Data Intensive processing with iRODS and the middleware CiGri for the Whisper project

Briand X.* Bzeznik B.†

Abstract

blabla

blabla

Keywords

Data-Intensive, grid computing, distributed storage, Seismic Noise, Whisper, Cigri, Irods.

1 Plan

1. Introduction, abstract (3§, 1 page)
2. Background (7-10§, 1-2 pages)
   (a) The Whisper project
   (b) Size of observational data
   (c) Size of computational data
   (d) General Big data/ science of universe
   (e) Coupling scientific objectives / IT constraints
   (f) Data grid paradigm
   (g) Data grid in Grenoble
   (h) Collaboration Whisper/IT infrastructure
3. Software for data-intensive processing (6-12§, 1-2 pages)
   (a) General IT codes whisper
   (b) Computer language and librairies
   (c) Structure of codes
   (d) Package for raw data
   (e) Package for correlations
   (f) First optimisation of correlation
   (g) Second optimisation of correlation
   (h) Codes for analysis of correlations
   (i) Where we can run the codes
4. IT infrastrcuture for grid computing (6-12§, 1-2 pages)
   (a) Needs of coupling storage computation
   (b) Presentation Ciment
   (c) Coupling Irods/Cigri
   (d) Resulting Data-Grid
   (e) Presentation Irods
   (f) General Infra Irods (diagram)
   (g) Effective Nodes Irods (diagram)
   (h) General Cigri
   (i) Mechanism with OAR
   (j) Mechanism resubmission/besteffort
   (k) Description of a campaign
   (l) Low Interaction Cigri/Irods
   (m) Cigri V2 and V3 functionalities
5. Results and feedback (28-31§, 6-7 pages))
   (a) Whisper Use Case
   (b) Infrastructure Ciment experiences

*Isterre, Cnrs, email xav.briand@gmail.com
†Ciment, Université Joseph Fourier

2 Background 7-10§, 1-2 pages

The *Whisper* * project is a an european project on seismology whose goal is to study properties of the earth with the seismic ambient noise such that evolution of seismic waves speed. This noise is almost all the signal continuously recorded by the seismic stations worldwide (Europe, China, USA, Japan), except earthquakes. It offers new observables for the seismologists, new types of virtuals seismograms that are not only located at the place of earthquakes. For instance, one can obtain wave paths that probes the deepest part of the Earth.

Accordingly, this is one of the first project in the seismological community that studies systematically the continuous recordings, which represents a large amount of seismological data, of the order of several tens of terabytes. For instance, one year of the Japanese Network is about 20 TB or 3 months of the mobile network USArray represents 500 GB (it depends on the sampling of the recorders).

In addition, the calculation operations downstream may produce even more data than the observation data. To give an order of magnitude, more than 200 TB has been managed by the Whisper project at the same time. A classical processing produces 8 TB in 5 days. An other one 'read' 3 or 4 TB and 'produced' 1 TB in 6 hours. Many tests of the signal processing are done and computational data are deleted as and when required.

Nowadays, the earth sciences or more generally, sciences of the universe are widely engaged in data-intensive processing. This leads to design scientific workflow, towards data-intensive discovery and e-Science.

Reflected by the Whisper project, we have to organize the science objectives with the computer constraints. We have to take into account the duration of postdocs and PhD theses, as well as the availability of computer infrastructures and their ease of access. This leads to many questions about software development, including the genericity of computer code and the technical support. But it also influences in terms of choice of appropriate infrastrucures.

Even if this project has his own ressources, such a problem of data-intensive requires specific tools able to organize distributed data management and acces to computational ressources: a data grid environment.

The University of Grenoble offers, thanks to the High Performance Computing (HPC) centre *Ciment*, this kind of environement with the distributed file system *Irods* and the middelware *CiGri*.

It is thanks to the close collaboration between IT ressources of Whisper and the infrastrucrures of the University that this project has been implemented as we show below.

*FP7 ERC Advanced grant 227507, see whisper.obs.ujf-grenoble.fr

3 Software for data-intensive processing (6-12§, 1-2 pages)

A part of the Whisper project is specificaly dedicated to the IT codes. This includes to design a specification, an implementation and some optimisations of a sequence of a data-management and of a computations [†]. This project uses some own IT resources (servers, dedicated bay) but also uses common IT infrastructure of the university. We developed also adaptations for the IT infrastrutures and we provide technical support for researchers.

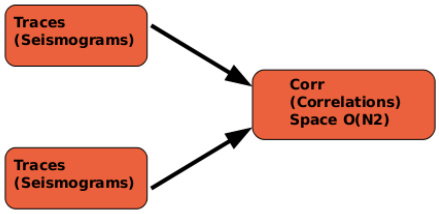
Most of the IT codes are writing with the *Python* language and uses intensively the scientific libraries *Scipy* (fortran and C embeded) and *Obspy* [‡] (essentially the 'read' function). The latest one is dedicated to the seismological community.

The IT codes consists of several tools described schematically at the figure 2 and grouped into three parts. The first one concerns the signal processing, the second part permits the computation of the correlations and the last part consists of codes for the analysis of the correlations.

A first package provides a flexible way to process raw data, to specify a pipeline of pre-processing of signal. The user starts by specifying a directory, a set of seismic stations and a set of dates. Then the codes scan the directory and extracts all pieces of seismograms (also called traces) and rearranges them in a specific architecture of files in order to calculate the correlations to the next step. We use here intensively the function 'read' of the library Obspy which allows to open most of format files seismogram. The user also define his own sequence of processings. He can use the functions predefined but also the Python libraries he needs. Moreover he can add eventually his own processing.

The second package concerns correlations. Roughly speaking, a correlation is an operation with two seismograms (for a given time windows) that provides the coherent part of the two seismograms (associated to the 2 stations) which is the seismic waves that propagate between the 2 stations. (Moreover, in some favorables cases, it converges to the Green's function). Thus, the code computes all the correlations and provides an architecture of files that corresponds to all the couples of seismograms (for each date).

Figure 1: Step of the correlations



Note that the space complexity is linear for seismograms processing but quadratic for the correlations. We have therefore to store the seismograms processing before compute the correlation in order to benefit of the good complexity. These quadratic space complexity can be critical and lot of effort was made in order to optimize the computation in two direction. First we improve the computation of the fast fourier transform by pre-calculating some "good" combinations of small primes numbers. With this method, we improve of forty percent the time computation in the favorable cases.

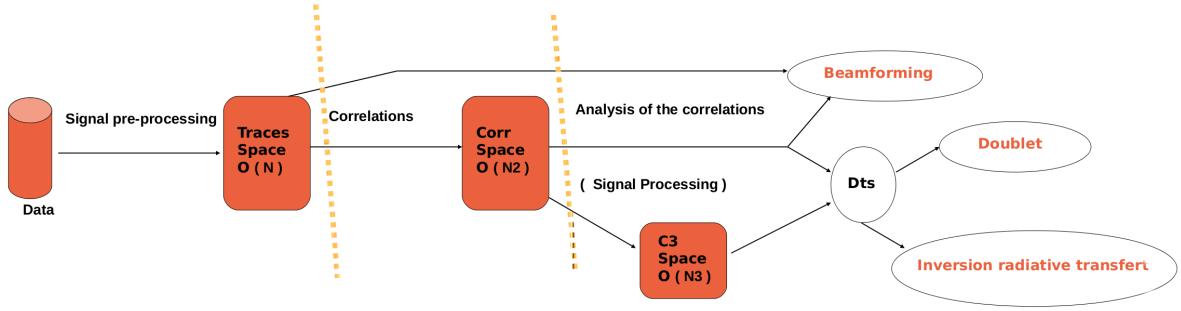
Nevertheless, the main optimization was made by testing the behaviour of the carbage collector of Python in order to follow the cache heuristics. More precisely, we do not use the gc module or the 'del' statement but we try to schedule and localize the line of code in order to find the good unfolding that uses the architecture optimally.

The last part of computer codes concerns the analysis of correlations (the virtual new seismograms) with methods such as beamforming, doublet or inversion. We also compute correlations of correlations C3 (also new seismograms). For example, we study the variations in velocity of seismic waves as we illustrate below.

These codes permit to process a dataset on a computer laptop. Nevertheless, to take advantage of IT infrastructure at the University of Grenoble, adjustments have been made for the grid computing as we shall see later.

[†] see code-whisper.isterre.fr/html/ (part of the design)
[‡] see www.obspy.org

Figure 2: Main sequences of processings of the Whisper Codes



4 IT infrastructure for grid computing (6-12§, 1-2 pages)

4.1 CiGri and iRODS synergy

The data-intensive processing needs obviously an IT infrastructure in order to couple storage and computation. In our cases, most of the processing are embarrassingly parallel. The amount of data and the location of compute nodes available suggests using a distributed storage system and a grid manager.

The IT infrastructure used here is provided by the *Ciment* [§] platform. Ciment is the High Performance Computing (HPC) center of the Grenoble University. It offers a partial pooling of computing resources (10 computing clusters, 6600 cores and some GPUS) and many documentations for users. Moreover, the computational resources are integrated in a local grid of supercomputers. Associated with a distributed storage, it provides a local data grid environement.

The distributed storage accessible by all the computing nodes of all the clusters is managed by iRODS [¶]. Nowadays it represents approximately 700 TB. The grid computing is managed by the *CiGri* [‖] *middleware*, that is part of the OAR [**] project (the Resource and Job Management System on which CiGri relies). CiGri and iRODS together build a complementary solution for embarrassingly parallel computations with large input/output distributed data sets.

Furthermore, whith unitary parametric jobs that are reasonnably short in time, CiGri can deal with the best-effort mode provided by OAR. In this mode, grid jobs are scheduled on free resources with a zero priority and may be killed at any time when the local demand of resources increases. This CIMENT organization (independant computing clusters glued together with a best-effort grid middleware and a distributed storage), in place for more than a decade, has proven to be very efficient, allowing near one hundred percent usage of computing resources thanks to small jobs being managed at the grid level.

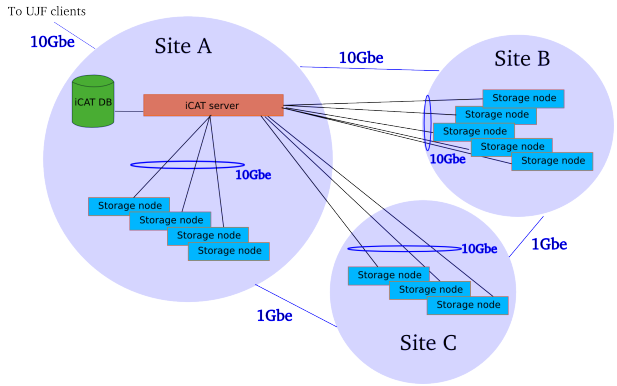
Furthermore, as the results of the grid jobs are stored into the distributed storage with a unique namespace, iRODS also acts to the user as a centralized controller with a total observation and thus allows the user to monitor its calculation.

4.2 iRODS infrastructure

The Integrated Rule-Oriented Data System (iRODS) is a Distributed File System (DFS) offering a single namespace for files that are stored on differents resources that may be on different locations. The administrator can set up rules (microservices) to perform some automatic actions, for example the storage of a checksum or an automatic replication to the nearest resource (staging). The user can control himself replications and create user-defined metadata. iRODS exposes a Command Line Interface (the i-commands), an API useable from several programmation languages (C, python, PHP,...), a fuse interface, a web gui, and a webdav interface. The meta-catalog is an SQL database, which makes it very efficient for managing additionnal meta-data or making advanced queries (see [1] for an illustration of use). It is not "block-oriented", and thus relies on underlying Posix filesystems. Performance is not the main goal, but when the files are distributed on different resources, the only bottleneck is the meta-catalog (which is centralized).

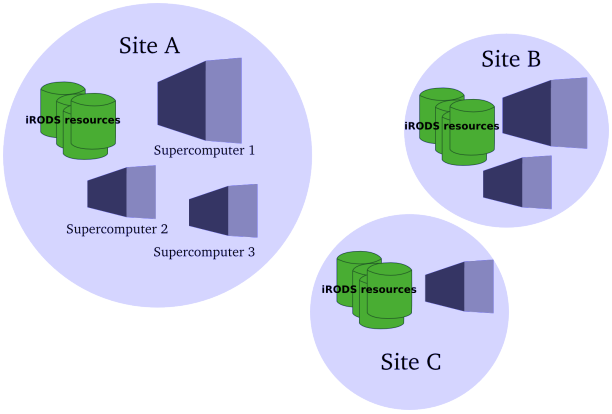
Figure 3: iCAT and storage nodes

[§] see ciment.ujf-grenoble.fr
[¶] see irods.org
[‖] see ciment.ujf-grenoble.fr/cigri/dokuwiki
[**] see oar.imag.fr



The iRODS storage infrastructure of Ciment consists of a unique zone with the iCat server and a dozen of nodes as illustrated on figure 3. The nodes are groupped inside 3 different locations, called site A, site B and site C, having heterogeneous WAN connexions. Each site has it's own 10Gbe local network switch.

Figure 4: iRODS resources close to supercomputers

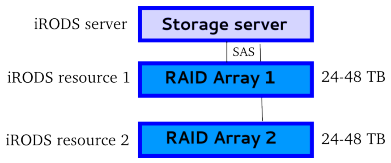


Those 3 sites are located into the 3 main datacenters where the CIMENT supercomputers live, so there are always close storage resources with the computing nodes (Figure 4). All resources of a given site are groupped into an iRODS resourceGroup and a rule tells that if a data is to be written from a computer of this site, then the data is written by default on a resource randomly chosen inside this group. So, data are always written to a local iRODS resource, using the LAN and not the WAN. Note that site C has only 1Gbe WAN connexions while A and B have 10Gbe WAN connexions (Figure 3). So, to optimize, we've set up automatic staging for site C: when a data is get from site C and the meta-catalog tells that the file is located on a site A or site B resource, then the file is automatically replicated to a resource of site C so that if it is accessed again later, it is not more transfered through the 1Gbe WAN link.

Capacity has now reached 700 TB and is constantly evolving and increases with investment in new projects, as iRODS offers a great scalability by simply adding new storage resources. Each node has currently 2 RAID arrays from 24 to 48 raw TB as illustrated at the figure 5

[††] see http://taktuk.gforge.inria.fr/kanif/

Figure 5: A storage node



iRODS nodes are running Debian GNU/Linux with Kanif [††] for easy synchronisation of the system administration. CIMENT has set up a web interface where the user can easily check the status of the ressources (figure 6).

Figure 6

4.3 CiGri infrastructure

The access to 6600 cores of the clusters of the Ciment platform is achieved through the middleware CiGri. Cigri launches embarrassingly parallel jobs on idle processors of every computing clusters and then optimizes the resources usage which are used for parallel jobs otherwise.

Each cluster of the University of Grenoble uses the resource manager OAR. Cigri acts, among other things, as a metascheduler of OAR. It retrieves the clusters states trough OAR and submits the jobs on free resources without exhausting the local scheduling queues.

While it may work in normal mode, CiGri is mostly used in best-effort mode and thus provides an automatic resubmission mecanism. CiGri offers a customizable notification system with a smart events management. With those mecanisms, the user can submit a big amount of small jobs, called a campaign, and forget about it until all the jobs of the campaign are terminated or CiGri notifies a serious problem.

Roughly speaking, in order to run a campaign, the user describes through a file (in the JSON format) the parameters of the campaign such as the accepted cluters, the needed resources, the maximum duration, the location of the codes, a prologue or epilogue script,... Codes and input data are retrieved from iRODS using i-commands into the prologue scripts and the jobs scripts (or using the iRODS API if the jobs are written into a supported language). So, there's no direct connexion between CiGri and iRODS, but the usage is totally complementary through the jobs scripts. Moreover, the user defines also a file where each line represents a value of the parameter for the user's code. Thus,

the number of line of these parameter file corresponds to the number of jobs of the campaign.

Users may monitor their campaigns and act on it whith a CLI or a REST API. Some statistics are provided, such as the completion percentage (in the term of number of terminated jobs), jobs execution rates, automatic re-submissions rate. When a job fails with a non-zero exit status, the user is notified by mail or jabber and requested for an action before submitting further jobs on the same supercomputer: simple aknowledge, aknowledge and re-submission or abort the campaign. Standard and error outputs of the jobs may be easily retrieved from the CiGri host without having to log on the underlying supercomputers. Users may even not be authorized to log-on to a specific supercomputer but allowed to start and manage best-effort jobs on it thanks to CiGri.

CiGri is now at the version 3, which represents a major evolution in terms of modeling and technology (Rest API, Ruby). It is structured around a PostgreSQL database and high level components (Ruby scripting language, Apache with SSL authentication, Sinatra,...).

4.4 Authentication and security

CIMENT has a centralized LDAP infrastructure. Users have the same login on all supercomputers and on the CiGri frontend. As iRODS does not offers a simple and direct LDAP authentication mechanism, we use the simple password method with an automatic synchronisation from our LDAP server to the iRODS database. We also have a script that automatically initializes the iRODS unix environment directly into the users home directory (.irods/.irodsEnv and .irods/.irodsA files) on every supercomputer, so that iRODS authentication becomes completely transparent to the users.

Each site has a filtering router acting as a firewall. As we want all communication schemes to be possible between each irods resource (a file might be transfered from a resource to another regardless of the site), we had to open some tcp and udp ports on those firewalls. The range of ports may be defined by the administrator into the iRODS servers configuration file, so that's not an issue.

5 Results and feedback (28-31§, 6-7 pages)

5.1 Whisper Use Case

1. Scientific whisper results: theses, articles, prices
2. Organisation with a DataSet: Evaluation local/Ciment, capacities adaptation students
3. Comparison data management, example Japan processing/USArray
4. Evolution specification, genericty of the codes, degree of automatization
5. First step: Conversion data format, distributions of datas on ressources: decrease the concurrency
6. Back to the first package signal processing for Japan
7. Identification seismogram: Dates, stations / Modelization distribution: sublist Dates, Stations
8. Adding python modules for data retrieve + Calls.
9. Focus on encapsulation queries
10. The file parameters
11. the file start.bash
12. The jdl file
13. Back to the package correlation: similar adds
14. Focus on the distribution problem (Space complexity/Transfer)
15. Mode besteffort: increase of datas transfert, it possibly adds also new steps of data storage.
16. Some order of magnitude for the Japan Computation.
17. On the Japanese results.

Whisper is one of the projects that have made it possible to ensure that the seismic noise brought new observable. This permits to carry out several scientific results including imaging and monitoring. Concerning monitoring, further studies provide news results about sligth variations of seismics waves induce by earthquakes. Many articles are in part due to this project [‡‡] as well as several post-docs and PhD.

Most of the time, on the computer part, the approach with researchers is as follows. After retrieving data from a data center or directly between person, we have to assess how this data can be processed. According to the computing time and storage capacity, either we perform operations on a dedicated bay (also host by Ciment), or either we use the Ciment infrastructure. It depends also on the ease of computer users and most of the time, at least the last treatments (less computationally expensive) are made locally (Some IT codes are also provided in order to retreive results on distributed storage).

We focus now on the part of processing that use the data grid environment of Ciment. But note before that an other aspect, and not least, is the evolution of the specification. Often, students and researchers have new requirements and the IT codes evolves with these specification. We try to be as generic as possible in order to, among other things, to achieve a sufficient level of automatization. However, sometimes we need to develop some parts specifically because of lack of time. These IT problems of specification and development time are among the most complex to evaluate for this type of project.

A first step, for the IT part of the scientific workflow, begins by storing data and also by checking their integrity. We also require that data be in a seismic standard formats. If conversion is necessary, it can be a large data intensive computing and specific codes are developed. To minimize concurrency, we have also to ensure that the data are well distributed on iRODS. Indeed, it happened that the data are too centralized on a resource, so that is truly diminished processing capacity. Some python codes are dedicated to this task and can replicate or spread randomly a Collection from a set of ressources to another set.

Let us come back to the package of signal processing of Whisper (the first part of the figure 2). Because each seismogram can be treated separatly it is the simplest case for data grid process. We treated for instance one year of the Japansese seismic Network (HiNet, Tiltmeter and FNet) around of the giant 2011 Tohoku-oki earthquake (6 months before and 6 months after). Note that first we need to convert 9 Terabytes of Japanese Data into around 20 terabytes of a standard format (here mseed or sac). Then we try many processing for the 20 TB (filter, whitening, clipping, ...) and store the results in seismogram with a duration of one day.

As almost all seismological data, the Japanese data are identified by the dates, names of seismic station and subnetworks. Therefore the choice of the modelization, in order to retrieve and distribute the data, follows these seismological metadatas. More precisely, the modelization of the distribution of the computation is made by setting a subset of dates and a subset of stations (This corresponds also to a normal use for researchers that wants to test some processing rapidly with a subset of seismograms). Note also that the distribution is constant with respect to transfer.

In order to get a set of seismograms from the iRODS storage to a computational node, we add Python modules to the Whisper package of seismogram processing. These modules contains classes that permits to retrieve a subset of seismograms (defined by the two subsets described above). More precisely, a first step is either to test directly existence of data or either building of hash tables in order to know the available seismograms. Then we provide an iter-

[‡‡] see whisper.obs.ujf-grenoble.fr, rubric publication

ator (in this case, a generator for Python) on available data in order to use other methods that performs the i-command 'iget'. The same approach is made for the storage of the results (with the 'iput').

Note that these commands of transfer between iRODS and computational nodes can become very difficult to achieve because of the concurrency of the queries. To take into account this obstacle, one develops a module that provides lot of encapsulations of the i-commands (number of try, waiting time, resubmission, with error, 'else' command, etc...). The IT infrastructure provide also wery usefull encapsulations.

We have also to adapt our process for the grid computation with CiGri. We first define a file of parameters 'param.txt' where each line correponds to the parameters of one job on the grid, for instance:

```
cat param.txt

traces160_0_8_0 160 0 8 0
traces160_0_8_1 160 0 8 1
...
traces160_1_8_0 160 1 8 0
traces160_1_8_1 160 1 8 1
...
```

Here we divide the dates in 160 sublists and the stations in 8 sublists. The line "traces160_0_8_1 160 0 8 1" correponds to a job named 'traces160_0_8_1' where we take the sublist of dates of index 0 and the sublist of stations of index 1 (each sublist have the same length +/-1). (There are also other parameters for components and networks not described here.)

For each of the 1280 jobs, a script is launched by CiGri , say 'start.bash', that take for arguments a line of the file parameter. The script, among other things, load necessary library like appropriate python, retrieve the code on iRODS, here 'main.py' and run the code (This is a diagrammatic view).

```
cat start.bash

#!/bin/bash
set -e
...
module load python
...
DirToCompute=$1
cd DirToCompute
...
iget -rf Codes
cd Codes
...
NumberDate=$2
IndexDate=$3
NumberStation=$4
IndexStation=$5
...
python main.py NumberDate
     IndexDate NumberStation IndexStation
...
cd
rm -rf DirToCompute
```

Then one defines the json jdl file (job description language), say 'processingSeismogram.jdl'.

```
cat processingSeismogram.jdl

{
  "name": "test_processing",
  "resources": "core=1",
  "exec_file": "$HOME/start.bash",
  "param_file": "param.txt",
  "type": "best-effort",

  "clusters": {
    "c1": {
        "prologue": [
        secure_iget -f /IrodsColl/start.bash,
        secure_iget -f /IrodsColl/param.txt,
        ... other lines of commands ],
        "project": "whisper",
        "walltime": "00:20:00"
        },
    "c2": {
        "prologue": [
        ... lines of commands ],
        "project": "whisper",
        "max_jobs": "450",
        "walltime": "00:30:00"
        },
    ...
    }
...
}
```

Here the campaign named 'test_processing' take only one core. It run the file 'start.bash' with the parameter file 'param.txt' in mode besteffort. It uses the clusters named 'c1' and 'c2' for the project 'whisper' with the duration defined by 'walltime'. In the prologue, we retrieve script and the parameter file on iRODS. One can add other parameters for each clusters such that the maximum number of jobs runing at the same time (the variable 'max_jobs').

Concerning the package for the correlations (the second part of the figure 2), we also add similar modules to retrieve on iRods the subsets of seismograms that have been treated. In order to compute all the correlations that corresponds to all the couple of seismograms we have two types of processes. For a fix subset of dates, either we retrieve one subset of stations and compute all the existing couples for this subset, or either we retrieve two disjoint subsets of stations and compute all the correlations where each seismogram is in a different sublist. This distribution for the computation of the correlations offers a good granularity.

With this kind of distribution for the stations, the transfer bewteen iRODS and the computational node becomes proportional to the distribution, i.e. the number of sublists of stations. Therefore we have to maximize the distribution of the dates and minimize it for the stations in order to obtain a reasonable walltime. Note that it is possible to improve the transfert. However, a fairly simple improvement have to effect that the distribution become dependent on the distributed architecture of computation. In order to keep genericity of the codes we do not change this aspect.

The mode besteffort also increase the transfert of data because some job are killed and submit in an other place. Moreover in some cases, such that a big walltime, we add new steps of iRODS storage for the process in order to store intermediate calculation (It is not the case for correlation because of the good granularity). Note that this may represent a significant development effort.

5.2 Infrastructure Ciment experiences

1. Whisper: an opportinuty to test and try to improve IT Infra for the data-intensive case
2. Other project which permits also some advanced? Or "In genral each scientific case brings its own needs of new specification, functionality.
3. Add new functionality with cigri V3: more control for the user (max job, env variable, customize cluster)
4. Intro Irods: Data-intensive processing adress specific questions for Irods compare to files system
5. i-commands encapsulating to deal with timeouts and connection control (secure_iget)
6. Specific rules for specific Irods server (dedicated data)?
7. Specific policies to acces files?
8. Irods server overload with a lot of small file tranfers (schema client/irods server/icat)
9. Network overload with automatic staging
10. Finish: Despite the problems, this allows processing of terabytes of data within hours.

References

[1] Gen-Tao Chiang, Peter Clapham, Guoying Qi, Kevin Sale and Guy Coates, Implementing a genomic data management system using iRODS in the Wellcome Trust Sanger Institute. *BMC Bioinformatics,12(1):361+, September 2011.*