



TOOLS FOR HIGH ENERGY PHYSICS

Gabriel Stoicea
Particle Physics Department
National Institute of Physics and Nuclear Engineering
IFIN-HH
Second High Energy Physics School in Măgurele
Măgurele-Bucharest, 22-23.10.2009

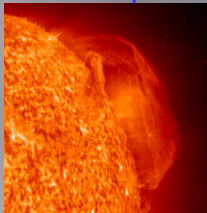
Outline

- Particle Physics
- Particle Physics Challenges
- Main Tools and Packages used in HEP offline software
- Grid Tool
- New/Old Tools
- Summary

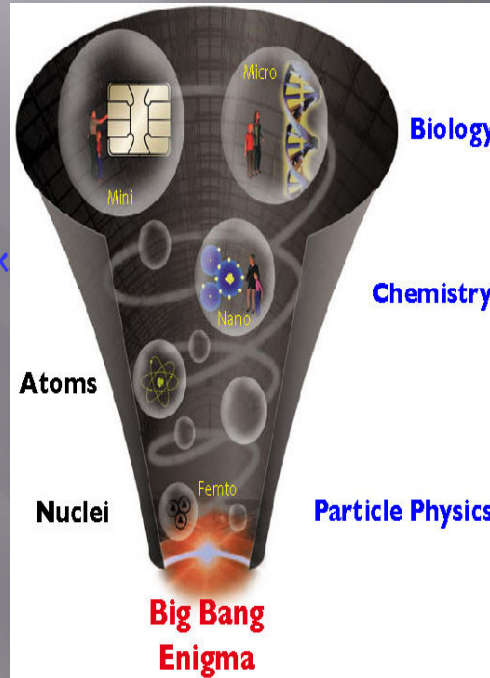
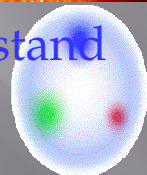
Particle Physics

	Quarks		Leptons	
Generation 3	t Top	b Bottom	τ Tau	ν_τ Tau-neutrino
Generation 2	c Charm	s Strange	μ Muon	ν_μ Muon-neutrino
Generation 1	u Up	d Down	e Electron	ν_e Electron-neutrino

Establish a periodic system of the fundamental building block

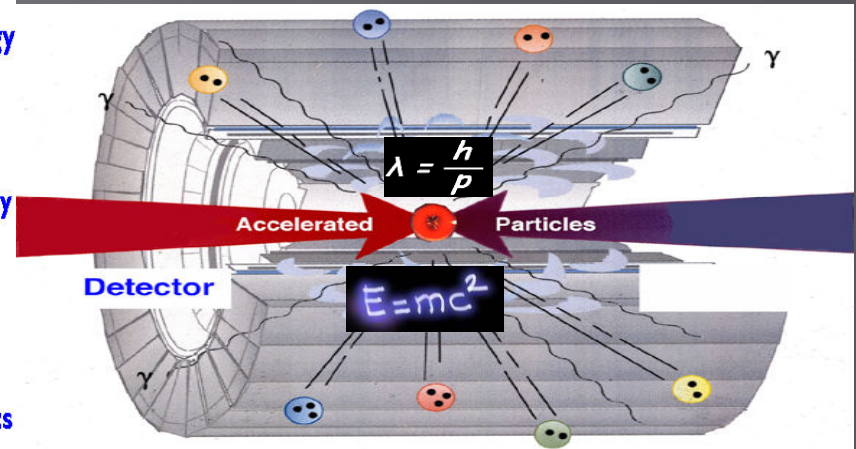


and understand forces



Methods of Particle Physics

The most powerful microscope

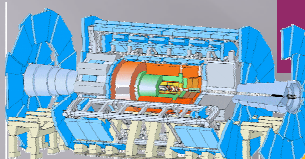


Creating conditions similar to the Big Bang


Particle Physics Challenges

Challenge 1: Large, distributed community

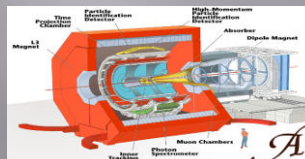
ATLAS



“Offline” software effort:
1000 person-years
per experiment



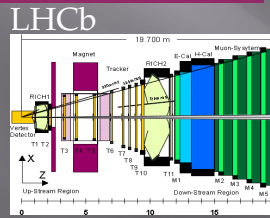
~ 5000 Physicists
around the world
- around the clock



ALICE



Software
life span:
20 years



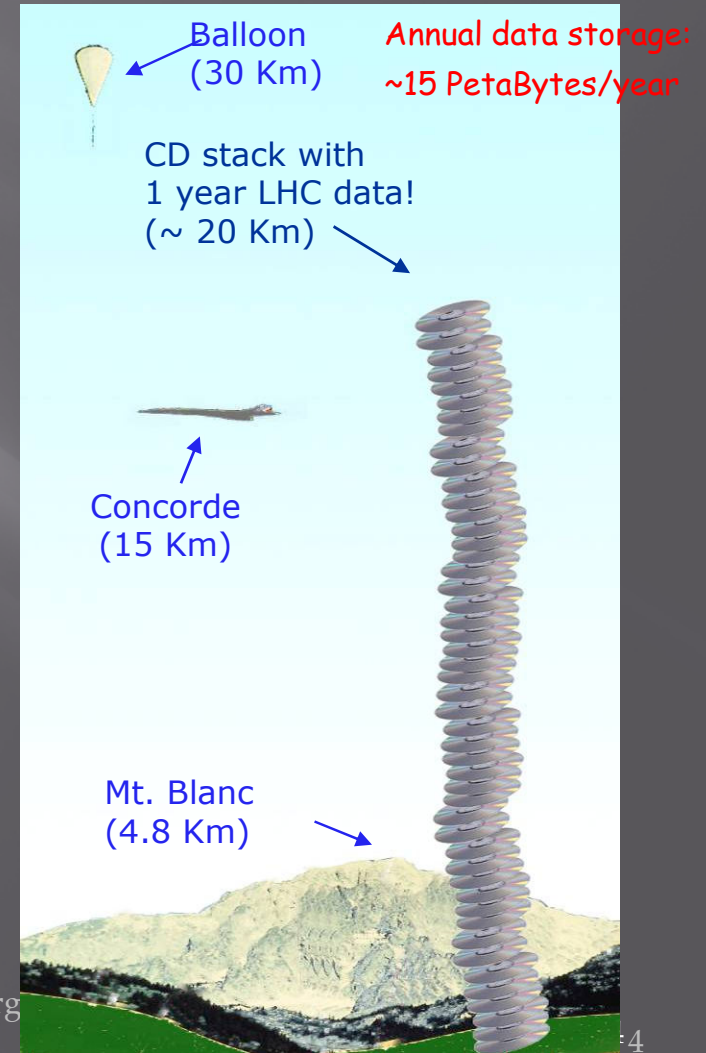
Challenge 2: Data Volume

Balloon (30 Km) Annual data storage:
~15 PetaBytes/year

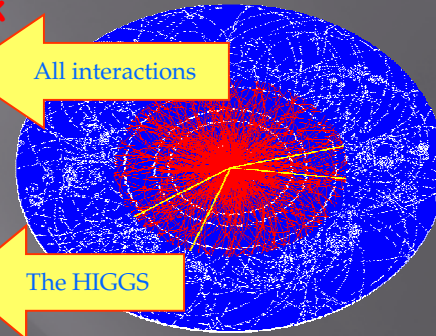
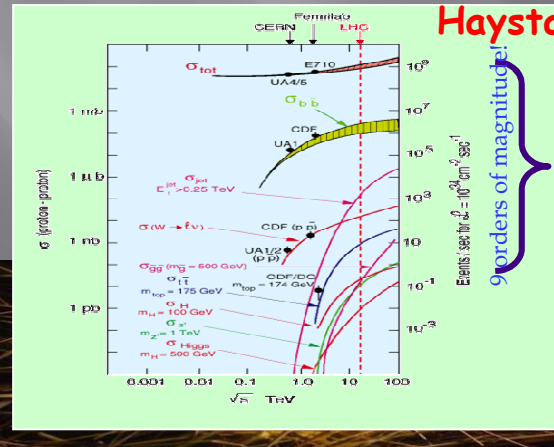
CD stack with
1 year LHC data!
(~ 20 Km)

Concorde (15 Km)

Mt. Blanc (4.8 Km)



Challenge 3: Find the Needle in a Haystack



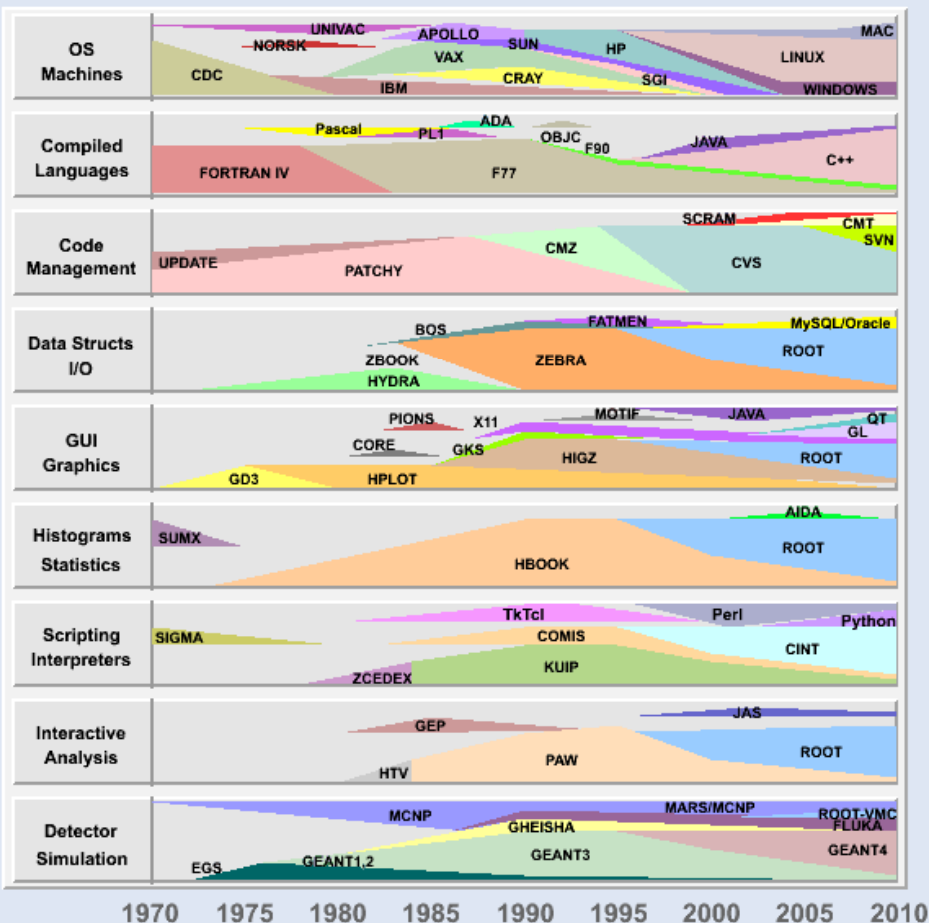
Rare phenomena - Huge background
Complex events

Gabriel Stoicea, Second High Energy Physics School in Magurele

Tools and Packages used in HEP

A Compilation of the main Tools and Packages used in HENP offline software since 1970

(Rene.Brun@cern.ch)



This compilation includes systems used by at least 4 experiments.

The Y scale indicates the relative importance of the system with time.

26.10.2009

(The author welcomes comments and additions)

- Starting with 1995 Linux was adopted like "de facto" OS for HEP - it comes in different flavors but in generally in our days most common is Scientific Linux (developed by Fermilab and CERN) which is a clone of RHEL (RedHat Enterprise Linux)

- In terms of programming languages most common is C++; Fortran77 is still in use powerful and free compilers provided by GNU included in Linux (gcc, g++ and gfortran/g77)

- Offline Software packages developed around Fortran and C++ languages:

- PAW (Physics Analysis Workstation) the old framework developed in Fortran and some C
- ROOT - An Object-Oriented Data Analysis Framework - developed in C++
- Detector Simulation packages: GEANT3 (Fortran77) and GEANT4 (C++)

Tools and Packages used in HEP

PAW Framework

PAW was conceived as an instrument to assist physicists in the analysis and presentation of their data. It provides interactive graphical presentation and statistical or mathematical analysis, working on objects familiar to physicists like histograms, event files (Ntuples), vectors, etc. PAW is based on several components of the CERN Program Library. Like CERN Program Library PAW usage and/or redistribution is granted under the terms of the GNU General Public License.

The CERN Program Library consists of a number of independent Fortran and C callable libraries and a collection of complete programs. The overall structure is described briefly below:

- KERNLIB - basic utility subroutines and functions
- MATHLIB - mathematical routines, including random number generators
- PACKLIB - the package library, including HEPDB, HBOOK, ZEBRA etc.
- GRAFLIBS - the graphics libraries, including HIGZ, HPLOT etc.
- PAWLIB - the PAW library, including COMIS, PAW and SIGMA
- Monte Carlo libraries - various Monte Carlo generators, usually as individual libraries
- Modules - such as PAW, FATMEN, etc.

GEANT3 - Detector Description and Simulation Tool. The principal applications of GEANT in High Energy Physics are: the tracking of particles through an experimental setup for simulation of detector response and the graphical representation of the setup and of the particle trajectories.

Tools and Packages used in HEP

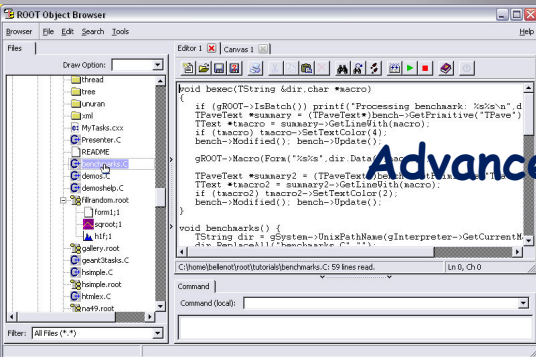
ROOT Framework

ROOT is a framework for data processing, born at CERN, at the heart of the research on high-energy physics. Every day, thousands of physicists use ROOT applications to analyze their data or to perform simulations.

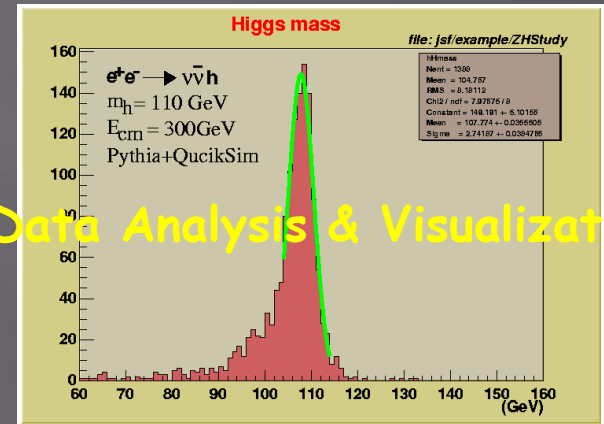
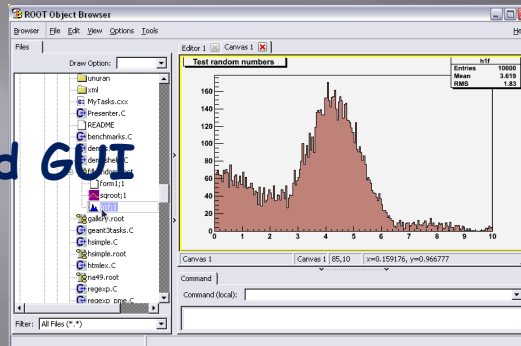
A quick overview of the ROOT framework ("Gentle introduction") is summarized below:

- **Save data.** You can save your data (and any C++ object) in a compressed binary form in a ROOT file. The object format is also saved in the same file. ROOT provides a data structure that is extremely powerful for fast access of huge amounts of data - orders of magnitude faster than any database.
- **Access data.** Data saved into one or several ROOT files can be accessed from your PC, from the web and from large-scale file delivery systems used e.g. in the GRID. ROOT trees spread over several files can be chained and accessed as a unique object, allowing for loops over huge amounts of data.
- **Process data.** Powerful mathematical and statistical tools are provided to operate on your data. The full power of a C++ application and of parallel processing is available for any kind of data manipulation. Data can also be generated following any statistical distribution, making it possible to simulate complex systems.
- **Show results.** Results are best shown with histograms, scatter plots, fitting functions, etc. ROOT graphics may be adjusted real-time by few mouse clicks. High-quality plots can be saved in PDF or other format.
- **Interactive or built application.** You can use the CINT C++ interpreter or Python for your interactive sessions and to write macros, or compile your program to run at full speed. In both cases, you can also create a GUI.

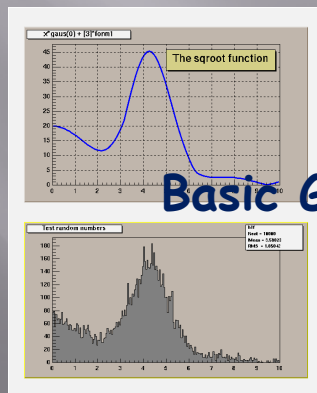
Tools and Packages used in HEP ROOT Framework



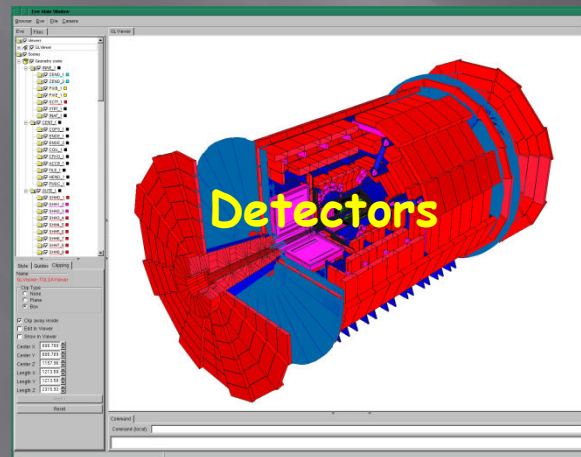
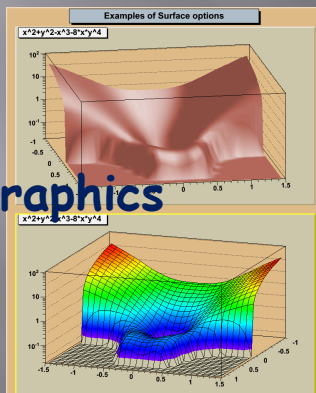
Advanced GUI



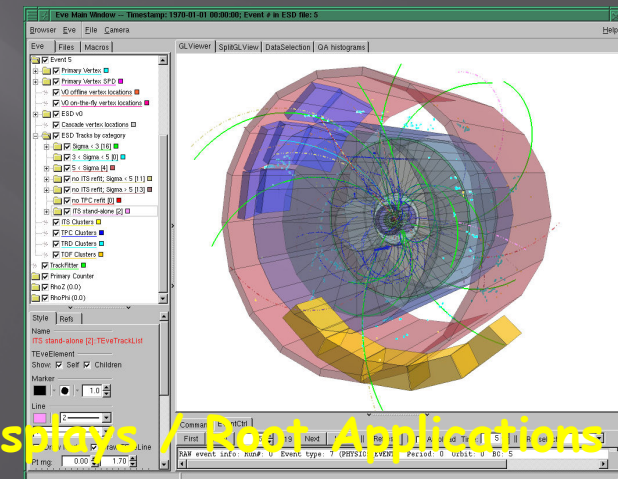
Data Analysis & Visualization



Basic Graphics



Detectors



Events Displays / Root Applications

Grid Computing

- ▣ Grid computing is a form of distributed computing whereby a "super and virtual computer" is composed of a cluster of networked, loosely-coupled computers, acting in concert to perform very large tasks.
- ▣ **WLCG (Worldwide LHC Computing Grid)**



What is the Grid? & How will it work?

- Resource Sharing
 - On a global scale, across the labs/universities
- Secure Access
 - Needs a high level of trust
- Resource Use
 - Load balancing, making most efficient use
- The "Death of Distance"
 - Requires excellent networking
- Open Standards
 - Allow constructive distributed development
- There is not (yet) a single Grid

The GRID middleware:

- Finds convenient places for the scientists "job" (computing task) to be run
- Optimises use of the widely dispersed resources
- Organises efficient access to scientific data
- Deals with authentication to the different sites that the scientists will be using
- Interfaces to local site authorisation and resource allocation policies
- Runs the jobs
- Monitors progress
- Recovers from problems

... and

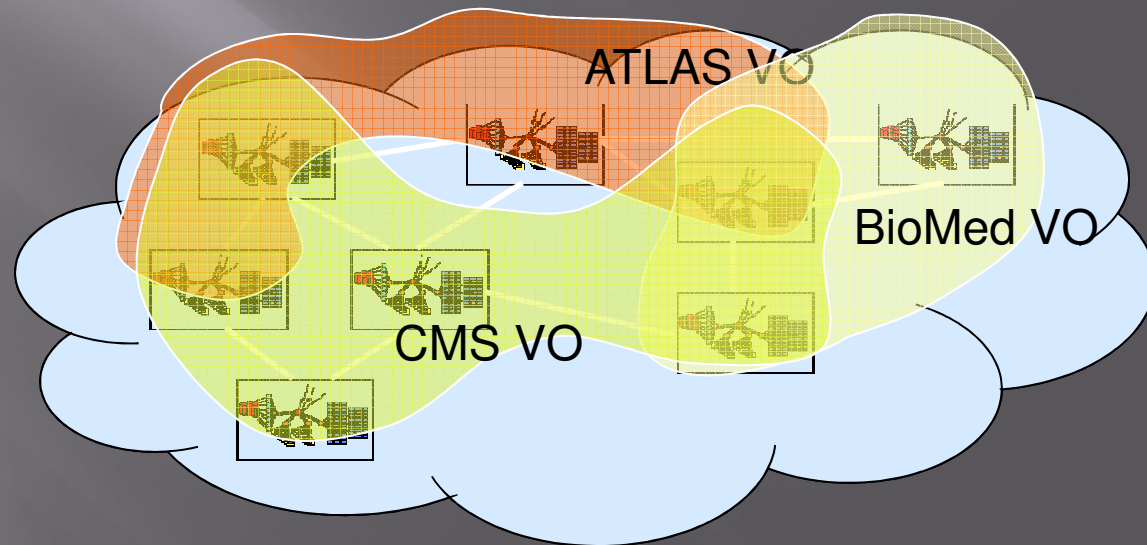
Tells you when the work is complete and transfers the result back!

Grid middleware

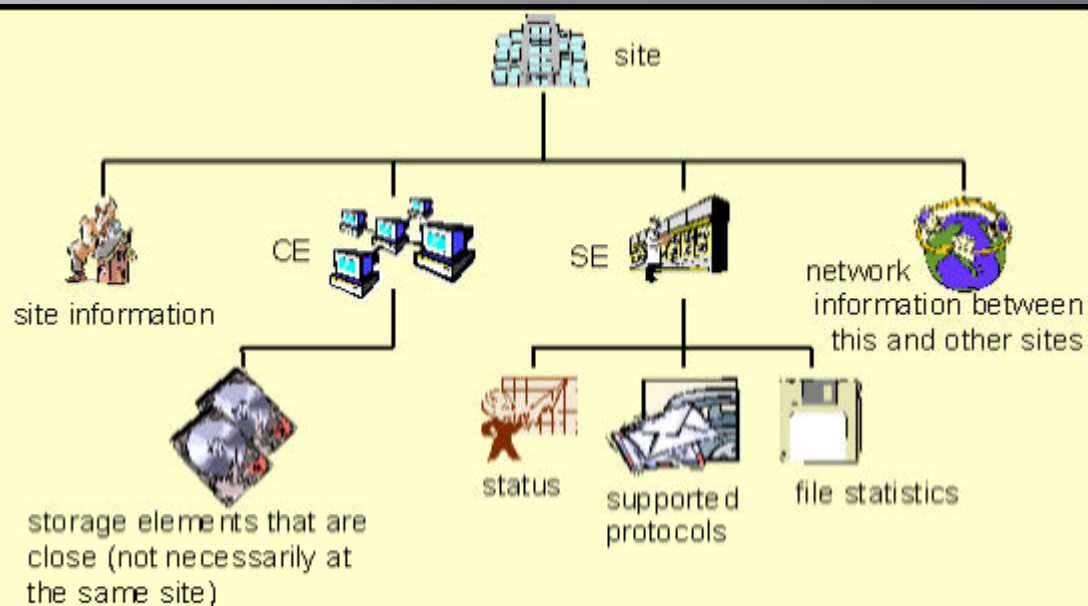
The gLite middleware comes from a number of Grid projects, like DataGrid, DataTag, Globus, GriPhyN, iVDGL, EGEE and LCG. This middleware is currently installed in sites participating in WLCG/EGEE.

In WLCG other Grid infrastructures exist, namely the *Open Science Grid (OSG)*, which uses the middleware distributed by VDT, and NorduGrid, which uses the ARC middleware.

The users of a Grid infrastructure are divided into *Virtual Organisations (VOs)*, abstract entities grouping users, institutions and resources in the same administrative domain.



Grid middleware



The **Information Service (IS)** provides information about the WLCG/EGEE Grid resources and their status. This information is essential for the operation of the whole Grid, as it is via the IS that resources are discovered. The published information is also used for monitoring and accounting purposes.

Data Management - The primary unit for Grid data management, as in traditional computing, is the *file*. In a Grid environment, files can have *replicas at many different sites*. Because all replicas must be consistent, Grid files cannot be modified after creation, only read and deleted. Ideally, users do not need to know where a file is located, as they use logical names for the files that the Data Management services use to locate and access them. Files in the Grid can be referred to by different names: *Grid Unique Identifier (GUID)*, *Logical File Name (LFN)*, *Storage URL (SURL)* and *Transport URL (TURL)*. The mappings between LFNs, GUIDs and SURLs are kept in a service called a *File Catalogue*, while the files themselves are stored in Storage Elements. Currently, the only file catalogue officially supported in WLCG/EGEE is the *LCG File Catalogue (LFC)*, although other catalogues exist.

Computing Element (CE), in Grid terminology, is some set of computing resources localized at a site (i.e. a cluster, a computing farm). A CE includes a *Grid Gate (GG)*, which acts as a generic interface to the cluster; a *Local Resource Management System (LRMS)* (sometimes called *batch system*), and the cluster itself, a collection of *Worker Nodes (WNs)*, the nodes where the jobs are run.

Storage Element (SE) provides uniform access to data storage resources. The Storage Element may control simple disk servers, large disk arrays or tape-based *Mass Storage Systems (MSS)*. Most WLCG/EGEE sites provide at least one SE. Most storage resources are managed by a *Storage Resource Manager (SRM)*, a middleware service providing capabilities like transparent file migration from disk to tape, file pinning, space reservation, etc.

Grid middleware

The access point to the WLCG/EGEE Grid is the **User Interface (UI)**. This can be any machine where users have a personal account and where their user certificate is installed. From a UI, a user can be authenticated and authorized to use the WLCG/EGEE resources, and can access the functionalities offered by the Information, Workload and Data management systems. It provides CLI tools to perform some basic Grid operations:

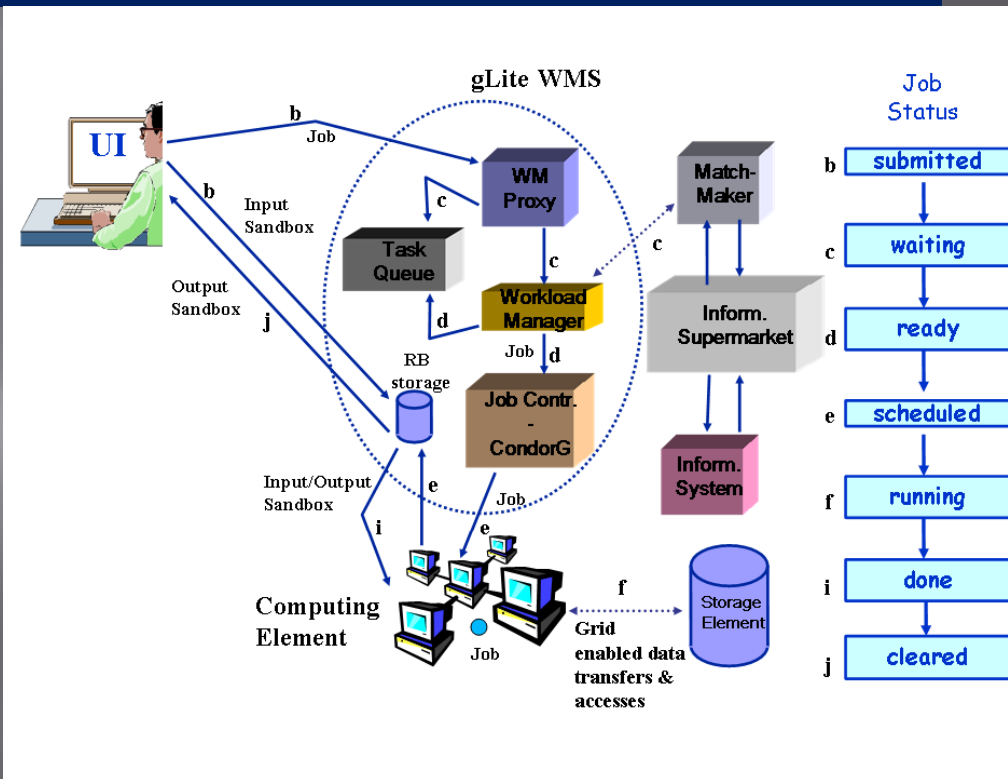
- list all the resources suitable to execute a given job;
- submit jobs for execution;
- cancel jobs;
- retrieve the output of finished jobs;
- show the status of submitted jobs;
- retrieve the logging and bookkeeping information of jobs;
- copy, replicate and delete files from the Grid;
- retrieve the status of different resources from the Information System.

In addition, the WLCG/EGEE APIs are also available on the UI to allow development of Grid-enabled applications.

The purpose of the **Workload Management System (WMS)** is to accept user jobs, to assign them to the most appropriate Computing Element, to record their status and retrieve their output. The **Resource Broker (RB)** is the machine where the WMS services run.

Jobs to be submitted are described using the **Job Description Language (JDL)**, which specifies, for example, which executable to run and its parameters, files to be moved to and from the Worker Node on which the job is run, input Grid files needed, and any requirements on the CE and the Worker Node. **Logging and Bookkeeping service (LB)** tracks jobs managed by the WMS. It collects events from many WMS components and records the status and history of the job.

File Transfer Service (FTS) provides a managed way to move large numbers of files between SEs.



The Worldwide LHC Computing Grid Project - WLCG

Collaboration

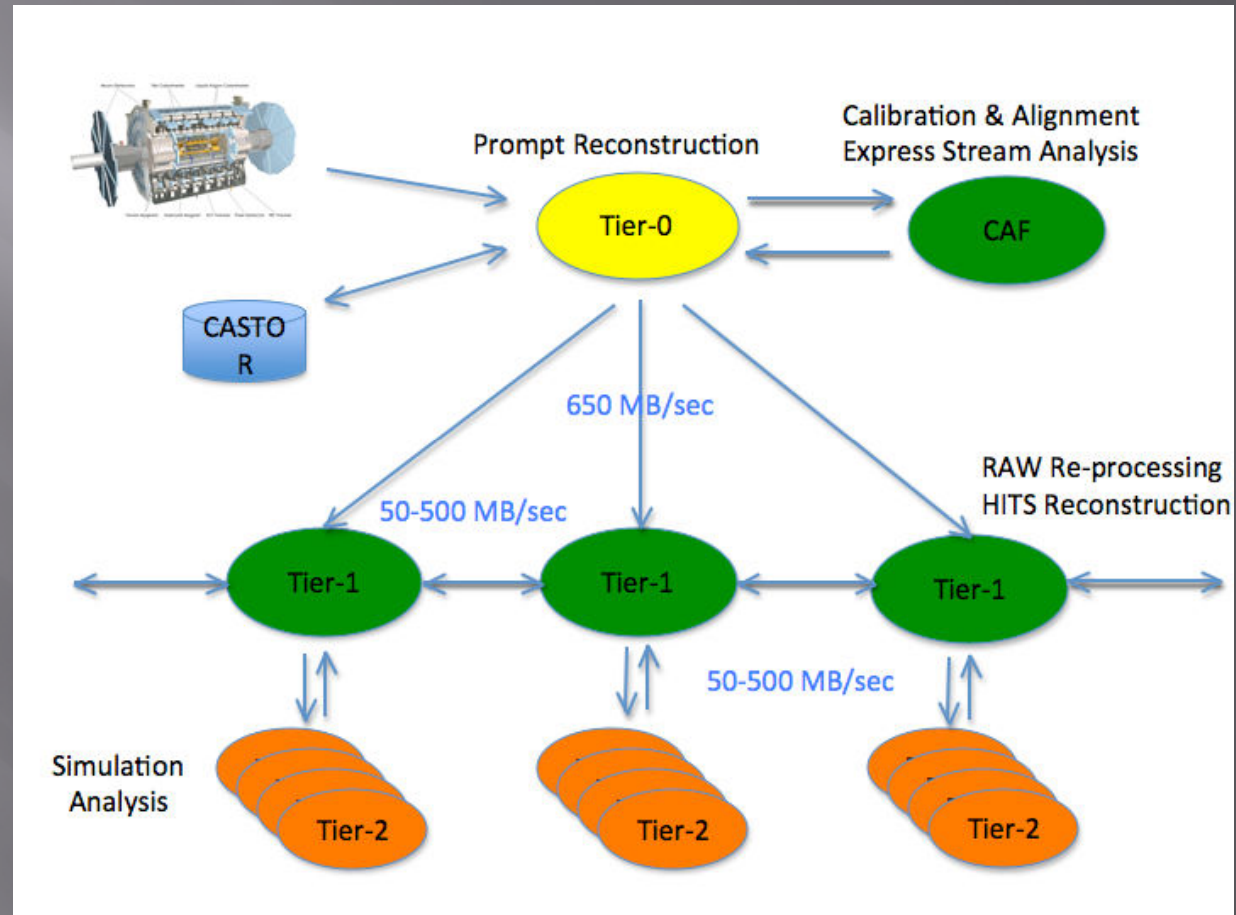
LHC Experiments
Grid projects: Europe, US
Regional & national centres

Choices

Adopt Grid technology.
Go for a "Tier" hierarchy.
Use Intel CPUs in standard
PCs
Use LINUX operating system.

Goal

Prepare and deploy the
computing environment to
help the experiments analyse
the data from the LHC
detectors.



New/Old Tools

OpenMP

an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C/C++ and Fortran on many architectures, including Unix and Microsoft Windows platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

Supported by Open Source Tools:

`GCC >= 4.2`

GPU Computing

With the increasing programmability of commodity graphics processing units (GPUs), these chips are capable of performing more than the specific graphics computations for which they were designed. They are now capable coprocessors, and their high speed makes them useful for a variety of applications.

Free of charge tools:

CUDA Programming Environment
from NVIDIA

Summary

- Grids offer a way to solve Grand Challenge problems for the new era of HEP Experiments.
- Grids offer a way of using the information technology resources optimally inside an organization. They also provide a means for offering information technology as a utility for commercial and non-commercial clients, with those clients paying only for what they use, as with electricity or water.
- In the light of new CPU and GPU architectures new ways of parallel computing could be developed for HEP community.