Distributed Computing for HEP - present and future

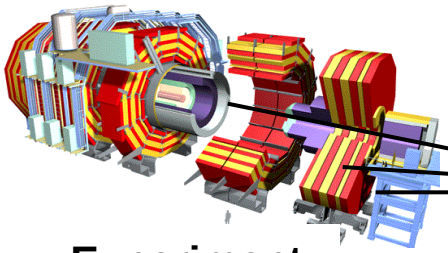
Claudio Grandi (INFN Bologna)

Outline

- Distributed computing architectures for HEP
- Security over a distributed system
- Storage Management
- Workload Management
- Infrastructure Management
- Conclusions

ARCHITECTURES FOR HEP DISTRIBUTED COMPUTING

MONARC



Experiment

Online System

Tier 0

CERN Center
PBs of Disk;
Tape Robot

Tier 1

IN2P3 Center

RAL Center

FNAL Center

CNAF Center

Tier 2

Tier2 Center

Tier 3

Institute

Physics data cache

Tier 4

Workstations

A hierarchical model was needed in order to keep the number of “connections” between computing centres under control

(HEP-)Grid

The number of “connections” is no more a limiting factor

- The grid tools hide the physical location of resources

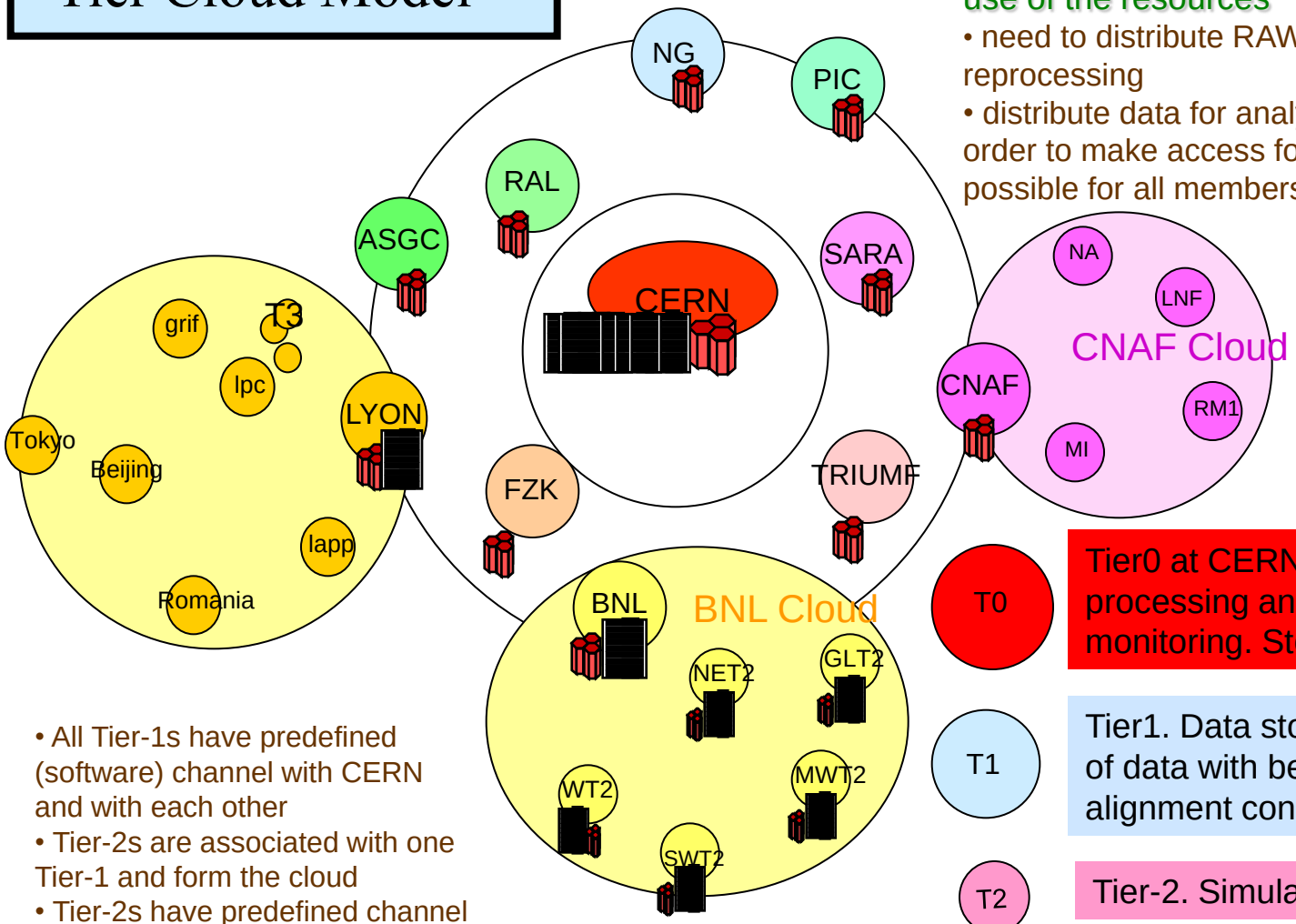
Any model is in principle possible...

"Role-driven" architecture

- Experiments decided to develop hybrid architectures
  - Even though all “connections” are possible only some are allowed
    - E.g. the ATLAS “cloud” model, CMS “regional” productions, etc…
- The driver in building the architecture is the role of sites
  - This was already in the MONARC model
    - Data archival, Data reprocessing, Monte Carlo production, Batch analysis, Interactive analysis, …
- A typical model:
  - Tier-0
    - First (cold) archival copy, first processing, (re-processing)
  - Tier-1
    - Second+ (hot) archival copy, reprocessing, (MC production), (analysis)
  - Tier-2
    - MC production, batch analysis
  - Tier-3
    - Mainly interactive analysis, access to the Grid
- More services are needed by the experiments and may find a well defined location (e.g. FTS servers, DBs, catalogues, …)

The ATLAS Cloud Model

"Tier Cloud Model"



Hierarchical Computing Model optimising the use of the resources

- need to distribute RAW data for storage and reprocessing
- distribute data for analysis in various formats in order to make access for analysis as easy as possible for all members of the collaboration

- All Tier-1s have predefined (software) channel with CERN and with each other
- Tier-2s are associated with one Tier-1 and form the cloud
- Tier-2s have predefined channel with the parent Tier-1 only.

T0 — Tier0 at CERN. Immediate data processing and detector data quality monitoring. Stores on tape all data

T1 — Tier1. Data storage and reprocessing of data with better calibration or alignment constants

T2 — Tier-2. Simulation and user Analysis

T3 — Tier-3. Non Grid user Analysis

SECURITY

Access Control

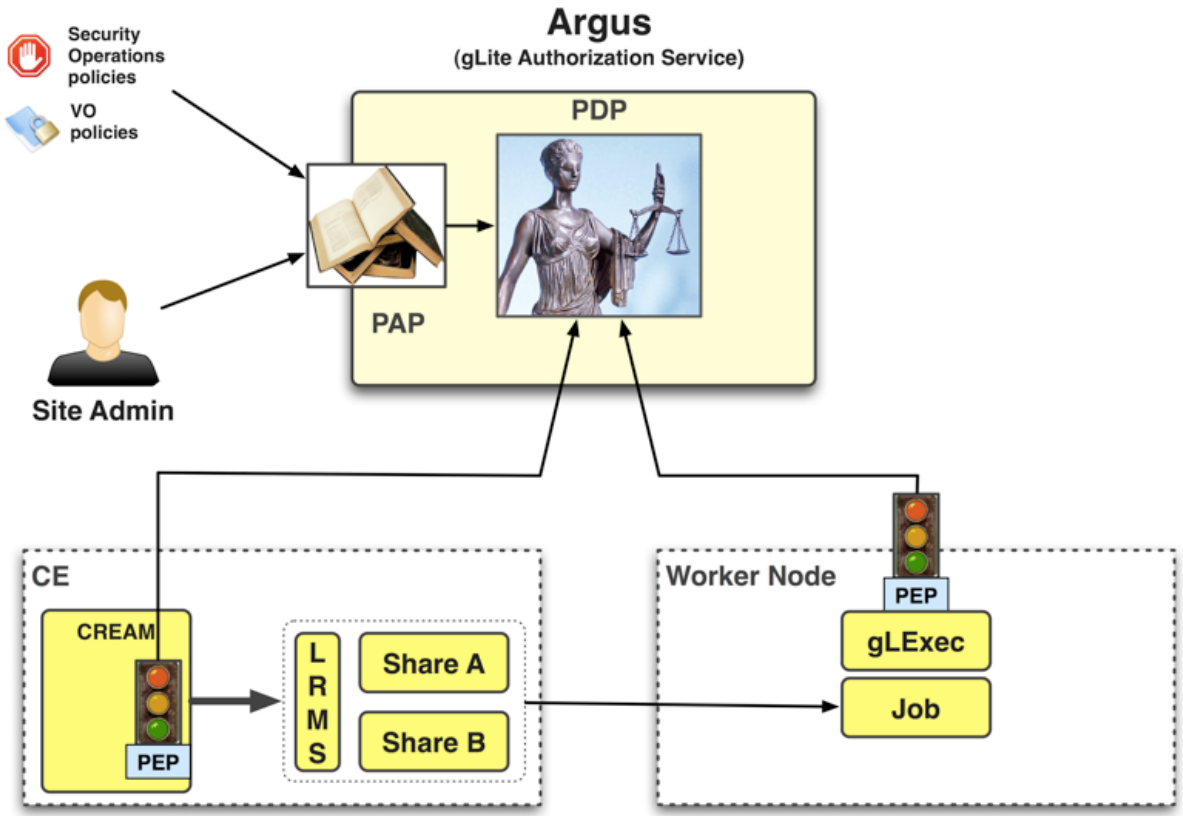
- Currently grid tools provide:
  - Authentication through X.509 certificates
    - Federation of Certification Authorities
    - Single sign-on (trust between users and resources without direct intervention of the organization in the process)
  - Authorization through attribute certificates
    - Based on Virtual Organizations (VO)
    - VOs define “groups” and “roles” for users: **VOMS**
- In general certificate proxies are used on the infrastructure
  - Limited duration – may be automatically renewed
  - May be dynamically delegated (in total or in part) to services that act on behalf of the user (e.g. workload management systems, etc…)
- VO’s may build their own services using the tools developed by grid middleware projects to implement proper delegation, renewal, etc…
  - The overall verification of the security of VO services is up to the VO themselves and the site administrators that host them

Moving away from certificates?

- Bridging of existing AuthN and AuthZ sources to X.509
  - Short Lived Credential Service
    - Shibboleth (glSLCS), Kerberos (kCA coming), …
- The “security” layers start to be modified to accept more AuthZ and AuthN means (e.g. SAML assertions, etc…)
  - There is also the need to address some intrinsic vulnerability related to the management of proxy certificates on the infrastructure
- It is probably premature to base a computing system for HEP on something different from X.509 certificates today

Next steps of authorization

- Important modifications to the authorization layer in gLite
  - Support for central policy management will allow experiments to have better control on how resources are used
  - Decouple the authorization step from the specific implementation on the resource
    - Today only through the mapping of a certificate (FQAN) to a local UID/GID
    - In future it may be extended to use other mechanisms, e.g. Virtual Machines
- More work will be needed for storage authorization
  - Quota's, interference with computing AuthZ, ...



DATA MANAGEMENT

Data granularity

- The data management framework depends heavily on the size of the smaller chunk of data that is addressable on the distributed infrastructure
  - Object level
    - Via OODBMS or RDBMS and object relational mapping
  - File level
    - Components of the fabric layer handle data at the level of files (OS, storage, ...)
  - File collection level
    - Aggregation at the application level, reduces the scale of the bookkeeping
- Grid tools work at the file level
  - Some attempts to use collections in catalogues, but the lower level tools are file based (SRM interface, gridFTP, FTS, ...)
- HEP experiments use:
  - Object level for conditions data and metadata (catalogue) (“mutable”)
    - Distribution models based on DB replication tools – see later
  - File (or file collection) level for event data (“immutable”)

File location catalogues

- Intended for logical to physical name translation
- “Flat” file catalogues don’t scale at the HEP scale
- A common approach is to use a hierarchical catalogue
  - Associate a list of sites (or storage systems) to a file (or better a file collection) in a global catalogue
  - Let a local site catalogue do the logical to physical name translation
- Local catalogues: DB vs. Algorithms
  - DB based: each logical name has an associated physical name
  - Algorithm-based: exploit a uniform local namespace
    - Define a rule to translate logical names to/from physical names
    - Simple, fast, reliable, does not require to run a service
    - Requires a unique namespace does not support multiple storage systems at a single site but for special cases
- Global catalogues: official vs. group or user catalogues
  - Official ones may be aggregated to Metadata or file transfer DBs, ...
  - “Local scope” DBs are needed for non official data

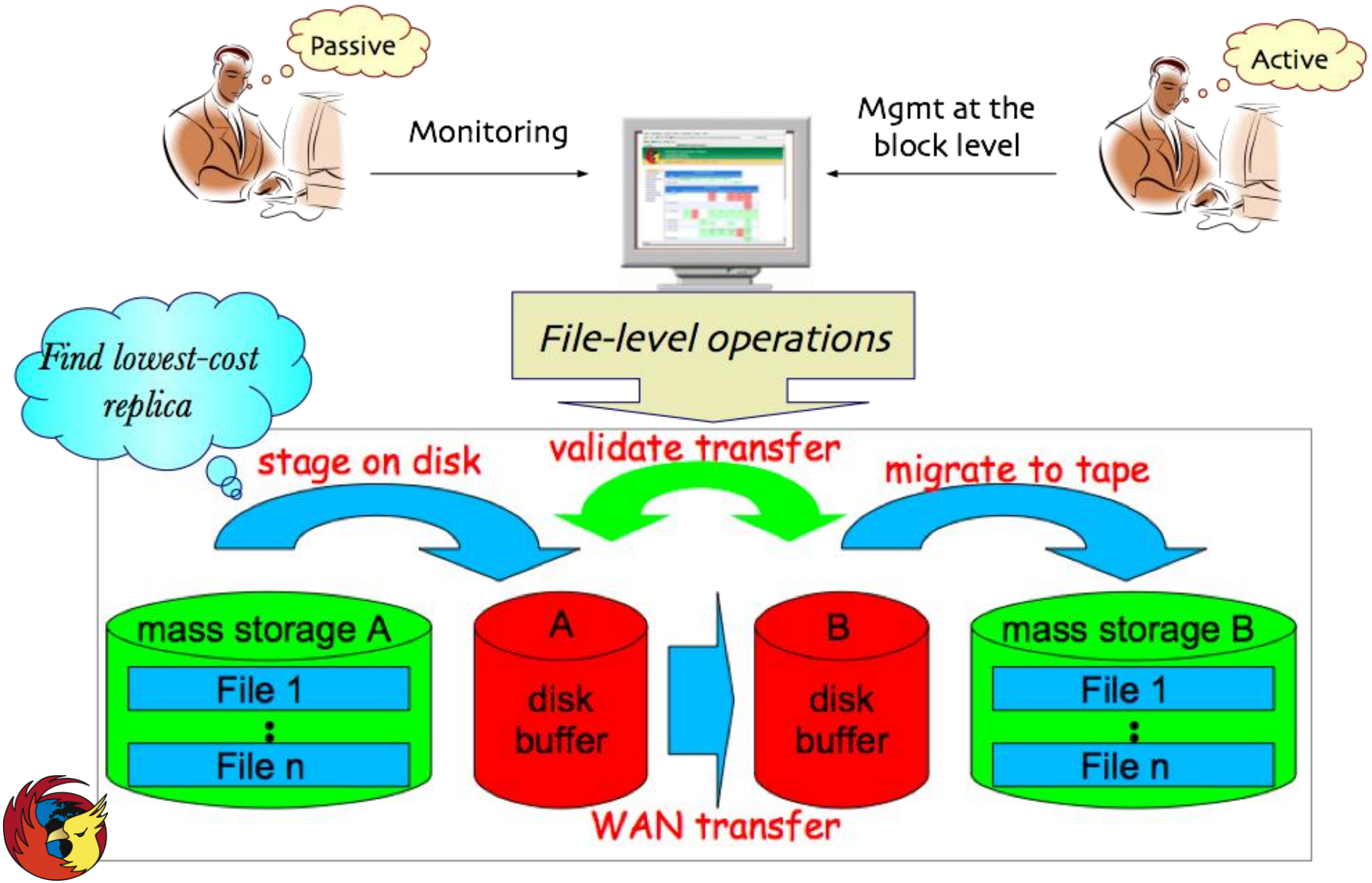
Databases

- Databases are needed to host data that needs to be accessible at the object level
  - Conditions data – accessed by running jobs: high frequency
  - Metadata – low frequency (?)
- Main middleware stacks do not include distributed databases
  - GREiC works with gLite but has never been tested at the HEP scale
- Experiments have been using databases with replication strategies based on read-only cashes at sites (avoid locking)
  - E.g. FroNTier server + Squid caches for conditions data
- Metadata catalogues are performance critical as typically are the first step in all data analysis processes
  - Often based on bare RDBMS (Oracle, MySQL, ...) with simple AuthN and a few read-only replicas
  - General purpose metadata catalogue, like gLite AMGA, have not been used by HEP experiments

File transfers

- Data placement is critical in all HEP experiments data models
- Basic functionalities provided by grid middleware stacks
  - gridFTP is the basic transfer mechanism
  - FTS has been produced by gLite/WLCG for bandwidth control
    - Are there more effective ways to do that using network tools (QoS, etc…)?
- Experiments build their own frameworks
  - E.g. PhEDEx by CMS
- Data transfer during processing
  - In general experiments use a data driven processing model
    - Jobs go to the data
  - Some experiments allow automatic job-input data transfer
    - The quality of the network allows this today, but these transfers are in general out of the control by the bandwidth control tools like FTS
  - Most experiments allow for automatic job-output data transfer
    - Critical, not only for the reason above, but also because failures in this phase are frequent and cost much in terms of CPU efficiency

PhEDEx operations model



File storage

- All HEP sites provide storage exposing the SRM interface
  - In some cases experiments require additional/alternative layers like xrootd
- SRM has hooks to interact with a Mass Storage System
  - Service classes define the strategies that the storage system implements for the interaction between tape systems and disk buffers
    - D1T0: disk only
    - DnT0: multiple disk copies for performance optimization (e.g. dCache)
      - Parallel file systems (e.g. GPFS, Lustre) do it through striping on multiple servers
    - DnT1: a copy on tape and 1 or more copies on disk
    - D0T1: a copy on tape; the disk is only used as a buffer
- Sites not offering archival services use D1T0/DnT0
- Archival strategies:
  - D1T1 is in general not feasible for HEP experiments
  - D0T1 with a large disk buffer: automatic recalls and garbage collection
  - D0T1 coupled with an independent D1T0 system used for reprocessing: controlled (manual) recall and deletion of files
    - Not all experiments are compatible with more than one storage system at a site

Storage for analysis

- Storage of official data is (easily) controlled by the data operation teams of the experiments
  - Processing-jobs input AND output are at the same site
- Storage for analysis-groups and end-users
  - Offered by Tier-2 and Tier-3 sites. Would require:
    - Appropriate authorization at the user- and group-level
    - Quota management on each storage system
    - Working “grid-wide” accounting system

    We are still missing most of these functionalities
  - In general analysis-jobs input and output are not at the same site, and sometimes the CPU is available at a third site
    - Errors in remote data operations are the main cause of failure of analysis jobs
    - Send the jobs where the input data are → non-optimal utilization of CPU
    - Copy the input data to temporary storage at the processing site → requires network bandwidth; requires large temporary storage; failures
    - Remote access → high latencies; requires network bandwidth; failures
    - Store job-output on temporary storage and collect it at the final destination with asynchronous file transfer systems → complex

WORKLOAD MANAGEMENT

Batch processing

- In HEP most of the processing power is exploited through batch jobs. Jobs enter a site through the Compute Element
- Currently a few flavours of CE are used to enter a site:
  - The LCG-CE (based on pre-WS Globus) on EGEE and OSG
  - The CREAM CE on many EGEE sites
  - The ARC CE on NorduGrid
- Currently the definition of shares, priorities and execution environment is done via the mapping to local UID/GID
  - Using the CREAM CE it is already possible to pass requirements to the batch system and customize the environment.
  - In future the use of ARGUS or other equivalent AuthZ systems may allow site managers to have a more fine grained control over the execution environment
    - E.g. The possibility to use a given VM, or to use N cores at a time, etc...
- Accounting is provided through the WLCG infrastructure
  - But it is possible to deploy an independent one (based on gLite DGAS)

Workload Management

- Via a service that dispatches jobs to sites
  - The choice is either done by the client (e.g. Condor-G) or done by the system based on the Information System data (e.g. gLite WMS)
  - If the choice of the site is non optimal the job may take longer than needed to complete
    - After 10 years of grid, the Information System data is often wrong ☹
    - There is an intrinsic inadequacy due to the delay of when the job starts w.r.t. When the information is produced
- Through pilot jobs (overlay network)
  - Central services or local factories submit placeholders that download the payload when they start on the allocated CPU
  - It is possible to do just in time scheduling on the central service
    - Implementing also VO policies
  - Concerns:
    - Scalability and fault tolerance (pilot frameworks rely on central servers)
    - Security, in particular for user analysis (multi-user payload) because the real identity of the user is hidden by the identity of the pilot submitter → use glexec
  - Most experiments use pilots for productions and many also for analysis

Experiment frameworks

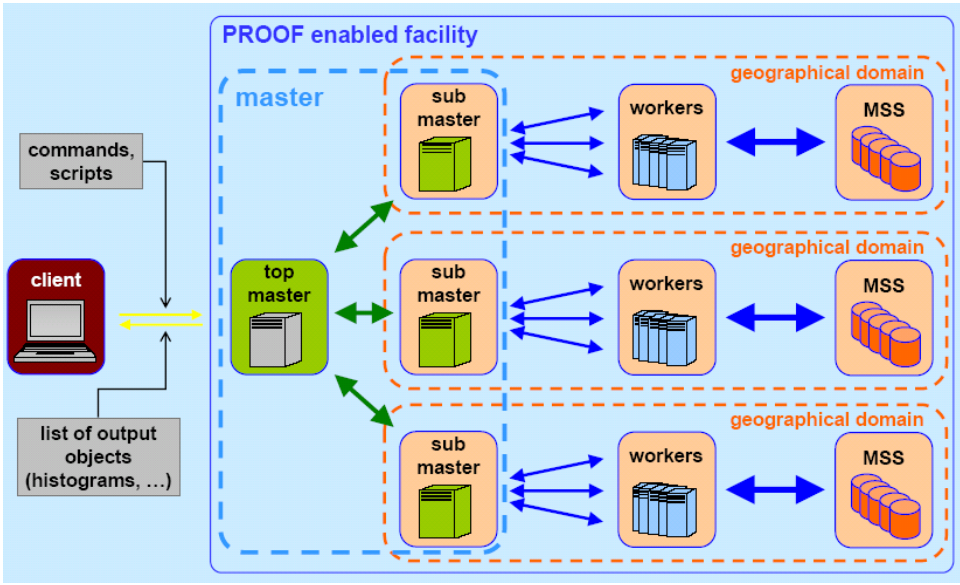
- The main tasks of the experiment frameworks are:
  - Interpret the high-level user request and define the parameters of the jobs to be submitted
    - Execution environment, executable and libraries, task partitioning, input and output data definition for each job
  - Submit the jobs (collections) to the grid workload management system
  - Track the job status and present it to the user in a user friendly way
  - Check job exit status and control resubmissions
  - Collect the user output, do the data management operations needed to aggregate the results into a form suitable for the following steps of the processing
- In addition many experiment frameworks implement part or all of the functionalities of the workload management systems
  - In particular if using a pilot job framework experiment frameworks implement task queues and match-making mechanisms to assign tasks to the correct pilots

Next steps for batch processing

- Support for pilot jobs is being improved
  - Mainly hardening the security (glexec on Worker Nodes)
- Experiment software starts to be able to exploit many cores
  - The Grid workload management is already able to handle parallel jobs. More flexibility will be needed e.g. to specify that the allocated cores must be on the same host
- Virtual Machines
  - Provide an effective way to isolate the execution environment
  - Sites already start using them to decouple the local and user environment (making easier to support many VOs)
  - Custom virtual machines can be requested by the users for specific tasks - CREAM can already pass this kind of requirement
- The key for more effective use of resources is:
  - A flexible way to pass requests to the local resource managers
  - A reliable authorization system to verify that requests are permissible
- gLiteWMS + CREAM + ARGUS are going in this direction

Interactive processing

- The last step of the user analysis is the repeated, interactive processing of the results produced by the batch jobs
- In the computing models this is supposed to be done at the user institutes, i.e. on Tier-3s
  - Not all LHC experiments in all countries are already equipped for this
  - Tier-3s are also supposed to be providing access to distributed resources (User Interface).
- PROOF is an example of a tool to distribute interactive work to many workers
  - It works well on local farms
- Exploiting distributed resources for interactive analysis is not effective because of the latencies



INFRASTRUCTURE MANAGEMENT

Infrastructure management

- Site resources are made available through Grid projects
  - HEP experiments use EGEE, NorduGrid, OSG (WLCG)
  - As part of the infrastructure VOs find basic functionalities
    - Basic monitoring, information systems, global accounting, support portals
- VOs still have to provide some functionalities
  - Specific support, specific monitoring, VO software installation
  - In addition may need to look after VO specific services that they decide to develop and use (e.g. pilot job frameworks, metadata catalogues, ...)

A few thoughts on "Clouds"

- “Clouds” are a way to outsource the maintenance of the hardware while keeping control of the software layer
  - No control over the interconnections of different components, in particular of computing and storage resources
  - Given the level of optimization that was needed for the interaction between storage and computing for HEP, I personally doubt that the same performance can be achieved by a general purpose infrastructure like commercial clouds
    - Yes: this is a problem of the providers, but clients may pay this in term of latencies and extra cost for resources
- The “Grid” was adopted and further developed by HEP with the aim of exploiting distributed resources and know-how available in the HEP community worldwide
  - This has a value per-se
  - Even if this is not considered something relevant it is still better reverting to the idea of one or two big computing centres where the experiments have some control rather than using a commercial cloud

Conclusions

- Join an existing infrastructure rather than building a new one
  - It may actually not be your choice
- Use as many services and tools provided by the infrastructure as possible rather than developing your own
  - If there are missing functionalities, ask
  - If you're able to develop something else, try to do it as extensions to existing tools (and contribute them to the basic stack)
  - Current LHC (and pre-LHC) experiments have been built in parallel with the distributed infrastructures and often this was not possible
- Concentrate on the high-level VO specific layers
- Plan for failures
  - Modular architecture; avoid single points of failure
  - Plan for responsibility transfers (e.g. a site is lost for some time)
- Think to end-user analysis since the beginning
  - If you concentrate on production activities only, it will be much harder