



**DELIVERABLE D1.3.1  
APPLICATION DESCRIPTION,  
INCLUDING USE CASES, FOR  
TASK 1.3**

**WP1, Task 1.3, Distributed Data Analysis in HEP**

---

Document Filename:	<b>CG-1-D1.1.3-CSIC003.doc</b>
Work package:	<b>WP1 Cross Grid Application Development</b>
Partner(s):	<b>CSIC, FZK, INP, INS, UAB</b>
Lead Partner:	<b>CSIC</b>
Config ID:	<b>CG-1-D1.1.3-CSIC003</b>
Document classification:	<b>PUBLIC</b>

---

Abstract: This document specifies the software requirements for CrossGrid Task 1.3 'Distributed Data Analysis in HEP'.



**Delivery Slip**

	<b>Name</b>	<b>Partner</b>	<b>Date</b>	<b>Signature</b>
<b>From</b>	Celso Martinez-Rivero	CSIC	08-May-2002	
<b>Verified by</b>				
<b>Approved by</b>				

**Document Log**

<b>Version</b>	<b>Date</b>	<b>Summary of changes</b>	<b>Author</b>
1-0-DRAFT-A	26-Mar-02	Draft versión	Jesús Marco, Rafael Marco, Celso Martínez-Rivero, Oscar Ponce & David Rodríguez
1-0-DRAFT-B	2-Apr-02	Description of services, use cases, HLT section	Jesús Marco, Celso Martínez
1-0-DRAFT-C	7-May-02	HLT section improvement	Tadeusz Szymocha, Piotr Malecki

## CONTENTS

<b>1. INTRODUCTION.....</b>	<b>4</b>
1.1. PURPOSE .....	4
1.2. SCOPE .....	4
1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS .....	5
1.4. REFERENCES .....	6
1.5. OVERVIEW .....	6
<b>2. OVERALL DESCRIPTION.....</b>	<b>7</b>
2.1. PRODUCT PERSPECTIVE .....	7
2.1.1. <i>System interfaces</i> .....	10
2.1.2. <i>User interfaces</i> .....	10
2.1.3. <i>Hardware interfaces</i> .....	11
2.1.4. <i>Software interfaces</i> .....	11
2.1.5. <i>Communications interfaces</i> .....	12
2.1.6. <i>Memory constraints</i> .....	12
2.1.7. <i>Operations</i> .....	12
2.1.8. <i>Site adaptation requirements</i> .....	13
2.2. PRODUCT FUNCTIONS.....	13
2.3. USER CHARACTERISTICS .....	15
2.4. CONSTRAINTS .....	15
2.5. ASSUMPTIONS AND DEPENDENCIES .....	15
2.6. APPORTIONING OF REQUIREMENTS .....	15
<b>3. SPECIFIC REQUIREMENTS.....</b>	<b>16</b>
3.1. EXTERNAL INTERFACES .....	16
3.2. FUNCTIONS .....	16
3.3. PERFORMANCE REQUIREMENTS .....	16
3.4. LOGICAL DATABASE REQUIREMENTS .....	16
3.5. DESIGN CONSTRAINTS.....	16
3.6. STANDARDS COMPLIANCE.....	16
3.7. SOFTWARE SYSTEM ATTRIBUTES .....	16
<b>4. APPENDIXES.....</b>	<b>16</b>

---

## 1. INTRODUCTION

### 1.1. PURPOSE

This document specifies the software requirements for CrossGrid Task 1.3, “Distributed Data Analysis in HEP”, including the two technical subtasks (1.3.1, namely Interactive Distributed Data Access, and 1.3.2, Data Mining Techniques on Grid), the Integration and Deployment task, 1.3.3, and the Application to realistic HEP examples described in task 1.3.4 (Application to HLT and Physics TDR).

The intended audience is the Task itself, Task 1.4.b on the two technical items, the WP2 and WP3 supporting middleware Tasks, and also WP4 regarding testbed setup requirements.

### 1.2. SCOPE

The products of Task 1.3 are:

- (a) An Schema for Interactive Distributed Database Access
- (b) Proposal for Distributed Data Mining Techniques
- (c) Realistic Prototypes for HEP Physics Analysis on Distributed Data

The two first products are proposed to be implemented in their final form in the framework of the OGSA, as Grid Services. This document also suggests possibilities for definition of an Interactive Service in the Grid framework.

#### *Application Perspective:*

The LHC High Energy Physics experiments located at CERN will start data taking in the year 2007. Amongst the four detectors that are being built right now, ATLAS and CMS are general purpose detectors that will be used by world-wide collaborations of around 2000 physicists each; LHCb and ALICE are dedicated to special topics in High Energy Physics; their collaborations are somewhat smaller (hundreds of physicists).

LHC will produce –running at high luminosity- around 800 million events per second. This huge number of events will be reduced to about 100 events that will be saved for future analysis. The filtering techniques that eliminate ‘uninteresting’ events to the appropriate rate are thus mandatory and its optimization will result crucial for the performance of analysis. Furthermore the selected ‘raw’ event will be next reconstructed and transformed in an Analysis Object Data (AOD) and finally into TAG data.

The average size of one raw event is 1 MB (reaching 25 MB for heavy ion collisions in ALICE) being around 10KB for AOD and just 1KB for TAG Data. Assuming  $10^7$  seconds data taking per year yields a total amount of 1 PB raw data recorded per experiment. One has to add to these figures the need of several data re-processing per year plus a still not fully determined amount of simulated events.

One of the main objectives of LHC is to look for the Higgs boson, a particle predicted to be the origin of mass through interaction with other particles. Its lower mass limit has been established by the previous LEP collaborations in  $114.3 \text{ GeV}/c^2$  at 95%CL within the SM assumptions. LHC detectors are optimized to look for this particle in a wide mass range up to 1 TeV.

Well known “Data-mining” techniques such as Neural Networks (a supervised learning technique in the sense that the network is trained over ‘signal’ and ‘background’ events) or clustering techniques like SOM (unsupervised learning due to the inexistence of predefined patrons to follow) are vital for extracting the desired signal events from the huge amount of background events present in data.

This application will thus address several challenging points: access to large **distributed databases** in the Grid environment, development of **distributed data-mining** techniques suited to the HEP field, definition of a layered application structure, flexible enough to adapt to different experimental set-ups and integration of user-friendly **interactive access**, including specific **portal tools**.

The software products proposed to be developed, as Grid services, are (see 2.1):

1. Interactive Session Distributed Database Access Service ( ISDB) and a...
2. Distributed Database Installation, Replication and Stripping Service (DBINST)
3. Interactive Session Distributed Processing (ISDP) implemented on top of Interactive Session Distributed Manager and Worker (ISM, ISW) using MPI or similar message passing interface.
4. Interactive Session Data Mining Service (ISDM), including Neural Network Algorithms.

**Interactive distributed HEP applications will be built with a Portal as UI and employing the previous services coordinated via an adequate high level job description language.**

### 1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

CrossGrid	The EU CrossGrid Project IST-2001-32243
DataGrid/EDG	The EU DataGrid Project IST-2000-25182
GRID	Grid framework for sharing of distributed resources.
DBMS	Database Management System.
O/R DBMS	Object/Relational DBMS.
SQL	Structured Query Language.
MPI	Message Passing Interface.
MPICH-G2	Grid-enabled implementation of MPI.
HTTP	Hypertext Transport Protocol.
XML	Extended Markup Language.
SOAP	Simple Object Access Protocol.
WSDL	Web Service Definition Language.
CE	Computing Element (EDG).
SE	Storage Element (EDG).
SM	Standard Model (in the High Energy Physics context).
NN	Neural Network.
BFGS	Broyden-Fletcher-Goldfarb-Shannon Hybrid linear method for NN's
PAW	Physics Analysis Workstation software.
ROOT	An Object-Oriented Data Analysis Framework.
SOM	Self Organizing Map.
ISM	Interactive Session Manager.
ISW	Interactive Session Worker node.
ISRB	Interactive Session Resource Broker.
ISDB	Interactive Session Database Server.
DBINST	Database Installation, Replica and Stripping Service.

---

## 1.4. REFERENCES

- CrossGrid      Project Technical Annex I (v 3.1)
- DataGrid/EDG   Public Deliverables in <http://www.eu-datagrid.org>
- Grid              Grid framework, Foster & Kesselman
- Grid Phys.      <http://www.globus.org/research/papers/ogsa.pdf>
- Globus           <http://www.globus.org/>
- MPICH-G2      <http://www3.niu.edu/mpi/>
- WSDL           <http://www.w3.org/TR/wSDL>
- MLPfit          <http://schwind.home.cern.ch/schwind/MLPfit.html>
- PAW             <http://wwwinfo.cern.ch/asd/paw/>
- ROOT           <http://root.cern.ch/>
- Data-Mining    T.Hastie,R.Tibishirani,J.Friedman,“The Elements of Statistical Learning”,Springer Series in Statistics, 2001.
- Sleuth: a quasi-model-independent search strategy for new physics; B. Knuteson; hep-ex/0105027
- Search for New Physics in e mu X Data at D0 using Sleuth:  
A quasi-model-independent search strategy for new physics;  
D0 Collaboration; hep-ex/0006011

## 1.5. OVERVIEW

Section 2, and in particular 2.1, provides the overall description with some detail, including the relation to existing and future software, and sets the basis for the Specific Requirements to be detailed in Section 3.

---

## 2. OVERALL DESCRIPTION

### 2.1. PRODUCT PERSPECTIVE

#### ***INTERACTIVE DATA ANALYSIS IN HEP***

Interactive data analysis has a long trajectory in HEP. PAW (Physics Analysis Workstation) was the “integrated product” used in the LEP experiments analysis, for the last 15 years. This product, used in a Fortran77 programming environment, includes several modules tightly coupled, from the I/O binary file access (based on the ZEBRA Fortran I/O product); an integrated script package, KUMAC, with Fortran “interpreter”; a histogramming, fitting and plotting package (HBOOK), etc. This “complete” approach has been continued in the Object Oriented framework adopted for the current (Tevatron at Fermilab) and future (LHC at CERN) Hadron Colliders with the ROOT package. It includes I/O based on an object streaming library, and a full object environment. As it is quite likely that the general I/O package for the LHC experiments will use ROOT as basis, the use of its interactive possibilities is very likely.

In parallel, the possibility to use O/R DBMS, widely extended in the commercial area, has been considered. The development of an independent User Interface, with separate Histogramming and Plotting packages, has been addressed by the LHC Collaborations (an example is the CERN LHC++ software, now referred as ANAPHE, including the Lizard tool for interactive analysis and visualization).

In the development of this HEP Interactive Data Analysis application, distributed in a Grid environment, both approaches will be considered, trying to get the best of both worlds.

A distributed ROOT prototype, PROOF, is being developed currently by the ROOT team. Our work in this line will try to collaborate directly with them, in particular considering the implementation of Distributed Data Mining techniques, like Neural Networks, in this framework.

#### ***USE OF DISTRIBUTED DATABASES (Task 1.3.1)***

On the other hand, the approach based on the use of O/R DBMS will attack in first place the use of distributed databases in the Grid framework. In the DBMS world this could be considered as a task including connection issues regarding security and access architecture, replication, and collective processing. Current commercial solutions ( like IBM Informix XPS ) solve the problem using sophisticated share-nothing architecture, so suited in principle to the Grid scheme with server nodes distributed across different sites. Other O/R DBMS like Oracle 9i require much more tightly coupled hardware and their clustering mechanism does not seem in principle suited to this direct use in the Grid environment.

The proposed solution is flexible and doesn't rely on proprietary techniques: following the approach used in the EDG project, it starts considering a METADATA catalog to provide the global information about the different DATASETS.

DATASETS are the basic information unit, and they are considered as read-only for our physics analysis applications. DATASET description will be based on an XML-Schema, XML-DDL, describing storing of semi-structured information in DB tables (so abstracting DDL for any SQL-99 compliant DBMS). The information stored itself is described also in XML format. This approach allows complete portability, vendor independence, and is supported by the correspondence between O/R features of the DBMS and the semi-structured information representation provided by XML.

Replication of a DATASET will be achieved dumping the DBMS information into an XML file, transferring the resulting compressed files via GridFTP, stripping to distribute homogeneously if needed, and reloading in the target distributed DBMS. This will be implemented as a DBINST (for Installation) service.

The METADATA catalog stores the description for each DATASET including a unique logical name and possibly several physical location names. These names will describe the “service location”. This METADATA catalog will reside itself on another database system, including a secure Grid access, and support for the corresponding Virtual Organization. The proposed service is being developed in the EDG WP2, in relation with the SpitFire project. Our aim here will be the collaboration to make this framework flexible enough to support our DATASET description, and work with the O/R DBMS employed in our project (in principle a generic one under the SQL-99 standard, in practice the first to be tried will be IBM IDS 9.2).

The actual access to the data itself will be described as another service, ISDB, and will work via XML flow both for the queries and the resultset (including in this case the possibility to refer to a local cache service or/and a virtual data service instead of the data itself for efficiency reasons).

### **INTERACTIVE SESSION SERVICES**

Interactive work in the Grid framework is another important issue. With the aim to contribute to a future discussion in the corresponding GGF work group, and based in past experience, we propose an interactive grid framework imposing several restrictions on the resource use, requiring the presence in the VO of an “Interactive Session Service Factory” (ISSF) that provides, via an Interactive Session Resource Broker (ISRB):

1. Allocation of an Interactive Session Manager (ISM) grid node, either in exclusive way to the user (or user group), or by requesting higher priority respect to other service processes (like those derived from a batch-like service, i.e., Computing Element service in the EDG notation). This ISM node plays a central role in the interactive session by:
  - a. establishing an outward connection to the User Interface (Portal) to get the corresponding user input.
  - b. running the main program, and distributing the work across the different INTERACTIVE SESSION WORKER nodes (ISW), via a mechanism like MPI
  - c. collecting and calculating global process quantities (like solving large linear systems for gradient estimation in NN, so requiring a powerful hardware configuration)
  - d. returning the output information to the User Interface
  - e. Accepting from the UI at least the following control directives about the interactive session itself: KEEP-ALIVE (renovation of reservation), CANCEL-CURRENT-PROCESS (the CTRL-C equivalent in current HEP applications), and BYE.
2. Allocation of the INTERACTIVE SERVICE DATABASE SERVERS (ISDB), including installation if required (DBINST, with replication and stripping to support distributed load partitioning)
3. Allocation of the INTERACTIVE SERVICE WORKER (ISW) nodes (one of them being the ISM in most cases)

Our experience indicates that in the interactive analysis in HEP, data location plays a central role: typical interactive requests will run on o(TB) distributed data, meaning transfer/replication times for the whole DATASET of the order of one hour at least or more (1 TByte is transferred in >8000 s. over a Gigabit network). So this operation, if needed, should take place once and in advance to the interactive session. This defines the need to allocate and install and load if needed, the corresponding database servers before the interactive session starts, and for efficiency consider ISM and ISW nodes as close as possible to them (if not the same, in the case of the ISW). Usually this will drive to an scheme where the Interactive Session Resource Broker starts by collecting the information from the UI about the DATASET to be used, locating adequate database servers, and then finding the ISM and ISW. This puts an extra constraint on the role of the ISM: it should be fixed after the database servers are allocated, as these constraint for performance the ISW location, and the communication between the ISM and the ISW should be as fluid as possible.

All this information should be considered by the Interactive Session Resource Broker (ISRB) to decide on an optimal configuration for the Interactive Session. It should be analyzed how such ISRB will extend or coordinate with the Resource Broker proposed in EDG. In this scheme, the role of Job Submission Services could be marginal as competitive shared direct (secure, using Grid authorization and authentication, but direct remote execution) access could be granted to interactive users, and the Scheduling Agent (task 3.2 in WP3) would be the key piece taking care of the technical work to be done by the Interactive Session Resource Broker to optimize the (pre-)allocation, including a possible pre-installation of DATASETS based on previous and future estimated use inside the V.O.

As an example, an analysis group in an LHC collaboration working on a common physics topic (i.e. Higgs search via muon signature) will usually work on a predefined DATASET with an original physical database plus a couple of replications distributed across their VO distributed resources structure (provided by the “testbed” in our project). One or several of these physical DATASETS will be distributed in one or several powerful clusters, allocated mainly for interactive use by the VO.

### ***DISTRIBUTED DATA-MINING TECHNIQUES (Task 1.3.2)***

Data-Mining services based on two different mathematical techniques, supervised and unsupervised learning (see reference) and two different architecture approach (processing in worker nodes vs processing in the DBMS side) are considered.

Neural Networks are a powerful method for signal versus background discrimination. Many different algorithms exist, and we will restrict initially to MLP and the BFGS method (allowing a simple parallelization based on load partitioning, as we will see). The final particular implementation is not yet chosen, for the first checks we will use the MLPfit package (ref MLPfit).

The process involves first several training steps, taking most of the time, to optimize the NN architecture, with control tests in parallel. After the NN is defined, its application is very simple: compute the NN output function defined by its weights, on the data considered. Unfortunately, the training takes a sizeable time for the data sizes considered. Already for the LEP era, a typical NN with a 10-15-15-1 architecture was trained on datasets with around 1M events, taking few hours to few days to reach a reasonable error level after  $o(10^3)$  epochs in a PIII running at 600 MHz. In an interactive work, and considering the increased size of the datasets for LHC, the objective of training the NN in few minutes makes mandatory the distribution of the work across  $o(10-100)$  nodes.

The process is implemented with a central program, running on the ISM node, that distributes via MPI among the ISW nodes the data processing (basically for calculating the partial “error” over the corresponding data sample given the NN architecture and the weight parameters transferred from the ISM). The ISM collects all the errors, estimates the gradient and proposes the next step as a variation in the NN weights. The iteration is repeated many times until convergence is reached (i.e. very small changes in the total error from one epoch to the following one).

Load balance is achieved through an intelligent partitioning, that in practice, given the completely arbitrary pre-filters that can be applied on the total dataset, can only be assured by stripping the data across the database servers.

Optimization of resources use is based on a compromise between node processing power and distributed information exchange over the network overhead. The NN parallelization scales perfectly from 2 to 100 nodes in a local cluster with fast ethernet, but requires low latency and reasonable high bandwidth when some of the nodes are not joined by local fast ethernet. An adaptive technique can be considered in this case.

Processing on the database server side is possible by implementing the NN error algorithm using an “stored procedure”. This will be tried for performance comparison, however the absence of an standard could result in higher database migration costs.

---

Unsupervised learning techniques, in particular Self-Organizing Maps, will be addressed in task 1.4.b, but the scheme follows closely what proposed here for the NN.

### 2.1.1. System interfaces

Interactive Session Access to Distributed Databases Service ( ISDB)

Query Catalog to get DATASETS and corresponding Database services

Installation, including replication and stripping to Interactive Session Resources (ISDBINST)

Query Database service in XML Format

Return ResultSet as

XML ResultSet if size < o(100 MB)

XML handle for ResultSet in cache

XML Query handle for virtual data

Interactive Session Distributed Processing (ISDP)

Process simple program in all ISW Worker nodes and collect in simple way output in the ISM Manager node, using information in a given DATASET.

Interactive Session Data Mining

Specialized version of the previous service, with external parameters reduced to the NN architecture or the SOM grid structure, and returning the NN weights.

Interactive Session Resource Brokering

This system interface will be hopefully developed in coordination with WP3, task 3.2.

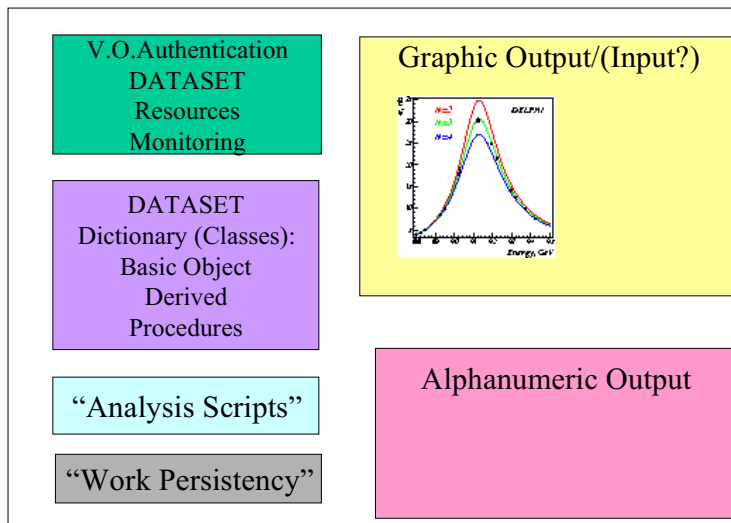
### 2.1.2. User interfaces

UI will be based on use of a Web Browser connecting to an Interactive User Portal (task 3.1), that will provide several panels/ applets using XML for both description and flow :

- Authentication panel
- Application Monitoring (in Grid) panel
- Data panel:
  - DATASETS, including access to the VO metadata catalog
  - Object/Class Dictionary, with base variables, derived, procedures, etc... subpanel.
- Scripts Panel (cascading to SQL/C/Java filters and procedures)
- Graphical Output (& Input?) Panel ( SVG based)
- Alphanumeric Output Panel (& Data Browser)

The initial connection to an VO Interactive Session Resource Broker will be handled by the UI, allowing user authentication and as first step the DATASET selection. After the resources are assigned, we expect a connection to be kept open with the ISM and used for the following interactions.

For a “user-desire picture” of the proposed User Interface, see next figure.



### 2.1.3. Hardware interfaces

Standard hardware proposed for use in LHC is based on IA32 with Linux RedHat as OS.

Two critical points are data storage and network connection (but for this last point see 2.1.5). Data is stored in O/R DBMS (IBM IDS, ORACLE, MySQL...) or in streamed object libraries (ROOT). Physically, it will reside on distributed on-line storage, mainly hard disks (we will not consider tape storage in interactive analysis; an adequate staging mechanism, completely transparent to the user and not addressed by this task is assumed). Efficiency in data access and processing depends upon the local hardware storage configuration: local SCSI or IDE disks, NAS or SAN servers, etc. We expect to get performance monitoring but specific hardware benchmarks are not yet defined. Same is also true for the pure processing (CPU) power, and in particular for dual CPU configuration.

More critical information on the hardware status is expected via the monitoring mechanisms (task 3.3), in particular for possible decision while waiting for a MPI collective regarding dead or extremely slow nodes.

### 2.1.4. Software interfaces

Globus (version 2.0 and future ones) is the underlying package providing basic services, from authentication and security to basic data transport (GridFTP).

MPI (MPICH-G2) will be used at basic level for distribution of processes, i.e., in the ISDP (Interactive Session Distributed Process service).

From EDG we expect to reuse part of the Resource Manager, and the Replica Manager (including the Spitfire mechanism) in the ISDB services.

From CrossGrid WP3 we expect to use the Portal tools (3.1), the Scheduling Agents inside the Interactive Session Resource Broker (3.2), the Monitoring tool for network and nodes operation (3.3).

---

Along development of the application, the tools from WP2 for MPI verification (2.2), and also for Monitoring and Benchmarking will be used.

### 2.1.5. Communications interfaces

Distributed processes will communicate over the usual local or WAN Ethernet, over TCP/IP. Fast Ethernet is assumed in the local clusters of testbed sites, even with Gigabit connections for Database servers (not critical).

It is quite likely that QoS regarding MPI messages will be considered.

### 2.1.6. Memory constraints

The initial tests will be based on existing LEP events with an average database size of  $o(100)$  Gb. NN's will be trained and tested over this sample, but their architecture will not reach very high dimension (typically 10-20-20-1 meaning less than  $o(1K)$  weights). So we do not expect any significant memory constraint.

Next prototypes will run on  $o(Tb)$  databases, and more complicated NN architectures will also be considered. We do not have yet estimated the possible memory constraints.

Usual hardware nodes have more than 512MB for RAM, and enough swap space to allow any user to run a typical HEP application.

### 2.1.7. Operations

In addition to what described in sections 2.1 and 2.1.2, we detail here a scheme of operations. First of all, the user identifies itself to the VO via the Portal. Once accepted, the user sends a request to use a given DATASET to the Interactive Session Resource Broker (step 1). The ISRB asks the Replica Manager (step 2) to find the physical DATASET locations. Using this answer (step 3a) selects Manager (ISM) and Working (ISW) nodes to maximize performance taking into account this location of the ISDB service. If no satisfactory resource configuration is found, the broker may ask for installation to the corresponding DBINST service (3b), knowing already that enough resources will be available for ISW and ISM nodes. Note that ISW and ISDB services could be implemented on the same machine, and also the ISM node can be used as ISW if needed. It could happen that a successful selection of interactive resources is not achieved (this should be notified to the UI (step 4)), but in a positive situation the ISRB transfers control to the ISM (Manager node) (step 5) which opens an outward channel to the UI, to handle XML input describing the user job (step 6). The ISM distributes the process to the ISW worker nodes via MPI (step 7) who in turn gets the corresponding data from the ISDB service (step 8). Upon completion, the ISM returns the output to the UI (step 9).

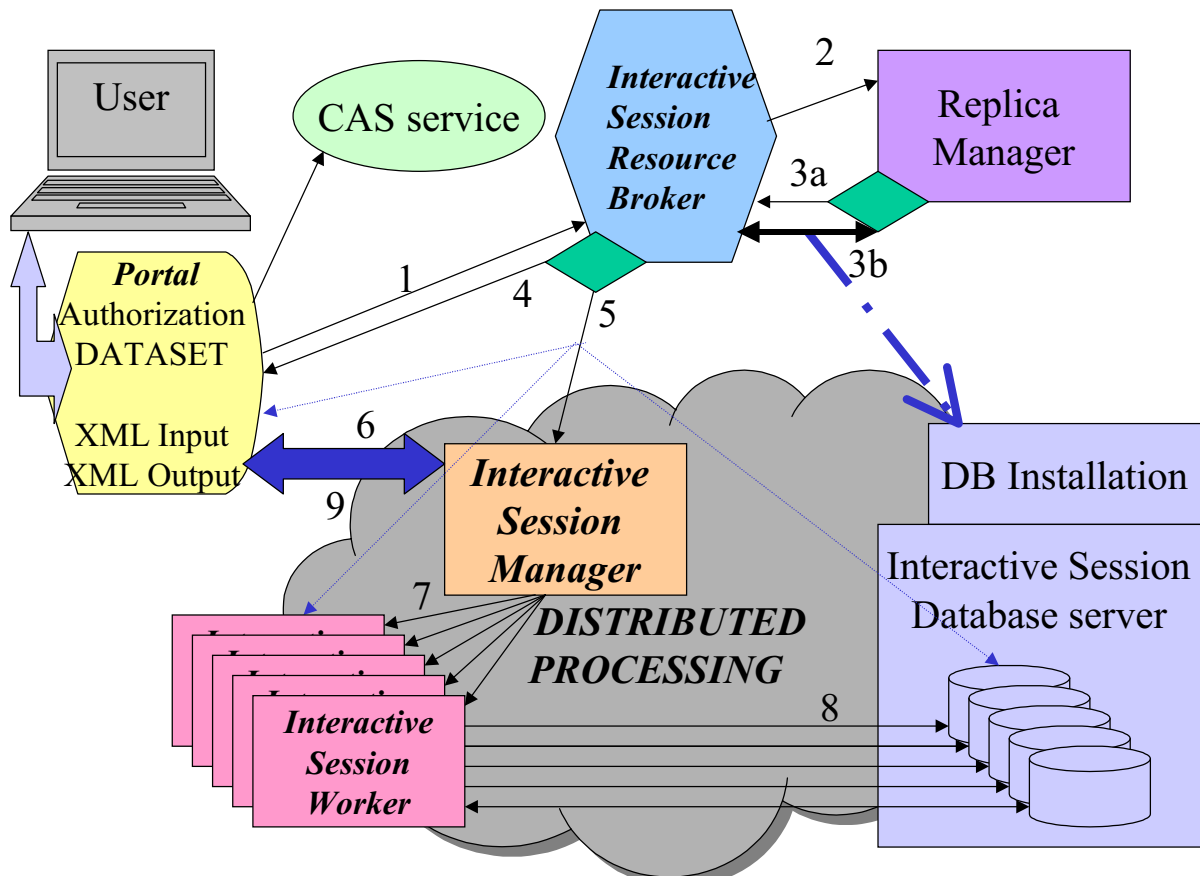


Figure 1.3.2: Description of operations (see text)

### 2.1.8. Site adaptation requirements

Grid nodes placed in a medium-large size cluster may not have public IP addresses needed for MPI. This in practice would require a ISM per cluster (similar so to a gatekeeper role), and the ISM “boss” would take care of the combination. Also service ports should be opened in the firewalls, or the application should be redirected to use the available ports.

## 2.2. PRODUCT FUNCTIONS (TASK 1.3.4)

Final user applications in HEP run on filtered set of events; the LHC collider will operate at an extremely high rate of almost  $10^9$  proton-proton collisions per second. Most of these correspond to

well-known processes and are, therefore, of low interest; only a small number of these interactions will be induced by new phenomena. A fundamental discovery is possible only if these rare processes are efficiently extracted from this huge background of ordinary interactions. The allowed event rate written to tape is about 100 Hz so that a reduction of  $10^7$  has to be done by the multilevel trigger system of each experiment. The low-level triggers are based on fast hardware solutions. Accepted events are processed subsequently by the high level triggers (HLT) where sophisticated algorithms have to be used in order to attain high efficiency for events showing the desired features. The quality of these algorithms has to be almost as good as the quality of the off-line analysis chain. It implies that a kind of simplified event reconstruction has to be applied at the trigger level before writing the event into tape. The time to make the decision at the last trigger levels is of the order of a few hundredths of milliseconds. Optimisation of these sophisticated algorithms is a mandatory step previous to the study of the final physics expectations from LHC experiments employing simulated data, the objective of the Physics Technical Design Report that should be completed by 2004. These results will require the input of DataGrid simulated and processed MC samples, while the corresponding final user analysis is a real challenge for our CrossGrid project.

In parallel, we plan to use the previous tools to search for new physics in HEP data using a quasi model independent method (the only assumption about new physics that will be made is that objects (like leptons, jets) produced will have a hard transverse momentum spectra) following the idea implemented in the D0 experiment (see references).

Another interesting issue is the time needed to process in these High Level Triggers a particular event. Though the average time is usually less than a second it is possible to have large queues with events being processed for more than 10 seconds. These particular events could possibly over-saturate and block the standard trigger farms being a solution to send these events for triggering to remote processing sites. In this way, the fast links leading to the blocked processors at CERN farms would be liberated.

In our project we will need to modify the software in charge of dispatching events to the processors in the farms. Currently, it is not clear whether this software will be able to directly manage processors from remote CrossGrid farms or it will have to communicate with the CrossGrid Resource Broker. To efficiently dispatch events, relevant info on CPU's and available bandwidth in both directions will be required.

When requesting processors from the CrossGrid certain conditions have to be fulfilled. To perform selection algorithms within limited time (few seconds) we will need fast processors and efficient on-line access to databases. Before the remote processors will be able to digest events, we have to make sure that the latest calibration and alignment databases are transferred to their locations, and executing algorithms will have full and fast access to them. Similarly the latest versions of the selection algorithms have to be transferred to the processors. All this preparatory work will be done using the CrossGrid tools.

Once we have demonstrated feasibility of remote HLT filtering, we could use it in the following cases listed below:

1. Events with long processing times can be sent to remote farms.
2. During installation, first working periods of the LHC require higher data acquisition rates for testing and commissioning purposes.
3. The accelerator luminosity (and consequently experiments event size and rate) is decreasing during the run. Some extra processors for processing algorithms at the beginning of the run can be bought at the CrossGrid farms periodically.
4. During exploitation of the LHC it might be desirable to change run parameters of an accelerator in searches for the new physics. Extra processors for longer and more sophisticated algorithms can be bought and used in the CrossGrid farms.

---

## 2.3. USER CHARACTERISTICS

The user characteristics are those corresponding to a research physicist in the field of High Energy Physics, aiming to perform an interactive analysis running over real and simulated data of an LHC detector. Technical expertise will vary from experts to novice users. Common feature is tendency to exhaust all available resources, and extreme impatience to get the results.

## 2.4. CONSTRAINTS

- a) Regulatory policies: None
- b) Hardware limitations: Interactive Response range is 1 second to 5 minutes
- c) Interfaces to other applications: common distributed data mining techniques with 1.4.b
- d) Parallel operation : MPI will be used over Globus.
- e) Audit functions: tbd
- f) Control functions: tbd
- g) Higher-order language requirements. C, C++, Java
- h) Signal handshake protocols: None of low level
- i) Reliability requirements. Weak, restart and run again is ok for an interactive physics analysis application as long as it usually provides a fast and reliable answer
- j) Criticality of the application: Weak
- k) Safety and security considerations. Data will belong to the different LHC collaboration and thus will be private to each of them. Same for original physics analysis methods, here groups could require sub VO access policies
- l) Network throughput and latency: minimal network bandwidth of 100 Mbps shared, for 10Mbps allocated, is mandatory for database replication; latency as low as possible (10-100 ms at most) needed between manager and worker nodes communicating via MPI.

## 2.5. ASSUMPTIONS AND DEPENDENCIES

The application relies first of all on the possible implementation of Grid Services in Globus 3 (OGSA) for the global scheme. It also relies on the EDG middleware, in particular for the REPLICA catalog integrated via the Spitfire project. These parts will be needed from month 12 on, to allow an easy integration with the general GRID middleware.

The main dependence however is on other tasks from CrossGrid, in WP2 and WP3, as stated before. Even so, simple dummy services replacing the Interactive Session Resource Broker and the Portal, could be used if these software pieces are delayed, and for the prototype development.

## 2.6. APPORTIONING OF REQUIREMENTS

Along the software development, we will start with the basic features, allowing the prototype to run in a local cluster in distributed mode by the end of the first year.

This applies in particular to the automatic database installation services, including replication and stripping, and the use of an automatic interactive session resource broker .

---

### 3. APPENDIXES

#### 3.1.1. APPENDIX A : Simple Initial Use Case

*(With underlying possible processes indicated in italic)*

The application will be used to analyse a pre-filtered real data sample with 10M events in the search for the Higgs boson, where 10 produced events are expected. For each event, about 10KB of relevant analysis information are available after offline reconstruction of the event. Background estimation requires 10 times the real data statistics. Signal simulation requires 1000 times the expected signal statistics times the number of different parameters (like Higgs mass). Total information is in the 1-10 TB range.

The data is stored in a database/file server at CERN or at any LHC Tier1 center, and accessible inside a V.O. framework corresponding to an LHC collaboration.

*An XML Scheme for the DATASET is prepared including all the relevant variables. The first instance of the DATASET in the O/R DBMS is created by reading the reconstructed data copied and processed file by file from the database/file server at a Tier1 to an interactive database server service (location provided by the ISRB). This will be a local cluster, managed by the physics analysis relevant group, with enough resources to keep on-line this DATASET. The DATASET description includes different labels for the real data, different background and different signal samples. The DATASET is registered in the METADATA catalog.*

*Example:*

*VO: CMS*

*DATASET description:Higgs signal: H into ZZ , with ZZ decaying into four muons, pythia 6.158 generator, A processing, year 2007, produced in IFCA(Santander), tag data level.*

*Logical file name: cms/tag/2007/procA/higgs/m130/ifca/signal/H4mu/pythia6158*

*Physical file names:*

*<https://grid021.ifca.unican.es/cms/tag/2007/procA/higgs/m130/signal/H4mu>*

*<https://test001.cern.ch/cms/tag/2007/procA/higgs/m130/signal/H4mu>*

*Nickname(unique): tag-2007-procA-Higgs-m130-H4mu*

An user (HEP researcher) joins the V.O. Portal, authenticates himself using his/her certificate.

The user selects the relevant DATASET (from a list built from the information in the METADATA catalog) in the corresponding Portal panel. He/she agrees with the definition of data, background and signal and corresponding cross-section estimations.

*The Interactive Session Resource Broker asks the REPLICA manager to provide the physical location of IS DB servers with the requested DATASET. It scans the list, to match adequate resources to process the DATASET in distributed way: available ISM and ISW nodes close (in network terms) to the DATASET location. In particular ISW nodes could match ISDB servers. If not such configuration is available, the ISRB considers either a new Installation or remote access from ISW to the ISDB nodes.*

The user selects the relevant variables in the DATASET Dictionary (basic like identified muon Pt, or derived like the invariant mass of two most energetic muons), with the action simple plot in the script, and sends the request.

*The ISM receives the XML input from the UI, and sends a simple distributed job that at each node simple fills a histogram using the answer to a single query requesting the relevant variable to the*

*corresponding ISDB server ; it collects the information from all nodes to build the global histogram. Then it sends this histogram back to the UI, either as XML or in graphical form (SVG).*

The user sees in the Graphical Panel the requested plot in less than 10 seconds.

Now the user decides to apply a filter to the data, specified also by conditions in the DATA Panel, like  $PTMuon1 > 100$  , and repeat the plot. It should take around 30 seconds.

*The ISM receives the XML input; the filter is included in the query to the ISDB servers from the ISW nodes.*

Now the user decides to try a Neural Network with 20 variables, with a 20-40-40-1 architecture. He picks the variables in the variables Panel, fixes the NN architecture in the script describing the job, and requests graphical output for the training and test process.

*The ISM receives the request, and runs the NN master program with MPI calls to the ISW nodes; the relevant variables are selected from the database in the first step, stored in memory, and used at each subsequent step. NN weights are received at each ISW node from the ISM at each step, and additive error transmitted back at the end of it. With the error value a new plot is produced and sent to the UI after each step. Also the error on the distributed test sample is obtained in the same way.*

The user sees the training plot in the graphical output panel. The plot does not seem to converge, and after 1 minute he cancels the job (CTRL-C).

*The ISM checks after each step for a message from the Portal to stop the process, if so the process is stopped . Similarly, the ISM checks after ending the job for the KeepAlive signal. If not present, the connection to the UI is released, and the allocation state changed in the Interactive Session Resource Manager.*

The user sends several more NN training jobs, each one taking 5-10 minutes at most, and finally after selecting the best one, checks the results on the real data. Eventually 5 real data events are selected by the NN as Higgs candidates. The user sends a request for detailed information on these events, and browses the relevant information in the alphanumeric output panel.

*The ISM receives the request and propagates it to the entire DATASET via the ISW, collecting the resultset in XML format and sending it back to the Portal. The user sees this output after being transformed via XSLT.*

The user is excited, and would like to save this information (plots, description of the events, the NN, etc) for later use. He is prompted to do so either in the VO, or to his own laptop/desktop/PDA...

*All XML flow between the UI and the ISM is stored as an XML session in the VO web server where the Portal is running. This XML file can be postprocessed or edited and then saved to store the session output. For each user, these files are automatically backed up, but can be also downloaded and saved in their own machine.*

The user starts to work for the paper, and needs tables, plots, etc. He can recover all this info from his private/group VO repository, and also work further from the previous session by revising the script and redoing or skipping the steps.