Ganga analysis framework: procedures, use case design and system capabilities

Cristian De Santis is presenting material of
Armando Fella and Andrea Galvani
on behalf of
SuperB Distributed Computing group

Presentation Layout

- Ganga framework overview
- Analysis framework design
- Use case modeling
- Ganga SuperB plugin
- Work planning and conclusion

Ganga overview

- **Ganga is a user-friendly job management tool.**
  - Jobs can run locally or on a number of batch systems and grids.
  - Easily monitor the status of jobs running everywhere.
  - To change where the jobs run, change one option and resubmit.

- **Ganga is the main distributed analysis tool for LHCb and ATLAS.**
  - Experiment-specific plugins are included.

- **Ganga is an open source community-driven project:**
  - Core development is joint between LHCb and ATLAS
  - Modular architecture makes it extensible by anyone
  - Mature and stable, with an organized development process

Thanks to J.Moscicki for stolen slides

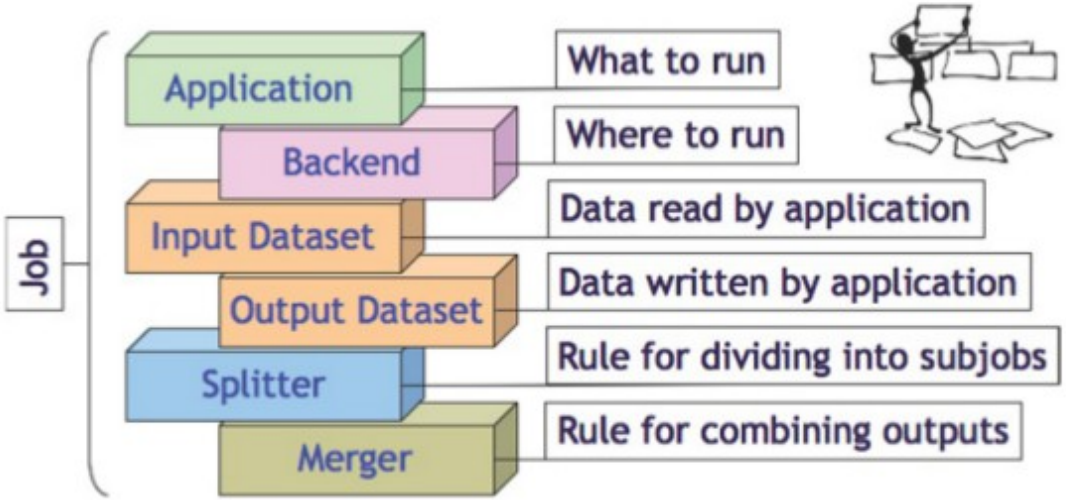
What Users Want

- **Users want:**
  - Development on the laptop; full analysis on "The Grid™".
  - To get results quickly, utilizing all of the resources available, wherever they are.
  - A familiar and consistent user interface to all of the resources.
- **Users don't want:**
  - To know the details of the grids or the resources.
  - To learn yet another tool in order to access some resources
  - To have to reconfigure their application to run on different resources.

"configure once, run anywhere"

Submitting a Job with Ganga

What is a Ganga Job?



Run the default job locally:

```
Job().submit()
```

Default job on the EGEE grid:

```
Job(backend=LCG()).submit()
```

Listing of the existing jobs:

```
jobs
```

Get help (e.g. on a job):

```
help(jobs)
```

Display the nth job:

```
jobs(n)
```

Copy and resubmit the nth job:

```
jobs(n).copy().submit()
```

Copy and submit to another grid:

```
j=jobs(n).copy()
j.backend=NG()
j.submit()
```

Kill and remove the n^{th} job:

```
job(n).kill()
job(n).remove()
```

Hello World Examples

- 1. Hello World Locally
  - j = Job()
  - **j.backend = Local()**
  - j.submit()
- 2. Hello World on Nordu Grid
  - j = Job()
  - **j.backend = NG()**
  - j.submit()
- 3. Hello World on EGI
  - j = Job()
  - **j.backend = LCG()**
  - j.submit()

You may use all features of a first-class programming language to write complex scripts

loops, ifs, variables files, math, network modules,...

Some handy functions

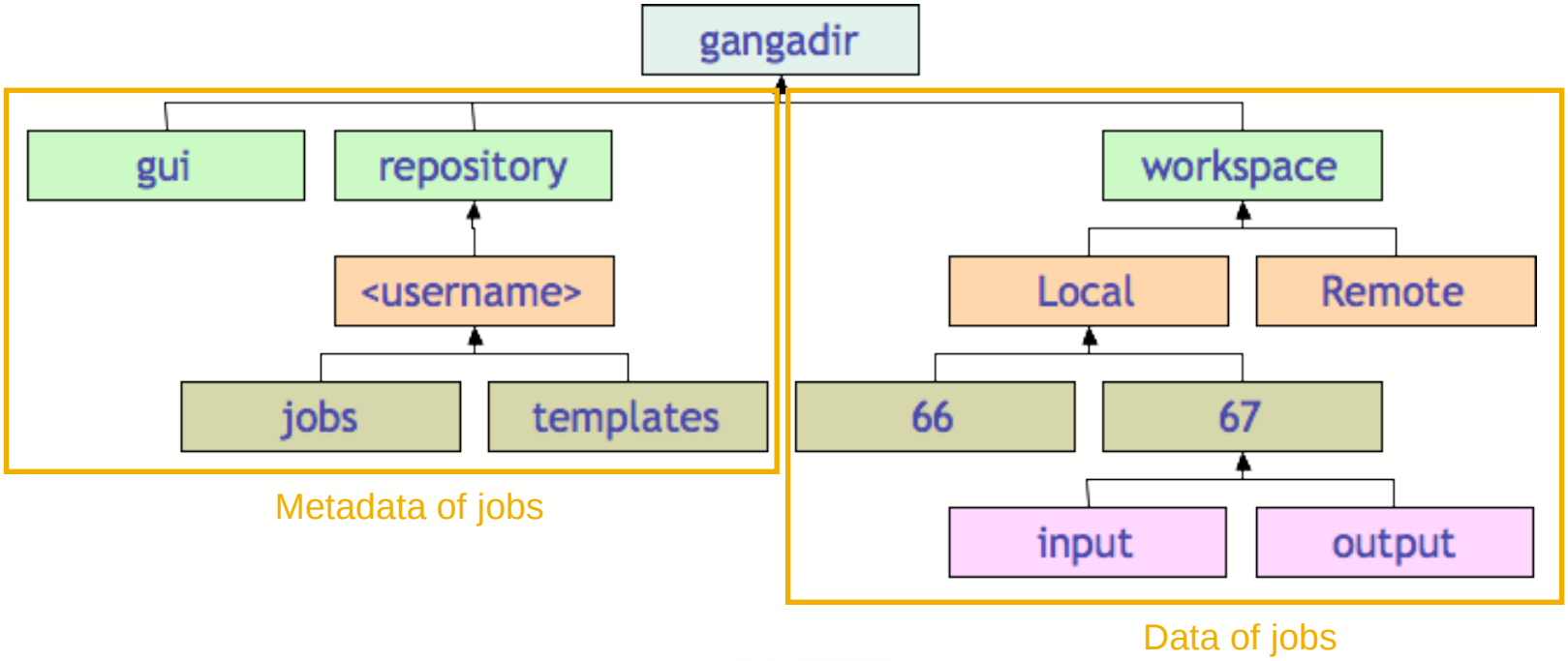
- <tab> completion
- <page up/down> for cmd history
- system command integration
- Job template
- In[1]: plugins()
  - plugins('backends')
- In[2]: help()
- etc.

```
In[1]: j = jobs[1]

In[2]: cat $j.outputdir/stdout
Hello World
```

```
In[1]: t = JobTemplate(name='lcg_simple')

In[2]: t.backend = LCG(middleware='EDG')

In[3]: templates
Out[3]: Statistics: 1  templates
--------------
#   id      status        name   subjobs
application          backend
    backend.actualCE
#    3    template  lcg_simple
Executable              LCG

In[4]: j = Job(templates[3])
In[5]: j.submit()
```

Job persistency: ~/gangadir

- Job-related files and information are kept organized in the Ganga work directory: ~/gangadir
- It is created at the first launch in user home directory



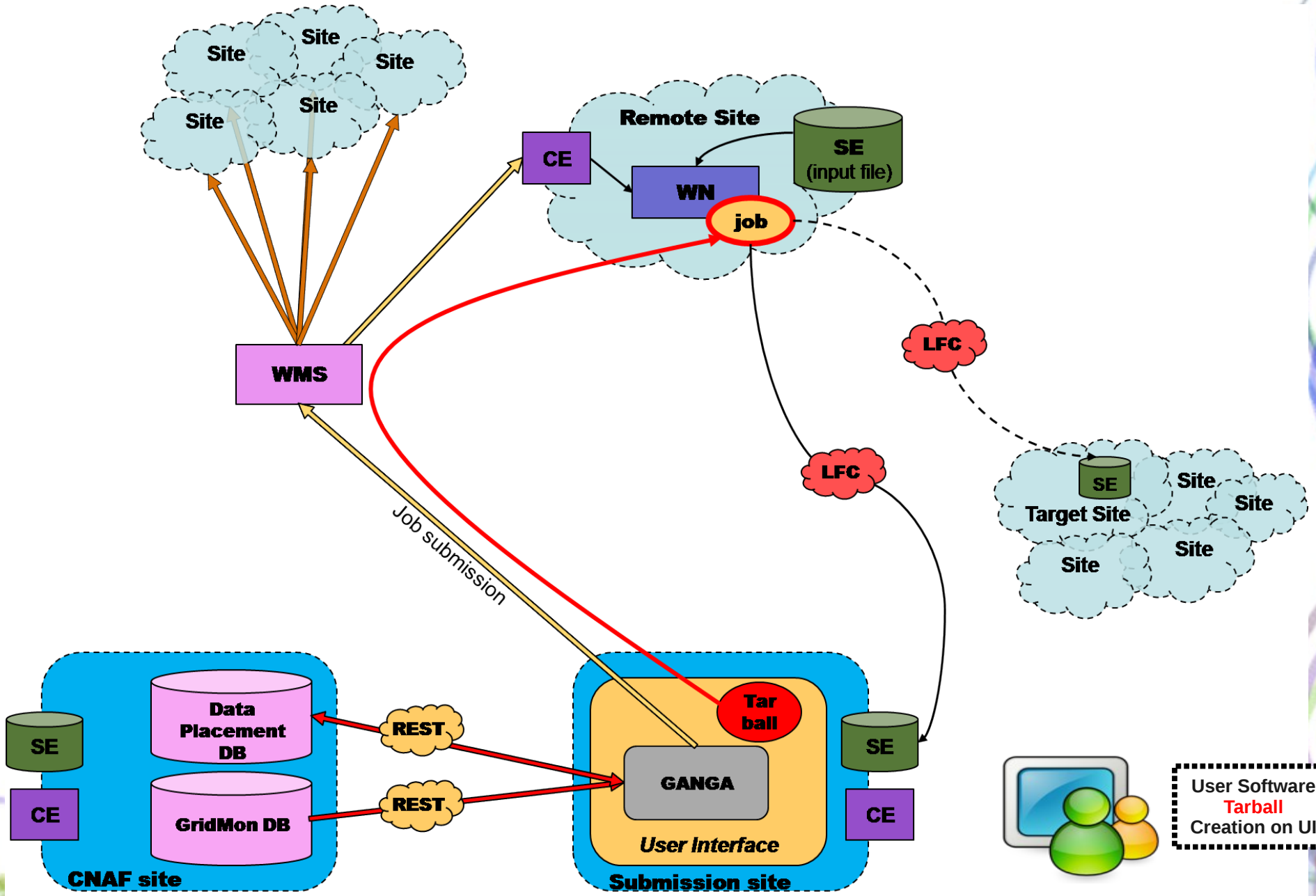
Analysis step by step I

- **Working on User Interface (UI)**
  - User creates his own working directory including the executable script, configuration files and all other things needed for analysis or personal production job
  - Executable script must comply some constraints explained later on input and output file placement on remote node
- **Job preparation from ganga interface**
  - User launches Ganga from UI. He/she can select the input dataset, declare the events to be processed per job, optionally can create the output dataset, define work dir and executable path, finally submit the bulk job. Ganga system will compress and tar the working directory.
- **Job run time and stage-in procedure**
  - SuperB job wrapper decompress user software package, checks the environment, transfers requested input files to the conventional area on worker node and launchs the user executable script.

Analysis step by step II

- **Stage out**
  - At job completion, all the output files residing into the conventional area will be transferred and registered in GRID to output site (default: submission site). A text file containing the list of logical file names (LFN) job output files is transferred to UI via outputsandbox together with the job logout file.
- **Job Monitoring and output retrivial**
  - From ganga interface users can:
    - check jobs status
    - manage jobs: kill, resubmit, copy, etc.
    - check output files and inspect gangadir
- **Data movement**
  - User can perform data transfer of its own dataset via a set of specific methods implemented into SuperB ganga plugin.

Analysis job workflow



Use cases

- Analysis/reduction:
  - Official FastSim/FullSim production dataset analysis
  - Personal FastSim/FullSim production dataset analysis
  - Generic analysis dataset analysis
- Personal Simulation Production (FastSim and FullSim)
- Dataset management
  - Monitor, research, creation, deletion, status management
  - Dataset transfer

Dataset Manager

- Provides the following methods:
  - **createDataset()** job output dataset creation. Interactive guide to dataset creation is available
  - **deleteDataset()** to delete empty ('prepared') dataset
  - **downloadDataset()** to retrieve all files belonging to a dataset from GRID to submission machine
  - **badDataset()** to set dataset status to 'bad'
  - **closeDataset()** to set dataset status to 'close'
  - **openDataset()** to set dataset status to 'open'
  - **showDatasetDetail()** show all metadata of your datasets

Dataset status

- Prepared
  - New dataset is in 'prepared' status
  - Can be deleted
- Open
  - Dataset with at least one output file associated
  - A dataset automatically become 'open' at first submission
- Closed
  - No more usable as output dataset
  - It is not possible append new output files
- Bad
  - Cannot be used as input dataset in analysis job
  - An automatic cleanup procedure will delete such a dataset

Official production analysis (FastSim)

- **Create a new job object and give it a name:**

```
j = Job()
j.name = 'myJob'
```

- **Assign SBApp as the application used by the job:**

```
j.application = SBApp()
```

- **Set the directory where all the job related files (executable, configuration, etc.) are located. Ganga will create a tarball and ship it in the inputsandbox:**

```
j.application.sw_directory =
'/storage/gpfs_superb/users/ganga_util/GangaSuperB/test/analysisSoftwa
re'
```

- **Executable *relative* path wrt user job package eg: ./analysisExe.sh**

```
j.application.exepath = 'analysisExe.sh'
```

- **Choose your work session type between analysis and personal production:**

```
j.inputdata = SBInputAnalysis() # j.inputdata = SBInputPersonalProduction()
```

Official production analysis (FastSim)

- **Input dataset selection:**

```
j.inputdata.getDataset()
```

- **You can filter the results interactively or identify univocally the dataset via dataset_id:**

```
j.inputdata.getDataset(
       prod_series='2010_September_311', prodscript='generic')
j.inputdata.getDataset(dataset_id='4f394214a328d55f2900003b')
```

- **Example of Ganga output:**

```
+----+--------------------+------------+--------------+------+---------------------+---------------+--------+
| id | prod_series        | prodscript | generator    | dg   | tcl                 | analysis_type | status |
+----+--------------------+------------+--------------+------+---------------------+---------------+--------+
| 0  | 2010_September_311 | Generics   | B+B-_generic | DG_4 | MixSuperbBkg_NoPair | HadRecoil     | closed |
| 1  | 2010_September_xyz | Generics   | B+B-_generic | DG_4 | MixSuperbBkg_NoPair | HadRecoil     | closed |
+----+--------------------+------------+--------------+------+---------------------+---------------+--------+
choose dataset: 0

Chosen dataset details:
analysis_type: HadRecoil                          |    generator: B+B-_generic
creation_date: 2012-02-13 18:10:49.885510         |    id: 0
dataset_id: 4f394214a328d55f2900003b              |    occupancy: 121915466273
dg: DG_4                                          |    occupancy_human: 113.5GiB
evt_file: 50000                                   |    prod_series: 2010_September_311
evt_tot: 94500000                                 |    prodscript: Generics
evt_tot_human: 94.5M                              |    status: closed
files: 1890                                       |    tcl: MixSuperbBkg_NoPair
```

Official production analysis (FastSim)

- **The framework permits to perform a unique bulk submission composed by n subjobs.**
- **The interface will ask for:**
  - **How many events need to be analyzed ?**

```
Insert the number of overall events (zero for all):
enter an integer or (q)uit: 510000

Total job input size: 673.8MiB
Total selected number of events: 550.0K
Total number of involved logical files: 11
```

  - **Adjust your job to GRID constraints:**

```
Insert the maximum number of events for each subjob.
- maximum output file size is 2GiB.
- suggested maximum job duration 12h.
- maximum input size job is 10GiB.
- at least 50000 (number of events per file).

enter an integer or (q)uit: 270000
```

Official production analysis (FastSim)

- **If you need to create an output bookkeeped dataset containing your submission output, instruct Ganga to assign that output files to an (existing) dataset.**

```
- create a new dataset using DatasetManager class
- j.outputdata = SBOutputDataset()
- j.outputdata.setOutputDataset(dataset_id, file_type)
```

- **Shell interactive example:**
  - **In the case each subjob creates multiple files owned by different dataset, the file name per dataset should be reported to the system:**

```
Enter a output file pattern: HadRecoil.root (wild card accepted)
```

  - **Ganga will print a summary of your 'prepared' and 'open' datasets, so you can choose one of them.**

```
Choose the dataset:
+----+-------------------+------------+--------------+------+---------------------+---------------+--------+
| …  | …                 | …          | …            | …    | …                   | …             | …      |
| 34 | myOutputDataset01 | Generics   | B+B-_generic | DG_4 | MixSuperbBkg_NoPair | HadRecoil     | open   |
| 35 | test_01_name      | Generics   | B+B-_generic | DG_4 | MixSuperbBkg_NoPair | HadRecoil     | open   |
+----+-------------------+------------+--------------+------+---------------------+---------------+--------+
enter an integer or (q)uit: 35
```

- **Then assign a backend and submit:**

```
j.backend=LCG()
j.submit()
```

Personal Production (FastSim)

Create a new job object and give it a name:

- `j=Job()`
- `j.name = 'myJob'`

Assign SBApp as the application used by the job:

- `j.application = SBApp()`

Set the directory where your sources are located. Ganga will create a tarball and ship it in the inputsandbox:

- `j.application.sw_directory = '/storage/gpfs_superb/users/<user>/FastSim/V0.3.1_test/'`

Executable *relative* path after software unpacking. eg: analysisExe.sh

- `j.application.exepath = 'workdir/pacBtoKstar0NuNu-dg4-3-ganga.csh'`

Personal Production (FastSim)

Assign SBInputDatasetProduction to the job.

- `j.inputdata = SBInputDatasetProduction()`

Set number of jobs.

- `j.inputdata.number_of_subjobs = 3`

This method try to guess your software version. If it finds FastSim software it asks for background frame as input.

- `j.inputdata.findSoftwareVersion()`

Local machine backend, really useful for testing purpose (with little number of jobs). Recommended.

- j.backend=Local()

Grid backend:

- `j.backend=LCG()`

Now you could type j for summary

- `j`

If it's all ok, type j.submit()

- `j.submit()`

Example 'j' output

```
In [4]:j
Out[4]: Job (
 status = 'running' ,
 name = 'myJob' ,
 inputdir = '/home/SUPERB/user/gangadir/workspace/user/LocalXML/50/input/' ,
 do_auto_resubmit = False ,
 outputdir = '/home/SUPERB/user/gangadir/workspace/user/LocalXML/50/output/' ,
 outputsandbox = [] ,
 id = 50 ,
 info = JobInfo (
    monitoring_links = [] ,
    uuid = '68574699-086-1331750007-8' ,
    submit_counter = 1 ,
    monitor = None
    ) ,
 outputfiles = [] ,
 inputdata = SBInputPersonalProduction (
    input_mode = 'dir' ,
    input_path = [u'lfn:/grid/superbvo.org/production/FastSim/bkgframe/1326811309'] ,
    runSite = ['INFN-T1', 'RAL-LCG2', 'IN2P3-CC'] ,
    number_of_subjobs = 3 ,
    source_sim_type = 'FastSim'
    ) ,
```

Example 'j' output

```
 merger = TextMerger (
    files = [severus.log, output_files.txt] ,
    compress = True ,
    ignorefailed = True ,
    overwrite = False
    ) ,
 inputsandbox = [] ,
 application = SBApp (
    exepath = 'workdir/pacBtoKstar0NuNu-dg4-3-ganga.csh' ,
    sw_directory = '/storage/gpfs_superb/users/<user>/FastSim/V0.3.1_test'
    ) ,
 outputdata = None ,
 time = JobTime (
    timestamps = '
         Time (UTC)   Status
- - - - - - - - - - - - - - - - - - - - -
2012/03/14 18:33:27 - new
2012/03/14 18:33:28 - submitting
2012/03/14 18:34:10 - backend_running
2012/03/14 18:34:10 - submitted
2012/03/14 18:34:14 - running

 splitter = SBSubmission (
    ) ,
 subjobs = 'Registry Slice: jobs(50).subjobs (3 objects)
--------------
    fqid |    status |      name | subjobs |    application |        backend |                          backend.actualCE |  exitcode
-------------------------------------------------------------------------------------------------------------------------------
    50.0 |   running |     myJob |         |     Executable |          Local |                 bbr-ui.cr.cnaf.infn.it |      None
    50.1 |    failed |     myJob |         |     Executable |          Local |                 bbr-ui.cr.cnaf.infn.it |         1
    50.2 |   running |     myJob |         |     Executable |          Local |                 bbr-ui.cr.cnaf.infn.it |      None
' ,
```

- **Backend = LCG()**

Analysis software setup

- User prepares his software directory containing executable script and all files needed for analysis or personal production purpose.
- At job run time on remote node the following environment variables should be used by user executable:
  - WN_INPUTFILES : points to the directory where input files are downloaded.
    - User executable should refer to such a directory for input file access
  - WN_OUTPUTFILES : points to directory where user executable should write its output files to be registered on grid.
  - WN_INPUTLIST : points to a txt file in the $WN_INPUTFILES directory which contains the input files absolute path list.
- The user executable script should comply these information to run correctly.

Stage Out

- Every SuperB subjob will return via outputsandbox
  - output_files.txt that contains the list of LFNs registered in the grid.
  - Job log output files
  - Whatever user need to be transferred back within 50MB of occupancy
    - Next command example will printout the output_files.txt belonging to each subjobs.
      - **!zcat $j.outputdir/output_files.txt.gz**

- If you wish to use the output files of one bulk job (eg: personal production output) as input of another analysis bulk job, you should create a 'dataset' which contains that job's output files.

Stage Out

Stage out operations are the same independently by use case:

- **Output sandbox** can be used for smaller files (< 50 MB):
  - **j.outputsandbox = ['graphs.root']**
  - 'graphs' directory residing in relative path with respect the node job home dir will be transferred to gangadir on User Interface
- **GRID registration**: every file your executable script put in WN_OUTPUTFILES will be registered in GRID file catalog.

  /grid/superbvo.org/analysis/<user_identity_from_certificate>/<date_idjob_tarballname_jname>/output/subjobid_filename

- On **CNAF** filesystem:
  - **/storage/gpfs_superb/sb_analysis/ <user_identity_from_certificate>/<date_idjob_tarballname_jname>/ output /subjobid_filename**

Monitoring

- Job registry: a list of all your jobs
  - fqid: ganga job id
  - status: current job status
  - name: job name (j.name = 'myJob')
  - subjobs: number of subjobs in a bulk submission
  - application: what job runs
  - backend: where job runs
  - backend.actualCE: hostname of the local machine, or 'hostname of the site' if backend is LCG
  - exitcode: job exit status

```
In [5]:jobs
Out[5]:
Registry Slice: jobs (1 objects)
--------------
    fqid |    status |      name | subjobs |    application |        backend |           backend.actualCE |  exitcode
-------------------------------------------------------------------------------------------------------------------
      84 | completed |HelloWorld |         |     Executable |          Local |     bbr-ui.cr.cnaf.infn.it |         0
      85 |    failed |      test |         |          SBApp |            LCG |     lpsc-cream-ce.in2p3.fr |         1
      86 |   running |     myJob |       3 |          SBApp |          Local |                            |      None
```

SuperB Ganga plugin: documentation

- Ganga official documentation:

  http://ganga.web.cern.ch/ganga/user/html/GangaIntroduction/

- Wiki: user guide

  http://mailman.fe.infn.it/superbwiki/index.php/Distributed_Computing/Ganga_setup_for_SuperB

- Epydoc: technical documentation from docstrings

  http://bbr-serv09.cr.cnaf.infn.it:8080/doc/ganga/

- SuperB plugin is integrated in Ganga's help.

  Inside a session simply type help() at the prompt:

```
In [1]:help()
***********************************
This is an interactive help based on standard pydoc help.

Type 'index'  to see GPI help index.
Type 'python' to see standard python help screen.
Type 'interactive' to get online interactive help from an expert.
Type 'quit'   to return to Ganga.
***********************************

help>
```

Conclusions

- SuperB Ganga Plugin is under heavy development. User feedback is essential to ensure Ganga has all functionality required for distributed analysis.

- We are looking for beta testers: participants are strongly encouraged to try using Ganga themselves, and to get in touch if they run into problems or have suggestions for improvement.
  - We are proposing to form a “Focused group” of users interested in collaborating in Ganga SuperB framework testing

- Send e-mail to SuperB ganga list:

  superb-ganga@lists.infn.it

Thanks all and backup slides

How to start Ganga at CNAF

- Obtain a certificate and setup your home on UI:
  http://mailman.fe.infn.it/superbwiki/index.php/CNAF_services/How_to_access_Grid_resources
- Add to your ~/.bashrc file the following line:
  - `alias ganga="${VO_SUPERBVO_ORG_SW_DIR}/ganga/bin/ganga_wrap.sh"`
- Then start ganga:

```
user@bbr-ui ~ $ ganga

*** Welcome to Ganga ***
Version: Ganga-5-7-7
Documentation and support: http://cern.ch/ganga
Type help() or help('index') for online help.

This is free software (GPL), and you are welcome to redistribute it
under certain conditions; type license() for details.

In [1]:
```

- **Ganga CLI: Ipython shell interface (http://ipython.org/)**

Ganga Scripting

```
#!/usr/bin/env ganga
#-*-python-*-
import time
j = Job()
j.backend = LCG()
j.submit()
while not j.status in [‘completed’,’failed’]:
    print(‘job still running’)
    time.sleep(30)
```

```
./myjob.exec

ganga ./myjob.exec

In [1]:execfile(“myjob.exec”)
```

GPI & Scripting

You may use all features of Python programming language to write complex scripts loops, ifs, variables files, math, network modules, ...

First Ganga Job

```
In [1]: !vi myscript.sh
```

```
#!/bin/sh
echo "hello! ${1}”
echo $HOSTNAME
cat /proc/cpuinfo | grep 'model name’
cat /proc/meminfo | grep 'MemTotal'
```

```
In [2]: !chmod +x myscript.sh
In [2]: j = Job()
In [3]: j.application = Executable()
In [4]: j.application.exe  = File(‘myscript.sh’)
In [5]: j.application.args = [‘ganga’]
In [6]: j.backend = Local()
In [7]: j.submit()

In [8]: jobs

In [9]: j.peek()
In [10]:cat $j.outputdir/stdout
```

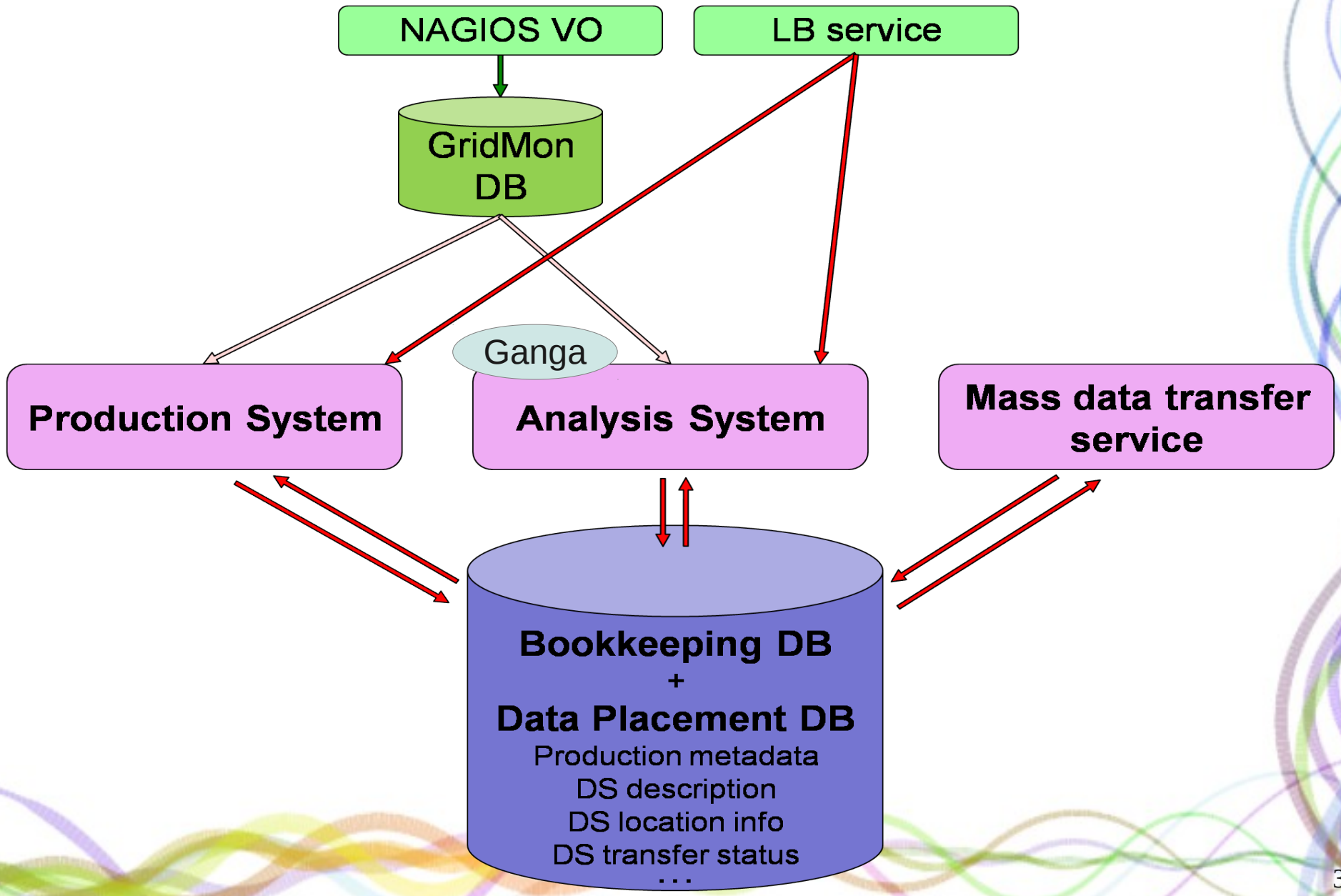
```
./myscript.sh ganga
```

First Ganga Job on the GRID

```
In [11]:j = j.copy()
In [12]:j.backend = LCG()
In [13]:j.application.args = ['grid']
In [14]:j.submit()
```

```
# job registry: a list of all your jobs
In [5]:jobs
Out[5]:
Registry Slice: jobs (1 objects)
--------------
    fqid |    status |      name | subjobs |    application |        backend |          backend.actualCE |  exitcode
-------------------------------------------------------------------------------------------------------------------
      50 |   running |     myJob |       3 |          SBApp |          Local |                           |       None
      84 | completed |HelloWorld |         |     Executable |          Local |    bbr-ui.cr.cnaf.infn.it |          0
```

High level scenario



Input and output sandbox

- You may specify additional files which will be copied to the worker node as the inputsandbox.
- The outputsandbox specifies which files should be copied from the worker node to the submitter machine (UI)
- This mechanism works for small files (less than 50 MB).

```
j = Job()
j.application.exe=File('spy')
j.inputsandbox = [File('~/extra_file')]
j.outputsandbox = ['b.dat','a*.txt']
j.submit()
```

Scripting mode

```
$ ganga athena \
--inDS myInputDataset.txt\
--outputdata myOutput.root \
--split 3 \
--maxevt 100 \
--lsf \
jobOptions.py
```

quick

CLIP mode

```
j = Job()
j.application=Athena()
j.application.prepare()
j.application.option_file='jobOptions.py'
```

application

```
j.inputdata=DQ2Dataset()
j.inputdata.type='DQ2_LOCAL'
j.inputdata.dataset="myInputDataset.txt"
```

inputdata

```
j.outputdata=DQ2OutputDataset()
j.outputdata.outputdata=['myOutput.root']
```

outputdata

```
j.splitter = AthenaSplitterJob(numsubjobs=3)
j.merger = AthenaOutputMerger()
```

Splitter & Merger

```
j.backend = LSF()
j.submit()

j2 = j.copy()
j2.backend=LCG( CE='ce102.cern.ch:2119/jobmanager-lcglsf-grid_2nh_atlas' )
j2.submit()
```

flexible