇ Conversion of FZ-SimHits in C++ objects
- ◇ New persistency is based on `ODBMS`: **Objectivity**
- ◇ Process very fast $\mathcal{O}(s/ev)$, high I/O: ~ 2 MB/ev
- ◇ Create "collection" of PU events: just pointers to real events data
- ◇ *ooHitFormat* both signal and PileUp events
- ◇ For each signal event, pick-up randomly *n* events from PU collection, and get SimHits for signal and PU
- ◇ Superimpose SimHits from signal and *n* PU and simulate sensitive detector response
- ◇ Create Digitized response of each sub-detector: electronic signal `DIGI` output
- ◇ Correspond to RawData as will come from CMS DAQ

Reconstruction and Analysis

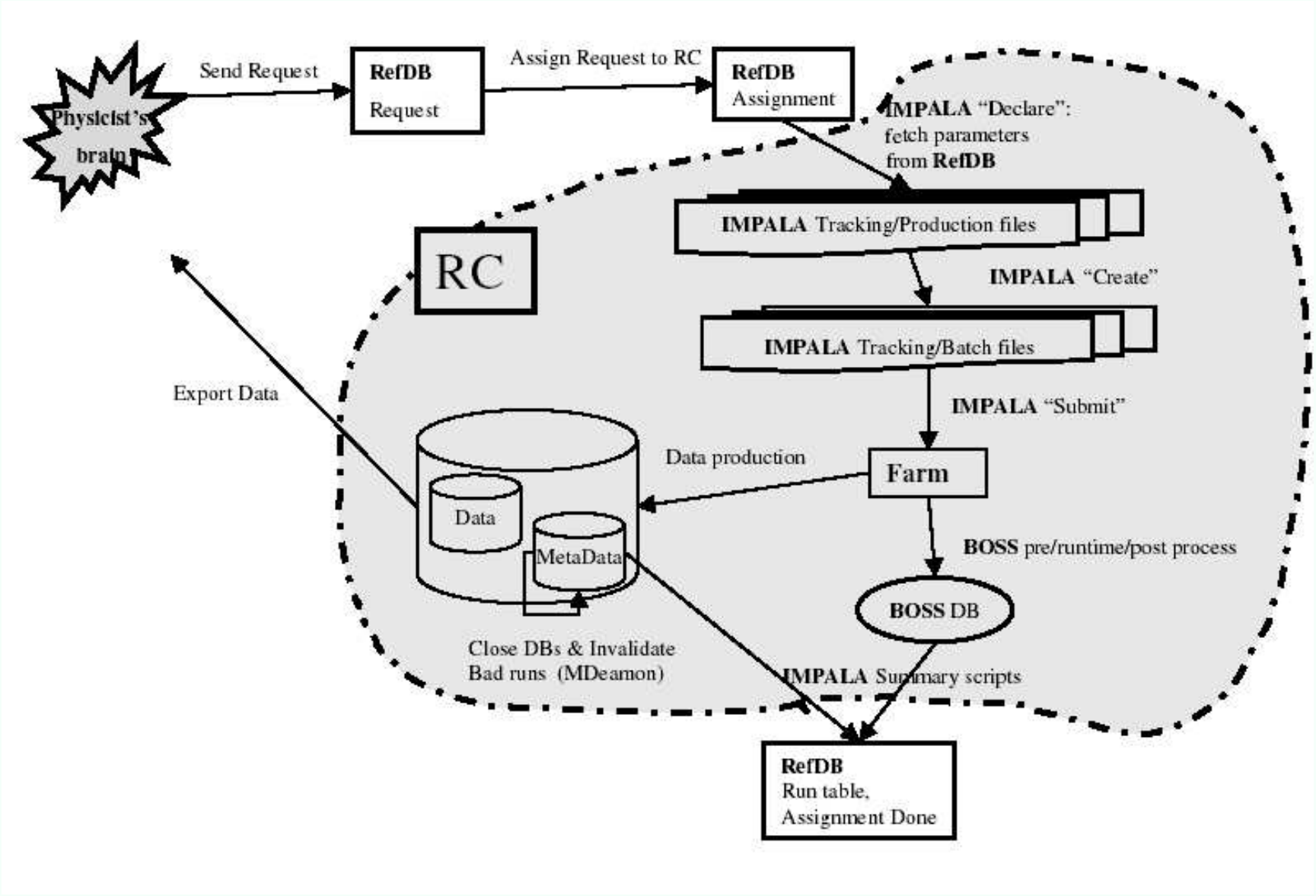


- Last step: Trigger simulation and reconstruction
- From RawData to physics objects: muon, electron, jets, photons, ...
- Repeated many times on small sub-sample to develop and test algorithm
- Reconstruction on all samples done once algorithm optimized (some iteration)
- **Input previously produced Digis**
- **Output format still not finalized**, many used by different groups/users (`root/hbook/...`): much smaller than RawData, just physical objects
- Analysis performed on reduced format: repeated many-many times by many users
- Very final result: pysical plots (such as Ugo's)

Data Flow


Pile Up events
Pythia
NTUPLE
CMSIM
FZ
ORCA
ooHitFormatting
Objy
Signal events
Pythia
NTUPLE
CMSIM
FZ
ORCA
ooHitFormatting
Objy
ORCA
Trigger simulation
and reconstruction
Root/ntuple
ORCA
Digitization
Objy
Other signal events ...

Information Flow



CMS Coordinated Production

- Production is coordinated worldwide by CMS from CERN
- In $2002 \sim 10$ reginal center (US, INFN, Russia, CERN, UK, ...)
- Some regional center subdivided in many sub-center (as INFN: Padova, Legnaro, Bologna, ...)
- Deployment of common tools for job creation, information retrieving from central DB, code distribution and versioning , bookkeeping, . . .
- Schematic information and data flow:
  - ◇ Physicist ask for a given “DataSet”, define input parameter, # events, . . .
  - ◇ DataSet is assigned to a given RC
  - ◇ RC executes the steps of the production (complete or partial)
  - ◇ Jobs runs on local farm, continuous local monitoring, asynchronous update of central bookkeeping at CERN
  - ◇ If partial production, data shipped back at CERN or other RC
  - ◇ Once completed, data shipped at CERN, (on RC for local analysis)

Padova Farm Configuration

Padova Farm pretty complex and heterogeneous

- ⋆ 8 dual PIII, 1.3GHz, 1Gb RAM, 1U, FastEth (CdE)
- ⋆ 10 dual PIII, 1GHz, 512Mb RAM, 2U, FastEth
- ⋆ 5 dual PIII, 650MHz, 512Mb RAM, desktop, FaseEth
- ⋆ 7 single PIII, 450MHz, 256Mb RAM, desktop, FaseEth (old)
- ⋆ (6 dual Xeon, 2.8GHz, 2Gb RAM, 1U, GigaEth (new))

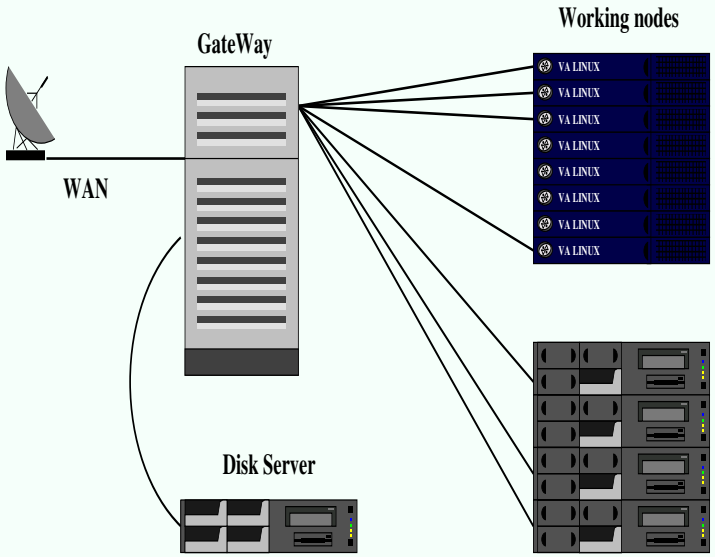
- ◇ Disk Server, $\sim 1\,TB$ `RAID 5`, GigaEth (CdE)
- ◇ Local disk on working nodes: total $\sim 1\,TB$ distributed

- Tape library (DLT) for backup and mass storage
- Switch GigaEth and FastEth

- ▶ OS Linux RedHat 6.3.1(CERN), kernel 2.2.19-6.2.1.1smp
- ▶ Batch management system `LSF`

Farm Configuration

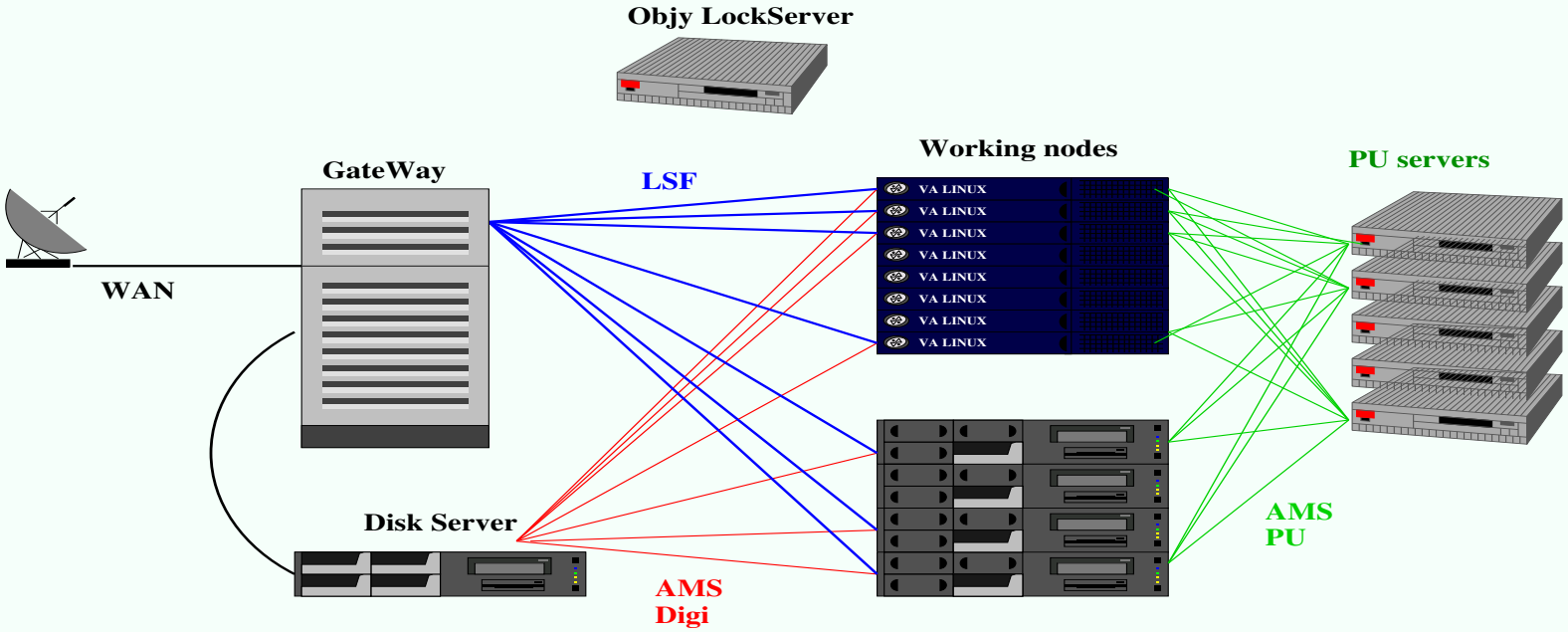
- Configuration is flexible for different steps
- For first three steps (Pythia, CMSIM, ooHitFormatting) "democratic" farm:



  - ⋆ Server acting as GateWay: user account, source code distribution (`nfs`), job submission, WAN access, monitoring, ...
  - ⋆ Other box running as worker nodes
  - ⋆ Jobs submission from GW to WNs via LSF batch system
  - ⋆ Output for NTUPLE/FZ on WNs local disk
- ⋆ for ooHitFormatting, read local Fz, write remotely Objy DB on DiskServer via `AMS` (Objectivity daemon for remote I/O)
- ⋆ ooFormatting has very high I/O, not run on all farm to not overload AMS/DiskServer (anyway fast process, so efficiency not crucial)
- ⋆ SimHits DB for PU produced centrally for all RCs, and then shipped

Farm Configuration Digitization

- Digitization stress the farm for both CPU and I/O
- CPU time $\sim 10\ s/ev$
  - need up to ~ 20 PU events, each $\sim 350kB$:
  - total I/O $\sim 0.7\ MB/s/CPU$
  - ~ 40 CPU running: PU I/O $\sim 30MB/s$ on LAN via AMS
- Note: almost pure random access!!! very critical for disk performance
- Reconfigure farm for optimal performance: CPU usage and I/O
- Use large LSF flexibility: configure different queues for Pythia/CMSIM, ooHitFormatting and Digitization
- Different configuration coexist at the same time: no hardware modification



- ⋆ PU DB splitted in several files (all attached to same Objy DB)
- ⋆ Distribute files among different dedicated nodes (PU servers)
- ⋆ Given random access to PU events, load shared among the PU servers
- ⋆ Node dedicated to Objectivity LockServer, and Journal server
- ⋆ Achieved best farm efficiency with 1 PU server each ~ 7 CPUs
- ⋆ Output (still Objy DB) written on DiskServer, not I/O demanding

Reconstruction and Analysis

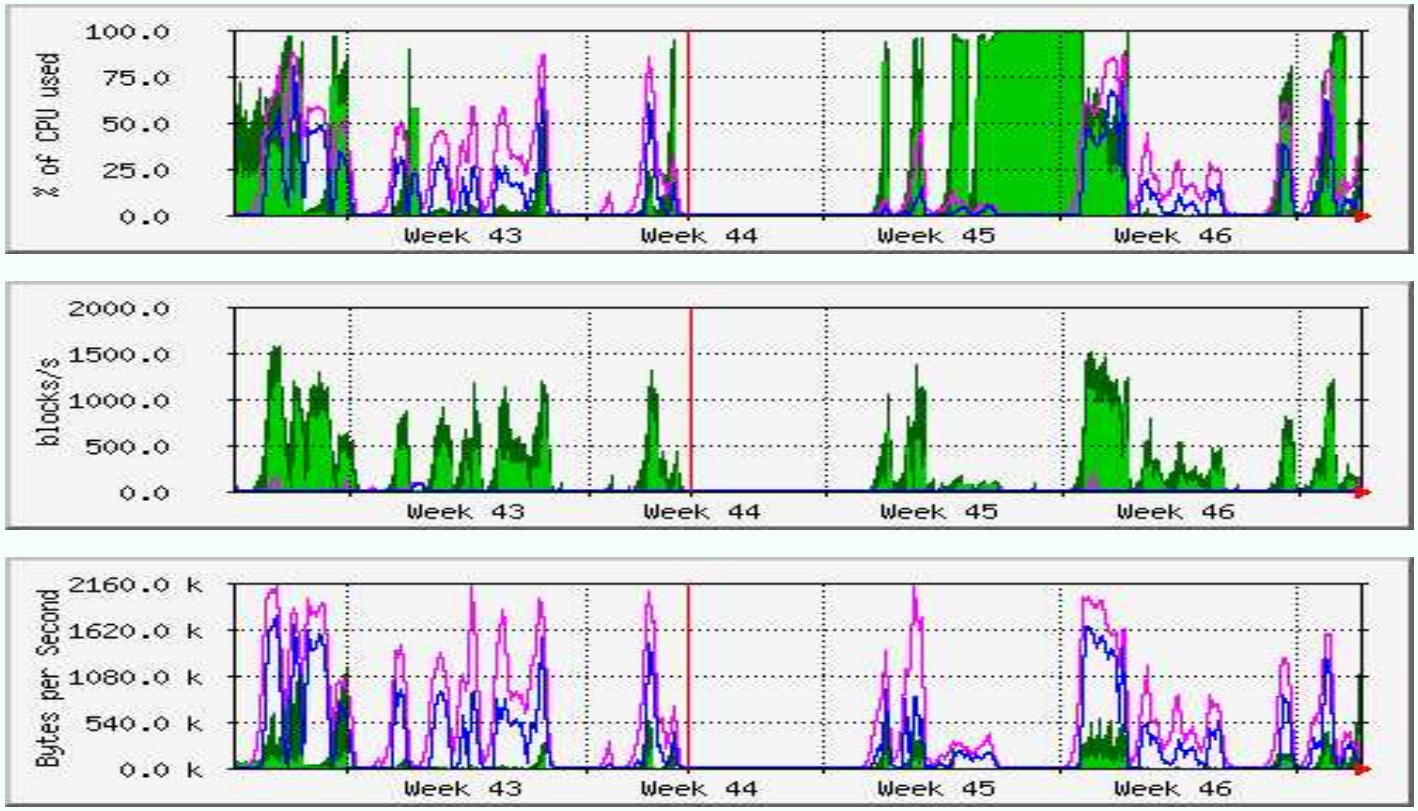
- Reconstruction under heavy development: one of major task of 2002 production!
- Many iteration on small sample to develop, test, debug and refine algorithm
- Chaotic data access from different user
- One “developing” box where user develops its algorithm , and runs small reconstruction jobs (interactively)
- Bigger samples processed using LSF: resources (fair) sharing via LSF functionality
- Once algorithm defined, process all datasets in a coordinate way, as for “production”
- Distribution of reconstruction code (based on certified central –CERN– software releases) among RC
- Final distribution (via CERN) of produced data

Data Produced

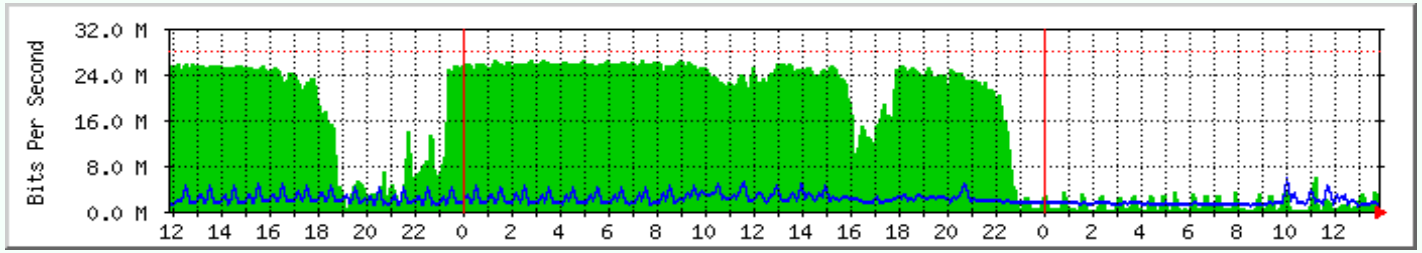
- During 2002 CMS has produced ~ 6 M events worldwide,
- plus additional ~ 1 M *private or semiprivate* for algorithm development
- In `Padova`:
  - ~ 0.3 M *official* production
  - ~ 0.3 M *semiprivate* production (access allowed to many CMS user)
- Disk usage:
  - ⋆ Diskserver: used ~ 0.5 TB (50%)
  - ⋆ Local Disks: used ~ 0.3 TB, (35%)

Farm Monitoring

Example of CPU Disk I/O and traffic on a PU server during “chaotic” data access



Example of data access through WAN to fetch produced data to CERN



Future Perspective for DC04

- End of 2002 major change on CMS software
- Baseline for persistency: Objectivity dropped, new model is mized
- C++ object I/O streaming using `ROOT`, HEP tool developed at CERN: able to write/read C++ object on binary files
- New tool `POOL` to manage file catalog and remote access
- Change almost completed, still waiting for "production" POOL software
- Remote file access via RFIO (CERN tool): alternative under study dChace (DESY) and *pnfs*
- New package for detector simulation: `OSCAR` based on Geant4, fully OO-C++: will use the same persistency model as following steps
- Develop new/improved tool for code distribution, production parameter retrieving, local and central bookkeeping, monitoring, ...
- First *large scale* test of GRID tools for either data production and, more complex, data access for user analysis