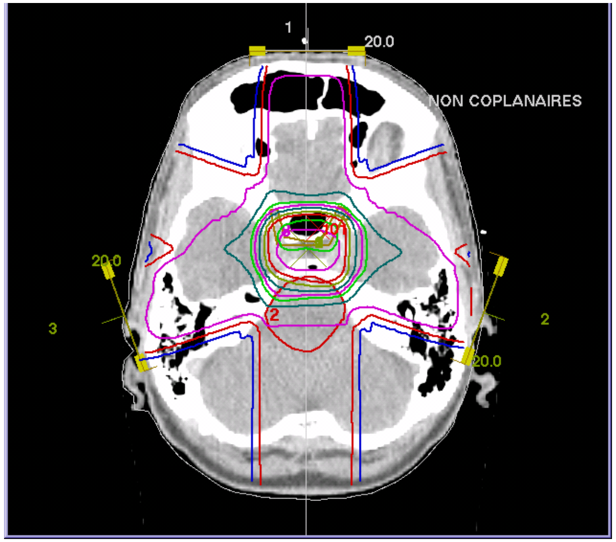
Online Application Demos

Bioinformatics
Earth Observation
High Energy Physics

Erwin.Laure@cern.ch
DataGrid Technical Coordinator

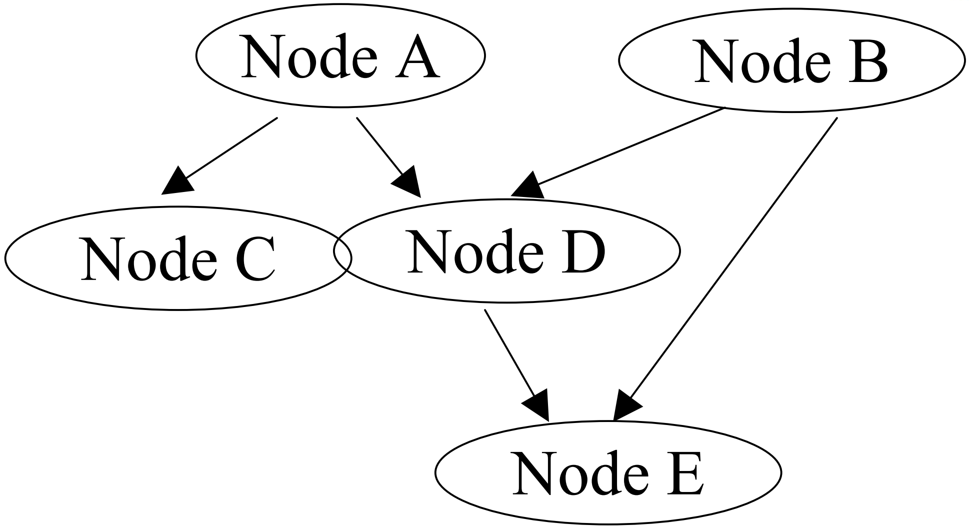
4 Application Demos

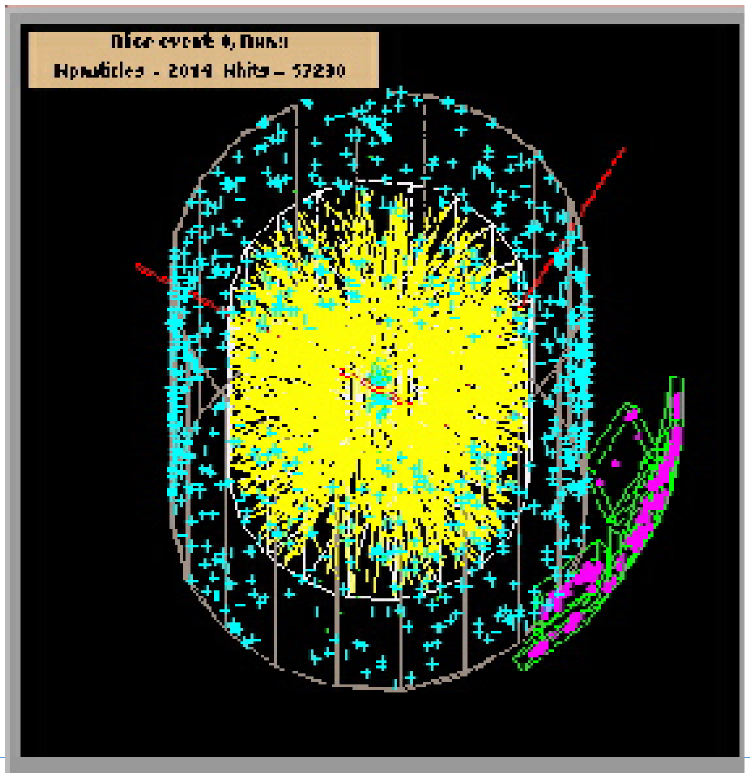
Monte-Carlo simulation for medical applications



Metadata usage in ozone profile validation

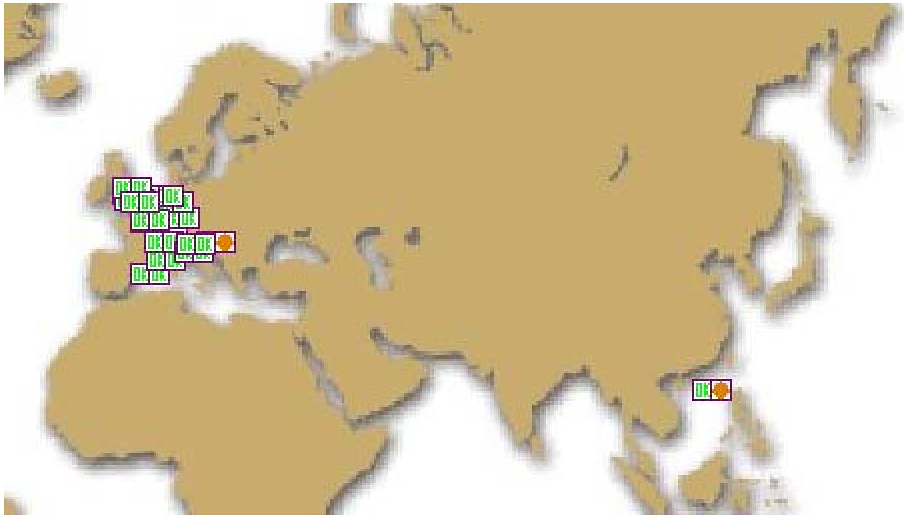
Advanced Scheduling in HEP Applications: CMS demonstrating DAGMan scheduling





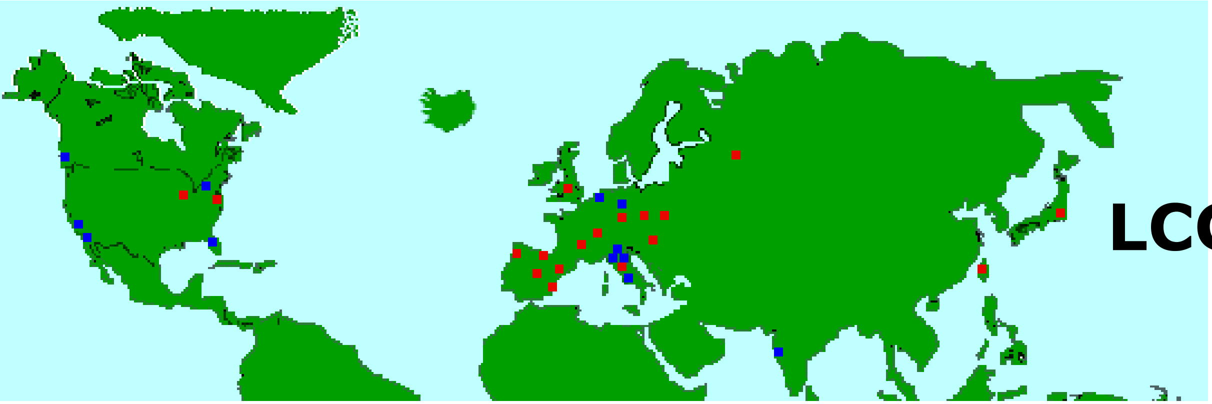
HEP production usage of Grid platforms: the ALICE project

Grid infrastructures used



EDG application testbed

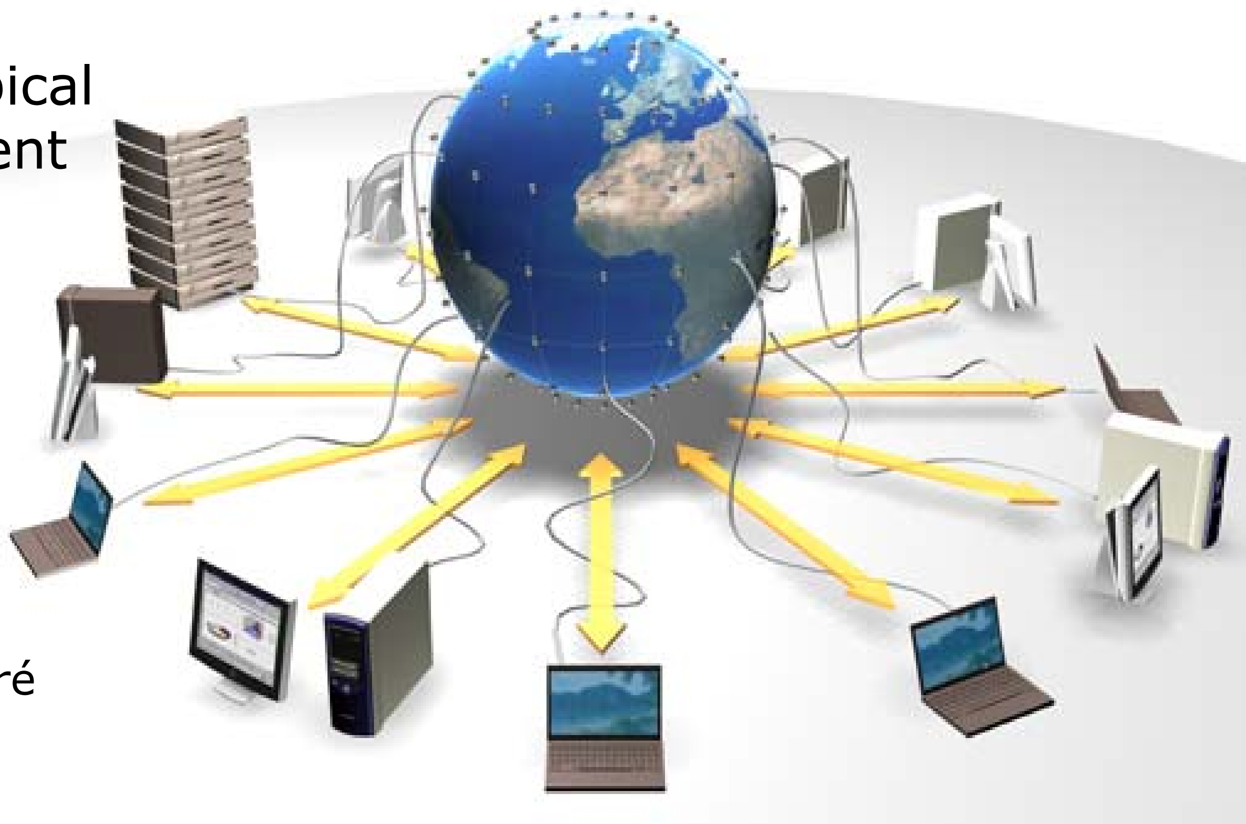
INFN testbed



LCG-2 infrastructure

Parallelization of Monte Carlo simulations GATE for medical applications

The scenario of a typical radiotherapy treatment



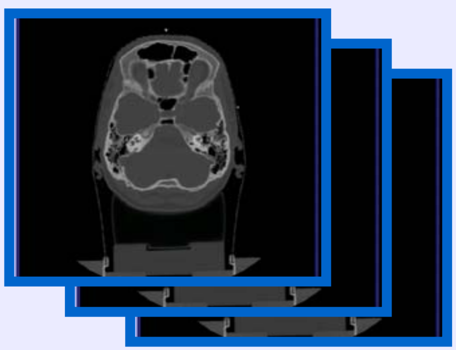
WP10

Lydia Maigne, Yannick Legré
maigne@clermont.in2p3.fr
legre@clermont.in2p3.fr

DataGrid is a project funded by the European Commission under contract IST-2000-25182

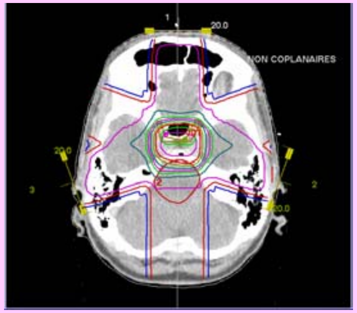
Radiotherapy is widely used to treat cancer

1°) Obtain scanner slices images



The head is imaged using a MRI and/or CT scanner

2°) Treatment planning



Calculation of deposit dose on the tumor (~1mn):
A treatment plan is developed using the images

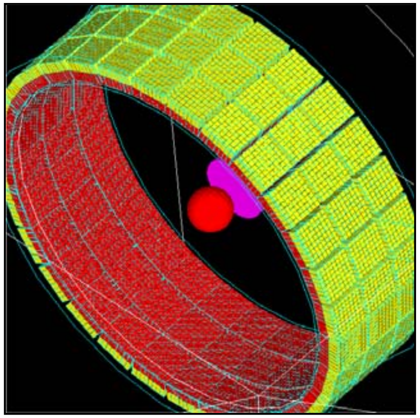
3°) Radiotherapy treatment

Irradiation of the brain tumor with a linear accelerator

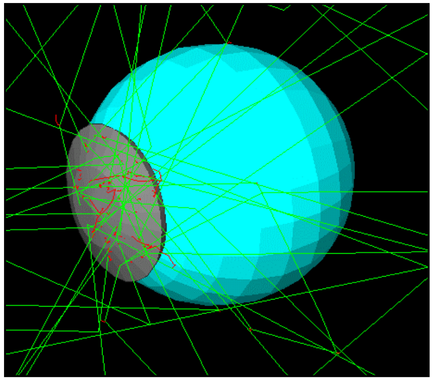
Better treatment requires better planning

- Today: analytic calculation to compute dose distributions in the tumor
  - For new Intensity Modulated Radiotherapy treatments, analytic calculations off by 10 to 20% near heterogeneities
- Alternative: Monte Carlo (MC) simulations in medical applications
- The GRID impact: reduce MC computing time to a few minutes

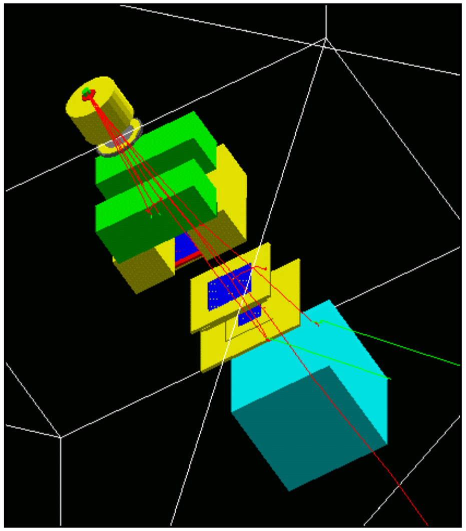
WP10 Demo: gridification of GATE MC simulation platform on the DataGrid testbed



PET camera



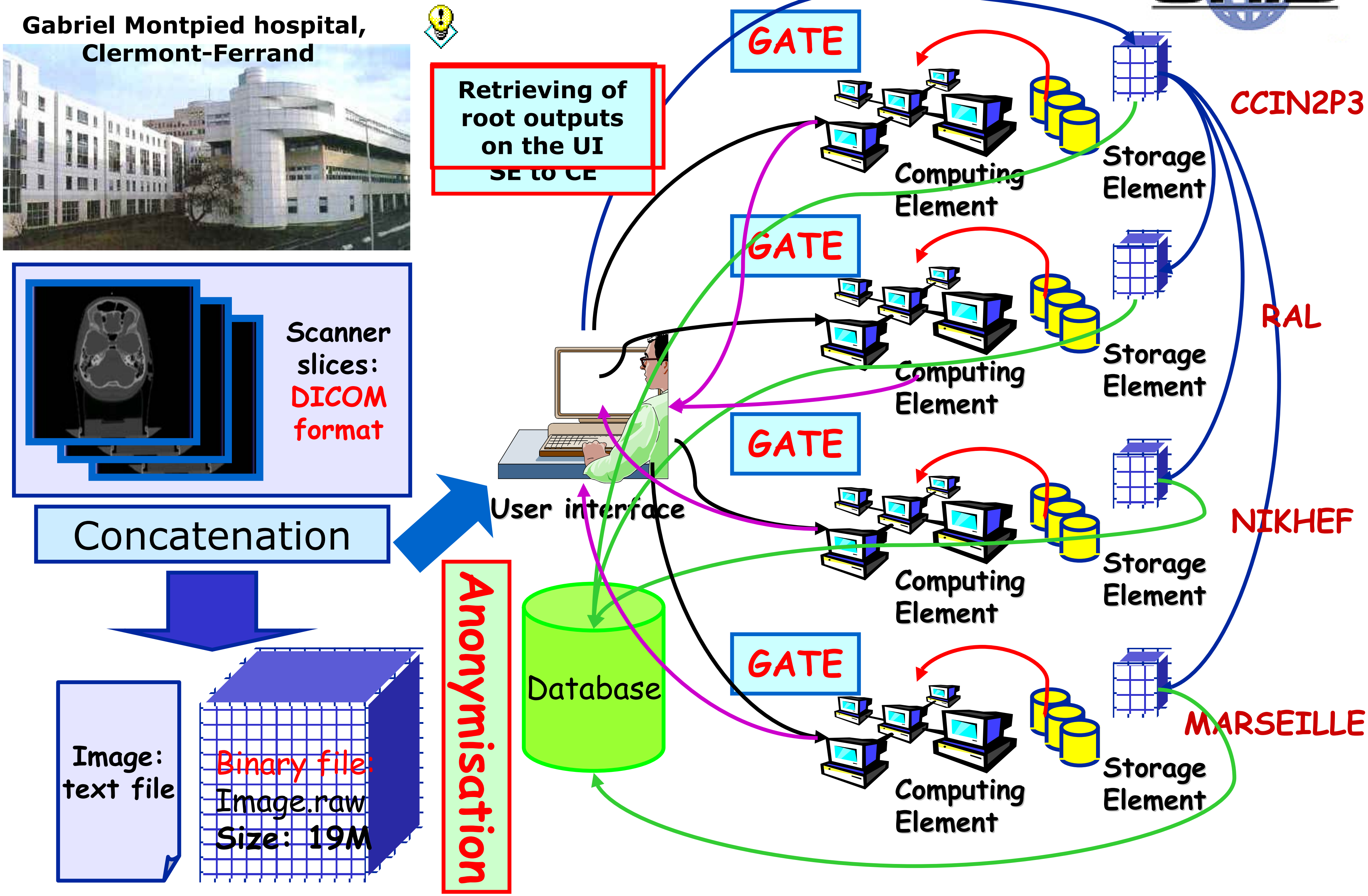
Ocular brachytherapy treatment



Radiotherapy

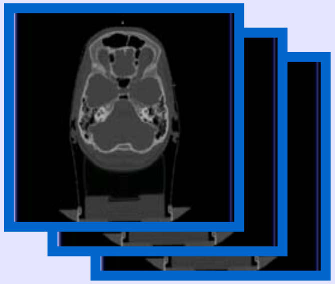
Computation of a radiotherapy treatment on the Datagrid:
Let's go....

GATE applications on the DataGrid



1°) Obtain the medical images of the tumor:

- 38 scanner slices of the brain of a patient are obtained



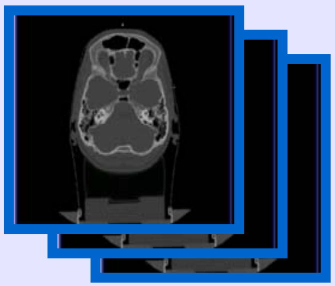
Scanner slices: DICOM format

- **Format of the slices:**
  512 X 512 X 1 pixels

- **Size of a voxel in the image:**
  0,625 X 0,625 X 1,25 mm

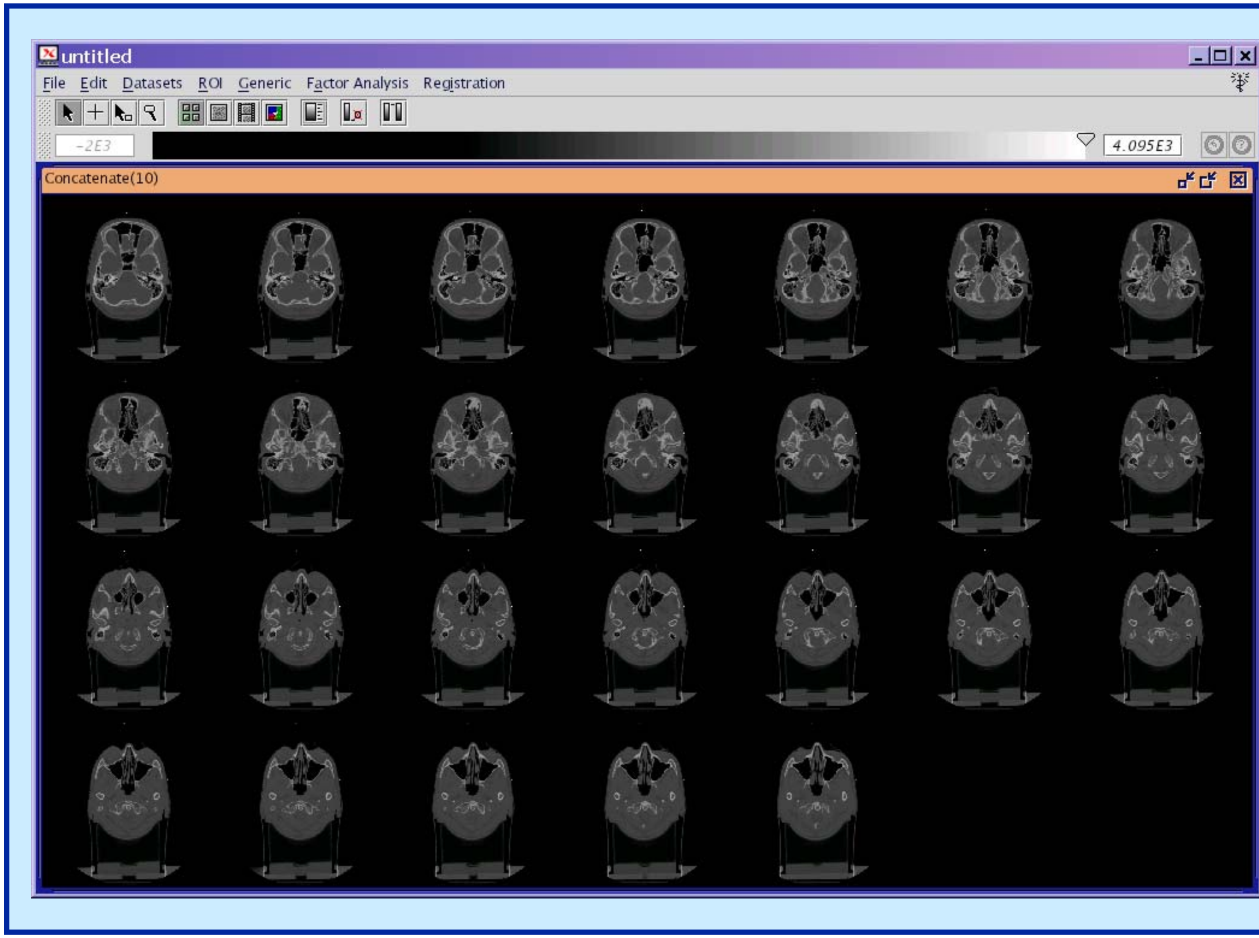
2°) Concatenate these slices in order to obtain a 3D matrix:

Pixies software

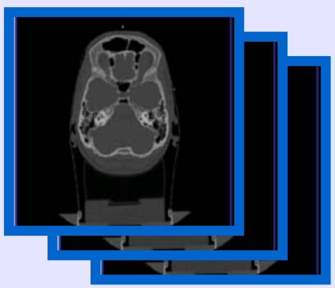


Scanner slices:
DICOM format

Concatenation

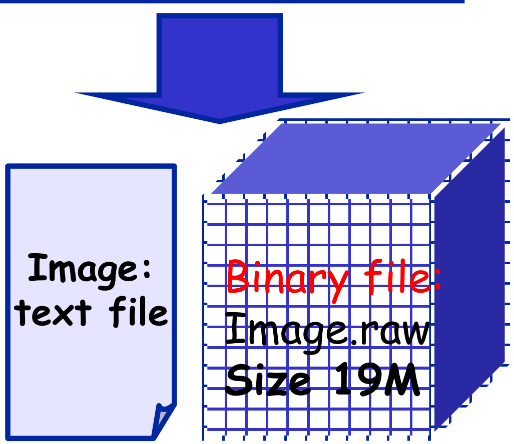


3°) Transform the DICOM format of the image into an Interfile format



Scanner slices: DICOM format

Concatenation



Anonymisation

Pixies software

```
!INTERFILE :=
!imaging modality := nucmed
!originating system :=
!version of keys := 3.3
conversion program := Pixies developer
program author := Apteryx
program version := 1.09pre
!GENERAL DATA :=
!data starting block := 0
!name of data file := image.raw
!patient id :=
!study id :=
!GENERAL IMAGE DATA :=
!type of data := Dynamic
!total number of images := 38
!imagedata byte order := BIGENDIAN
number of energy windows := 1
!energy window [1] :=
!DYNAMIC STUDY (General) :=
!number of frame groups := 1
!Dynamic Study (each frame group) :=
!frame group number := 1
!matrix size [1] := 512
!matrix size [2] := 512
!number format := unsigned integer
!number of images := 38
!scaling factor (mm/pixel) [1]:= 0.625
!scaling factor (mm/pixel) [2]:= 0.625
!slice thickness (pixels):= 1.25
!number of bytes per pixel := 2
!number of images this frame group := 38
!image duration (sec) :=
!maximum pixel count in group := 4095.0
!END OF INTERFILE :=
```

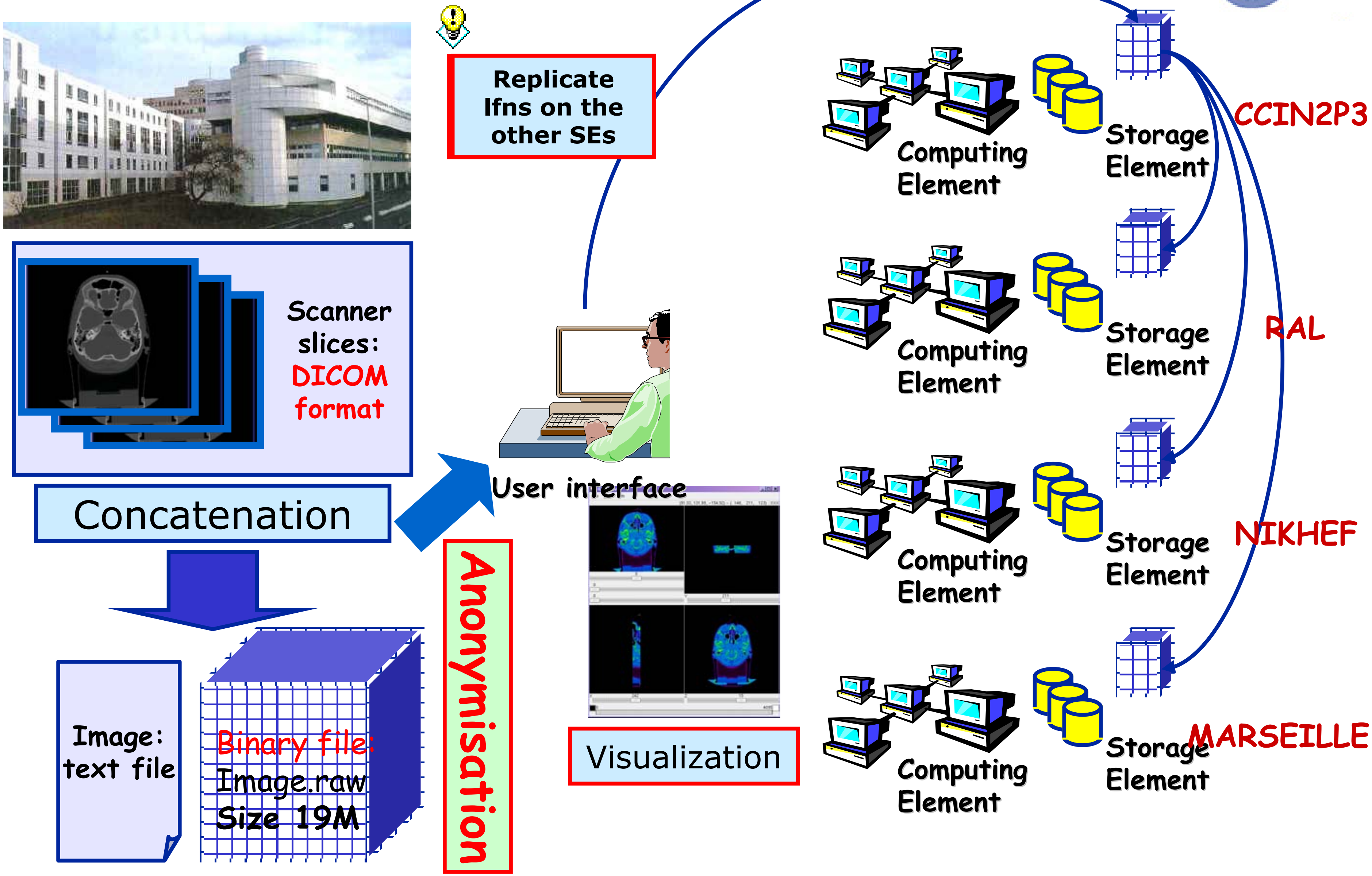
Binary image of the scanner slices

Size of the matrix

Size of the pixels

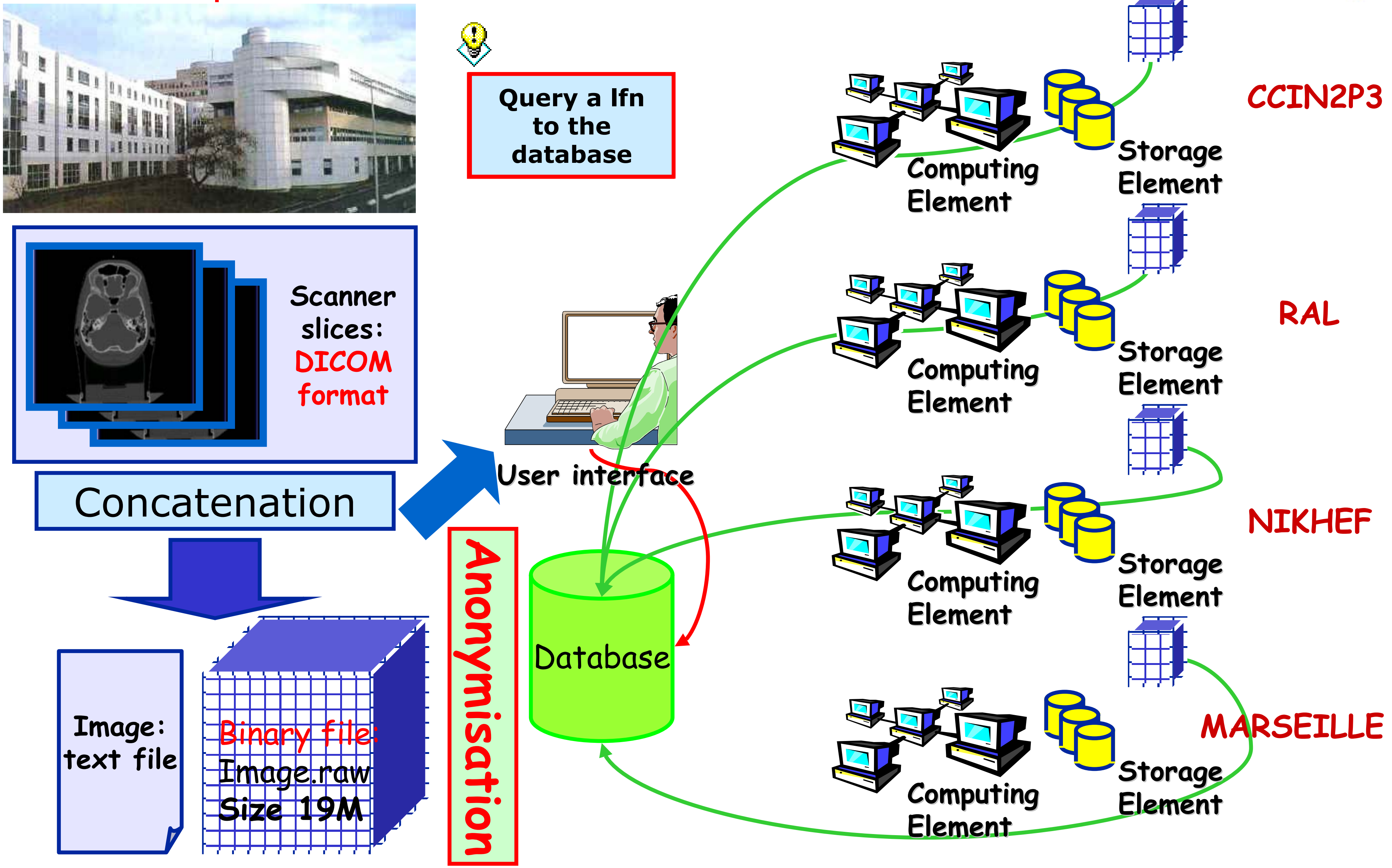
Number of slices

4°) Register and replicate the binary image on SEs:



5°) Register the lfn of an image:

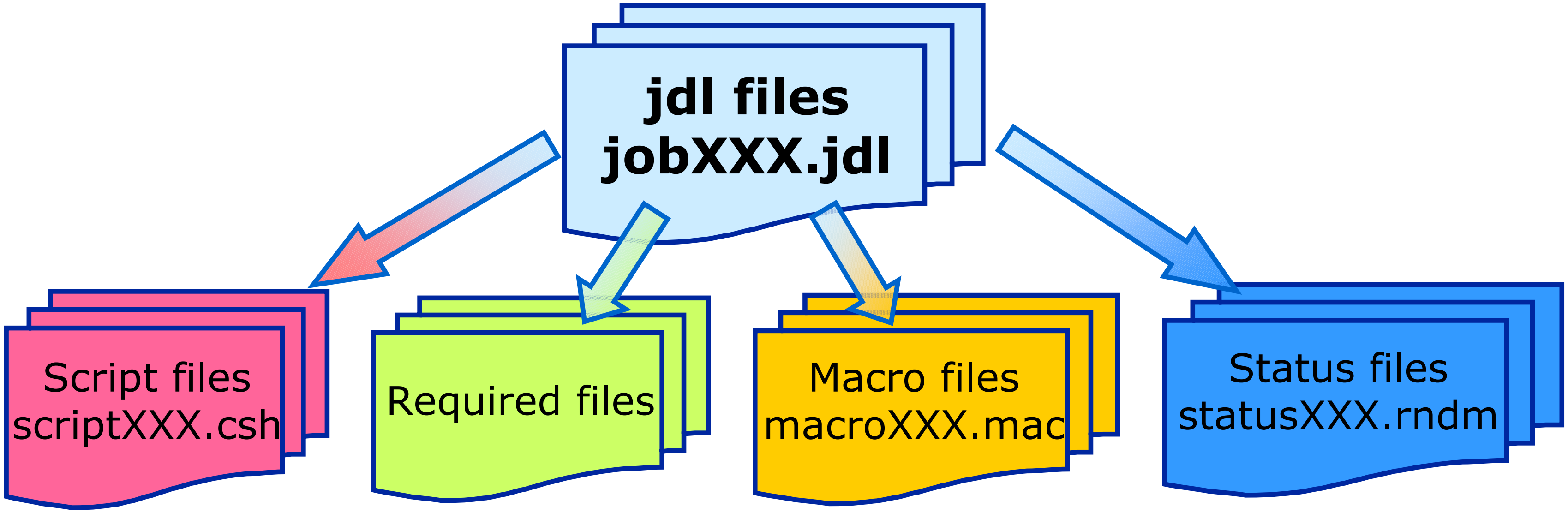
- WP2 spitfire or local database



6°) Split the simulations:

JobConstructor C++ program

- A GATE simulation generating a lot of particles in matter could take a very long time to run on a single processor
  - So, the big simulation generating 10M of particles is divided into little ones, for example
    - 10 simulations generating 1M of particles
    - 20 simulations generating 500000 particles
    - 50 simulations generating 200000 particles ……
  - All the other files needed to launch Monte Carlo simulations are automatically created with the program.



A typical jdl file:

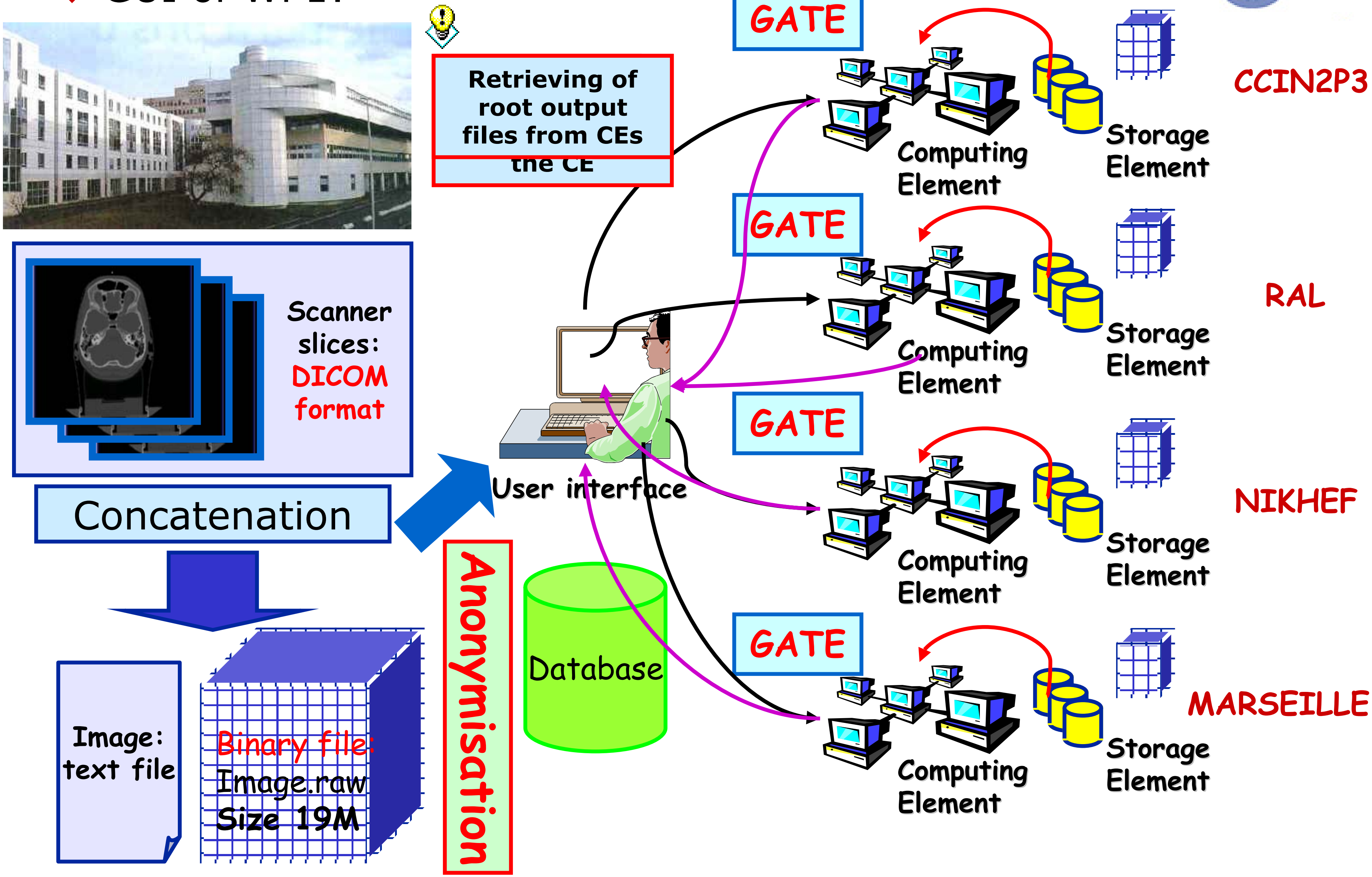
```
[
    VirtualOrganisation = "biome";
    Executable = "/bin/tcsh";
    Arguments = "./script000.csh";
    StdOutput = "std.out";
    StdError = "std.err";
    OutputSandbox = {
        "std.out",
        "std.err",
        "Brain_radioth000.root"
    };
    RetryCount = 3;
    InputData = "lfn:maigne_BrainTOT_demo";
    DataAccessProtocol = {
        "file",
        "gridftp"
    };
    JobType = "normal";
    Type = "Job";
    InputSandbox = {
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/script/script000.csh",
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/macro/macro000.mac",
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/status/status000.rndm",
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/required/prerunGate.mac",
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/required/rangeInterfile2.dat",
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/required/CJP_BrainTOT",
        "/afs/cern.ch/user/l/lmaigne/JOBS/jobGate_5/required/GateMaterials.db"
    };
    rank = (-other.GlueCEStateEstimatedResponseTime);
    requirements = (Member("GATE-1.0.0-3",other.GlueHostApplicationSoftwareRunTimeEnvironmen
t)&&(other.GlueCEStateStatus=="Production"))
]
```

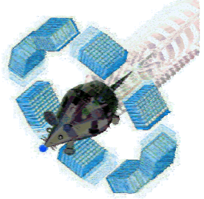
A typical script file:

```
#!/bin/tcsh

#Script de lancement de simulation Gate sur DataGrid
#Auteur :Lydia Maigne
#Date:
#Version :

#############################################################
###Mise en place de l'environnement pour l'exï¿½ution de Gate#
#############################################################

#Get the LFN passed in arguments

#flist="$@"
#for lfn in $flist; do

#echo "Get File"
#edg-rm --vo=biome copyFile $lfn file://$PWD/image.raw
edg-rm --vo=biome copyFile lfn:maigne_BrainTOT_demo file://$PWD/image.raw

#list content of PWD
ls -l $PWD

############################
#Lancement de la simulation#
############################

eval `${EDG_LOCATION}/bin/edg-vo-env --shell=csh biome`
source ${BIOME_ROOT_DIR}/gate_env_main.csh
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${BIOME_ROOT_DIR}/gate/lib/root
${BIOME_ROOT_DIR}/gate/bin/Linux-g++/Gate macro000.mac
```

7°) Submission on the DataGrid:

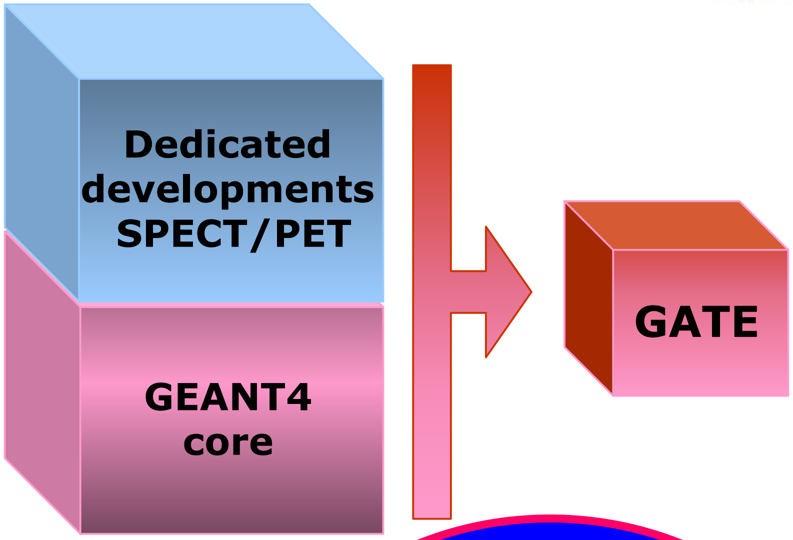
- GUI of WP1:



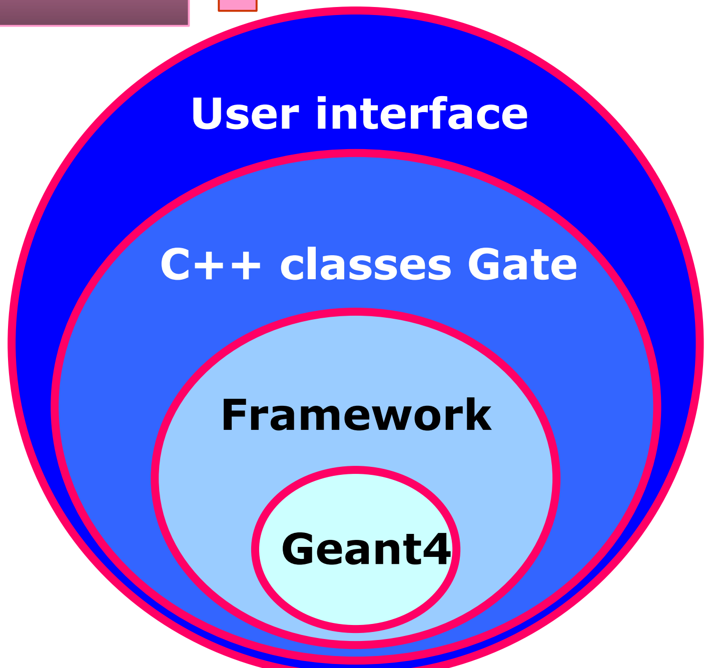


GATE: Geant4 Application for Tomographic Emission

- **Develop a simulation platform for SPECT/PET imaging**
  - **Based on Geant4**
  - **Enrich Geant4 with dedicated tools SPECT/PET**
  - **User friendly**
- **Ensure a long term development**
  - **Effort of shared development**
  - **Collaboration: OpenGATE**

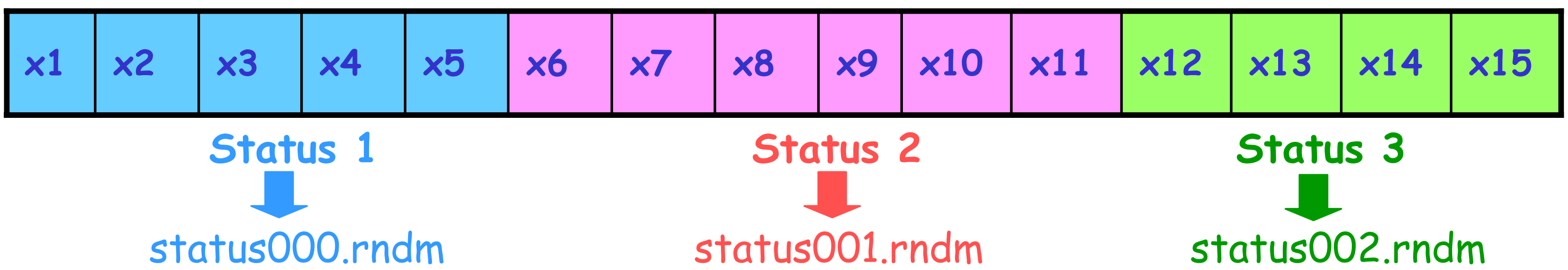


- **Based on Geant4**
  - **C++ object oriented langage**
  - **Reliable cross sections**
- **Framework: interface**
- **GATE development**
  - **modelisation of detectors, sources, patient**
  - **movement (detector, patient)**
  - **time-dependent processes (radioactive decay, movement management, biological kinetics)**
- **Ease of use**
  - **Command scripts to define all the parameters of the simulation**



Parallelization technique of GATE

- The random numbers generator (RNG) in GATE
  - CLHEP libraries: HEPJamesRandom (deterministic algorithm of F.James)
    - Characteristics:
      - Very long period RNG: 2^{144}
      - Creation of 900 million sub-sequences non overlapping with a length of 10^{30}
  - Pregeneration of random numbers
    - The Sequence Splitting Method



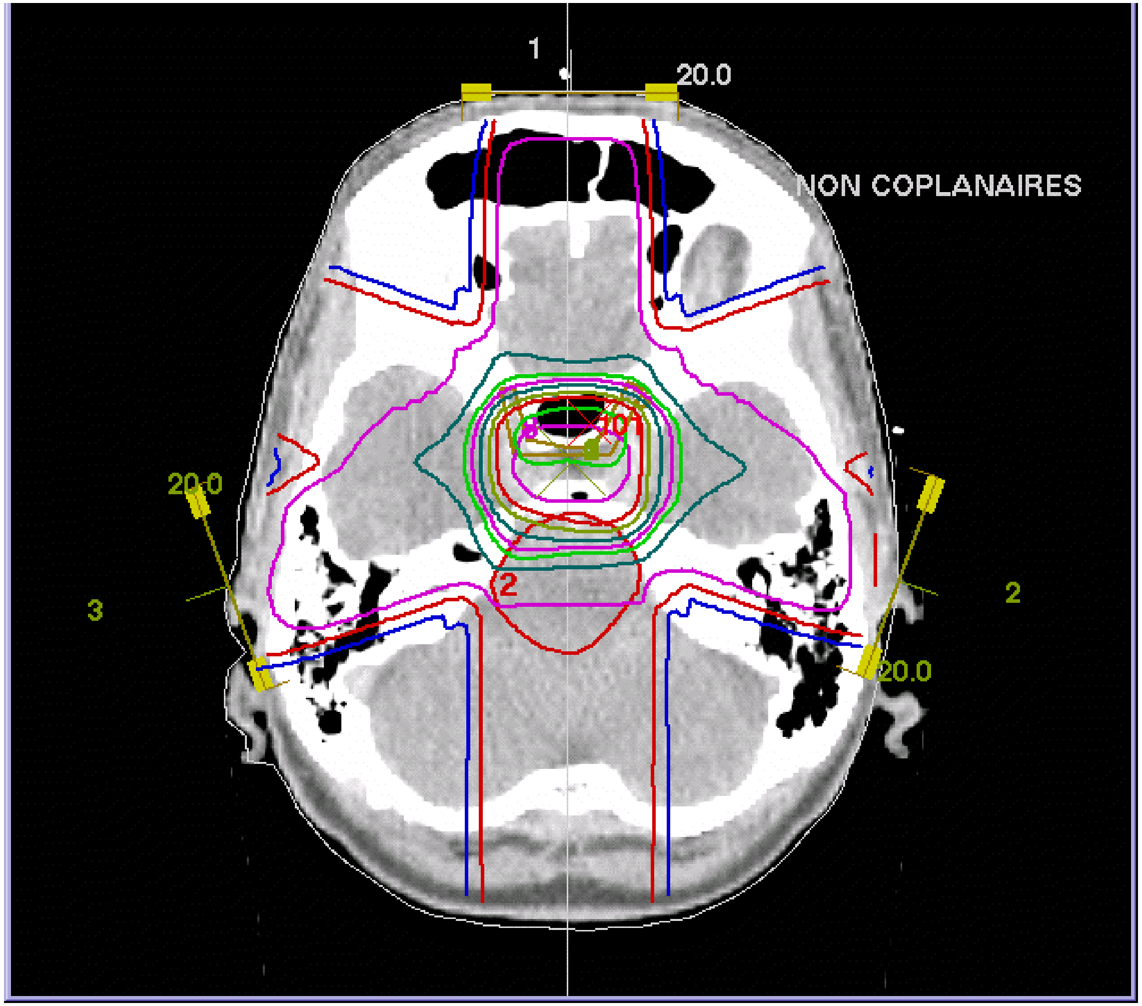
    - Until now, 200 status files generated with a length of 3.10^{10}

Each status file is sent on the grid with a GATE simulation

8°) Analysis of output root files

- Typical dosimetry:
  - Merging of all the root files
  - Computation of the root data

Brain_radioth000.root
Brain_radioth001.root
Brain_radioth002.root
Brain_radioth003.root
Brain_radioth004.root
Brain_radioth005.root
Brain_radioth006.root
Brain_radioth007.root
Brain_radioth008.root
Brain_radioth009.root

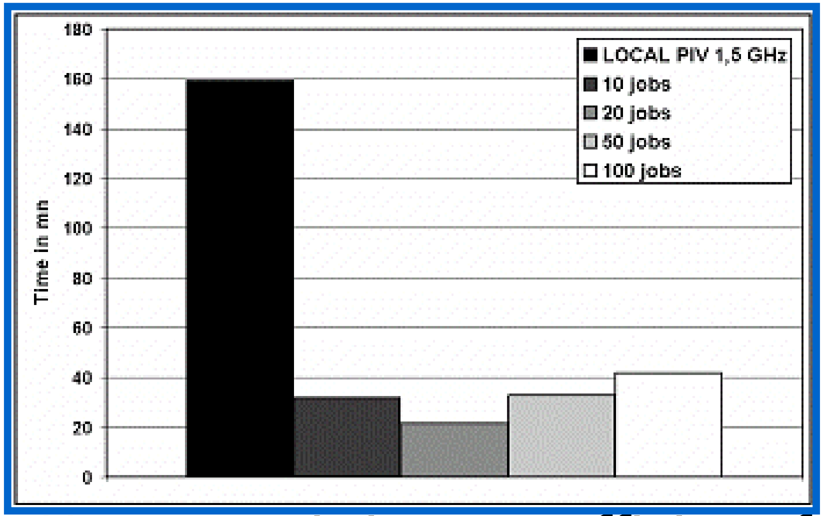


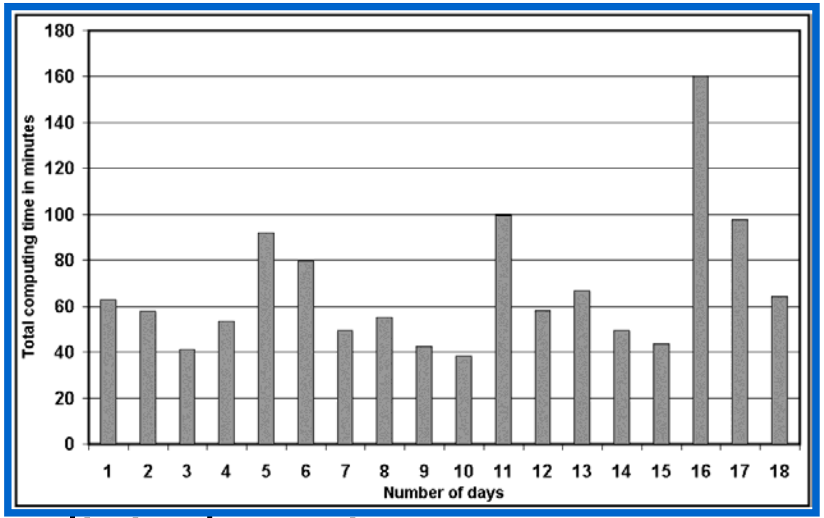
transversal view

Centre Jean Perrin
Clermont-Ferrand

Conclusion and future prospects

- The parallelization of GATE on the DataGrid testbed has shown significant gain in computing time (factor 10)





- However, it is not sufficient for clinical routine
- Necessary improvements
  - Dedicated resources (job prioritization)
  - Graphical User interface

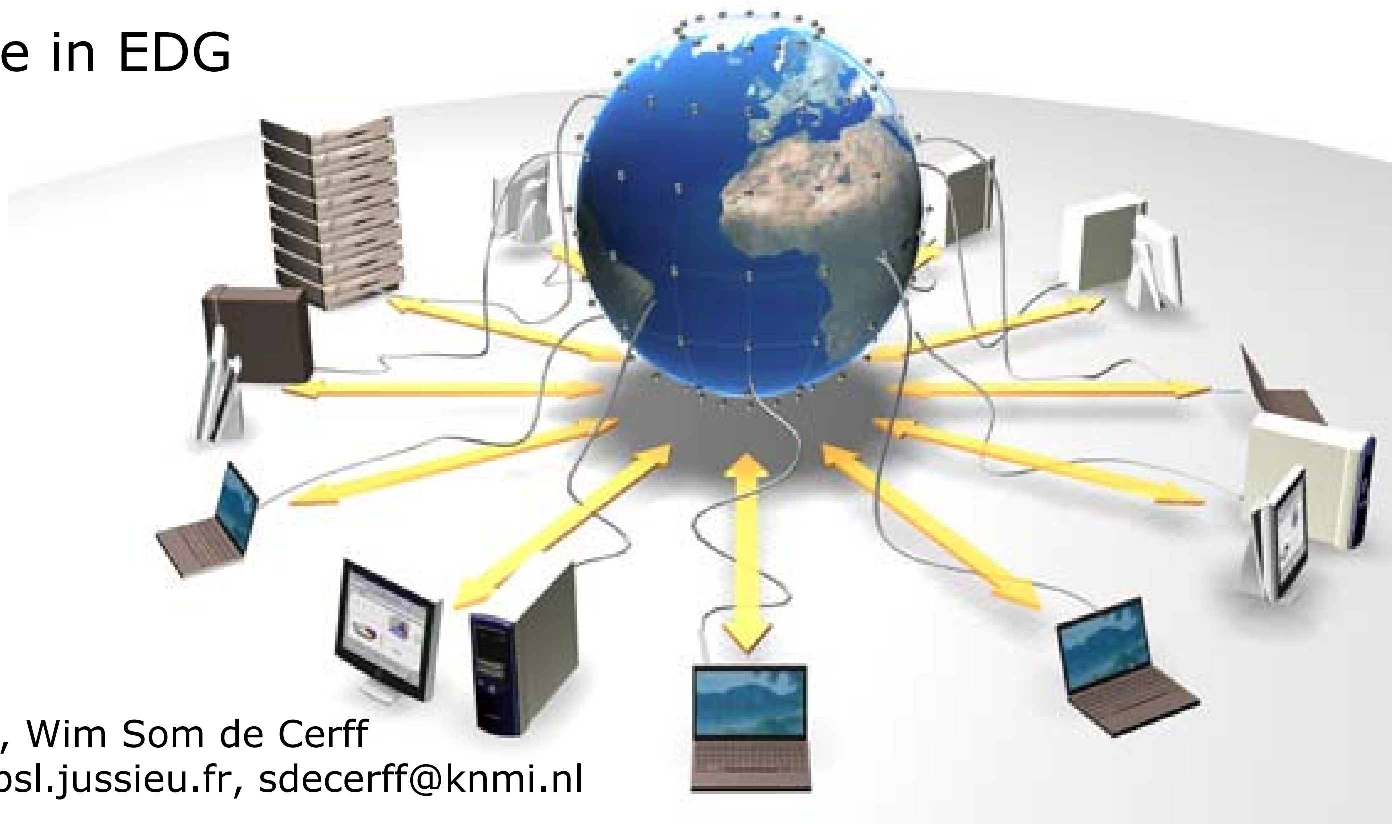
Aknowledgements

- WP1:
  - Graphical User Interface, JobSubmitter
- WP2:
  - Spitfire
- WP6:
  - RPMs of GATE
- WP8
- WP10:
  - 4D Viewer (Creatis)
  - Centre Jean Perrin
  - LIMOS
- System administrators
  - Installations on UIs

EDG Final Review Demonstration

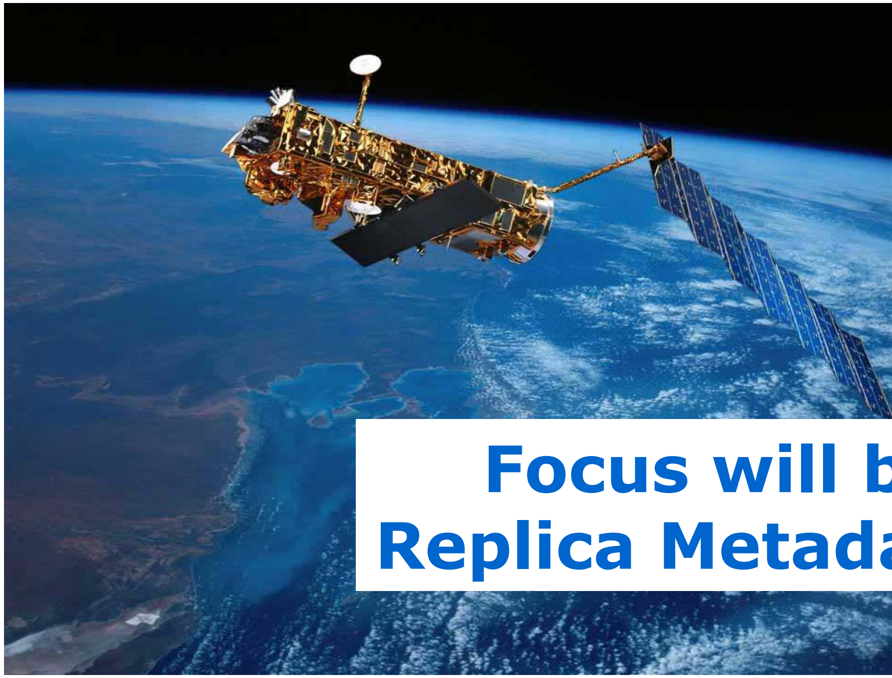
WP9 Earth Observation Applications

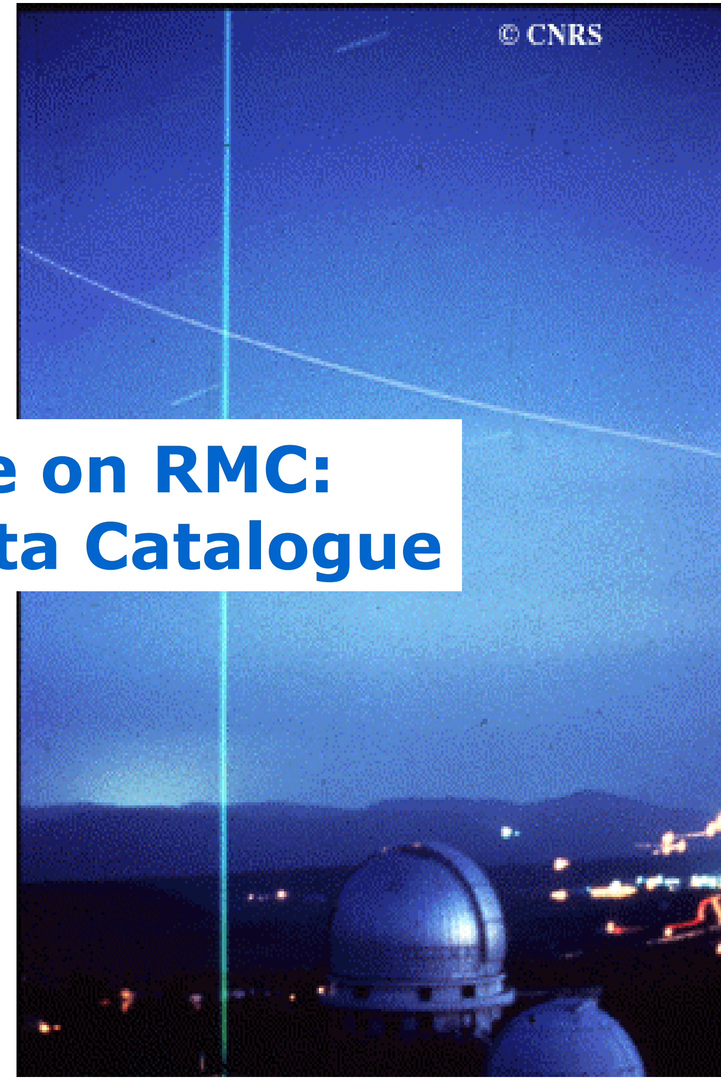
Meta data usage in EDG



Authors: Christine Leroy, Wim Som de Cerff
Email:Christine.Leroy@ipsl.jussieu.fr, sdecerff@knmi.nl

Earth observation Meta data usage in EDG





Focus will be on RMC: Replica Metadata Catalogue

- Validation usecase: Ozone profile validation
- Common EO problem: measurement validation
- Applies to (almost) all instruments and data products, not only GOME, not only ozone profiles
- Scientists involved are spread over the world
- Validation consists of finding, for example, less than 10 profiles out of 28,000 in coincidence with one lidar profile for a given day
- Tools available for metadata on the Grid: RMC, Spitfire

Demonstation outline

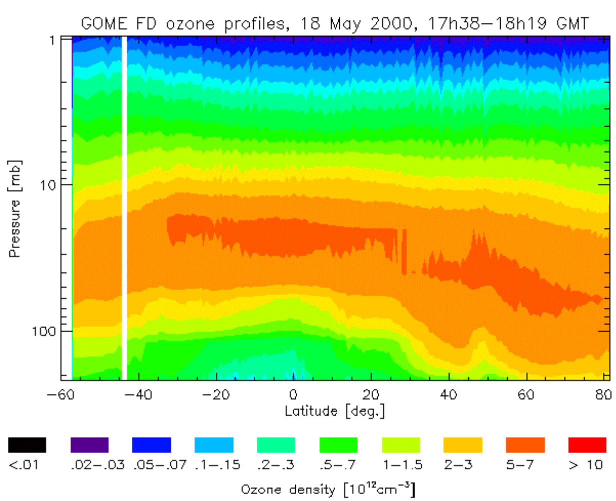
Replica Metadata Catalogue (RMC) usage

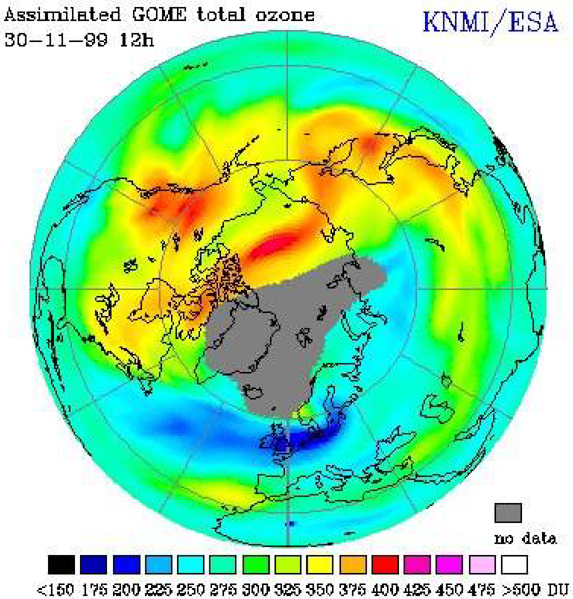
1) Profile processing
   Using RMC to register metadata of resulting output

2) Profile validation
   Using RMC to find coincidence files

3) RMC usage from the command line
   Will show the content of the RMC, the attributes we use.

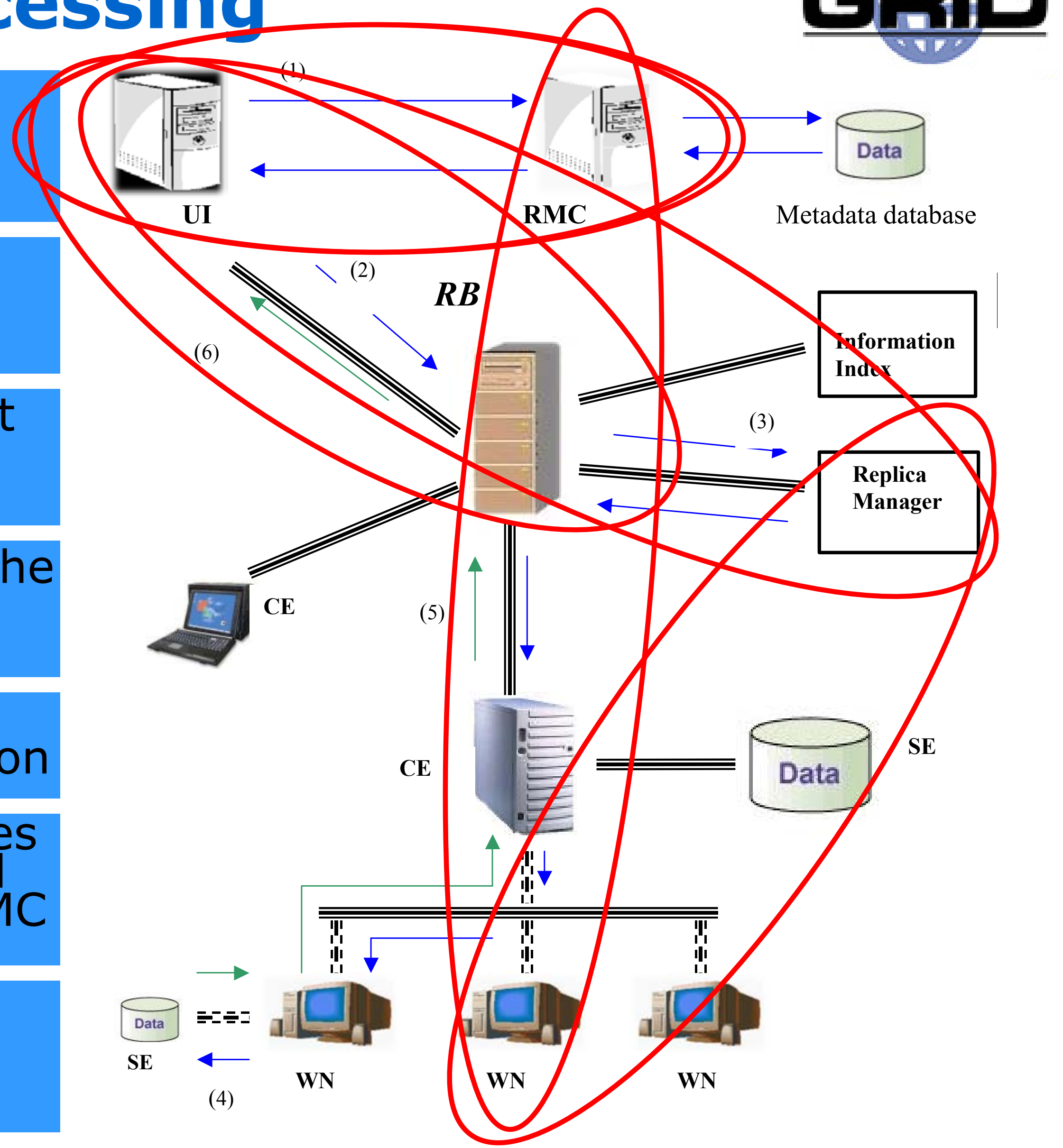
4) Show result of the validation





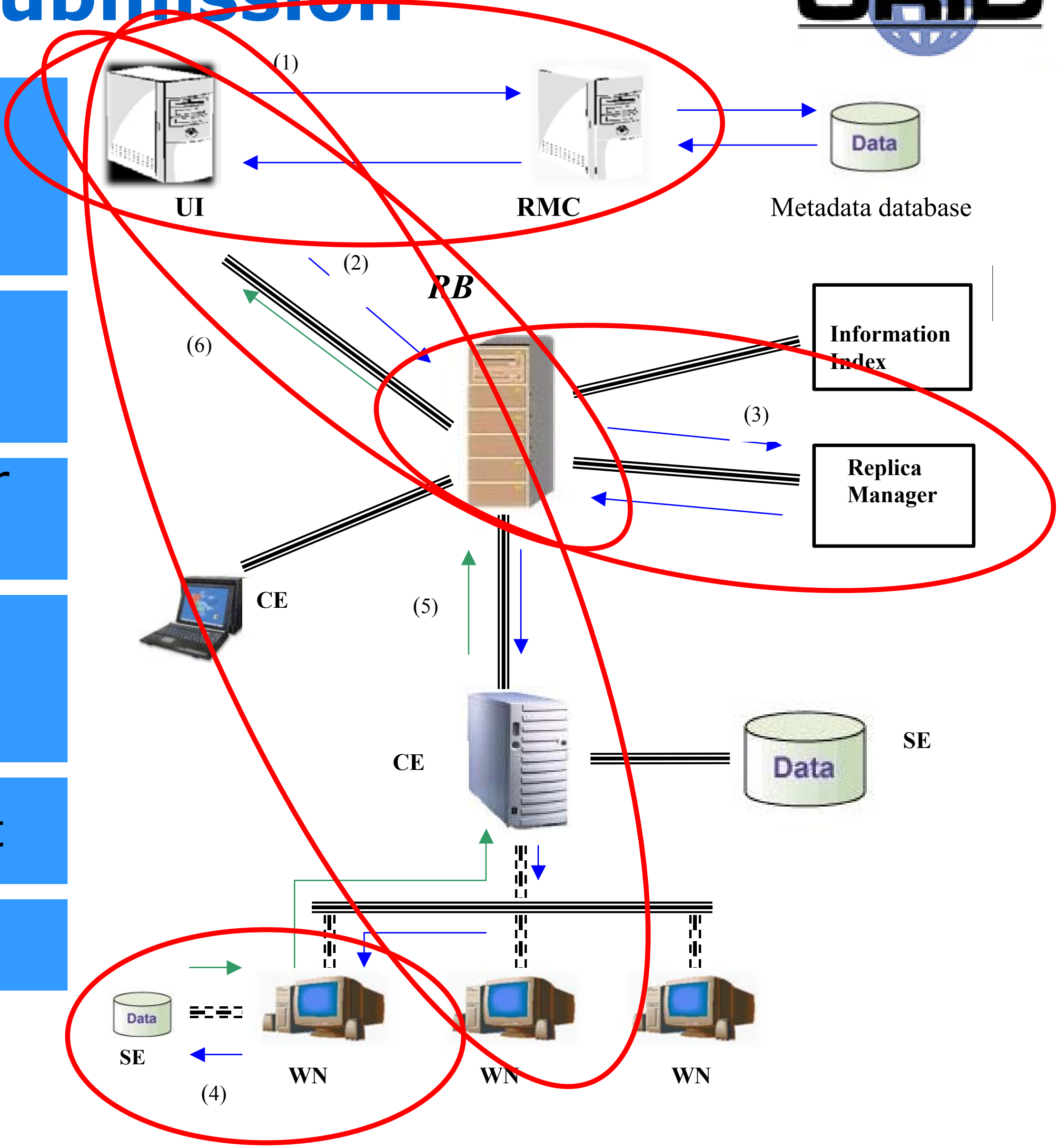
GOME NNO Processing

1. select a LFN from precompiled list of non-processed orbits
2. verify that the Level1 product is replicated on some SE
3. verify the Level2 product has not yet been processed
4. create a file containing the LFN of the Level1 file to be processed
5. create a JDL file, submit the job, monitor execution
6. During processing profiles are registered in RM and metadata is stored in RMC
7. query the RMC for the resulting attributes

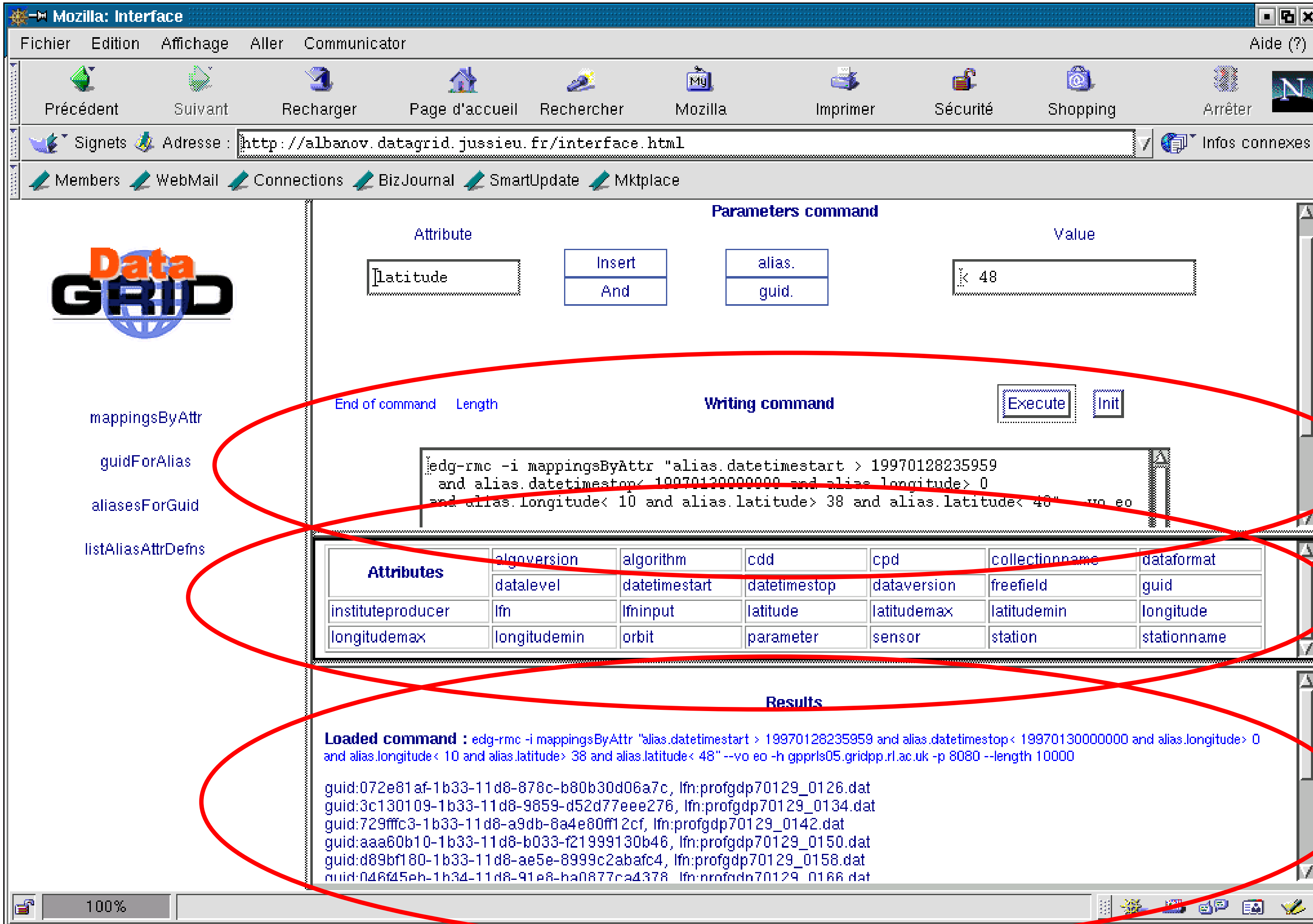


Validation Job submission

1. Query RMC for coincidence data LFNs (Lidar and profile data)
2. Submit job, specifying the LFNs found
3. Get the data location for the LFNs from RM
4. Get the data to the WN from the SE and start calculation
5. Get the output data plot
6. Show the result



RMC usage: attributes



Command area

All attributes Of WP9 RMC

Result area

Metadata tools comparisons

Replica Metadata Catalogue Conclusions, future direction:

- RMC provides possibilities for metadata storage
- Easy to use (CLI and API)
- No additional installation of S/W for user
- RMC performance (response time) is sufficient for EO application usage
- More database functionalities are needed: multiple tables, more data types, polygon queries, restricted access (VO, group, sub-group)

Many thanks to WP2 for helping us preparing the demo

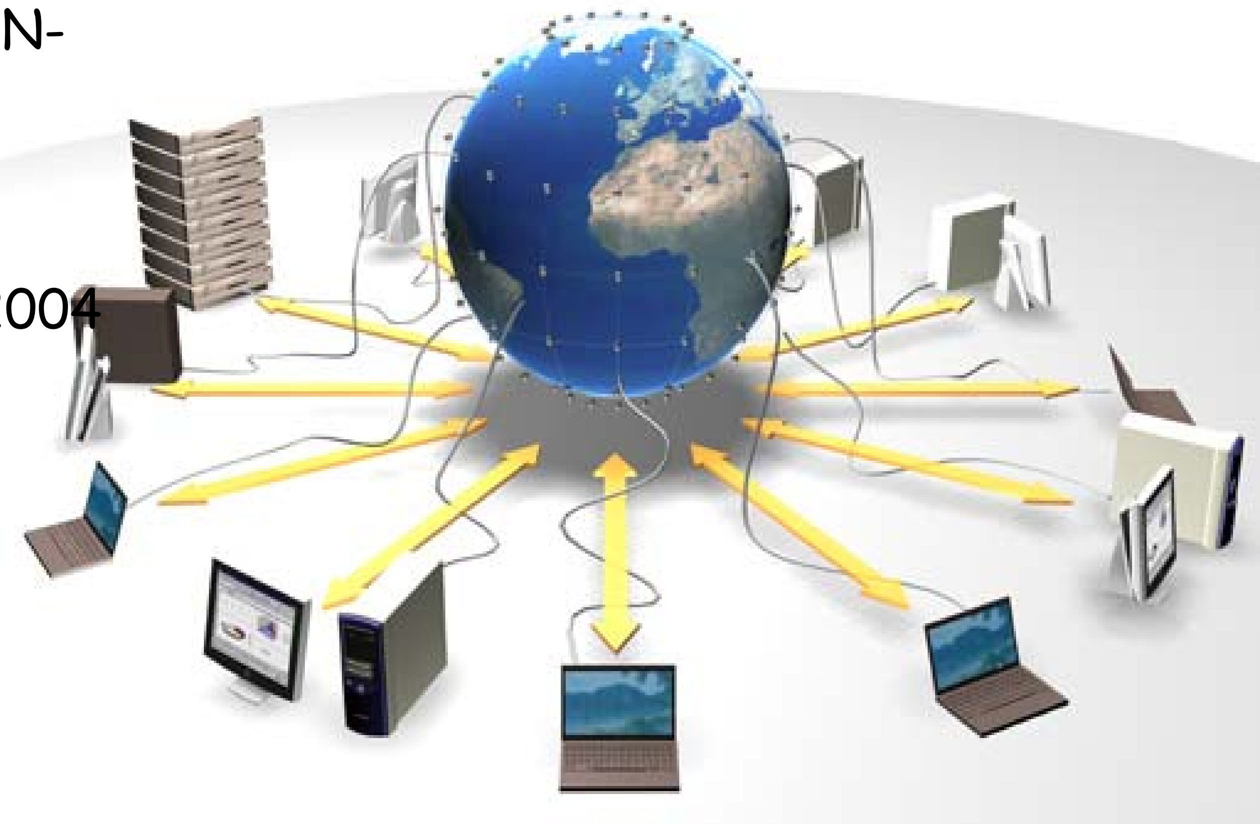
Advanced Scheduling in HEP Applications:

CMS demonstrating DAGMan scheduling

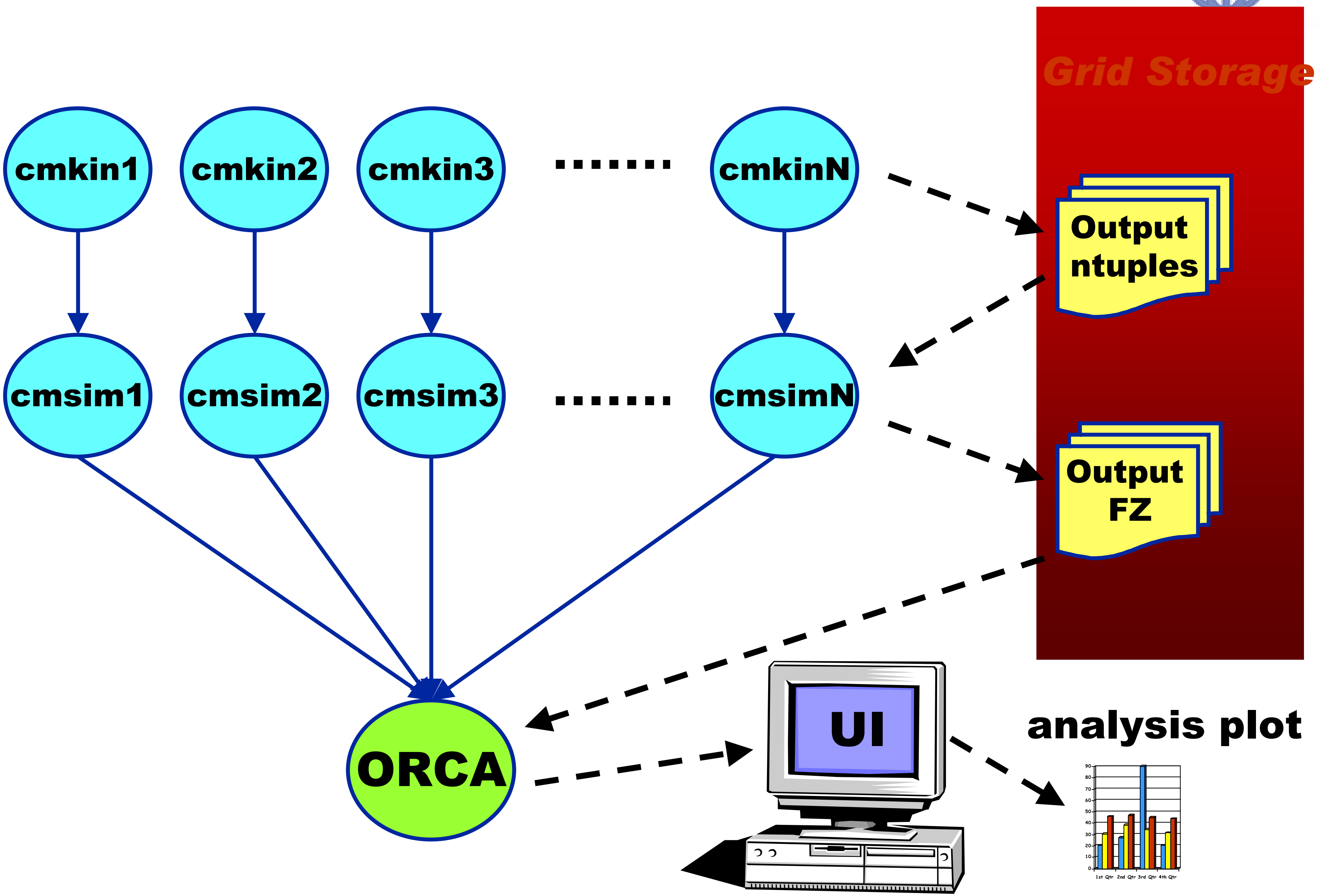
Marco Verlato (INFN-Padova)

EDG Final Review

CERN, February 19, 2004

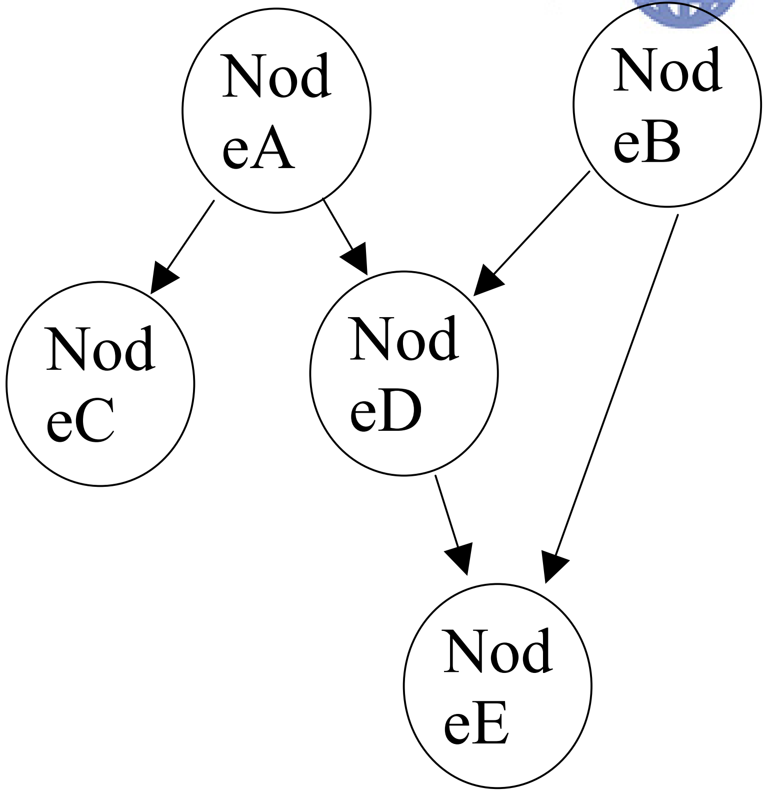


What we are going to see…



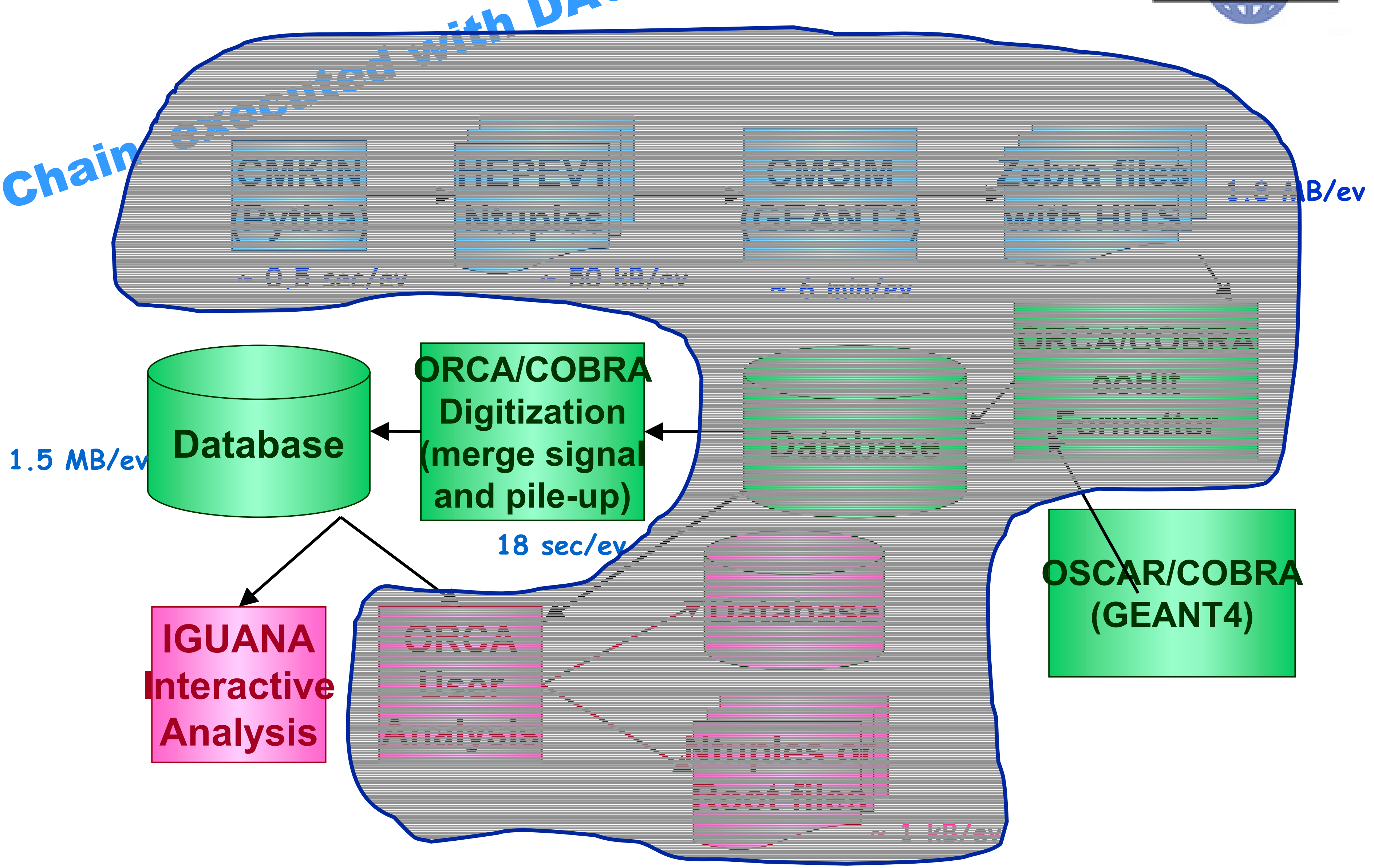
What is a DAG?

- Directed Acyclic Graph
- Each node represents a job
- Each edge represents a (temporal) dependency between two nodes
  - e.g. NodeC starts only after NodeA has finished
- A dependency represents a constraint on the time a node can be executed
  - Limited scope, it may be extended in the future
- Dependencies are represented as "expression lists" in the ClassAd language




```
dependencies = {
   {NodeA, {NodeC,
NodeD}},
   {NodeB, NodeD},
   {{NodeB, NodeD}, NodeE}
}
```

CMS Production chain



DAG support in Workload Management System

- The revised architecture of all WMS components for Release 2 (see D1.4) accomodates the handling of job aggregates and the lifecycle of DAG request
- Definition of DAG representation as JDL and development of an API for managing a generic DAG
- Development of mechanisms to allow sub-job scheduling only when the corresponding DAG node is ready (lazy scheduling)
- Development of a plug-in mapping an EDG DAG submission to a Condor DAG submission
- Improvements of the ClassAd API to better address WMS needs

JDL for CMS-DAG demo

```
[
    type = "dag";
    node_type = "edg-jdl";
    max_nodes_running = 100;
    nodes = [
       cmkin1 = [
        file =“~/CMKIN/QCDbckg_01.jdl”
       ];
       ...
       cmsim1 = [
        file =“~/CMSIM/QCDbckg_01.jdl”
       ];
       ...
       ORCA = [
        file =“~/ANA/Analisys.jdl”;
       ];
    dependencies = {
        { cmkin1, cmsim1 },
        { cmkin2, cmsim2 },
        { cmkin3, cmsim3 },
        { cmkin4, cmsim4 },
        { cmkin5, cmsim5 },
        { {cmsim1, cmsim2, cmsim3, cmsim4, cmsim5}, ORCA}
      }
    ];
  ]
```

Implementation:

- Uses DAGMan, from the Condor project
- The JDL representation of a DAG has been designed by WMS group and contributed back to Condor
- A DAG ad is converted to the original Condor format and executed by DAGMan

Conclusion

- CMS experiment strongly asked for DAG scheduling in WMS, MonteCarlo production system for creating a dataset could greatly improve its efficiency with DAGs
- WMS provided DAG scheduling in Release 3
- We successfully exploited DAG scheduling executing the full CMS production chain (including analysis step) for a "QCD background" dataset sample

Summary

- Demonstrated Grid usage by all application areas
- Focused on 3 general themes
  - Grid support for simulation
    - Medical simulation
  - Advanced functionalities in EDG
    - Metadata handling
    - DAGMan scheduling
  - Grid in production mode
    - ALICE HEP data challenge

Backup slides WP9

EO Metadata usage

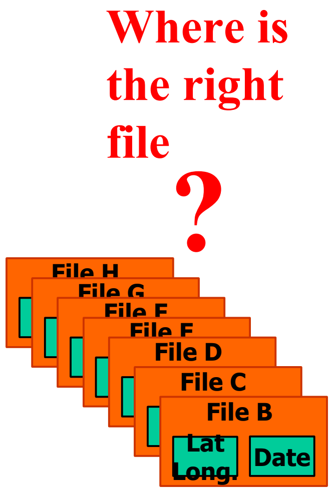
Questions adressed by EO Users:
How to access metadata catalogue using EDG Grid tools?

Context:

- In EO applications, large number of files (millions) with relative small volume.
- How to select data corresponding to given geographical and temporal coordinates?
- Currently, Metadata catalogues are built and queried to find the corresponding files.

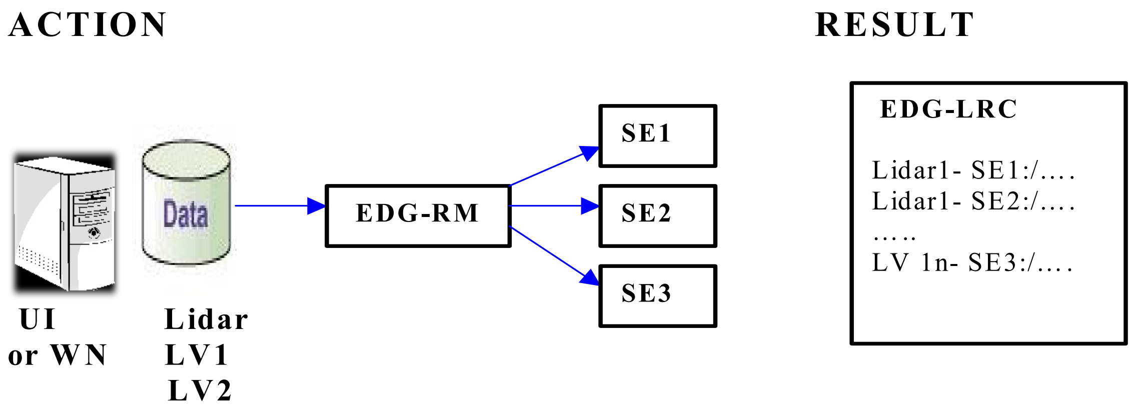
Gome Ozone profile validation Usecase:

- ~28,000 Ozone profiles/day or 14 orbits with 2000 profiles
- Validation with Lidar data from 7 stations worldwide distributed
- Tools available for metadata on the Grid: **RMC**, **Spitfire, Muis** (operational ESA catalogue) via the EO portal



Data and Metadata storage

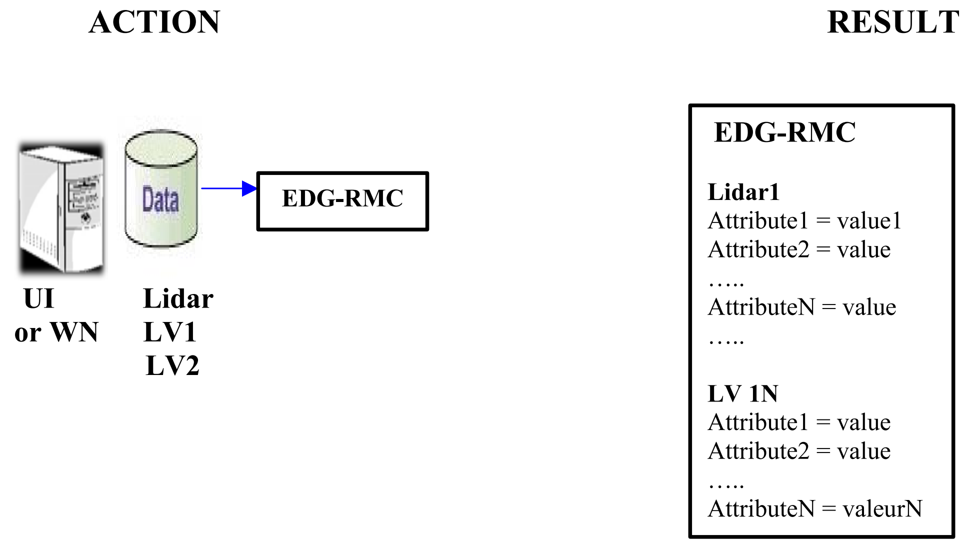
- Data are stored on the SEs, registered using the RM commands:



- Metadata are stored in the RMC, using the RMC commands

Link RM and RMC:

Grid Unique Identifier (GUID)



Usecase: Ozone profile validation

Step 1:	Transfer **Level1 and LIDAR** data to the Grid Storage Element
Step 2:	Register **Level1** data with the Replica Manager Replicate to other SEs if necessary
Step 3:	Submit jobs to process **Level1** data, produce Level2 data
Step 4:	Extract metadata from level 2 data, store it in database using Spitfire, store it in Replica Metadata Catalogue
Step 5:	Transfer **Level2** data products to the Storage Element Register data products with the Replica Manager
Step 6:	Retrieve coincident level 2 data by querying Spitfire database or the Replica Metadata Catalogue
Step 7:	Submit jobs to produce Level-2 / LIDAR **Coincident** data perform **VALIDATION**
Step 8:	Visualize Results

Which metadata tools in EDG?

Spitfire

- Grid enabled middleware service for access to relational databases.
- Supports GSI and VOMS security
- Consists of:
  - the **Spitfire Server** module
    Used to make your database accessible using Tomcat webserver and Java Servlets
  - the **Spitfire Client** libraries
    Used from the Grid to access your database (in Java and C++)

Replica Metadata Catalogue:

- Integral part of the data management services
- Accessible via CLI and API (C++)
- No database management necessary

Both methods are developed by WP2

Focus will be on RMC

Scalability (Demo)

- this demonstrates just one job being submitted and just one orbit is being processed in a very short time
- but the application tools we have developed (e.g. **batch** and **run** scripts) can fully exploit possibilities for parallelism
- they allow to submit and monitor tens or hundreds of jobs in one go
- each job may process tens or hundreds of orbits
- just by adding more LFNs to the list of orbits to be processed
- **batch –b** option specifies the number of orbits / job
- **batch –c** option specifies the number of jobs to generate
- used in this way the Grid allows us to process and register several years of data very quickly
- example: just 47 jobs are needed to process 1 year of data (~4,700 orbits) at 100 orbits per job
- this is very useful when re-processing large historical datasets, for testing differently 'tuned' versions of the same algorithm
- the developed framework can be very easily reused for any kind of job

GOME NNO Processing – Steps 1-2

Step 1) select a LFN from precompiled list of non-processed orbits

```
>head proclist
70104001
70104102
70104184
70105044
70109192
70206062
70220021
70223022
70226040
70227033
```

Step 2) verify that the Level1 product is replicated on some SE

```
>edg-rm --vo=eo lr lfn: 70104001.lv1
```

```
srm://gw35.hep.ph.ic.ac.uk/eo/generated/2003/11/20/file8ab6f428-1b57-11d8-
    b587-e6397029ff70
```

GOME NNO Processing – Steps 3-5

Step 3) verify the Level2 product has not yet been processed

```
>edg-rm --vo=eo lr lfn: 70104001.utv

Lfn does not exist : lfn:70104001.utv
```

Step 4) create a file containing the LFN of the Level1 file to be processed

```
>echo 70104001.lv1 > lfn
```

Step 5) create a JDL file for the job

(the **batch** script outputs the command to be executed)

```
>./batch nno-edg/nno -d jobs -l lfn -t

run jobs/0001/nno.jdl -t
```

GOME NNO Processing – Steps 6-7

Step 6) run the command to submit the job, monitor execution and retrieve results

```
>run jobs/0001/nno.jdl –t
Jan 14 16:28:45 https://boszwijn.nikhef.nl:9000/o1EABxUCrxzthayDTKP4_g
Jan 14 15:31:42 Running grid001.pd.infn.it:2119/jobmanager-pbs-long
Jan 14 15:57:36 Done (Success) Job terminated successfully
Jan 14 16:24:01 Cleared user retrieved output sandbox
```

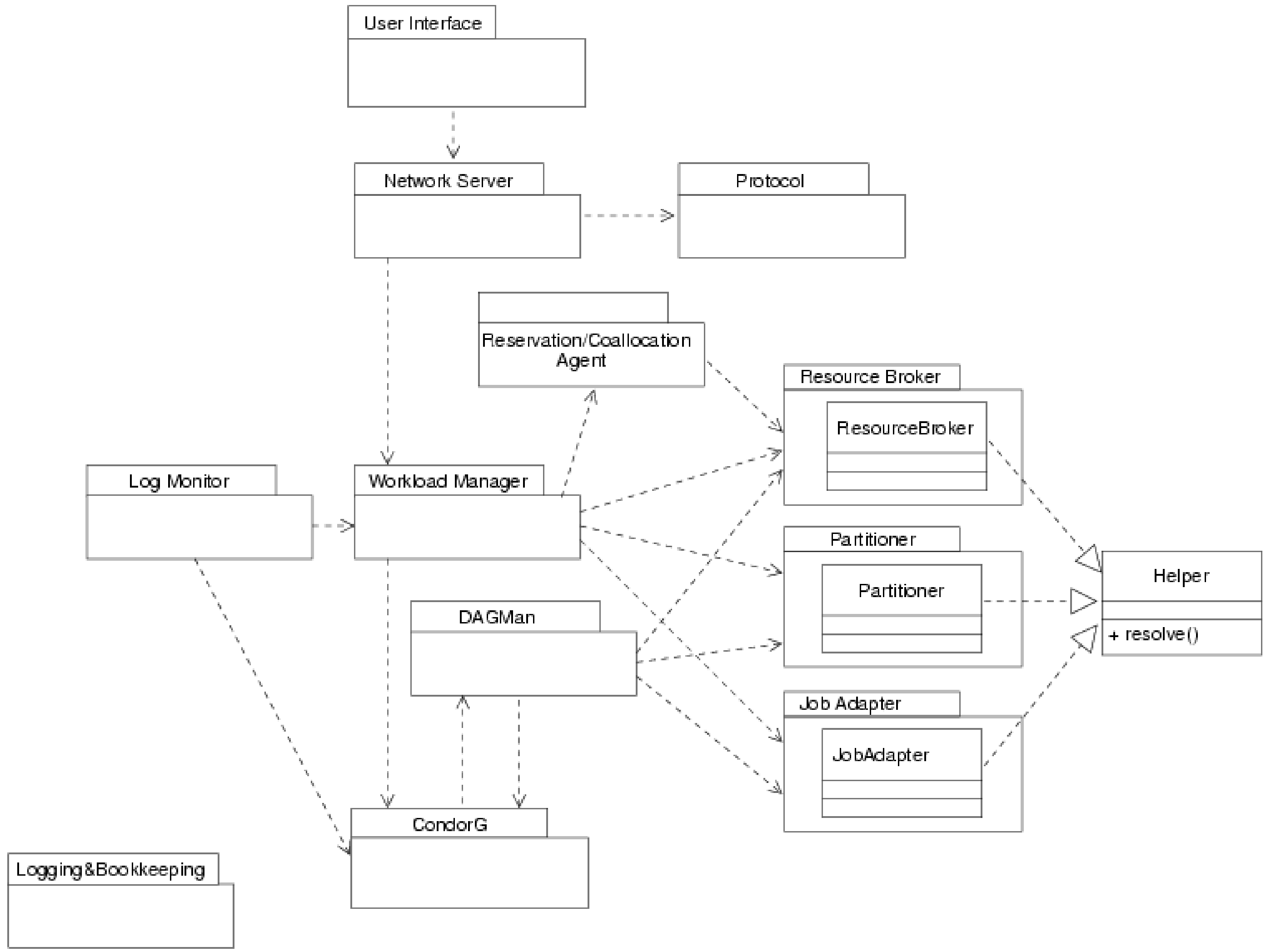
Step 7) query the RMC for the resulting attributes

```
./listAttr 70517153.utv
lfn=70517153.utv
instituteproducer=ESA
algorithm=NNO
datalevel=2
sensor=GOME
orbit=10844
datetimestart=1.9970499E13
datetimestop=1.9970499E13
latitudemax=89.756
latitudemin=-76.5166
longitudemax=354.461
longitudemin=0.1884
```

BACKUP SLIDES

DAGMan

WMS architecture



Job Workflow and Test-bed Layout