CNRS
Centre National de la Recherche Scientifique

INFN
Istituto Nazionale di Fisica Nucleare

The Advanced Virgo Computing Infrastructure's Implementation Plan. Draft Version 0.1

TDS number: VIR-0177A-14

Authors: The Virgo Collaboration
Date : April 6, 2014

Abstract: We present here the "Implementation Plan" which describes technical solutions for the Computing model of Advanced Virgo

VIRGO Collaboration
EGO - Via E. Amaldi - I-56021 S. Stefano a Macerata, Cascina (Pisa)
Secretariat: Telephone (39) 050 752 511 - FAX (39) 050 752 550 - e-mail perus@ego-gw.it

Contents

Chapter 1

Structure of this document

1.1 Introduction

This document is the Advanced Virgo Computing Model Implementation Plan. It describes the way as we would like to build up our new, improve our current and arrive to the planned Advanced Virgo computing infrastructure.

The document is organized in sections corresponding to bigger areas of computing.

In each chapter we repeat some of the decision and planning written in the Computing Plan. In order to make the argument behind various decision as clear as possible and to make it possible that one can always refeer back to the discussions, decisions, we always list the requirements which motivated, guided our investigations and on which our decision is based.

To make the whole process trackable and managable and to ensure that various dependent activities are not blocking each other we assign a Redmine Tasks (see later) to each of the steps for a specific issue and we update this document and the corresponding redmine entry whenever anything related is changed.

This way one can immediately get a picure of readiness reading any part of this document.

"A plan without breakdown of tasks, specification of the timing, or without manpower estimation will always remain a dream...."

1.2 For the contributors to this document

This document will be edited by several people. As such we need to ensure that the style, logic and structure will be the same from the beginning to the end. This is why all the contributor should follow the following 'template' when writing up or contributing to same part.

Each section / issue should follow the following layout:

1. List the requirements, references to CM
2. Explain the decision
3. List the necessary steps needed to reach the gols (and introduce them to Redmine)
4. Define a deadline
5. List and update the status of the tasks

Chapter 2

Project management, communication, web

2.1 Introduction

In order to face the challanges set up by the increased and more complex computational problems, some development and changes has to be put in place regarding project management of computing issues.

Currently there are various groups in the Collaboration handling, or involved in computing issues, but unfortunately their activity is just loosely synchronized.

With the changes described in this section, our goal would be to ensure that these activities get grouped, strengthening each other and saving a lot of duplicated work.

When defining the necessary steps for the above golas we have the following considerations in mind:

1. Collaborators has to know about each other activity
2. Work dependencies has to ba handled, bottlenecks and manpower limitations has to be identified.
3. Better and more frequent communication channels has to be establised
4. Minimize the number of custom developments, use industry standard solutions whenever possible.

2.2 Project management tool

Any computing project needs a tool which helps people plan, track, update, follow the activities and make it easily overviewable for other collaboration memebers. So far there was no such tool used in Virgo for computing issues, only a bug tracker (SPR [3]) was used which is - after examining it - turned out to be insufficient for the above purpose.

We have examined, tested and evaulated possible tool to serve as a project management tool including Trello [4], LogBook [5], Redmine [12], SVN Trac [?].

The evaulation of these tools has been performed the results is presented on VDASC meeting 2012/12/03 and Redmine has been choosen, installed and now is in use.

Once the Redmine use mode will be better understood after some experience with it, the possibility of merging (and existing entries database porting) with the SPR system will be evaluated.

Documented in **rt#46**.
Deadline: 2014/01/30
Status:

- **rt#46**: Completed.

2.3 Teleconferencing tool

So far SeeVogh [8] was used as a remote communication tool for VDASC meeting, however many of us offten experienced problem in installing, using it. Also the sound quality was in some cases not satisfactory.

For this reason we (for the moment only the VDASC group) investigated the possibility of using an other tool. Apart from Skype [7] which is not robust enough for large number of participants TeamSpeak [9] was considered.

We have organized a series of test during the VDASC call and we found that TeamSpeak is far more robust and has much better sound quality. We have choosen TeamSpeak to be the regular telecommunication tool for VDASC meeting. **rt#47**

Currently we are using LIGO TeamSpeak server installation, but on the long term Virgo also needs an own installtion of the service **rt#48**

TeamSpeak is currently lacking functionalities such as the desktop sharing. Investigation will be done to check with TeamSpeak development team if such funtionalities are part of their foreseen future extentions.

Any further and wider use of TeamSpeak within the Ligo - Virgo Collaboration will of course be synchronized and aligned with the recommendation of the LSC-Virgo Remote Participation Group.

Deadline: 2014/03/15
Status:

- **rt#47** Completed
- **rt#48** Completed
- **rt#51** In progress

2.4 Web page reorganisation

2.4.1 Motivation

Virgo is suffering from serious lack of manpower. There is no sufficient funding to hire people, thus the only way out is to attract volunteers. Volunteers, students will come only if they feel that this is an attractive (both in term of science and possibilities), cool experiment. The most straightforward way to judge this is to check out the web page of the experiment

2.4.2 The issue

Virgo experiment web pages are not attractive they are out-of-fashion in terms of design, usability and up-to-dateness. Information for collaboration members, outsiders and media furthermore internal and public web pages are mixed. It is difficult to navigate, find, edit and organize information. Different sub-web pages have different stlyle and organisation not easy to follow for somebody not checking them every day. There is a lot of unnecessary - or publicely known - password, languages, styles and hierarchy levels mixed. The issue is much more severe and important than it sounds for the first time. Actually, it is not about the web page itself but instead about the future manpower of Advanced Virgo, as such it must to be solved immediately !

2.4.3 Actions needed

1. Set up a work group which defines a single top level domain and the structure of the new web pages.
2. The work group should collect the requirements for: a.) web page update, b.) and access use cases of various working groups and member institutions.
3. A professional (group of) web expert should select a (content management) tool which fulfils these requirments.
4. All the interested group should upload and regularly update the content.
5. There should be a person in charge who regularly checks (let's say once a month) the quality of the web pages.

The above steps are introduce into Redmine as **rt#49**. Further issues will be added when consensus on the specific steps has been achieved.
Deadline: 2014/12/30
Status:

- **rt#49** In progress

2.5 Other web improvements

As of today a large fraction of the scientist is regularly using mobile devices to connect to web pages, email boxes, news, etc... As such it would be really useful to have a mobile / smart phone friendly version of the most important EGO - Virgo web pages, cluster and detector monitoring pages availabe. Apart from its practical imporance this would also contribute a lot to the 'Virgo experiment is cool' imadge, especially if implemented in a fancy way. Though, of course its priority is low, should be a side-project of some volunteer student instead of taking human resources from other more important tasks.

Related tasks:

- **rt#79** Create smart phone friendly versions of most important EGO - Virgo web pages.

Deadline: 2015/04/30
Status:

- **rt#79**: New

Chapter 3

Software maintenance

3.1 Introduction

The Virgo Collaboration use a huge number of various computer software including control, commissioning, data acquisition, data analysis, monitoring, etc. The software development is done using various architectures, programming languages, level of expertize and is distributed geographically, administratively and in time. In order to make the pieces of such a heterogenous software environment process smoothly working together the development and maintenance of the various components has to be coordinated, synchronized. Furthermore, in almost each of the above development / application area it is crucial that the software complies with a set of well defined criteria.

For what concern the timeline of the upgrades discussed below, we should consider that the Detector Control Software will need to support the initial critical phase of AdV subsystems precommissioning which will soon start. Because of this very tight schedule it is considered preferrable to postpone those changes which may be quite impacting existing well established procedures. To be considered also that already foreseen changes such as the introduction of the virgoStaging area are still to be fully introduced and are based an the existing infrastructure. The current Revision control (SVN) + Build (CMT) toolchain will then at least support the initial AdV phase. The possibility to experiment some of those upgrades for a Data Analysis software subset not at the moment in the critical path is being evaluated.

3.2 Requirements for experiment software

Virgo software of any kind should comply with the following criterias:

- Is running on the operating systems supported by Advanced Virgo.
- Must be readable, easy to maintain and as least error prone as possible
- Coding standards should be applied systematically
- Udergoes software review procedures including manual and automatic code inspections. Tools measuring standards metrics (like McCabe cyclomatic complexity) will be used to identify which software packages are more prone to exhibit faulty behavior, and should therefore be tested more thoroughly.
- Undergoes nightly tests, unit testing, stress testing
- Has a proper and regular versioning

- Is stored in the Virgo revision control system
- For the installation and distribution one can use linux standard tools, there should be no need to learn a special installation methods separately
- Has proper packaging and installation instructions
- Should be part of the Virgo software repository
- As efficient as possible, no waste of computing cycles
- Is well documented
- The responsible developer can be contacted

3.3 Supported operating systems by Advanced Virgo

The variety of computing architectures is so huge that it is impossible to ensure that our software is running on all of them. For this reason it is essential that a minimal subset of so called 'supported operating systems' is defined, so the developers are provided with clear guidelines about the platform. Furthermore these platforms has to be binary compatible with the ones used by LIGO and the EMI middleware. The following steps necessary:

- rt#54: Check which OSes will be supported by LIGO and by the EMI middleware
- rt#55: Select two OSes and evaulate their future usability, make a decision
- rt#56: Check the usability of existing Virgo software on the selected architectures

Deadline: 2014/03/30
Status:

- rt#54: New
- rt#55: New
- rt#56: New

3.4 Revision control system

To help the software development and ensure the proper storing, back-upping and versioning of the software we need a revision control system in place. For a long time Virgo was using the Concurrent Version System, (CVS) [13] tool until the recent transition to Subversion (SVN) [14]. Transition to SVN was mainly driven by its compatibility with CMT. Unfortunately it turned out that SVN is not as powerful as expected and in fact did not bring much improvement respect to CVS.

Based on the experiences with CVS and SVN we have set up a set of requirements that must be met by the to-be-selected revision control system.

3.4.1 Requirements of a revision control system

- Has to have the concept of froozen 'Tags'
- Has to be compatible with the build system of choice
- Has to be compatible with the software distribution system
- Has to allow for a distributed development path

- Has to be compatible with the LIGO choice
- Should have better conflict resolution than that of SVN
- Should be a widely used well supported main-stream, future proof solution

We have examind some of the alternatives such as Mercurial, Bazaar, Fossil, Perforce and GIT [15] and having the above requirements in front of our eye we intend to further investigate GIT as the revision control system for Advanced Virgo. In order to ensure a possible seamless transition the following steps would be necessary:

- **rt#57 (New):** Design a more clean, more logical and safer repository layout. The current layout has been driven by the CMT use mode adopted by Virgo. Repository reshuffling would imply a review of such use mode.
- **rt#58 (New):** Test installation of GIT and mapping the designed layout to actual repositories
- **rt#59 (New):** Transition of active SVN contents to the new repository
- **rt#60 (New):** Archiving of the dead / unused SVN content to a spearate repository
- **rt#61 (New):** Incorporate, interface GIT into / with the build and distribution system.
- **rt#62 (New):** Set up a secure web browser for GIT

Deadline: 2014/11/30

3.5 Build system

Any serious, complex software development need to have a proper build and testing system in place. The build system ensures that the software components can be built on specific platforms (i.e. the ones supproted by Advanced Virgo) the dependencies are met and the installation is possible. A unit test and a regression test system is necessary to spot and identify new or re-appearing bugs in the software. In Virgo to support the testing phase the tool TAT (Tool for Automated Test) has been adopted and altough some of the most critical Detector Control software (such as Cm) is now equipped with regression test suites, the tool has found limited use. For the Advanced era a build system integrating testing support is deemed necessary. Such a system tremendously helps the job of the code reviewers, as well, which is currently done on a volunteer, best effor basis, instead of a regular, semi-automatized way.

The currently used build and configuration tool CMT[17], while working fine, has also some limitations because of which its usage might be reconsidered in the future.

- Usually one would like to install software using linux standard solutions. Adding an other layer of necessary step for any installation method highly reduces the number of people actually using the software. This is especially true when external collaborators (such as LIGO) would like to use Virgo software.
- The developer and user basis of CMT is realtively small compared to other solutions for the same problems

In order to improve the situation we should go through on the following steps.

- **rt#63** Examine the possibility of using CMake [18] based configuration, packaging and testing for Virgo software components

- **rt#64** Convert some simple selected software package and create CMake configuration file for them
- **rt#65** Select a build system which is robust enough for nightly builds and can be interfaced with GIT and CMake. Being a complex problem, this task probably will have a series of subtask.
- **rt#66** Set up a nightly build server
- **rt#67** Set up and configure reporting web pages and email notifications for nightly builds
- **rt#68** Set up and test a unit and regression test framework.

Modern build systems are so called Continous Integration Servers. This working principle ensures that all the commit to a software is buildable and the identification of errorneous commits become really easy. Several other aspects of this working principle brings significant improvements for software development processes, that we are not going to discuss here. In case of further interest one can consult with [19]. The choiche for a build system is still an ongoing process. We are evaulating various solutions such as Jenkins[20], Hudson [21], JetBrains[22], CruiseControl [23] etc...
Deadline: 2015/01/30
Status:

- **rt#63**: In progress
- **rt#64**: New
- **rt#65**: New
- **rt#66**: New
- **rt#67**: New
- **rt#68**: New

3.6 Distribution of the software, repositories

For the distribution of Virgo software currently we are using CMT. With all the advantages and convenience provided by CMT we should reconsider looking for other solutions. The main reasons are, that if we would like Virgo software to be used by a wider community we should ensure that it is possible to configure and install it following widely used linux standards familiar for the majority of average user.

The suggestion is that we should move towards using standard linux repositories such as APT [24] and YUM[25], and standard linux package formats such as RPM[26] and Deb[27] however the way is not that straightforward. Tha main reason for this is that the compute resources used by Virgo are not controlled and administered by Virgo, so installation (for example that of rpm packages) of software which requires privileged access is not possible. We need to investigate how this can be circumvented, there are several possibilities. Each of them has to be evaulated and considered.

- **rt#69** Set up standard repositories and ask external site admins to add and use those repositories to the node installation.
- **rt#70** Perform user space installation of packages
- **rt#71** Run virtual machines on remote resources, which would solve many of the above problems.

Deadline: 2014/11/30
Status:

- **rt#69**: New
- **rt#70**: In progress
- **rt#71**: In progress

Chapter 4

Data management, access, distribution and file catalogs

4.1 Data management

4.1.1 Data management at EGO-Cascina

For file handling, Data Management at Cascina will follow the Virgo/Virgo+ model in which the production DAQ data streams from the Virgo Detector and the Online Processing are written on dedicated redundant storage circular buffers. To insure the needed reliability the access to these buffers is limited to specific on-line operations. From these buffers the files are migrated to a larger area, the storage farm, where data are kept on site for the time requested by the on-site workflows, currently not larger than 6 months except for some selected data. The storage farm space is also the endpoint of the data transfer activity and of the tape backup where needed. The file handling software for the production streams was developed internally for the Virgo data flows and processes in input and output the file lists in the standard FFL DAQ format but internally uses a local locator database to track the file locations and status and to manage the migration/transfer queues.

No particular new functionalities have been spotted in Advanced Virgo for DAQ streams, except for the increase of the data rate; therefore in the baseline scenario the main work in the short term isto adapt the file handling software to the new storage system that is going to be installed in 2014 and will remove the necessity to handle a high number of small (1TB) volumes thanks to larger volumes and more flexible storage provisioning capabilities. One requirement, namely the archiving of the "interesting data segments" (DS) for a time longer than the standard buffer length in Cascina, will be handled manually as in the Virgo/Virgo+ procedures, but will be integrated in the local locator database in a later phase. An organizational procedure needs to be setup in order to determine the lifecycle of these data to make space when becoming "uninteresting data segments".

It has to be noted that all the other data files (on-line/in-time analysis by-products or end-products, Ligo production data) are unmanaged in the baseline scenario, and they are moved either manually orvia ad hoc procedures either by the users or by the storage managers at Cascina. If during the development of the Computing Model it will come out that some workflows could benefit from a centralized management, they will be integrated in the file handling system.

The databases on-site will be as much as possible centrally managed. The shortest term target is to align all the MySQL installations and decouple the application clients from the database servers in order to centralize the storage backend following the upgrade of the storage system.

4.1.2 Data management at CNAF

4.1.3 Data management at Lyon

The model and basic rules for this item is described in Part III, section 4.2 of the AdV Computing Model. The software version and milestones are shown in Part IV, sections 5.9 and 5.8.4

4.2 Data access

4.2.1 Local Data Access

4.2.1.1 At EGO/Cascina

The access requirements to the files for the workflows at Cascina can be divided in at least 2 categories: rawdata browsing, mainly for Commissioning interactive work, and data read/write processing on smaller subsets from all the data categories. From the past experience the workload of the various activities is very dynamic and changes frequently according to the number of people on site, the presence of interesting events in the data, the on-line/off-line data analysis tests, but in any case it can involve any time periods over the Cascina buffer length. Therefore the files need to reside all on disks, but the majority of the space (for the rawdata browsing category) can be served by nearline disks not optimized for high number of write IOs per second but with acceptable throughput for quasi-sequential reads. To insure the needed flexibility, the placement of the data according with the I/O categories will be optimized exploiting the policies allowed by the new storage farm that will allow to manage a large (scalable) pool of storage according to the needs.

From the point of view of the applications and end users the local data access will be POSIX based, NFSv4 in the baseline scenario. The next step will be to select the backend filesystem and to evaluate the benefits of a parallel filesystem up to the clients.

No provision for remote data access, either toward Cascina or toward the external CCs, is foreseen in the baseline scenario.

4.2.1.2 At CNAF

INFN-CNAF, the Italian Tier-1 located in Bologna, is one of the LHC Tier-1 and it houses computing and storage resources for many other particle physics and astrophysics experiments, including Virgo.

Storage at CNAF amounts to more than 10 PB of tape space and 6 PB of disk space. The centre has recently developed a new mass storage system called GEMSS (Grid Enabled Mass Storage System) which proved to be an efficient solution to manage data archiving between disk and tape.

The main components of the system are:

- a filesytem layer implemented by the GPFS (General Parallel File System) framework;
- the IBM TSM (Tivoli Storage Manager) software which manages the tape layer access;
- the StoRM layer that is used in conjunction with the GridFTP servers to provide remote Grid access.

GEMSS services manage data flow between disk layers and tape in an automatic and fully transparent way. Files created on the disk layer are automatically copied to tape; when the disk occupation is over a defined threshold, the system replaces the disk copy of the ”old” files (files not being accessed for the longest time) with a pointer to the copy on tape (”stub-file”). If the file is later requested GEMSS automatically recalls the file back on disk.

There are currently three storage areas at Cnaf locally accessible by Virgo users:

- `/storage/gpfs_virgo4` (labelled in the following as **/virgo4**)
- `/storage/gpfs_virgo3` (labelled in the following as **/virgo3**)
- `/opt/exp_software/virgo`

The /virgo4 area

This is the main storage area, which hosts the Virgo bulk (Raw, h(t), 50 Hz, etc.) and LIGO h(t) data. It is a Disk-Tape area managed by GEMSS. It is writable only with Grid tools (see below) and mounted read-only on the user interfaces and on the computing nodes.

The base path to Virgo and LIGO data is:

- `/storage/gpfs_virgo4/virgo/virgoD0T1/Run (Virgo data and Ligo h(t))`,
- `/storage/gpfs_virgo4/virgo/virgoD0T1/DATA (50 Hz.  INGV_turbine, trend)`

Note: the directory `/storage/gpfs_virgo4/virgo/virgoD0T1` is not user readable, but inner folders, e.g. `/storage/gpfs_virgo4/virgo/virgoD0T1/Run` are!

To list the content of `/storage/gpfs_virgo4/virgo/virgoD0T1` use lcg-ls (LCG tools):

```
lcg-ls -v --vo virgo -l srm://storm-fe-archive.cr.cnaf.infn.it:8444/virgod0t1/
```

Note that the path `/storage/gpfs_virgo4/virgoD0T1/` is mapped by SRM into /virgod0t1/.

/virgo4 area is writable uniquely using LCG utils, and is accessible RO via POSIX and via LCG utils. An example of how to download data from /virgo4 to a Grid user interface is given below.

```
lcg-cp -v -b --vo virgo -T srmv2 srm://storm-fe-archive.cr.cnaf.infn.it:8444/srm/managerv2?SFN=/v
file:‘pwd‘/V-raw-864088320-240.gwf
```

Related tasks:

- **rt #138 Migration from LCG-utils to gfal2-utils**
- **rt #139 Set appropriate ACLs to Virgo storage area at Cnaf (/gpfs_virgo4)**

The /virgo3 area

It is a user scratch area, mounted on the user interface only. It has no tape backend (Disk-only area). It is Write accessible via Posix and via Grid (LCG tools, see below).

The main /virgo3 areas are:

- `/storage/gpfs_virgo3/home (user area, home/<user> writable)`
- `/storage/gpfs_virgo3/scratch (scratch, writable)`
- `/storage/gpfs_virgo3/virgo (Writable with Grid tools)`

The /opt/exp_software/virgo area

It is the Virgo software area, mounted R/W on the user interfaces and R/O on the worker nodes. Virgo and LIGO FFLs are available in `/opt/exp_software/virgo/virgoData/ffl/`.

Related tasks:

- **rt #140 Investigate CVMFS to distribute Virgo software**

4.2.1.3 At CCIN2P3

Marie Anne

The CCIN2P3 is a Tier-1 computing center located in Lyon, France, whose resources (farm and storage) are mutualized and shared among all experiments/users. The storage system is a 2-level system. The main permanent storage media are tapes. Data on tapes are accessible through a high mass storage system (HPSS) that is composed of several tape libraries, robots, disks and servers. Virgo data (raw data and the different processed data streams) are stored permanently in HPSS. As of March 2014, the total capacity of HPSS is 23 PB. Virgo data amounts for 883 TB.

In addition to tapes, more than 10 PB of disks are used to store non permanently experiments' data. Virgo data are accessible through the XrootD cache disk system (total capacity $\sim$ 1 PB). The access is totally transparent for users, using POSIX I/O libraries as long as the XrootD environnement is launched before data is read. The CCIN2P3 is maintaining the XrootD environment. Here is an example of a script that reads a Virgo frame format file stored in HPSS:

```
#! /bin/sh

. /usr/local/shared/bin/xrootd_env.sh

if [ $# -eq 0 ]; then
  echo 'file name missing. Example:'
  echo '/hpss/in2p3.fr/group/virgo/Run/VSR4/raw/991/V-raw-991135200-150.gwf'
  exit 1
fi

${FRROOT}/${FRCONFIG}/FrDump.exe -i $1

exit 0
```

To list Virgo and LIGO data files, on can simply use RFIO commands. All RFIO commands are described in **http://cc.in2p3.fr/docenligne/292**. For instance one can access to the Virgo HPSS repository using the rfdir command.

```
rfdir cchpssvirgo:/hpss/in2p3.fr/group/virgo
ccage017:tcsh[88] rfdir cchpssvirgo:/hpss/in2p3.fr/group/virgo
drwxr-xr-x  11 mabizoua virgo                512 Jun 14  2004 .
drwxr-xr-x  62 root     root                1024 Apr 24  2003 ..
drwxr-xr-x   8 virgdata virgo                512 Oct 12  2006 DATA
drwxrwxr-x   2 mabizoua virgo               2048 Sep 22  2006 VBMDC
drwxrwxr-x   8 mabizoua virgo                512 Oct 12  2006 USERS
drwxrwxr-x  44 mabizoua virgo               1024 Jun 14  2004 Run
drwxr-xr-x   2 mabizoua virgo                512 Apr 25  2005 ROG
drwxrwxrwx   7 mabizoua virgo                512 Sep 03  2003 MDC
drwxrwxr-x   7 mabizoua virgo                512 Apr 27  2005 VLMDC
drwxrwxr-x   6 mabizoua virgo                512 Jul 12  2012 NINJA2
drwxr-xr-x   2 virgdata virgo                512 Apr 11  2012 TAROTZadkoQUEST
```

Virgo and LIGO data are stored according to the run and the type of the data set. Raw and h(t) data sets are in **cchpssvirgo:/hpss/in2p3.fr/group/virgo/Run**. More information on past

Virgo and LIGO data at Lyon is available in [41]. The XrootD path is identical to the HPSS path, except that the prefix "cchpssvirgo:" should be dropped. Actually the XrootD path is understood by RFIO commands and "cchpssvirgo:" prefix can be dropped. Finally, all *frame format list* (ffl) files have been generated by hands and are available on the cluster in $GROUP_DIR/BKDB/. These files are organized in folders named after the runs.

```
dir $GROUP_DIR/BKDB/
total 42
drwxr-xr-x 17 root     root  4096 Oct 26  2011 .
drwxr-xr-x  2 root     virgo 2048 Nov 18 14:23 ..
drwxr-xr-x  2 mabizoua virgo 2048 Aug 18  2005 C5
drwxr-xr-x  2 mabizoua virgo 6144 Oct 15  2005 C6
drwxr-xr-x  2 mabizoua virgo 2048 Mar 16  2011 C7
drwxr-xr-x  4 sentenac virgo 2048 Apr  7  2008 DATA
drwxr-xr-x  2 virgdata virgo 2048 Jul 29  2008 MDC
drwxr-xr-x  2 mabizoua virgo 2048 Apr 29  2008 S5
drwxr-xr-x  2 mabizoua virgo 2048 Jun  3  2013 S6
drwxr-xr-x  2 virgdata virgo 2048 Oct 27  2011 VA3
drwxr-xr-x  2 mabizoua virgo 2048 Mar 12  2012 VA4
drwxr-xr-x  7 mabizoua virgo 4096 Jan  6  2007 VL
drwxr-xr-x  2 mabizoua virgo 2048 Apr 19  2011 VSR1
drwxr-xr-x  2 mabizoua virgo 2048 Oct 28  2011 VSR2
drwxr-xr-x  2 mabizoua virgo 2048 Mar 18  2011 VSR3
drwxr-xr-x  2 virgdata virgo 2048 May  5  2012 VSR4
drwxr-xr-x  2 mabizoua virgo 2048 Jun 26  2009 WSR
```

To be done:

- **rt #XXX Clean up the HPSS archive from uncontrolled transfers)**
- **rt #XXY Rewrite the CCIN2P3 data storage and local data access webpage)**

4.2.2 Remote Data Access

Gergely

4.3 Data transfers

4.3.1 Low latency data transfer

4.3.2 Bulk data transfer

The bulk data transfer framework is used an will be used to distribute data around the computing centers. The framework that we used to distribute data in the previous work is working fine and usable, however we would like to improve the situation by reducing the number of different tools and protocols used. Currently there is at least 3 different type of solution for different endpoints which causes a significant amount of overhead in terms of required manpower.

4.3.2.1 Requirements for bulk data transfer

The currently implemented and any new solution and the following requirements,
in terms of latancey:

- Cascina -¿ Virgo CCs
  - raw data: 1 day
  - trend data: 1day
  - h(t): 1day
- LIGO -¿ Virgo CCs
  - RDS data: 1 day
  - h(t): 1 day
- Cascina -¿ LIGO CCs
  - RDS data: 1 day

and in terms of functionality and service quality, architecture:

- data consistency should be checked before distribution in order not to distribute bad data
- remote physical location of files transferred has to be registered int the site-wise file locator database and also in the file catalog of choiche (LFC)
- identical physical files stored at different CCs should be registered with the same logical name in the file catalog, as such they should being each others replicas
- checksum of any kind should be calculated for the files transferred and periodically (once a week) checked
- integrity of File Locator Database, LFC and the physical files has to be checked automatically, periodically. There has to be an entry for each physical file transferred in the file catalogs and vica versa.
- data transfer framewrok should be easily extendable to include additional target sites, not only CNAF and Lyon. Smaller sites with capability to store a fraction of the data could be usefule for various distributed analysis.
- should be able to arrange the files in order of time
- should be able to join smaller files into bigger ones in order to make space allocation more effective on hierarchical mass storage systems
- there has to be a documentation for administrators, developers and users who are running the actual transfer
- if possible data should be transferred directly to its final destination, no data moving should happen on the destrinaiton site (apart from automatic staging of data)
- file transfer should not require the installation and maintenance of software on each of the target / or source site, but should be managable using a single instantiation of the service and communicating with services and interfaces. Installing any kind of software on the destination sites always generates lot of maintenance and security question on both side
- transferred files should appear in the file catalog and in the web based monitoring tool almost real time, i.e with a maximum latency of a few minutes
- data transfer should have a web based monitoring tool

4.3.2.2 Possible choices for bulk data transfer

As part of the tasks below, we have examined various possible solutions fulfilling these requirements including

1. **(rt#117; Completed)** The File Transfer Service (FTS)[35] of the EMI middleware
2. **(rt#118; Completed)** The BitTorrent[36] protocol
3. **(rt#119; In progress)** Dirac file transfer service[**?**] and utilitites
4. **(rt#120; In progress)** Custom developed file transfer toolkit
5. **(rt#121; In progress)** Ligo Data Replicato[38]

Despite we strongly beleive that none of the solution will work for us out of the box, we do perform these evaulation studies because many good idea and experience can be collacted and implemented / adjusted to our custom data transfer solutions that we are very probably is continuing to use and extend.

4.3.2.2.1 EMI File Transfer Service, FTS As part of task rt#117 we have examined FTS as one possibility for Virgo bulk data transfer. As a result of our investiagtions we found, that while FTS has greatly improved over the last years - for example the concept of channels has been replaced by a much flexible solution which involves on the endpoints - it would probably be not the best choich for Virgo. Reasons include

- FTS has extensively been tested only with the Oracle backend which is far too heavy for a small Collaboration to install and maintain.
- Using an existing FTS installation somewhere could involve a lot 3rd party administration and could easily be the bottleneck of debugging and development.

4.3.2.2.2 Torrent protocol As part of task rt#118 we have examined the bitTorrent protocol as a possible file transfer solution for Virgo. While torrent has a lot of appealing features, such as

- multi source copy,
- built-in checksum verification
- plenty of well testes client on all platform
- light-weight, programmable
- very-high level of failure tolerance
- cool,
- LIGO has also considered this choice
- could easily and transparantly cover site down-times
- etc, etc..

we have found that **while for redistributing of the data it could be very useful**, i.e to copy data from the CCs to a job under execution in the distributed job submission framework, for bulk data transfer there are better or more precisely simpler solutions.

The reasons are that other protocols could have

- much faster rump-up time

- possibly better bandwith utilisation
- and in principle the bulk data transfer always happens between a few number of static site
- the fragmented file parts might not be optimal for mass storage systems, but this could be easily circumvented (staging only after copy is completed)

as a result, there is no real need for something sophisticated.

However we keep in mind that for data redistribution or P.R. purposes torrent can be a good solution and we will continue investigating this possibility.

4.3.2.2.3 Dirac file transfer service and toolkit Dirac was primarily developed as a distributed job submission framework. For the jobs to access data, Dirac also includes a data management system. This system is being studied for Virgo. Several features are under investigation:

- the transfer sevice between Dirac Storage Elements,
- the file integrity checks,
- the file catalog,
- the file catalog meta-data.

These investigations are reported under task rt#119.

4.3.2.2.4 LIGO data replicator The LIGO Data Replicator based on Globus [39] and RLS[40] is developed and used by the LIGO collaboration to perform various data transfer tasks. As a result of our investigations we found that LDR is definitely a good and capable tool of performing various data transfers for Virgo use case it has the following limitations

- it cannot handle iRods enpoints, as such even if we use it there is a need to maintain the Lyon case if we cannot manage to use gridftp enpoint in Lyon
- any customization and Virgo specific chnages in the code could be difficult to push back to the development source
- there is not too much installed instance around

While we don't think that Virgo should have its own LDR server installtion, as currently done LDR can be an optimal choice to transfer Virgo data to LIGO resources in the following way. On Virgo storage there is always a well defined dataset published seen by the LIGO LDR server which automatically does the transfer. In this way Virgo only has to maintain the client side publishing of the files which is not so teadious.

4.3.2.2.5 Custom developed Virgo transfer framewrok As part of task rt#120 we are in investigation how the current framework used for bulk data transfer can be extended and simplified. This is the most probable scenario and we refeer it as the "baseline scenario". Our tool is able to satisfy all the requirements and we can definitely make use of it also for Adanced era. However some testing and evaulation still might to come. Some of the necessary steps are:

- **(rt#122; In Progress)** Check the possibility of how to use gridftp protocol to copy data from Casinca to Lyon iRods storage.
- **(rt#123; New)** How to copy files using gridftp from LIGO clusters (for ex Hannover) to Cascina
- bf (rt#124; New) Check how the framework can be extended to additional sites, such as Wigner.

4.3.2.3 Data transfer considerations

A functional model needs to be built from the Virgo applications requirements that details the workflows for the following categories:

- The in-time bulk data transfer of production data from/to the interferometers, differentiated between rawdata and the other lightweight data. It is characterized by an in-time requirement, possibly with different priorities among the data categories/DApipelines.
- The off-line data transfer between the various CCs. This has a management component (for replication/migration) and a component related to data access from users/jobs.

In the baseline scenario the in-time bulk data transfer will be accomplished using the same model of Virgo/Virgo+, i.e. star-centered fluxes from Cascina to the CCs using the data transfer protocols supported by each CC, and the same for Virgo data sent from Cascina to LIGO.

For the data going from LIGO to Virgo CCs a task has been setup in order to manage the transfer and the compaction of the files directly in the datacenters before the final storage.

For the off-line data transfer, it will be triggered in the first step by management needs, like replicating the production datato be near to the computing jobs accessing them locally.

In addition, in parallel to the work on the data access, we will need to converge to a lightweight subset of data that can be transferred transparently and accessed by the computing jobs for example using one or more selected distributed storage system.

4.3.2.4 Data transfer from LIGO to Virgo

As discussed above the best candidates are a.) Virgo custom developments b.) Dirac file transfer servcice. Still not decided.

4.3.2.5 Data transfer from Virgo to LIGO

As discussed above the best candidate is to continue using the current practice with LDR.

4.3.2.6 Virgo only bulk data transfer

As discussed above the best candidates are a.) Virgo custom developments b.) Dirac file transfer servcice. Still not decided.

The model for this item is described in Part 3, chapter 4.1 of the AdV Computing Model.

4.4 File catalogs

Since every site of a distributed computing environment has different storage solution and non-default access path to experiment data the usage of a file catalog is unavoidable. A file catalog is a service which translates logical file names to physical access paths for a given computer center. It can also store information about the available replicas of the same data chunk and metadata information.

4.4.1 Requirements for a file catalog

- Should be simple, lightweight
- Should support the concept of replicas
- Optionally could have some metadata information
- Should be able to query it from outside of any job

- Preferably has a web interface
- Its long term support or easy migration is guaranteed
- Should be compatible with the storage systems and protocols used

Work is needed to collect the functional and detailed requirements for the file handling and access and the corresponding metadata to be able to manage the bookeeping of the file locations irrespectively of the underlying storage system. Once this functional model is ready a thorough study of the current distributed storage frameworks supported by the Virgo CCs must be started to converge on the most agreed solution that requires the least development, at the same time providing the most stable API to the Virgo applications.

In the baseline scenario the files will be handled as in Virgo/Virgo+ by various file catalogs, each supported by the local computing center, with hybrid access provided locally by the CC, and the location of the files will be tracked by maintaing the corresponding FFL lists in a more automatized way.

4.4.2 Possibilities

There are several alternatives which are still under evaulation, namely the followings:

- LFC[31], the Logical File Catalog - is a well-tested lightweight solutions interfaced with various data manipulation utilitites such as lcg-tools or gfal. It is used by big HEP collabrations, as such its future or at least a solution for a future migrations seems to be guaranted.
- Dirac[32], the Dirac file Catalog - a fully-fledged solution for and inside in the Dirac job submission framework. Exhibit all the feature we need, however available only inside the Dirac framework, with Dirac client tools. This fact might not necesseraly impose any limitation.
- Custom file catalog - It is planned that in any case we develop a custom file catalog, File Locator Service, to serve as an implementation independent backup solution for Virgo software not using any Grid-based solution.

For the successfull evaulation and decision process we need to go through on the following tasks:

- task #XXX

Deadline: 2014/08/30
Status:

- task #XXX

4.4.3 Data Bookkeping

Gary, Didier
Data Bookeeping information allows to retrieve channels and interesting time periods from a set of data files.

The DQSEGDB database (Data Quality Segments Database) contains time segments where the interferometer is in science mode and time segments for which data should not be analysed.

The Channels DB provides various information (sampling rate, description, etc...) about the channels acquired and stored in the raw data files.

The raw.ffl file contains the list of raw data files stored on Cascina site and allow a direct access to the channels selected in the time periods selected.

4.4.4 File locator Data Base and File Locator Service

Despite having a decent grid-based file catalog in place there always have to be a backup solution which is independent of any actual implementation of the file catalog used by the jobs in the distributed job submission framework.

This is the very purpose of the File Locator Service. This approach has several advantage:

- serves as a backup file catalog
- ensures easier interaction and query of data for jobs nost using grid tools
- any new file catalgo can be easily repopulated using it

Since the various data transfers will have well defined target locations it is very easy to put these information to the File Locator Service.

The following tasks are necessary to implement the service:

- task #XXX

Deadline: 2014/08/30
Status:

- task #XXX

Chapter 5

Local job submission methods

The chapter should include the implementation steps which are necessary that Virgo members become able to submit jobs locally to the clusters listed in the following sections. Task for each computer center should include amon others:

- enabling uniform, certificate based authentication
- documentation, including support pointers and example submission
- anything else which is missing

5.1 Local job submission at EGO - Cascina

The computing resources in Cascina will be mainly devoted to the on-line and in-time computations, like in the past, but the flexibility of the allocation will need to be more flexible, avoiding to assign statically the CPUs except for selected on-line critical tasks. The planned hardware is a set of homogeneous multi-processors machines of the order of 300-500 cores from which to carve pools of virtual machines or bare metal hosts which will be assigned to the various tasks/groups or to the IT infrastructure activities. Either for bare metal hosts or for virtual machines pools, in order to optimize further the CPU usage, some of the in-time tasks, those to be accomplished with a delay from some hours to 1 day, will be executed via a job submission system.

In the baseline scenario the job submission, based on Condor, will be used locally for in-time Noise Analysis jobs (for example Noemi) and also for Noise investigation jobs submitted from the users via the DNMAPI web-based framework.

5.2 Local job submission at CNAF - Bologna

CNAF implements LSF as its local batch system. The "virgo" queue is reserved to the Virgo collaboration. The resources reserved to the virgo queue, currently of the order of 1000 CPU or 10K HS06, should be negotiated with CNAF every year.

Users can submit jobs to LSF from the two Virgo user interfaces (**ui01-virgo.cnaf.infn.it** and **ui02-virgo.cnaf.infn.it**). Logging into the UI is done via SSH, through a "bastion" host (**bastion.cnaf.infn.it**).

Below is an example command to submit jobs to LSF, which shows how to send input files and retrieve output files using the "-f" option:

```
bsub -f "inputdata.tgz > /home/VIRGO/<user>/inputdata.tgz" -f "outputdata.tgz < /home/VIRGO/<user
-f "run.log < run.log" -o run.log script.sh <args>
```

In this example script.sh writes the output in the user's home on the worker node (**/home/VIRGO/¡user¿** - NOTE: it is not the home directory on the UI!) and the log file in the **/tmp** of the worker node.

Big files should be stored using Grid tools on the main storage area (**/storage/gpfs_virgo4**, see Par. 4.2.1.2); they can be accessed via POSIX from the worker nodes. Software and libraries can be stored on the Virgo software area (**/opt/exp_software/virgo**).

5.3 Local job submission at IN2P3 - Lyon

Marie Anne
The computing resources of the IN2P3 computing center (CCIN2P3) are grouped on a unique farm and are presented as a mutualized power: all computers are not dedicated to an experiment or group but are normalized and mutualized. All of them are accessible by all users. Computing resources request are collected every year from all experiments. In case of large computing demands, special discussions and arbitrations between experiments are mandatory. This has never concerned Virgo demands which have always been fulfilled by the CCIN2P3. This management policy allows an excellent and optimal usage of the farm and the resources.

The batch farm is composed by different operating systems. For each of it, it can be possible to have different materials and hardware configurations. The hardware is replaced and upgraded constantly. For instance in March 2014 there are 704 workers for a total of 19 144 processors (16, 24 or 32 multi-core) all running SL6 for a overall computing power of 200 049 HS06-hours. Details are available in **http://cctools.in2p3.fr/mrtguser/info_sge_parc.php**.

The whole farm, with its diversities, is managed by a single batch scheduler developed at the Computing Centre: Grid Engine. Grid Engine can be accessed through GRID resource brokers or directly by users form the interactive farm (**ccage.in2p3.fr**). Different queues are available for users to cover all types of computing resource needs (long duration jobs, large memory size, etc ...). All queues' information are available at **http://cctools.in2p3.fr/mrtguser/info_sge_queue.php**. Locally submitted jobs are accessing Virgo and LIGO data stored in HPSS through the XrootD environment. As explained in section 4.2.1.3 data are read in a transparent way through XrootD and the *Frame library.*

The Virgo and LIGO software is installed in $THRONG_DIR/virgoApp. The software environment is defined by the Virgo czar in Lyon and any user is invited to source $THRONG_DIR/group_login file in their login script.

Jobs' outputs (ans some users' input files) can be stored in the semi-permanent disk area /sps/virgo/USERS/<user-id>. This disk area is a semi-permanent disk space. When 95% of the disk are full, the oldest files will be deleted.

Grid Engine commands are available in **http://cc.in2p3.fr/docenligne/970**. Here is a very script that can be submitted to the farm with GE qsub command. The script is just trying to dump some metadat out of a frame file. The script trend.sh is including all GE options in the header:

```
#! /bin/bash -f
### ct: cpu time
### os: operating system
### fsize: worker tmp disk size
### vmem: memory required
### sps=1: access to sps
### xrootd=1: use of xrootd
### hpss=1 use of HPSS
### -N name: job name
### -P P_virgo: project name

### Merge stdout et stderr in a single file
#$ -o /sps/virgo/USERS/mabizoua/RDS/merge_3.o
#$ -e /sps/virgo/USERS/mabizoua/RDS/merge_3.e
#$ -P P_virgo
#$ -l os=sl6,s_rt=171000,fsize=1.5G,vmem=1G,xrootd=1,sps=1
#$ -N merge_3

source /usr/local/shared/bin/xrootd_env.sh

${FRROOT}/${FRCONFIG}/FrDump.exe -i /hpss/in2p3.fr/group/virgo/DATA/trend/2011/T-993254400-48F.gwf

unset LD_PRELOAD

exit 0
```

To submit the script: *qsub <script>*
To check the jobs status: *qstat*

Currently, to submit locally a job at Lyon, users need to ask for an account on the **ccage.in2p3.fr** farm. The CCIN2P3 is providing user support through a ticketing system. The CCIN2P3 web site **http://cc.in2p3.fr/** provides the most updated information about the cluster status. Questions specific to Virgo should be addressed to the Virgo czar in Lyon. Information pertaining how to get an account, user support, computing resources, job submission and Virgo/LIGO data access is available from the Virgo CCIN2P3 web page: **http://virgo.lal.in2p3.fr/CCIN2P3/accueil.html**.

- **rt #XXX Remote data access in Lyon)**
- **rt #XXY Rewrite the CCIN2P3 Virgo webpage)**

5.4 Local job submission at INFN - Roma

The Roma site is used only for grid job submission, as such no local job submission method is allowed, available.

5.5 Local job submission at Wigner - Budapest

The Wigner Research Centre for Physics of the Hungarian Acadamy of Sciences currently offers 3 isolated cluster for Virgo users.

- A Tier-2 cluster part of the EMI infrastructure (SGE)

- An SGI ICE Altix computer with 512 processor (Condor)
- A small-scale GPU cluster with 128 core and 16 GPU (Condor)

The Tier-2 cluster will be part of the Distributed Job Submission Framework but also supports local job submission. The resources are in everyday operation, however further customization is necessary. In order to make this computing resources available for the LIGO - Virgo gravitational wave community we need to complete the following tasks:

- (**rt#111; New**) Set up certificate based auth for the GPU cluster
- (**rt#112; New**) Set up certificate based auth for the SGI ICE
- (**rt#113; New**) Check Virgo VO configuration for the Tier-2 cluster
- (**rt#114; New**) Test native job submission for the Tier-2 cluster
- (**rt#115; New**) Test job submission using the DJSF for the Tier-2 cluster
- (**rt#116; New**) Set up or update documentation and support pointers for the clusters

Deadline: 2014/09/30

Chapter 6

Distributed job submission framework

6.1 Introduction

In order to extract all the possible physical signal from the detector the various analysis pipelines has to be able to run in their full potential. In many cases the computational power available for a specific analysis can directly be translated to analysis / measurement sensitivity. However many of the analysis workflow - for various reasons - is computationally bounded. There is in principle three way to overcome this difficulty

- Purchase more computing resources
- Write far more efficient analysis code and make use of new technologies, solutions
- Enable already existing and usable computing resources

Given an upper limit on the computing budget we can exclude the first and concentrate on the second and third possibility.

The work on porting and more efficient possibly paralell reimplementation of various analysis software is widely discussed in other parts of the CM and also in this document.

This chapter aims to provide an answer to the third scenario, i.e. setting up a job submission infrastructure with the help of which we can enable otherwise unreachable geographically and administratively distributed heterogenous resources.

In order for such a system to be as effective as possible it has to satisfy a set of well defined requirements.

6.2 Requirements of distributed job submission framework

In order for such a system to be as effective as possible it has to satisfy a set of well defined requirements.

- Should be future-proof, i.e. it has to have a development and maintenance roadmap ansured for at least the following 5 years.
- Should be easy to use for an average user and pipeline developer.
- Should be compatible with any data transfer system to be developed for bulk data transfer
- Should allow various data access mechanisms available on the centers where it enables the execution of the jobs

- Has to have a strict autehntication and authorization system in place
- Should be able to handle scientific, relational workflows
- Should be able to directly use or to submit to the biggest european computing infrastructure to the EMI middleware.
- It should be possible to port Various LIGO pipelines without or with minor modifications.
- Should enable job execution, monitoring, logging, etc..
- Should not be specific to any of the target execution site but flexible enough to be extended to any (or many) future - currently unseen or unexpected - computing architexture.
- Should be easy to operate, maintain

Taking into consideration all the above requirements we examined a lot of possible solution including Glide-in WMS, Condor pilot-pool, Alien framework, Dirac, Pegasus, etc...

Out of the above alternatives we have left with two promising candidate namely Dirac[10] and Pegasus[11]. We will refeer to these two choice as job submission framework candidates in the rest of this chapter.

Naturally it would make sense to use only one of them, but for this decision one should go through on a well defined set of tasks and tests to varify their usability for our purpose.

6.3 Verification and testing process for the job submission framework of candidates

Here we define the necessary steps to be performed before making the final selection of the distributed job submission framework. These steps are

1. (**rt#9, rt#27**) Contact the developers and get a statement about the future development roadmap
2. (**rt#10, rt#28**) Check whether the software is available for the selected OSes and platforms of AdV
3. (**rt#11, rt#29**) Download, install and configure the software
4. (**rt#12, rt#30**) Check which computing resources can be reached / enabled with the given solution
5. (**rt#13, rt#31**) Check job management capabilities, such as submit, remove, stop, hold, logging, etc..
6. (**rt#14, rt#32**) Check up-to-dateness of documentation
7. (**rt#15, rt#33**) Check the active user bases, forums, support possibilities
8. (**rt#16, rt#34**) Try to set up and submit a "Hello world" job for various resources.
9. (**rt#17, rt#35**) Examine the user experience of the software
10. (**rt#18, rt#36**) Try to submit a more complex workflow
11. (**rt#19, rt#37**) Test data access from within the submitted job
12. (**rt#20, rt#38**) Test file catalog, job sub submission system interaction if there is any

13. (**rt#21, rt#39**) Test the job submission with a real Virgo analysis workflow
14. (**rt#22, rt#40**) Test security, authenticationa and authorization features
15. (**rt#23, rt#41**) Make an estimate about the FTE necessary for maintaining theframework for the whole collaboration
16. (**rt#24, rt#42**) Write up short tutorial for Virgo users and gave a presentation on VDASC
17. (**rt#25, rt#43**) Submit bulk-data challanges
18. (**rt#53, rt#52**) Executing virtual machines within the job submission framework
19. (**rt#26, rt#44**) Do stress testing of the framework

When all the above tasks are completed for both of the candidates then we have to make a decision and select the one which will be used thorough the Advanced Detector Era.

The decision has to be made by end of June, 2014, see **(rt#45)**.

Many of the task listed above has to be repeated with the selected job submission system in full-scale

The task are introduced into our project management system Redmine[12], (redmine task number marked with (rt#X, rt#Y) for Pegasus and Dirac respectively) where the exact description, testers, due-dates, results and problems will be updated and can be followed.

Chapter 7

Commissioning, detector characterisation and scientific data analysis workflows

7.1 Introduction

This part of the document gives the implementation plans for all the workflows detailed in Part I of the AdV Computing Model. It is hence divided into the two main branches, which are 1) the Commissioning needs which have relevant impact on data analysis activities and detector characterization and 2) Data analysis pipelines (divided as usual into CBC, BURST, CW and STOCH).

7.2 Commissioning and operation workflows

This section is for Loic, DAQ chair

The Computing model for this item is Part I, Sect. 1.2 and Part II, Sect. 2.2.
Don't forget:
how many and which channels in the raw data, how many in the RDS ?
References to web pages for details, pages which we know will always be maintained, are fine.
Why do we need a buffer length of 6 months in Cascina ?
It is very important to have and maintain web pages with detailed info.

7.3 Detector characterization workflows

7.3.1 Detector Characterization: Data Quality

The Data Quality work includes glitch studies, online vetoes production, offline vetoes production and the development of tools for monitoring, investigations and commissioning help. The main axes of this work were described in the Computing Model: mainly a trigger generator (Omicron), an online veto production (UPV), a storage of data quality segments (DQSEGDB), a set of in-time monitoring and investigation tools for glitch studies and DQ flags safety and performances (MonitoringWeb, DMS, dataDisplay, DQperf, DQsafety, UPV matrix, Omiscans, ...). What follows is a description of the implementation of the tools developed for those strategic axes (most of computing and storage numbers have been already quoted in tables 8.2 and 8.6 of the Computing Model document):

7.3.1.1 Omicron pipeline

This trigger generator will run in-time over the h(t) channel and over hundreds of auxiliary channels, taking as input the raw data files. It will produce trigger files in ROOT format, stored on disk and to be used for various features of data quality and glitch investigation. This will require about 60 computing nodes with Gbit interface. Each computing node will analyse about 15 channels sampled at frequencies between 1 kHz and 20 kHz for a total of about 10 Mbytes per second read from the raw data files. About 1-2 TB per year will be needed to store the output triggers.

7.3.1.2 On-line vetoes

An online version of Omicron will run on a set of a few tens of selected channels in order to produce the online veto segments. It will use thresholds and channels determined by the UPV algorithm running on the offline Omicron triggers. It will take as input the raw data stream provided by DAQ and will send the veto segments within frames down to the SegOnline process which will write DQXML files containing the information to be inserted into DQSEGDB. This will be the main source of online veto segments. Other processes like Excavator, BRMSMon or VetoMon will produce online DQ flags. They will also take as input the raw data stream and will send their output to SegOnline. We plan to put also in the online vetoes a selection of the DQ flags produced by the DMS (Detector Monitoring System). All those processes are written in C and use the FdIO library to access data. About 4 computing nodes will be needed to run Omicron online and 4 computing nodes will be needed to run the other online veto producers.

7.3.1.3 Detector Monitoring System (DMS)

The DMS is a set of processes taking as input data the DAQ raw data stream and producing DQ flags used to show in control room a complete online view of the status of interferometer's subsystems, DAQ and online processing. This system is made of a set of processes running on one computing node and a set of web pages produced by a php script running on one computing node. An archive of those web pages is available but no large disk storage is needed.

7.3.1.4 MonitoringWeb

This is a general framework which handles monitoring information and plots produced in-time by bash scripts and ROOT macros. It takes as inputs various data (raw data, trend data, spectro data, Omicron triggers or DQSEGDB entries). They give information on the interferometer status and on all the ongoing on-line data quality and data analysis. To allow all the web pages to be updated with a latency below 30 mn, it requires about 6 computing cores (without taking into account the spectrograms computation). A daily archive of all the plots (except spectrograms) requires about 600 GB per year. In addition, about 4 computing cores will be needed to set up a dataDisplay server which will be the main provider of data for dataDisplay online connections.

7.3.1.5 Spectrograms

This part of the MonitoringWeb pages provide, for several channels and several frequency bands, a set of spectrograms computed over one hour, one day or one week. A dedicated process (C code using FdIO library), running on one computing node, computes spectra online and get input from the DAQ raw data stream. Those spectra are saved under frame format in a specific "spectro" data stream, stored on a dedicated disk area under /data/procdata. This requires about 0.5 TB per year. A set of ROOT macros then create the spectrogram plots from those spectro. They require about 16 computing nodes and a archiving disk space of about 200 GB per year.

7.3.1.6 DQ developments

Any development of a new DQ flag or a new tool to help commissioning and glitch investigations will require to do some tests using off-line raw data or Omicron triggers. Such tests will need a set of 4 to 6 computing nodes not used by any online task. The output of those tests will require about 0.5 TB of disk space to be stored temporarily.

7.3.1.7 DQ flags reprocessing

This is mainly a set of scripts to easily manage the reprocessing of the data quality flags and the reprocessing of the Omicron triggers. This will be done at the Lyon computing center and will require a easy and reliable access to the raw data stored there, as well as the use of 600 computing nodes over one week or two. In addition, a reprocessing of the Omicron triggers may be needed and will require access to raw data at Lyon, and the use of 600 computing nodes for several weeks and at least 2 TB of disk space to store the Omicron triggers over 1 year.

7.3.1.8 DQSEGDB

This is a mySQL database to store the Data Quality (DQ) flags from LIGO and Virgo. This database is supposed to contain Science flag, Lock flag, Injection flags, and the offline and online data quality flags. It should be able to manage up to 250 millions of data quality segments and will be accessed through a central server, using dedicated client or web interface. The machine hosting the central server will require a large memory (more than 16 GB?) and the capacity to answer to various queries coming from tens of users in parallel.

7.3.2 Detector characterization: Noise Studies

The Noise Studies goals are to characterize the detector data, to identify the main source of noise and possibly the path the noise use to come into the detectors, to catalogue the identified noise, either identifying spectral lines or the non linear coupling ranking.

In this section is described how we implement these goals. We have a set of noise monitor (NM) pipelines and a common framework NMAPI (Noise Monitor Application Program Interface), as described in the CM, in which NMs are integrated. This means that in the NMAPI web interface are reported the daily results of the NM pipelines or it is possible to launch scripts linked to each NM, on user demand.

Each NM relies on its own software enviroment. Most of them needs only free software, integrated in the standard Adv Virgo software environment, others can require the use of commercial software as Matlab.

Some noise studies tools (NMAPI, WDF, NoEMi) rely on the use of MySQL database for the insertion of results of processed data. The databases, described in Computing Model document, are located in Cascina CC and we use NMAPI as web interface to perform query to them. We setup a Batch Queuing System (BQS) for the 'in-time' analysis, which at the moment is Condor based. Most of the noise analysis pipelines do not need to give results on-line, that is in few seconds as the data are acquired, but it is useful to have the results in time in such a way to give feedbacks information to commissiong and search group. This is true for the identification of spectral lines, for the ranking of non linear coupling, for the coherence. We implement the in-time analysis trough the use of BQS.

NMAPI contains summary pages for each pipelines which are hourly or daily updated, using cron jobs.

Using NMAPI web interface it is possible to launch scripts on the Cascina BQS (which in the testing set-up is based on Condor system), and have the results back as html pages.

7.3.2.1 NMAPI

It is extensively described in Computing Model document. It will be implemented on a single node, where its database is and will interface to BQS. **rt#129 (In progress)**

7.3.2.2 NoEMi

NoEMi is a tool for the in-time discovery and follow-up of frequency noise lines and narrow band disturbances in the ADE data. It analyzes raw frame files (the h(t) channel, the raw Dark Fringe (DF) channel and a subset of environmental monitoring sensors) looking for matching frequencies and similar patterns between the lines found in the science data and the environmental sensor data.

It runs every night on the data collected in the previous day. It generates daily web pages reporting on the run data quality and it feeds the Lines database, which is used in the vetoing procedures of the CW and Stochastic searches.
It runs on BQS, using cron jobs to produce summary pages or NMAPI for the script launching. **rt#126 (In progress)**

7.3.2.3 SILeNTe

SILeNTe is a non-linear system identification technique developed to identify linear and non-linear noise coupling mechanisms. It is an in-time analysis.The input are raw frame files It is implemented in Matlab, but it can run also trough NMAPI on Cascina BQS. It produces a list of rank values for the non linera coupling among channels. **rt#127 (In progress)**

7.3.2.4 Regression

It is an in-time analysis. This monitor allows to survey the bilinear coupling between different auxiliary channels and the data channel. The input are raw frame files. The analysis runs on the local Condor batch system. The plots produced by daily analysis are inserted in NMAPI framework. Also Regression has associated scripts to be launched on demand using NMAPI. **rt#130 (In progress)**

7.3.2.5 WDF

WDF finds triggers associated to transient signal events. It analyzes data in the time domain, using a wavelet transform, to find an excess of power in the data and identify the trigger. The input are raw frame files. The transient signal events are produced on-line and all the parameters which characterize the event are stored on line in a MySQL database in Cascina. It relies on cron jobs for the production of daily and hourly summary page, and on NMAPI BQS launching system to execute scripts on demand. **rt#125 (In progress)**

7.3.2.6 Coherence

This analysis in in-time analysis. The input are raw frame files, the output MySQL entries and plots. It is not fully integrated in NMAPI. The summary pages are integrated in NMAPI, but not the script launching **rt#130 (In progress)**

7.3.2.7 Non stationary monitoring

The NonStatMoni pipeline run on-line to monitor band-limited RMS in many bands BRMS and showing slow variations. The inputs are selected raw data channels, from the DAQ shared memory, the output are html summary pages and plots. It is not fully integrated in NMAPI. The summary pages are integrated in NMAPI, but not the script launching

rt#131 (In progress)

7.4 Science data analysis workflows

This section describes those necessary software related tasks and plans which is required to get the various data analysis software running. As such it should not contain search plans or various configuration of physical parameters, but only those architectural, software and hardware related tasks which are necessary to be done in order to get the pipeline running.

The description should include

- the current situation
- any problem or issue to be solved, improved
- the development path with breakdown of the tasks
- a status report

As a short example please have a look to the CBWaves description.

7.4.1 CBC

7.4.1.1 MBTA

7.4.1.2 Testing GR pipeline

TIGER (Test Infrastructure for GEneral Relativity) is an analysis pipeline built on LALInference, the joint LIGO-Virgo toolbox for CBC model selection and parameter estimation. It aims to do two things:

- Provide reliable model selection between a non-GR model (which assumes that one or more phase coefficients are not as predicted by GR) and the model that GR is correct;
- If on the basis of model selection it turns out that there is no reason to doubt GR, provide stringent constraints on the phase coefficients through parameter estimation.

Model selection requires the calculation of a background distribution, *i.e.* the distribution of the "detection statistic" for GR violations when GR is correct. Currently this can be done reliably with binary neutron star coalescences, because for that case excellent waveform models are already available. Moreover the long BNS waveforms are not very prone to glitch-induced problems. However an important goal is to extend TIGER to all stellar mass CBC events: BNS, NSBH, BBH. Currently the background calculation is computationally expensive; this could be mitigated by an interpolation technique such as reduced order modeling.

Future development of TIGER is envisaged to proceed as follows.

- Arrive at a clear picture of the behavior of background for the case of BNS by using time slides
- On the basis of this, formulate data quality requirements tailored to the problem at hand
- Implement an algorithm for simultaneous fitting of signals and glitches
- Implement reduced order modeling, to begin with for BNS
- Implement semi-analytic waveform models for NSBH/BBH (precessing-spin EOBNR, PhenSpin, or IMRPhenomB)
- Implement reduced order modeling for the general CBC case

Deadline: 2015/03/30

7.4.1.3 GWTools - CBC@Home

GWTools [33]- The Gravitational Wave Data Analysis Toolkit is a collection of various routines widely used in the GW community. The elements of the library can be used as building block to create a data analysis application of any kind. Currently all the building block is implemented which is necessary to do physical analysis, we are lacking manpower and time to complete the various tasks. It is intended the GWTools will have two main application:

- CBC@Home - Running CBC analysis on Boinc resources
- GWTools4CW- Implementing the Rome Frequency Hough algorithms

The CBC@Home project's aim is to use the resources of the Boinc volunteers comunity to perform CBC data analysis. This has never been done before, as such it is really a challanging task. There is a long development road, as such here we just describe some of the first necessary tasks.

- (**rt#86 ; In progress**) Create a Visual Studio project for GWTools on Windows
- (**rt#87 ; In progress**) Compilation of GWTools library under windows
- (**rt#88 ; New**) Create a X86_64 statically linked minimal test executable
- (**rt#89 ; New**) Try submitting the test application to Boinc resources
- (**rt#90 ; New**) Set up the server side of the CBC@Home project
- (**rt#91 ; New**) Test client - server communication from inside the application
- (**rt#92 ; New**) Data download and processing in the application
- (**rt#93 ; New**) Data upload testing from within the application
- (**rt#94 ; New**) Number crunching - test the correctnes of the results
- (**rt#95 ; New**) Add meaningful calcualations, for exexample filtering a single template
- (**rt#96 ; New**) Compare results of the same computation on different computers
- (**rt#97 ; New**) Scheduling, data distribution, result collection and display

Deadline: 2015/03/30

Of course these are just the main milestones, many of the above tasks will give birth to tens of subtask impossible to foreseen and exactly define right now.

7.4.1.4 CBWaves

CBWaves [34] - the Compact Binary Waves is a state-of-the-art gravitational wave generator simulating the waves emmited by generic configugartion spinning, eccentric black holes and / or neutron stars. It implements all the currently known post-Newtonian contribution valid for the motion and waveform emission of generic configuration binary stars.

It is a generic tool based on C++ and having CMake based built system, and its usability is much broader than the special generators used in Ligo - Virgo CBC groups. As such it cannot directly be embedded into the otherwise automake and C based lalsuite software packages.

In order to make it available for the community there is a need for some reorganisation:

- (**rt#80 ; Completed**) Internal reorganisation of the code
- (**rt#81 ; Completed**) Creating a shared library
- (**rt#82 ; In progress**) Creating a C wrapper

- (**rt#83 ; New**) Creating a C++ wrapper
- (**rt#84 ; New**) Creating an example application
- (**rt#85 ; New**) Test linking agains LALsuite and burst group software

Deadline: 2014/04/30

Once these are done it will be easily usable for the members of the collaborations.

7.4.2 CW

7.4.2.1 Rome PSS pipeline

The CW Rome group has developed analysis methods for two different kind of search, the targeted search for known isolated neutron stars, with the narowband search extension, and the all-sky search for unknown isolated neutron stars. While the first has a relatively small computational load, the second poses important computational issues, which we have considered in the scheme of the procedure (which is thus hierarchical), and we are addressing by some codes optimizations, which might imply to use GPUs for some parts of the procedure.

Below we detail the organization and architecture of the pipelines. They have a preliminary part which is the same: the construction of the SFDB (short FFT date base) files, which is done by usingin input, as explained in the CM, only the frames files of calibrated data, h(t). The code for the production of these files is written in C, is embedded in NoEMi and thus runs every night together with the noise online analysis. If needed it can run offline, using local batch systems. We have run it in the Rome farm and at CNAF.

7.4.2.1.1 The targeted search of known neutron stars Starting from SFDB files a small, fraction of Hertz, band is extracted and saved in a `.sbl` file. On these data the barycentric and spin-down corrections are applied, data are down-sampled a final cleaning step, in which outliers are removed, is done and the detection statistic is computed. This is used to assess detection significance and then to compute an upper limit or to estimate signal parameters. All the codes are written in Matlab. The analysis is typically run in background on the Rome farm. For a single target pulsar the full analysis takes about 2 days per year of data on a 8-core node.

Recently the analysis method has been extended to narrow-band searches in which a small range of frequencies and spin-down values is analyzed. The main difference with respect to the targeted search case is that a number of intermediate corrected time series is constructed and used to build the detection statistic for each value of the frequency and spin-down. This part of the analysis is the most computationally demanding. At the moment the corresponding Matlab code is compiled and run on the Grid, with one job for each spin-down value (and the full frequency band). Each job takes about 3 days on a 8-core node for 3 months of data. No further development/change is foreseen at the moment.

7.4.2.1.2 The All-Sky search of unknown neutron stars From SFDB files we produce peakmap files, which are the input to the search. The code is a C code which can run under local batch systems, without any particular architectural need. We have run it in the Rome farm and at CNAF.

The next steps are done with Matlab compiled codes. It can run interactively or, as we typically do, under the GRID, using Python scripts.

The documentation of the functions is at:
`http://www.roma1.infn.it/rog/astone/doxygenMATLAB/output/html/index.html`.

We have a small executable, shipped in the input sandbox. Libraries are installed in the Software areas at CNAF and Rome, reachable by Grid jobs with the same path ($VO\_VIRGO\_SW\_DIR/matlab/..$)

Peakmap files are preprocessed and splitted in a number of files so that each job reads only two files (frequency bands and sky region). Data preparation in the low frequency region, [10-128] Hz, takes a few hours on a single machine. The data set is then transferred to CNAF storage and registered to lfc.

We are working to have that each job will read the input files with LCG tools and produces two output files, which we plan to register on the GRID storage system with the same tools (work in progress).

JDL files are automatically produced using Python scripts. We have also a command-line tool to: submit the jobs, monitor their status, download output files of completed jobs, identify and re-submit failed jobs.

This framework can be used to submit jobs to any GRID site, where we only need to install Matlab runtime libraries on the software area. Work is in progress to implement in the analysis codes the writing on and reading from the GRID of files in a completely trasparent way.

To summarize the work in progress (we still have to add the tasks to Redmine)

- enable the jobs to read the input files with LCG tools and produce two output files, which we register on the GRID storage system with the same tool. This will allow to submit the jobs to every GRID site, where we only need to install Matlab runtime libraries on the software area;
- Test on GPUs of the most computationally heavy part of the pipeline.

Documentation under the INFN wiki (password protected):
`http://wiki.infn.it/strutture/roma1/experiments/virgo/cwgridtools`

7.4.2.2 Polgraw pipeline

The Polgraw pipeline is a time-domain, narrowband in frequency standalone code consisting of two parts: the first one, search for candidate periodic signals is written in standard C (the only requirements from the shared libraries point of view are `GSL`, `GSLBLAS` and `FFTW3`; the code was tested on various compilers, but the most of runs was performed with the GNU compiler, `gcc`). The compilation is maintained with the standard `make/gmake` command (type `make` or `make search` in the command line). By current design, limited mostly by the performance of processors on the FFT, the most important and computationally-expensive part of the analysis, the input data consist of three files: narrowband time series of the detector data `xdat_d_b.bin`, ephemeris of the detector `DetSSB.bin` and grid generating matrix `grid.bin`. The data `xdat_d_b.bin` spans the length of 2 sidereal days and it is sampled at 0.5 s and thus consists of $N = 344656$ double precision numbers. In total, the input data for every 1 Hz frequency band `f` and the two day time slot `d` is about 10 MB. The output depends on the number of candidate signals found in the data and for a given (`f`, `b`) may be of the order of 1 GB. The output consists of a binary file with sequences of five numbers: position of the candidate signal on the grid, frequency and frequency derivative of the signal and the SNR (from our experience, this may produce a bottleneck in the analysis data flow, when too many processors are trying to write the output to the same location; this may be mediated by writing the data locally on the processing unit and gathering them later). A typical call from the command line is as follows:

```
./search -d .  -o ./candidates -i 42 -b 271 --whitenoise,
```

where the test input data (`xdat_42_271.bin`, `DetSSB.bin` and `grid.bin`) is located in the directory `42/` and the output is place in the directory `candidates/`.

Second part of the pipeline that takes into account the post-processing (vetoing and looking for coincidences among the signals from the two-day stretches at the corresponding frequency) is written in Pascal, compiled using `freepascal`, but will be rewritten in standard C. The search for coincidences is much less demanding in terms of computational power than the search for candidate signals.

The source codes of the pipeline can be found at

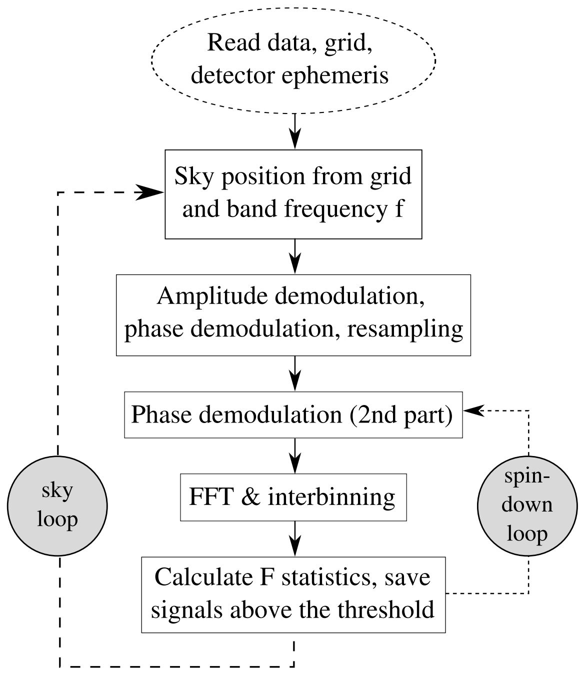
Read data, grid, detector ephemeris
Sky position from grid and band frequency f
Amplitude demodulation, phase demodulation, resampling
Phase demodulation (2nd part)
FFT & interbinning
Calculate F statistics, save signals above the threshold
sky loop
spin-down loop

Figure 7.1: Flow diagram for the Polgraw all-sky pipeline search for candidate signals (serial version).

`https://wwwcascina.virgo.infn.it/cgi-bin/cvsweb/cvsweb.cgi/PolgrawAllSky`

This directory contains the documentation (`doc/`), sample data for tests (`42/`), the source files (`src/`) as well as the slightly modified sources used for software injections in order to estimate the sensitivity of the search (`src/Sensitivity/`). The description of the procedure and the first science run with the VSR1 data is available at `arXiv:1402.4974` (see references therein, especially Astone et al. 2010 for the construction of the optimal grid and the detailed description of the search algorithm for candidate signals). The pipeline design allows for dividing the grid of search parameters for every frequency and time stretch (`f,b`) into smaller regions on the sky, thus allowing for efficient parallelization, since the sky positions are completely independent (`-r <grid-range.dat>` option). The pipeline was tested on the available queuing and scheduling systems (Torque PBS, LoadLeveler) and Condor.

The development plans for the Polgraw all-sky search pipeline are as follows:

In progress: generalization of the pipeline for the analysis of data from a network of detectors (the input and output data will scale up like the number of detectors),

Completed: massively parallel version of the code, using the MPI framework (with an internal, scalable scheduler taking into account e.g., different execution times for the instances of the code for different search frequencies), that can be launched in the grid environment as a single job. Current version scales up to 50000 cores,

In progress: researching for possibilities of scaling to more than 10^5 cores without the penalty from the Amdahl's law and with a proper weak scaling (since the algorithm does not need much of inter-core communication, the most possible bottleneck is the output writing),

In progress: GPU version of the code. Working version of the CUDA Nvidia code is already available (current speedup w.r.t. the serial version of the search code is $\gtrsim 30$), works are mainly done concerning the optimization and documentation,

In progress: Hybrid version of the code, combining the MPI and GPU approaches that will be optimal w.r.t. speed over the relevant parts of the pipeline,

in progress: researching for an alternative approach at the massively-parallel computing; tests were performed with the DAKOTA tookit (A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis, `http://dakota.sandia.gov`). Such approach, or a similar, could be used to manage the workflow of many small jobs created by e.g., dividing the sky parameter space conveniently for a single unit (`f,b`) job (first stage, search for candidate signals) into sufficiently many small-enough processes. Such an approach provides an additional benefit for the second part of the pipeline (search for coincidences among the candidate signals), since the data is already sorted into data blocks w.r.t. sky positions.

7.4.2.3 Pisa pipeline

7.4.2.4 GWTools4CW

For a description of GWTools please see the CBC subsection. Since the algorithms used in the Frequency Hough search method are very well paralallizable using many-core architectures, it is straightforward to do a reimplementation using GWTools.

The project is half ready now, we are lacking of time and manpower to complete. Anyway we envisage the following roadmap for the forthcoming year which will lead us to a working prototype of the analysis.

- **(rt#98 ; Completed)** Implementing the peak map file reading from p12 format.
- **(rt#99 ; Completed)** Differential Hough-map creation
- **(rt#100 ; Completed)** Integral Hough-map creation
- **(rt#110 ; In progress)** Peak finding
- **(rt#111 ; New)** Peak map reading using VOB files
- **(rt#112 ; New)** Pre-cleaning of stationary lines
- **(rt#113 ; New)** Generating the sky grid
- **(rt#114 ; New)** Including doppler modulation
- **(rt#115 ; New)** Finding hardware injections
- **(rt#116 ; New)** Correctness test on CPU, GPU
- **(rt#117 ; New)** Various optimisations
- **(rt#118 ; New)** Local job usbmission
- **(rt#119 ; New)** Using the distributed job submission framework

Deadline: 2015/03/30

7.4.3 Burst

7.4.4 Stochastic

Chapter 8

The Virgo Virtual Organisation

8.1 Obtaining an X509 digital certificate

8.1.1 Recognized CAs

8.2 Registering and maintening user credentials

8.3 Using Grid services with the certificate

8.4 The information system, supporting sites

Chapter 9

Authentication and user credentials

Solid and secure authentication againts and authorization for various computing and other services is crucial from almost all point of view.

9.1 Accessing LIGO resources

So far LIGO was using x509 certifiacte [?] based authentication and authorization solution for dedicated LDG clusters. Virgo users could equally connect to these resources by registering in the LIGO Roadster. In order to ensure the smooth continuation of the practice we needed a confirmation from our LIGO colleagues that this system will be continued in the Advanced Era. This has been done and as part of task **rt#72** it is confirmed that LIGO will indefinitely support certificate based auth for LDG clusters and will also provide username
pass based access to XEDE resources[29].
Deadline: 2014/01/30
Status:

- **rt#72** Completed

9.2 Accessing external Computing Centers

While it is planned that external CCs will be used mainly remotely as part of the distributed job submission framework, it is very useful to have the possibility of direct login to the user interface machines of various clusters for developing, debugging purposes.

However, unfortunately different sites are using completely different authentication systems and this requires lot of user name and password to be used by the user. Our goal would be to minimize the number of credentials necessary for an average user to access to external computing centers without lowering the security considerations. This work requires lot og negotiations and careful planning with the administration of the external computing centers.

One of the solution is to use certificate based auth for all the external computer center. This would have the following benefits

- Absolutely compatible with the LIGO auth/authZ system
- Uses a dual security protections based on 'property and knowledge', i.e. the private key of the certificate and its password.

- Does not require from the site to store user credentials, i.e. no password hases are necessary to be stored anywhere. Very safe from the user point of vew.
- Compatible with Grid authentication systems
- Many web service are also supporting it

We should at least allow certificate based auth for the bigger computing centers. Necessary steps for this are

- **rt#73** (In progress) Negotiating with Lyon
- **rt#74** (On hold) Negotiating with CNAF
- **rt#75** (New) Negotiating with ROME
- **rt#76** (New) Negotiation with Wigner
- **rt#77** (New) Negotiation with Nikhef

Deadline: 2014/06/30

Certificate based auth is only one of the possibility and also comes with several disadvantages. Other solutions such as AAI/SAML based international or organisation federations are also considered, though requiring significantly more work to make it happen.

9.3 Accessing EGO services

Various authentication and authorization solutions are used by EGO services. Currently there are far too many necessary credentials that a user must know to be able to use these services in practice, includeing

- public web passwords
- firewall authentication
- Active Directory credentials
- linux cluster credentials
- Workarea user names and passwords

The above situation should be consolidated and a more uniform auth system to put in place. Again, there are various solutions to be considered.

- Should require minimal number of user credentials
- Should not be too complicated, should not involve big internationals federations since that would involve too many organisational and administrative bottleneck
- It should be possible also for LIGO colleagues to use the service
- Required maintenance should be as low as possible

The EGO site is working on the upgrade of the local AAI system in order to rationalize the accounts needed in Cascina for the various services and provide more security in the unix domain. The upgrade will occur along these lines:

1. designing the user profiles and roles, among which the various kind of Virgo users personalities.

2. designing the LDAP schema with the attributes compatible with the current AAI federation systems(such as for web-based SAML federation with Italy, Europe and LIGO) and the most of the local services.
3. integrating the schema either directly in Active Directory or in a general user provisioning system and setting up one attribute authority for authorization
4. upgrading the authentication and authorization services in Cascina to support the above mentioned profiles, in particular the unix domain with Kerberos authentication and LDAP authorization and the web domain with SAML-based authentication and authorization

Realistically, even if succeeding to bring up the infrastructure at the end of 2014, the conversion of the applications to the new AAI system will occur on a longer time period, at worst following their natural evolution.

9.3.1 Accessing web services

We dedicate a subsection to this issues, not because of it technical difficulty or complexity, but because its effect on Virgo's apperaence on the world wide web.

- **rt#78** EGO - Virgo web serves should have proper commercial or academic (EUGridPMA recognized) [30] certificate instead of using self-signed certificate.

Deadline: 2014/12/30
Status:

- **rt#78**: New

Chapter 10

An alternative computing architecture

10.1 Introduction

The very goal of any work related to the Virgo experiment should be to extract as many science out of the data as possible. There is many necessary ingredient of such a goal, among which computing is a very important one. Any computing strategy, computing model should support this goal in its full power.

The Computing Model - in its original and current form - and the first part of this Implementation Plan presented so far, describes a computing strategy which tries to fullfill the requirements of the commissioning and scientific data analysis workflows by mapping the problems onto, and providing solutions using our currently available computing hardware and middleware infrastructure.

However, this approach suffers from many artifacts and has major deficiencies which impose serious limitations on the quality and efficiency of our scientific explorations.

This section is dedicated for an alternative Implementation Plan which does not try to fit the problems into the restrictions of our existing infrastructure, but suggest a solution which is much more optimal from the science, work effificency, cost and manpower point of view.

10.2 Problems

10.2.1 Limitations of the current computing architecture

Without the desire of completnes here we list some of the major limitation of our current model. The current computing hardware architecture consist of the Cascina online farm and the 32 french, italian, hungarian and dutch site of the EMI Grid which supports the Virgo VO, with the two most important being CN2P3 and CNAF, located in Lyon and Bologna.

- Virgo computing reources are too much fragmented with all the consequences of this fact: Different sites are
  - having different authentication system
  - using different batch systems
  - having different and isolated support and ticketing systems
  - different data access methods

  - different, incompatible local job submission solutions

- The sites are in a different administrative domain, a fact which multiplies the amount of communication necessary for any kind of development and changes or just for get the things running properly.
- Too much emotional interest in using or favouring various (sometimes custom made) solutions in some cases putting the technically optimal or simpler solution in the background.
- Administrators of the computing resources are not part of the everyday work of the LIGO - Virgo collaboration as such it is naturally much more difficult for them to understand and handle the various problems in a level where they are belonging to.
- Too much people and too much (at least 9!!!) different body is needed to interact with each other even in the simplest computing questions (VDASC, EGO-Computing, CTCC, ECC, JECC, Lyon administration, CNAF administration, DASWG, EMI middleware developers, etc)
- With the help of the Grid middleware the Grid sites can be used uniformly, however
  - Grid middleware job submission is not compatible with LIGO solution
  - there are no really powerful tools available to monitor and manage grid workflows submitted using the native grid commands
  - more complex job submission frameworks such as for example Dirac[10] can solve lot of these problems, but again, Dirac is incompatible with any workflow used so far. It would be very tedious to convert the common LIGO-Virgo workflows to Dirac, and continously maintain multiple version of the same code would be a waste of manpower
- Grid sites are not owned and not administered by the Virgo collaboration, as a consequance,
  - In order to exploit the full potential of the computers we have a user should know 5 different kind of job submission method, such as SGE for Lyon, LSF for CNAF, Condor for LIGO clusters, grid job submission methods for the Grid. Of course nobody takes the effort to build a pipeline which is compatible with all these, and nobody learns all the diferent job submission, as such everybody is using only a fraction of our infrastructure. One could of course say that let's use grid submission is the only method, however local job submission methods have valuable advantages for example in terms of debugging, speed and closeness of local administrators.
  - Virgo software of any kind have to follow and be compatible with the operating system and software environment enforced by the grid middleware
  - Virgo members have no administrative privileges on the execute machines, as such virgo software have to be installed user space. For this to work easily Virgo uses the CMT[17] configuration manager, which - despite being convenient to use for ones using it every day - does not compatible at all with the standard way of linux packaging and distribution. This seriously limits the usability of Virgo software by our LIGO colleagues and by any other external collaborator. Furthermore it is too slow to be used as a nightly build system.
  - LIGO usage of Virgo resources vs Virgo usage of LIGO resources is completely assymetric and creates a very inconvenient situation from many perspective for both LIGO and Virgo.
- Different sites have different storage systems, as a result of which
  - They use different protocols and we have to duplicate the number of tools used for data transfer.

  - They use different data access methods, as a consequance despite a job can be submitted transparently to both of them using the grid middleware the job has to be customized to all diferent environment, which is again just a duplication of work.
  - Data cannot really be accessed from outside, from the end-user laptop without using special client tools.

- In order to find and access the files on the Grid we need to use File Catalogs. These catalogs (such as for example LFC[31]) are not compatible with the way as one of the most intensively used and complex CBC workflows are finding data. Again, this results a duplication of work since an LDR compatible file catalog format is necessary for these workflows...
- Many common (smaller) LIGO-Virgo workflow assumes a shared file system over the execute nodes. The is not the case in general for the grid, as such it is not possible to easily port them to our framework.
- Lyon and CNAF has the full data stored on site - even if it requires different access method - to run jobs on other Grid sites requires continous data transfer and staging which is not a problem for less data intensive jobs but is a limitation for some of the pipeline.
- Cloud services and virtualisation solutions are nor readily available on these sites. (CNAF supports some level of virtualisation, that probably could be modified in the future to meet our needs. Has to be checked.)

10.2.2 Summary of problems

Summarizing the difficulties in major groups would result the following list.

1. **computing efforts are difficult to focus**
2. **incompatibility with LIGO**
3. **fragmented, heterogenous resources**
4. **lack of manpower**
5. **communiction overhead**

Checking on the the list of the previous subsection item by item one could say that these are not major issues - which is true - however the ensemble of the problems results a really inefficient and expensive computing environment for Virgo. Of course there are collaborations that are living together all these problems, but either because they don't have other choice or because they have enough money and manpower to face these problems. For Virgo none of these cases is true and fortunately there is a way out. A future implementation of the proposal of the next section would solve almost all (at least many) of the above problem.

10.3 Proposal for a new computing architecture

The Virgo Collaboration should have a single, dedicated, LIGO and Grid compatible computing cluster and a few storage resources (for backup) all which is owned by the Collaboration and managed by collaboration members located closely and actively working together with data analysis groups !

In the following subsection we will explain and make clear all the benefit of this approach.

10.3.1 Specification of the ideal computing and storage resorurces

In this section we explain better how the computing model and requirements can be mapped to a newly implemented architecture.

- The new Virgo computing infrastructure would consist 3 site: Cascina and site A,B.
- Site A is only for (tape based) data backup, while site B both for data and for all the compute task.
- All kind of low latency analysis is running in Cascina as before.
- The data from Casinca is transferred to sites A, B. It is written to tape at Site A and stored (at least the data of the actual year) on disk at site B. (A third copy of the data will be at LIGO resources.)
- The dataset from LIGO is copied to site B only.
- Site B should have the following architecture
  - Uniform hardware (or virtualized resource) dedicated for Virgo (and LIGO).
  - Condor batch system for local job submission with Condor-C and Condor Flocking enbled
  - Disk based shared file system with POSIX access
  - Various virtualisation techniques available
  - A possible GPU extension of the nodes enabled
  - Gridftp interface for LIGO and Virgo data transfers
  - LIGO compatible authentication and login system
  - Possibility for EMI Grid submission
  - Single entry point for support, ticketing, etc.
  - User friendly job monitoring and reporting tools

10.3.2 Advantages of the new approach

The proposal presented would immediately solve a huge number of problems:

- All the LIGO pipeline could run seamlessly without any kind of modification and waste of manpower.
- LIGO colleagues would not need to use different authentication system to access Virgo resources
- There is a need to use only one kind of job submission method (instead of the above listed 4) in order to be able to use our resources in its full power.
- Tremendous reduction in communication while allowing everybodies needs and requirements to be fullfilled.
- No need for additional file catalogs. The file catalog populated on data transfer can directly be used
- No need for user space installation of the software (still will be always possible)
- Any kind of virtualisation is possible
- Far easier collaboration and sharing of results with LIGO colleagues

- Disk based archived data can be exported to end user via industry standard linux solutions such as for example NFS4.
- No more fragmentation of efforts and resources
- Inclusion of new hardwares (natively or with bare-metal virtualsiation) such as for example GPUs will be much easy, and accessible for everybody !
- Virgo computing could play a much important role in a possible future discovery.
- Uniform administration and high-end hardware allows for a much cost-effective operation.
- Virgo software and Virgo developments will be much more easier accessible for LIGO colleagues
- No need to develop and maintain alternative job submission frameworks
- Virgo software can be properly packaged and published in standard linux repositories...
- ...as a consequance Virgo can use proper nightly build systems which are able to build Virgo software stack.
- The interconnection with various LDG sites would be much more easier, which is good for load balancing
- Data intensive jobs can also easily run with no problem
- No need for special data access methods, native POSIX access possible
- Many problem, tasks listed in the implementation plan would not be necessary, as such we could save valuable manpower for scientific data analysis instead of computing.
- Users using the same tool and the same site will be able to very effectively help each other (see the example of LIGO clusters), in contrary with the current situation where multiple methods on multiple sites are used.

As a result of the the benefits above, the problems listed in 10.2.2 subsection would be resolved in many extent. The new, suggested solution

1. would allow for a much more focused, coordinate computing effort
2. would be absolutely compatible with LIGO solution
3. would be homogenous, uniform and integrated, no fragmentation of resources
4. would require far less manpower to maintain
5. would require far less communication to work effectively

10.3.3 Possible disadvantages of the new approach

This section will contain a list about the possible disadvantages and / or dangers - if any - of the new, suggested computing architecture.

10.3.4 Transition

Of course such a big change cannot and should not be done from one day to the other. It requires careful evaulation and testing period. We should also be careful moving or leaving our existing resources and switching to new ones. However this should all happen with our goal in mind that we should create a more uniform and more integrated computing architecture. Also beside having a single homogenous officially supported solution Virgo should have a ”second leg”, i.e. we should always be able to use the Grid submission methods. This will even more strengthen the solution and can also be very useful for load balancing in peak periods. A breakdown of necessary task for this evaulation and testing period will be given in this section.

10.3.5 Technical specification

The previous subsection gave a high-level description of an optimal computing infrastructure for Advanced Virgo. We also have a very precise technical specification in mind for such a solution which - after a series of consultation with all the interested / affected parites - will be described in this section.

10.4 Conclusion

The above thoughts, recommendations and suggestions are based on our experience and inputs from experts and power users of the LIGO - Virgo community. A set up of such a new comuting architecture means a lot of work and some investment, but we beleive that it will definitely pay out in terms of a.) operation cost b.) scientific efficiency c.) future potential and d.) reputation, already in the short term ! As such we would definitely like to see this transition starting to happen.

Chapter 11

Man power tables for all workflows

In these sections we detail the FTEs involved in each activity and we also give names of responsibles and collaborators at the actual date (March 2014). We indicate also the foreseen needed FTEs to reach the goal by the year 2015. The goal has been indicated, for each workflow, in Part IV of the CM. The sections here have been divided using the same classification introduced in the Part I of the CM (workflows description) and used through the document. Some information on the workflows needs and their main characteristics has been given in Part IV (Software management) of the CM. It should be clear that the manpower indicated here refers only to the activity present in the CM at today (2014) and thus any other data analysis activity not related to the workflows of the CM has not been mentioned here.

11.1 Detector characterization (Detchar)

11.1.1 Detchar/Data Quality

Table 11.1 gives the information for the Detector characterization/data quality activities.

Pipeline	Responsible and Collaborators	FTEs (March 2014)	FTEs needed (2015)
New veto development (including Omicron, UPV and Excavator)	F. Robinet 0.2, B. Swinkels 0.1	0.3	0.9
On-line vetoes production checks and reprocessings	Verkindt 0.3, F. Robinet 0.1, N. Leroy 0.1, G. Hemming 0.2, M.A. Bizouard 0.0	0.7	1.5
Generic tools (like channels DB) (DQSEGDB)	G. Hemming 0.1, D. Verkindt 0.1 M.A. Bizouard 0.1	0.3	0.3
DMS	F. Berni 0.2, V. Dattilo 0.05, D. Verkindt 0.05,	0.3	0.3
Spectrograms	D. Verkindt 0.1	0.1	0.1
Monitoring tools (MonitoringWeb, dataDisplay, DQperf, Omiscan...)	D. Verkindt 0.2 F. Robinet 0.1	0.3	0.6
DQ categorization, links with DA, glitch studies, glitch classification...	F. Robinet 0.2, D. Verkindt 0.2 M.A. Bizouard 0.0 N. Leroy 0.1 F. Marion 0.1 B. Swinkels 0.1	0.7	2.0
TOTAL		3.2	5.7

Table 11.1: Detchar/Data quality manpower information

11.1.2 Detchar/Noise studies

Table 11.2 gives the same for Detchar/Noise studies workflows

Pipeline	Responsible and Collaborators	FTEs (March 2014)	FTEs (2015)
NoEMi	A. Colla (not staff)	0.3	0.3
SILeNTe	F.Piergiovanni (staff)0.1, G.M.Guidi (staff) 0.1	0.2	0.2
Regression	M.Drago (not staff), G. Vedovato (staff) V. Re (not staff)	0.15	0.3
WDF	E.Cuoco (staff)	0.3	0.3
NonStatMoni	-	0	0.2
Coherence	-	0	0.2
TOTAL		0.95	1.5

Table 11.2: Detchar/Noise studies

11.2 Scientific analysis

11.2.1 Low-latency searches

Table 11.3 gives the FTEs status and needs for each pipeline.

Pipeline	Responsible and Collaborators	FTEs (March 2014)	FTEs (2015)
CBC low-latency (MBTA)	G. Guidi 0.5, F. Piergiovanni 0.4 F. Marion 0.4 B. Mours 0.1 D. Buskulic 0.5	1.9	3.5
TOTAL		1.9	3.5

Table 11.3: Low-latency (the one which runs at AdV site). The table includes also the development work which is done and run offline

11.2.2 Burst offline

Table 11.4 gives the manpower for Burst searches.

Pipeline	Responsible and Collaborators	FTEs March 2014	FTEs 2015
cWB offline	G. Vedovato 0.9, M. Drago 0.35 V. Re 0.35, Lazzaro 0.35, M. Tringali 0.7, G. Prodi 0.25	2.9	?
cWB-GPU offline	E. Chassande-Mottin 0.5 Lebigot 0.6	1.1	?
STAMP long bursts	Franco 0.75, M.A. Bizouard 0.15, P. Hello 0.2	1.1	?
X-pipeline	N. Leroy 0.25 Was 0.1	0.35	?
Cosmic String	F. Robinet 0.25	0.25	?
TOTAL		5.7	??

Table 11.4: Burst workflows

11.2.3 CBC offline

Table 11.5 gives the manpower for CBC workflows.

Pipeline	Responsible and Collaborators	FTEs March 2014	FTEs 2015
ihope, GWTools	C. Van Den Broeck 0.1 G. Debreczeni 0.2	0.7 0.5	1.0 0.5
LALinference	C. Van Den Broeck 0.25 Agathos 0.4 Meidam 0.4 Vasuth 0.4 Nelemans 0.2(*) Ghosh 0.2(*) Shah 0.5(*)	2.35	3.0
TIGER	Chris Van Den Broeck 0.25 Agathos 0.4 Meidam 0.4	1.05	2.0
EMGW joint with Bursts	Marica Branchesi 0.3 G. Guidi 0.1 F. Piergiovanni 0.1 A. Chincarini 0.2 L. Rei 0.4	1.1	2.0
TOTAL		5.2	8.0

Table 11.5: Table for CBC (offline). (*) indicate a work on Parameter Estimation strictly connected to the EM follow-up activity

11.2.4 CW

Table 11.6 gives the manpower for CW searches.

Pipeline	Responsible and Collaborators	FTEs March 2014	FTEs 2015
Frequency Hough (PSS)	P. Astone 0.8 A. Colla (not staff) 0.1 S. D' Antonio (not staff) 0.8 S. Frasca 0.9 C. Palomba 0.4	3	3.5
Polgraw AllSky	A. Krolak, M. Bejger, K. Borkowski, O. Dorosh	0.8	1.3
Rome Targeted (PSS)	C. Palomba 0.55 P. Astone 0.1 , A. Colla (not staff) 0.1, S. D'Antonio (not staff) 0.15, S. Frasca 0.1	1	1.4
Polgraw Targeted	A. Krolak, M. Bejger	0.5	0.8
Direct searches	I. Ferrante, O. Torre, G. Cella	1	2
Polynomial	Jonker Reiner	1	1
TOTAL		7.3	10.0

Table 11.6: Table for CW pipelines.

11.2.5 Stochastic

Table 11.7 gives the manpower for stochastic work.

Pipeline	Responsible and Collaborators	FTEs March 2014	FTEs 2015
Isotropic	T. Regimbau 0.2 G. Cella 0.5 F. Di Renzo 0.5 D. Meacher 0.25 L. Martellini 0.2, JD Fournier 0.1	1.75	2.2
Spherical Harmonic	T. Regimbau 0.2, D. Meacher 0.25,	0.45	1
TOTAL		2.2	3.2

Table 11.7: Stochastic

11.2.6 GWTools GPUs project

Table 11.8 gives the manpower for GPUs work.

Pipeline	Responsible and Collaborators	FTEs March 2014	FTEs 2015
GWTools	Gergely Debreczeni 0.2 (*)	0.6	0.6
GWTools (CW)	Gergely Debreczeni Pia Astone 0.1 S. D' Antonio 0.05 C. Palomba 0.05	0.6	0.6
TOTAL		0.8	1.2

Table 11.8: Table for GWTools. (*) already listed in the CBC manpower for ihope, thus not added to the total here

11.3 Summary table of 2014 FTEs and needs for Detchar and Scientific analysis

Pipeline	FTE (March 2014)	FTE (2015)
DETCHAR/VDQ	3.2	5.7
DETCHAR/Noise	0.95	1.5
CBC Low Latency (MBTA)	1.9	3.5
CBC	5.2	8.0
BURST	5.7	?
CW	7.3	10
STOCH	2.2	3.2
GWTOOLS	0.8	1.2
Total	27.25	33.1

Table 11.9: Summary of the FTEs and needs for Detchar and Science analysis.

11.3.1 Data transfer

Table 11.10 gives the manpower involved in DT work.

Pipeline	Responsible and Collaborators	FTE (March 2014)	FTE (2015)
Low-latency			
Bulk	L. Salconi (EGO)		
aLIGO to AdV	EGO or Virgo, depending on the strategy		
AdV to aLIGO	EGO with LIGO colleagues		

Table 11.10: Table for Data transfer tools.

11.3.2 Data management work

Table 11.11 gives information on the data access manpower.

Pipeline	Responsible and Collaborators	FTE (March 2013)	FTE (2015)
Scientific pipelines			
Others,			

Table 11.11: Table for Data management tools, having divided the item into two main sub-parts

11.4 Summary table of 2014 FTEs and needs for data trasfer and data management

Pipeline	FTE (March 2014)	FTE (2015)
Data transfer		
Data management		
Total		

Table 11.12: Summary of the FTEs and needs for data transfer and data management work

Bibliography

Bibliography

[1] Virgo coll.
Virgo note VIR-xxxx-13 (October 2013):

[2] mettere l' ultimo. LSC and Virgo coll. VIR-0271A-12 (May 2012): *https://tds.ego-gw.it/itf/tds/file.php?callFile=VIR-0271A-12.pdf*

[3] Software Problem Reporting *http://sprserver.ego-gw.it/mantisbt/login_page.php*

[4] Trello board *http://trello.com*

[5] The Virgo LogBook *https://tds.ego-gw.it/itf/osl_virgo/index.php*

[6] The SVN Trac project *http://trac.edgewall.org/wiki/TracSubversion*

[7] Skype internet phone *http://skype.com*

[8] SeeVogh *http://seevogh.com*

[9] TeamSpek *http://teamspeak.com*

[10] The Dirac framework *http://www.diracgrid.org*

[11] The Pegasus framework *http://pegasus.isi.edu*

[12] Redmine - a project management system *http://redmine.ego-gw.it*

[13] Concurrent Version System *http://www.nongnu.org/cvs/*

[14] Subversion *http://subversion.apache.org/*

[15] Alternatives to GIT *http://alternativeto.net/software/git/*

[16] The GIT revision control system *http://git-scm.com/*

[17] Configuration Management Tool *http://www.cmtsite.net/*

[18] CMAKE - Cross platform make *http://www.cmake.org/*

[19] Continous Integration Solutions *http://en.wikipedia.org/wiki/Continuous_integration*

[20] Jenkins - AContinous Integration Server *http://jenkins-ci.org/*

[21] Hudson - A Continous Integration Server *http://hudson-ci.org/*

[22] JetBrains - Build system *http://www.jetbrains.com/*

[23] CruiseControl - *http://cruisecontrol.sourceforge.net/*

[24] Advanced Packaging Tool - *http://en.wikipedia.org/wiki/Advanced_Packaging_Tool*

[25] Yellow Dog Updater - *http://en.wikipedia.org/wiki/Yellowdog_Updater,_Modified*

[26] RPM Package Manager - *http://en.wikipedia.org/wiki/RPM_Package_Manager*

[27] Deb Package format - *http://en.wikipedia.org/wiki/Deb_%28file_format%29*

[28] X509 certificate - *http://en.wikipedia.org/wiki/X.509*

[29] XEDE resources - *https://www.xsede.org/*

[30] EUGridPMA - *https://www.eugridpma.org/*

[31] The Logical File Catalog - *http://www.eu-emi.eu/emi-2-matterhorn-products/-/asset_publisher/B4Rk/content/lfc-1*

[32] The Dirac File Catalog - *http://diracgrid.org/files/docs/UserGuide/Tutorials/FileCatalogBasic/index.html*

[33] GWTools - Gravitation Wave Data Analysis Toolkit - *http://gwtools.org*

[34] CBWaves - Compact Binary Waves - *http://gravity.wigner.mta.hu/cbwaves*

[35] File Transfer Service - *www.eu-emi.eu/*

[36] The BitTorrent protocol *http://en.wikipedia.org/wiki/BitTorrent*

[37] Dirac File Transfer Service *http://diracgrid.org/files/docs/Overview/index.html?highlight=pilot*

[38] Ligo Data Replicator *http://www.lsc-group.phys.uwm.edu/LDR/*

[39] The Globus Toolkit *http://toolkit.globus.org/toolkit/*

[40] The Globus Replica Location Service *http://toolkit.globus.org/toolkit/data/rls/*

[41] Book-keeping of the data set transfered to CCIN2P3 and CNAF *https://tds.ego-gw.it/ql/?c=7867*